CodeWalk: Facilitating Shared Awareness in Mixed-Ability Collaborative Software Development

VENKATESH POTLURI*, Paul G. Allen School of Computer Science and Engineering at University of Washington, USA

MAULISHREE PANDEY*, University of Michigan School of Information, USA

ANDREW BEGEL, Microsoft Research, USA

MICHAEL BARNETT, Microsoft Research, USA

SCOTT REITHERMAN, Microsoft Research, USA

COVID-19 accelerated the trend toward remote software development, increasing the need for tightly-coupled synchronous collaboration. Existing tools and practices impose high coordination overhead on blind or visually impaired (BVI) developers, impeding their abilities to collaborate effectively, compromising their agency, and limiting their contribution. To make remote collaboration more accessible, we created CodeWalk, a set of features added to Microsoft's Live Share VS Code extension, for synchronous code review and refactoring. We chose design criteria to ease the coordination burden felt by BVI developers by conveying sighted colleagues' navigation and edit actions via sound effects and speech. We evaluated our design in a within-subjects experiment with 10 BVI developers. Our results show that CodeWalk streamlines the dialogue required to refer to shared workspace locations, enabling participants to spend more time contributing to coding tasks. This design offers a path towards enabling BVI and sighted developers to collaborate on more equal terms.

CCS Concepts: • **Social and professional topics → People with disabilities**; • **Software and its engineering → Collaboration in software development**; • **Human-centered computing → Synchronous editors**.

Additional Key Words and Phrases: software developers, collaboration, blind or visually impaired, accessibility, sound effects, workspace awareness

ACM Reference Format:

Venkatesh Potluri, Maulishree Pandey, Andrew Begel, Michael Barnett, and Scott Reitherman. 2022. CodeWalk: Facilitating Shared Awareness in Mixed-Ability Collaborative Software Development. In *The 24th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '22), October 23--26, 2022, Athens, Greece.* ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/3517428.3544812

1 INTRODUCTION

Synchronous software engineering activities like pair programming and code walkthroughs are useful for developers to share knowledge, improve and refactor the source code, and debug the code together. Developers have to remain *closely synced* to achieve effective collaboration and communication in these activities. If a developer moves to a new location in the code *i.e.* line, function or file, their collaborator should follow them immediately; real-time edits should

*Both authors contributed equally to this research.

Manuscript submitted to ACM

become apparent right away to enable quick feedback. When collocated, sighted developers work together on one system to observe and discuss the source code without expending additional effort to stay on the same page. However, referencing a collaborator's screen is inaccessible for blind or visually impaired (BVI) developers, often requiring them to drive the collaboration on their computers [53].

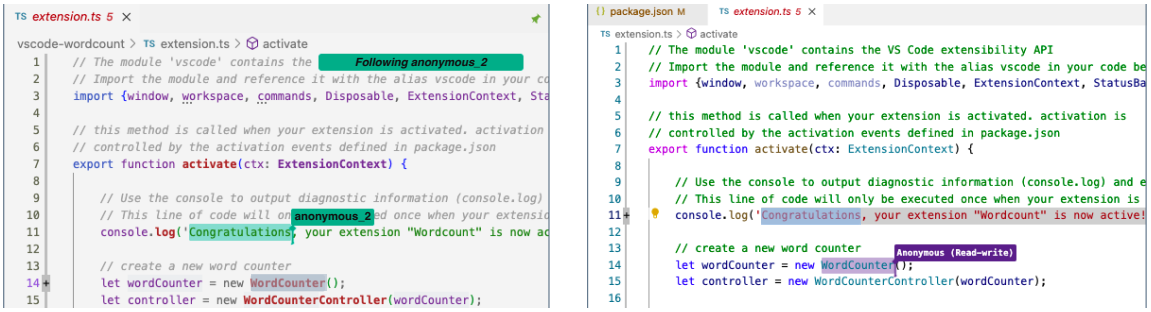


Fig. 1. VS Code is an IDE that offers integrated collaboration support through its Live Share extension. Live Share enables developers to work together on source code through document sharing and co-editing in their respective IDEs. It represents collaborators' location and selection in the source code through colorful cursors.

This screen inaccessibility is magnified in remote synchronous collaboration. Developers typically either use screen shares or integrated development environments (IDEs) with integrated collaboration support (e.g. VS Code, JetBrains, Floobits, CodeTogether, etc.) to work synchronously (see Figure 1). These approaches assume that everyone can see their screens [53, 73]. However, BVI developers cannot access the screen share video or the visual awareness cues in IDEs through assistive technologies such as screen readers. They have to constantly request that their sighted colleagues speak code locations, such as line numbers, functions, file names, etc., out loud, in order to stay in sync. Much like collocated collaboration, BVI developers end up driving the activity. Sometimes, they even hand off their computer's control to sighted colleagues in refactoring and debugging tasks, which reduces their own agency.

That task of providing accessible awareness information lies at the heart of facilitating effective remote collaborations in mixed-ability contexts [25, 34, 53]. Research has begun to explore making shared workspaces accessible to BVI users [18, 38]. However, these solutions are intended for general-purpose document co-editing; they do not cater to the unique needs of software engineering tasks like pair programming and code walkthroughs.

Prior to our work, no programming environment with *accessible*, *remote*, synchronous co-editing support was publicly available. We have created and released such an environment, in accordance with four design criteria (see §2.4), which include maintaining the agency of BVI developers and reducing their burden to drive the collaboration.

In this paper, we present *CodeWalk*, a set of features added to Microsoft's Live Share VS Code extension[1] (available to all Live Share users since November 2021), with support for remote, synchronous code review and refactoring tasks. Our design derives from an investigation of relevant literature in remote collaboration and a set of formative design activities led by a BVI developer and researcher on the team. During our design process, we compared techniques for capturing a collaborator's navigation, editing, and referential (i.e., pointing at or highlighting parts of the code) activities and presenting them to BVI users using a combination of sound effects and speech (see §4.1). We evaluated CodeWalk in a within-subjects controlled experiment involving 10 BVI professional developers (see §5). Our results show that CodeWalk increased study participants' awareness of their collaborator's actions and reduced the coordination overhead

[1] https://docs.microsoft.com/en-us/visualstudio/liveshare/use/enable-accessibility-features-visual-studio-code

required to sync on code locations. Participants strongly preferred CodeWalk over the baseline — the unextended version of VS Code with Live Share which provides awareness cues visually (see §6).

Our work is an end-to-end demonstration of how to improve the accessibility of an IDE's remote collaboration features. We make the following contributions to the HCI, accessibility, and software engineering design communities.

(1) A design for supporting tightly-coupled synchronous programming activities (see §3 and §4).
(2) CodeWalk, an implementation of a set of features added to Microsoft's Live Share VS Code extension that makes synchronous programming tasks accessible to BVI developers (see §4).
(3) Validation of our design's capability to increase shared awareness and facilitate efficient synchronization during remote collaboration, meeting our design criteria (see §5 and §6).

The COVID-19 pandemic has exacerbated the need for collaborative programming environments that can enable BVI developers, one of the largest physical disability groups of software developers [67], to participate in remote work at par with their sighted peers. CodeWalk addresses this timely need and provides a foundation for future software engineering tools to facilitate accessible collaborations.

2 BACKGROUND AND RELATED WORK

2.1 Awareness for Sighted People in Remote Collaborations

Groupware is only effective when it supports collaboration across time and space constraints in a shared workspace [32]. Buxton describes a shared workspace in terms of (1) person space [14], (2) task space [14], and (3) reference space [13]. Person space offers a strong sense of copresence with remote collaborators. For instance, teleconferencing platforms combine video, audio, and even chat to convey facial expressions, gestures, and spoken messages. Sharing the task space refers to being copresent in the context of the task itself. In collaborative software development activities, the shared source code forms the task space. Reference space is where the person and task overlap [13], allowing remote participants to gesture and point to reference one another as well as the task at hand. An example is using text highlighting during screen shares to direct collaborators' attention to specific details.

Dourish and Bellotti identified two approaches to present awareness information in shared workspaces — active and passive [20]. Active approaches include role assignment (e.g. owner, reviewer, editor, etc), audit trails, annotations, and messaging. Active mechanisms require explicit action on collaborators' part (e.g. leaving a comment on the shared artifact like document or source code). Conversely, conveying a collaborator's whereabouts and edits automatically in real-time is a passive approach — collaborators do not have to make explicit efforts to communicate their actions.

Scholars and practitioners have blended different kinds of shared workspaces with active and passive approaches to communicate awareness information in remote synchronous software development. Consider the example of real-time co-editing of source code. A primary concern is that the shared activity should not introduce bugs, preventing the code from successfully compiling [23]. IDE plugins like FASTDash [7] and Syde [26] summarize the real-time activity of collaborators to help them avoid editing conflicts. They also allow developers to leave annotations to inform collaborators of their actions. Systems like AtCoPE [21] and Collabode [23] allow programmers to concurrently edit the source code, enabling collaboration in a shared task space. CodePilot [76] extends the activities supported in the shared task space to allow collaboration across the software development process — editing, testing, debugging, and even version control.

Real-time activities like pair programming and code walkthroughs impose an additional requirement on developers to remain *closely* coordinated [74]. These activities often rely on explicit role assignment. In one common form of

pair programming, one programmer drives (*leader)* the session and writes or explains the code; the other programmer follows (*follower*) to offer feedback or heed the explanations. To collaborate efficiently and maintain mutually recursive awareness of one another (also known as *shared intentionality* [74]), the participating developers must look at the same regions within the source code. Saros [60], an Eclipse plug-in, displays each programmer's text cursor to communicate their location in the source code and provides a *Follow* mode for programmers to sync their IDE viewports during pair programming and code walkthroughs. D'Angelo and Begel used a novel gaze visualization technique to communicate the lines of code a collaborator was looking at during pair programming [16]. The visualization changed color when both programmers' gaze overlapped, communicating awareness and co-presence passively in shared task space. Their evaluation of the gaze visualization technique revealed that when the collaborators' eye gaze was in sync, they more efficiently spoke about their source code using deictic references (*e.g.* terms such as this, here, that, etc.)

The research discussed above has focused solely on collaborations among sighted developers; they do not report anything about the needs of BVI developers. The tools noted above rely heavily on visual information, which leads to significant accessibility problems in mixed-ability contexts [36, 54], including software development [53].

2.2 Inaccessibility for BVI Programmers in Software Development

Initial empirical studies identified accessibility issues with programming tools, like seeking information in IDEs and challenges in doing UI development [1, 42, 50]. Albusays *et al.* found that IDEs rely on visual aids like indentation and syntax highlighting to structure the source code, which favor navigation for sighted developers [2]. On the other hand, command line interfaces (CLIs) present text in unstructured form [61], which limit BVI developers in navigating the text efficiently. Researchers have designed tools to improve navigation [3, 4, 30, 62] and address the accessibility challenges in tasks like code editing [57] and debugging [68]. Potluri *et al.* designed CodeTalk, a Visual Studio extension, to improve code glanceability, navigation and debugging by combining UI enhancements, speech, and sound effects [56].

Accessibility challenges are further complicated by workplace dynamics and project management practices [28, 58]. BVI developers often reach out to their sighted colleagues to solve breakdowns in programming tools, especially selecting teammates who understand workflows with assistive technologies [71]. However, help-seeking in the workplace can be complicated by the team and organization's attitudes towards accessibility and inclusion [53].

Accessibility and HCI research has primarily studied BVI developers' as individual contributors, resulting in limited understanding about the accessibility of collaborative programming activities. The exceptions are the research efforts focused on creating inclusive learning experiences for novice BVI programmers [29, 35, 41, 48, 49, 69]. Pandey *et al.* were the first to report on the accessibility of collaborative software development [53]. They highlight the sociotechnical challenges in activities like pair programming, code reviews, and code writing that limit the contributions of BVI developers during collaboration. The paper reported that BVI developers have to perform additional work, often invisible to their sighted colleagues [12], to address the accessibility challenges in the workplace [53].

2.3 Accessibility for Mixed-Ability Programmers in Remote Software Development

Given the limited research on awareness needs of BVI developers, we turn to the larger accessibility literature to identify how sighted and visually impaired people achieve *real-time*, remote collaboration. A common approach is asking collaborators to describe their actions, but sighted people often forget to verbalize the relevant details, resulting in incomplete collaborator awareness [17, 53]. BVI people hesitate to repeatedly request information to avoid slowing down the pace of the collaboration or imposing on their sighted collaborators [17].

In another workaround, collaborators work on their respective computers with the BVI developer sharing their screen using a video calling application to enable the sighted developer to follow them [53]. Collaborators rely on chat features to copy-paste text and share line numbers, etc., to collaborate more accessibly. While this workaround allows a sighted developer to track a BVI developer's location, the information is not reciprocated to that BVI developer. It also places the onus of driving the collaboration session on BVI developers. BVI people occasionally have to relinquish control of their computers to let the sighted colleague control their screen and make changes in real-time [73]. Since BVI and sighted computer users navigate interfaces differently [8, 55], the approach causes the screen reader to change focus unexpectedly without feedback to the BVI person, impinging on their agency, and raising privacy concerns [73]. Latency issues also lead to unwieldy drag and click interactions for sighted collaborators.

Another strategy is to use NVDA Remote [75] or JAWS Tandem [63], screen reader addons that transmit announcements instead of simply relaying the video of the collaborator's screen during screen share [9, 73]. Unfortunately, these addons often suffer from long latency, causing the BVI person to receive announcements after 15-30 seconds [73]. Plus, sighted collaborators have to set up the screen reader and the addon with matching configurations at their ends, giving them additional invisible coordination work and adding to the total collaboration time [53]. Tools like Sinter address the latency issues and strict configuration requirements of remote screen reading but have not yet been evaluated in collaborative contexts [10].

Das *et al.* designed auditory representations to support *asynchronous* collaborator awareness for activities such as commenting and editing in shared text documents [18]. The evaluation of their design elements revealed that the use of non-speech audio, voice modulation, and contextual presentation could improve awareness of BVI authors. Recently, CollabAlly [38] and Co11ab [19], both Google Docs extensions, have been designed to support *synchronous* collaborator awareness in shared document editing. The extensions use spatial audio and voice fonts to represent the actions of collaborators joining and leaving the document, addition and deletion of comments, and movement of their cursor into or away from the BVI user's paragraph. Co11ab also uses a variant of Follow mode [60] to sync *collaborators' viewports*. Shared document editing differs from pair programming and code walkthroughs in an important way, however. Shared document collaborators need real-time awareness to actively *avoid* each other's cursors in order to prevent overwriting collisions. Software development collaborators, on the other hand, intentionally work on the same lines of code *together*, requiring close and immediate coordination for extended periods of time.

In summary, the workarounds discussed above insufficiently convey collaborator awareness, suffer from delays, place a disproportionate burden on BVI people for driving the collaboration, and compromise their agency. Furthermore, some of the proposed non-visual techniques [18, 38], which are designed for collaborative writing, will not fulfill the unique needs of synchronous collaborative programming. We begin to address accessibility of collaborative software development by designing design criteria for CodeWalk.

2.4 Design Criteria

Based on this literature review, we synthesize the following design criteria for CodeWalk. We annotate each criterion with citations to the literature that inspired them.

- D1. *to minimize the cognitive load on the BVI developer* [18] (e.g., maintaining accessible workspace awareness [19, 38] while minimizing conflict with collaborators' conversations) during synchronous programming activities
- D2. *to maintain agency of BVI developers* [65] in mixed-ability collaboration [17, 53]
- D3. *to reduce the burden on BVI developers* [73] of driving the collaboration session to accessibly collaborate [53]

Table 1. Descriptions of our code walkthroughs. Each walkthrough occurred between a pair of sighted and/or BVI developers along with a sighted observer watching a shared screen or listening to a BVI developer's screen reader.

ID	Leader	Follower	Sighted Observer Task
CW1	Sighted	BVI	Watched sighted developer's screen
CW2	BVI	Sighted	Watched sighted developer's screen
CW3	Sighted	BVI	Listened to BVI developer's screen reader
CW4	BVI	Sighted	Watched BVI developer's screen

- D4. *to support tightly-coupled collaboration* between all collaborators [74]

3 DESIGN

In this section, we describe the formative design activities we conducted that led to the design of CodeWalk.

3.1 Formative Design Activity 1: Choosing a Baseline IDE

Our first design activity was to choose a good baseline IDE to build upon, one that was already accessible by BVI developers and had facilities for collaborative co-editing support. Several popular IDEs that offer collaborative co-editing support, such as JetBrains CodeWithMe [33], Sublime [24] and Atom [22], are unfortunately difficult to use by BVI developers. A few require BVI developers to perform additional setup steps and others even lack accessibility support for screen reader users to be able to perform basic code editing [72].

By contrast, we found Microsoft's Visual Studio Code IDE (VS Code) to be both accessible and easily extensible. Its accessible command palette makes it easy for screen reader users to find commands. In addition, we learned that the VS Code team speaks with the BVI developer community regularly to improve its accessibility [44, 45]. VS Code supports collaborative work through its Live Share extension. Similar to Saros [60], Live Share supports synchronous collaboration through a Follow mode feature, which draws the leader's cursor in all of the followers' IDEs and keeps it in sync as the leader moves around the document. Live Share also supports co-editing, keeping a shared view of the source code in sync between the connected parties. Though there is little information on the accessibility of Live Share's features for BVI developers, there are enough features to make it a good choice for our project's baseline IDE.

3.2 Formative Design Activity 2: Code Walkthroughs

To assess the accessibility of VS Code with Live Share for teams with BVI and sighted developers, three of the authors (one of whom is also a BVI developer) conducted four code walkthroughs (see Table 1). Two walkthroughs were led by a sighted researcher and two were led by the BVI researcher-developer. We tried to cover all combinations of abilities in a pair along with each taking on a leader or follower role. All of the walkthroughs involved mixed ability teams (CW1, CW2, CW3, and CW4). In all of the walkthroughs, a third sighted researcher observed the shared screen, however in code walkthrough CW3, the sighted observer simply tried to listen to and comprehend the (slowed down) screen reader audio used by the BVI developer.

Each code walkthrough looked at different example source code from VS Code's library of extensions. Each example extension consisted of multiple files written in TypeScript, several files of which were read through during each walkthrough. Each walkthrough was 60 to 90 minutes long.

In addition to recording the code walkthrough sessions, the sighted observer took detailed notes during each code walkthrough, noting down accessibility breakdowns and workarounds. They minimized their interruptions, limiting

questions to clarifications of leader and follow actions to locate one another in each file and to help work around any non-obvious accessibility barriers. In total, the sighted observer took six pages of notes. Immediately after each code walkthrough, both the leader and the follower memoed, reflecting on their experiences [59]. As a group, the entire research team rewatched the sessions from the recording and discussed the memos and notes in their weekly meetings.

We observed that conversations between leader and follower primarily focused on code discussions and clarification questions during breakdowns in accessibility. When in Live Share's Follow mode, the IDE drew each developer's cursor and synced their viewports on everyone's screen. Unfortunately, since this information was only visual, the BVI follower was not aware of any of it and was frequently lost. Consequently, the sighted leader had to speak their location out loud to the BVI follower to help facilitate tightly-coupled collaboration (Design Criterion D4). This active approach was error-prone because the sighted leader sometimes forgot to mention their location, especially when they were navigating quickly around the source code. The BVI developer initiated another workaround, asking clarification questions to sync with the leader. This often put the burden on the BVI developer (Design Criterion D3) to request enough accessible information to follow along in the code walkthrough.

When the BVI developer led the code walkthrough, they never got lost. However, they often became unsure whether the Follow mode had really synced the pairs' viewports, and had to ask their sighted follower to confirm they could see the expected code in their window. Finally, the sighted collaborator often talked at the same time as the BVI developer was trying to listen to their screen reader. This made it difficult for the BVI developer to listen to either audio stream. The research team discussed that some of the audio overlaps could be avoided if the sighted developer knew when the BVI developer's screen reader was speaking. But, revealing the use of AT is a sensitive issue for many screen reader users, thus we decided to designed CodeWalk's features to judiciously and carefully make use of audio effects and speech to reduce the cognitive load (Design Criterion D1) experienced by the BVI developer.

3.3 Formative Design Activity 3: Synthesizing Code Walkthrough Scenarios

Inspired by our literature review and our code walkthroughs, we created 15 scenarios comprising short events that occurred (or we wished had occurred if the IDE were more accessible) across our code walkthroughs. In addition, we considered both sighted and BVI developers as leaders, but skipped scenarios involving solely sighted developers. Some scenarios explore possible communication mechanisms between leaders and followers (e.g., non-verbal, notifying the leader, notifying all collaborators, or not notifying at all and syncing up after the session). All of these scenarios are listed in Table 2.

Here is an illustration of Scenarios 1 and 2, which expose some inaccessible features of the baseline IDE and the design features we explored to address them. Blake, a sighted developer, wants to refactor a piece of code. He asks Mia, a BVI developer and his colleague for advice. They set up an audio call to verbally discuss the code as they view the code in a Live Share collaboration session hosted by Blake. Blake shares the session link with Mia, who joins the session and is presented with Blake's code in her IDE. Mia invokes the Follow mode command to stay in sync with Blake's viewport. As Blake navigates in the IDE, Mia's IDE shows a copy of Blake's cursor in a distinct color, which unfortunately is not accessible to Mia. Though the viewport changes, Mia's cursor remains untethered from Blake's. Therefore, Mia has to occasionally interrupt Blake to ask him to speak his line numbers and keywords out loud so that she can navigate there herself and use her screen reader to read the code that Blake is referring to.

This scenario exposes the limitations and asymmetry of current IDEs in supporting tightly-coupled collaboration and shared awareness (Design Criterion D4) among sighted and BVI developers. To address the asymmetry, CodeWalk automatically tethers a BVI developer's cursor to the leader's (section 4.1), so that their cursors move in unison whenever

Table 2. Mixed ability code walkthrough scenarios that informed the design requirements for CodeWalk. Each scenario was inspired by at least one code walkthrough. Sighted+ and BVI+ indicates more than one developer. Following or watching "on the side" splits the VS Code editor and puts one in Follow mode.

Scenario	Leader	Follower	Walkthrough	Activity
1	Sighted	BVI	CW1, CW3	Follower joins collaboration session hosted by leader.
2	Sighted	BVI	CW1, CW3	Follower tethers cursor to leader.
3	Sighted	BVI	CW1	Follow leader ''on the side'' without tethering.
4	Sighted	BVI	CW1	Follower restarts tethering after watching leader ''on the side''
5	Sighted	BVI	CW1, CW3	Follower tells leader they are lost.
6	Sighted	BVI	CW1, CW3	Follower takes notes during collaborative session.
7	Sighted	BVI	CW1, CW3	Follower fails to notice what command the leader just used.
8	Sighted	BVI	CW1, CW3	Follower asks leader about the command they just used.
9	BVI	Sighted	CW2, CW4	Leader invites follower to join collaboration session.
10	BVI	Sighted	CW2, CW4	Leader jumps to follower's cursor, answers the follower's question, and jumps back.
11	BVI	Sighted	CW2	Leader asks follower to show them something.
12	BVI	Sighted	CW2	Leader asks follower a question to test if they are lost.
13	BVI	Sighted	CW2, CW3	Follower asks for help using a ''I need help'' command.
14	BVI	Sighted, BVI	CW2, CW3, CW4	Leader gets follower's cursor location from VS Code.
15	BVI	Sighted+, BVI+	CW4	Leader gets approximate location of multiple followers from VS Code.

the leader initiates the navigation action. However, this only happens in Follow mode. Now, Mia should normally have no doubts about being in sync with Blake, but can still detach (i.e., turn off Follow mode) from the leader if she wants to explore the code on her own. The feature can also be useful in Scenarios 9 through 13 where she leads the collaboration. She does not have to worry about the correct code segment being displayed in Blake's IDE.

Furthermore, to reduce the burden on BVI developers (Design Criterion D3), to preserve their agency (Design Criterion D2), and minimize cognitive load, we designed several features in CodeWalk to convey a collaborator's location, navigation, and edit actions accessibly to BVI developers using a passive, automated approach. We describe the detailed implementation of these features next.

4 CODEWALK

CodeWalk is a set of features released with Microsoft's Live Share VS Code extension that supports accessible, remote, synchronous code review and refactoring activities. We describe the cursor tethering and audible feedback features that power its capabilities and discuss some of the implementation details that we found needed careful design.

4.1 Features

4.1.1 Cursor Tethering. Live Share's Follow mode yokes each collaborator's editor viewport together, a passive visual mechanism that is inaccessible to BVI developers. CodeWalk facilitates tightly-coupled collaboration (Design Criterion D4) by tethering BVI collaborators' cursors with the host of a Live Share session. In designing this feature, we explored several options to toggle tethering along with various levels of autonomy, ranging from always tethering cursors to only tethering cursors when the user toggles Follow mode.

Always tethering the BVI developer's cursor to their collaborator minimizes their cognitive load (Design Criterion D1), but reduces their agency (violating Design Criterion D2), as the sighted colleague would have total control over

Table 3. Use of audio cues to convey awareness indicators in Follow mode (unless specified otherwise in the row)

Awareness Information	Non-Speech Indicator	Non-Speech Indicator Frequency	Speech Indicator	Speech Indicator Frequency	Built-in Visual Indicator
Viewport scrolls	Click wheel sound	Every scroll event	None	None	Screen scrolls
Scroll direction	Falling or rising tone depending on direction	When scrolling stops	None	None	Can be inferred from the scrolling viewport
Current Viewport	None	None	''Lines X to Y on screen''	When scrolling stops	Visible on screen
Cursor moves by single line to line N	Keyboard click	Every cursor move	''Line N''	1.5 seconds after cursor moves end	Cursor moves on screen
Cursor moves multiple lines to line N	Falling or rising tone depending on move direction	Every event	''Line N''	1.5 seconds after cursor moves end	Cursor moves on screen
Cursor moves by multiple lines to line N	Falling or rising tone depending on move direction	Every event	''Line N''	Every event	Cursor moves on screen
Selection	Depends on selection (keyboard/mouse)	Every event	''Selection on line N''	1.5 seconds after selection is made	Selection visible on screen
Edits on follower's line	Keyboard type	For every character typed	None	None	Cursor moves on screen; edits visible on screen
Edits on follower's line (*Follow mode off*)	Keyboard type	For every character typed	''<collaborator> is editing the same line as you''	As long as edits continue on the same line	Cursor moves on screen; edits visible on screen
Edits within 5 lines of follower (*Follow mode off*)	Proximity sound	For every character typed	''<collaborator> is editing nearby''	As long as edits continue on the same line	Cursor moves on screen; edits visible on screen
Follow status	Pull and push sound	When follower starts and stops following leader	''You are now following <collaborator>''	Every event	None

their BVI colleague's cursor. BVI and sighted developers navigate code and interfaces differently [2, 55, 56]; they may want to read a part of the code that their sighted collaborator is talking about by character or by word, a kind of fine-grained navigation a non-screen reader user has no idea about. To support this need, we support temporarily untethering the BVI follower's cursor whenever they move it around, giving them control to move the cursor to the code they want to read. After 10 seconds of inactivity, CodeWalk retethers the cursors.

4.1.2 Conveying Collaborator Actions via Audio. CodeWalk uses a combination of sounds and speech to passively communicate a tethered collaborator's location and their navigation and edit actions, reducing the burden on BVI developers to ask about them (Design Criterion D3). We explored several sound designs, drawing inspiration from audio cues used by popular accessible navigation apps and operating systems. We experimented with futuristic artificial sound effects as well as skeumorphic sounds of keyboard clicks and scroll wheels. We felt that since BVI developers

were already familiar with the sounds of standard computer hardware, the skeuomorphic sound effects would be the best one to convey navigation actions. Navigation distance and direction lacked obvious skeumorphic analogs, so we designed a set of artificial rising and falling tone sound effects to be played a short time after navigation activity ends. If the user clicks the mouse somewhere else in the codebase, CodeWalk plays an artificial ''teleportation'' sound instead of a mouse click to make it more obvious that something drastic has happened to the cursor location, which may invalidate the mental model the BVI developer has of the region where they thought the cursor was located.

In designing using speech and sound effects, our primary focus is to minimize cognitive load relative to the frequency and specificity of the information to be conveyed. We draw inspiration from accessible data visualization and programming efforts [40, 66, 70] and use speech to announce highly specific information like line numbers, which is needed less frequently. We use sound effects to convey less specific information, such as the actions performed and navigation direction - these actions occur at a much higher frequency during collaboration. The use of speech and sound effects has shown to improve awareness between collaborators [43]. Sound effects minimize cognitive load on BVI developers (Design Criterion D1) because they do not require conscious interpretation and can be heard even when a screen reader is actively speaking. However, they do not give enough information about a collaborator's location. To address this, after navigation activity has stopped for 1.5 seconds, CodeWalk uses computer-generated speech to tell the BVI developer what line of code (and file name if it changed) they are now on. Most sound effects are around 200 ms (though one is longer, at 550 ms). Similarly, speech announcements are kept short and precise. The complete business logic for CodeWalk's sound effects and speech can be seen in Table 3.

Fig. 2. Image shows BVI developer's code editor as she follows a sighted leader. CodeWalk tethers the cursors of collaborators in Follow mode. When the sighted leader uses arrow keys to navigate, CodeWalk plays skeuomorphic keyboard sounds for each line moved. When they stop navigation at line 19, CodeWalk plays an artificial falling tone to indicate downward movement followed by line number announcement. Similarly, when they highlight a word, CodeWalk announces the selection.

Sighted collaborators commonly use visual reference space gestures such as cursor location, text highlighting, and mouse waving to refer to code [16], gestures largely unavailable to BVI developers [36, 54]. CodeWalk supports selection awareness by speaking the portion of code highlighted by a collaborator. This simplifies the process of understanding what a collaborator wants to talk about and reduces the burden on BVI developers to ask sighted colleagues to verbally announce their selections. An example illustrating these sound effects and speech can be seen in Figure 2.

When a collaborator edits the code with their keyboard, sharper, shorter key click sound effects are played. If the collaborators are untethered (i.e., Follow mode is off), then they may both be editing the document simultaneously. The baseline VS Code Live Share gives no indicator that collaborators' edits may collide, other than drawing the two cursors near one another. This, of course, is inaccessible to BVI developers. In CodeWalk, whenever the collaborators are editing within 5 lines of one another, CodeWalk speaks a warning, ''your collaborator is editing nearby.'' If the collaborator is on the same line, the warning repeats, ''your collaborator is editing the same line as you,'' which should hopefully cause the collaborators to stop what they are doing and negotiate their next actions together, verbally.

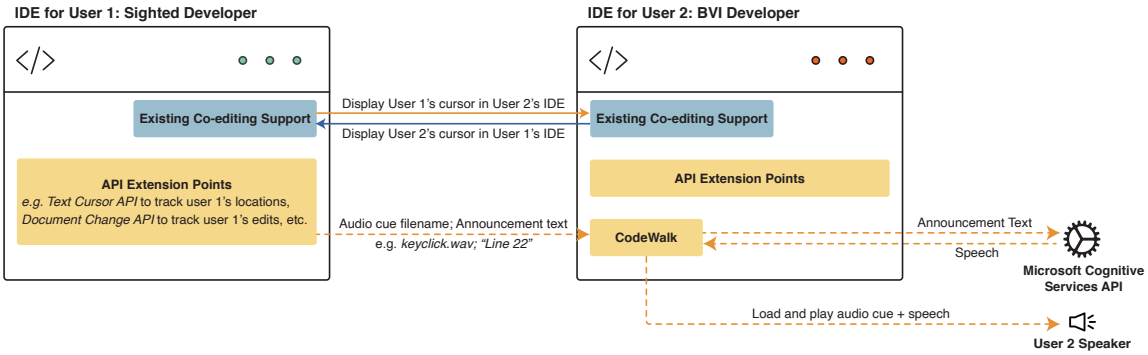


Fig. 3. System Architecture Diagram for CodeWalk

4.2 System Implementation

The basic architecture of CodeWalk can be seen in Figure 3. Each developer runs an instance of the VS Code IDE, extended by CodeWalk. CodeWalk extends four extensibility points provided by VS Code and Live Share (i.e., programming APIs enabling third-party developers to enhance specific features of the IDE), which we illustrate in the following scenario walkthrough.

Mia, a BVI developer (User 2 in Figure 3), installs CodeWalk along with Blake, a sighted colleague (User 1 in Figure 3). Blake and Mia enter into a joint collaboration session facilitated by VS Code Live Share's existing co-editing support. Various extension points are triggered as they collaborate. The first triggers when Blake changes his cursor location. It sends the new location to Mia's IDE along with a tag explaining what action caused it. CodeWalk then runs through its business logic (described in Table 3) to determine the kind of audio feedback to play (sounds and/or speech) and queues them for playback on Mia's computer. Each of Blake's navigation actions may trigger a tuple of one or two sounds along with a spoken message, each separated by a delay. Typically, the first sound is skeuomorphic (*i.e.* for key clicks, mouse clicks, or the scroll wheel). It is followed by a 1.5-second delay, a falling or rising tone (to indicate navigation direction), and an spoken announcement of the new line number. The 1.5 second delay avoids spamming Mia with additional sounds and speech if Blake pauses momentarily during his actions (e.g. pausing to adjust mouse wheel when scrolling through a file). As there are no cross-platform APIs for asking screen readers to generate custom announcements, we generate CodeWalk's spoken announcements using the Microsoft Azure Cognitive Services Text-to-Speech (TTS) API. If too many sounds are requested to be played in a row, queued sounds and speech may be delayed. If they are delayed over one second, it is considered out of date and CodeWalk ejects it from the playback queue. Additionally, CodeWalk categorizes sounds into *notifications* and *warnings*. Events associated with the former are interruptible, meaning if a second event comes in before the first one is done playing, it will cancel the first and start the second right away. *Warning* sounds are uninterruptible. They are reserved only for edit actions to prevent co-editors from overwriting one another's changes.

The second extension point tethers the co-editors' cursors together. When Mia *follows* Blake, her cursor will move automatically wherever Blake's cursor goes. When tethering is turned on, all of Blake's edits will always happen on the same line as Mia's, so we suppress any spoken warnings. When tethering is turned off, edit sounds and announcements only play when Blake is editing within five lines of Mia, else the sheer quantity of sounds would overwhelm her.

Table 4. Demographic characteristics of the participants and their study session details.

ID	Gender	Age	Country	Profession	Condition 1	Condition 2	Programming Language
P1	Male	34	India	Senior Software Engineer	CodeWalk (Set B)	Baseline (Set A)	JavaScript
P2	Male	24	India	Software Development Engineer	CodeWalk (Set A)	Baseline (Set B)	Python
P3	Male	27	India	Technology Analyst	CodeWalk (Set B)	Baseline (Set A)	JavaScript
P4	Male	47	USA	Software Engineering Manager	Baseline (Set B)	CodeWalk (Set A)	JavaScript
P5	Female	29	USA	Data and Applied Scientist	CodeWalk (Set A)	Baseline (Set B)	Python
P6	Male	44	USA	Senior Program Manager	Baseline (Set B)	CodeWalk (Set A)	C#
P7	Male	21	USA	Software Engineering Intern	Baseline (Set A)	CodeWalk (Set B)	Python
P8	Male	46	Sweden	Software Developer	Baseline (Set B)	CodeWalk (Set A)	C#
P9	Male	35	USA	Senior Software Developer	Baseline (Set A)	CodeWalk (Set B)	Java
P10	Male	29	Netherlands	Freelance Developer	CodeWalk (Set A)	Baseline (Set B)	Python

A third extension point tracks and conveys selection actions between the co-editors. Mia hears a verbal announcement of the selection whenever Blake selects some text in his editor, as long as she is tethered to Blake.

The final extension point queues sounds to be played whenever Mia toggles Follow mode on or off. Similar to what happens in Zoom or Microsoft Teams, a sound is played whenever a co-editor joins or leaves the collaboration session, followed by an announcement of the co-editor's name and their cursor location.

5 EVALUATION STUDY

We conducted a within-subjects study to understand and compare the effectiveness of VS Code Live Share with CodeWalk features against our *baseline*, plain VS Code Live Share [46]. Our study aimed at answering the following research questions: (1) How well does CodeWalk improve coordination during remote synchronous collaboration between sighted and BVI developers? (2) How does it affect the communication between developers about the source code? (3) How does it shape BVI developers' perceptions of their collaborative experience?

5.1 Participants

Eligible participants had to be 18 years or older, identify as blind or visually impaired, be comfortable with using screen readers, have at least a year of programming experience in one of the following languages: C, C++, C#, Python, JavaScript, TypeScript, or Java (*i.e.* the programming languages into which we translated our tasks), have collaborated on code, and be able to communicate about code in spoken English. We recruited participants by posting on social media platforms and mailing lists (e.g., Program-L) that primarily comprised BVI developers.

Our study accepted 10 BVI developers (P1–P10). Nine participants identified as male; one as female. Participants were between 21 and 47 years old (average age 33.6; median age 31.5). Table 4 summarizes the details of participants' demographics, country of residence, and current job title. Each participant was compensated with USD $100 (or its equivalent in local currency) for their participation in the study.

5.2 Tasks

We employed a 2x2 within-subjects experimental design. Each participant, in collaboration with a sighted confederate (one of the authors and the study coordinator), performed a series of tasks without CodeWalk (the baseline condition) and another set of tasks with CodeWalk (experimental condition). Like prior HCI studies [11, 31, 51], the confederate

was instructed to strictly and consistently follow the study protocol with all participants, which is known to lead to more generalizable results [27].

We developed two sets of tasks (henceforth, set A and set B) for the study. We randomized the order of task sets and the conditions across the participants. Each set comprised three tasks, resulting in a total of six tasks. We designed the tasks to range from easy to difficult within each set, enabling participants to ease into the programming environment. In both sets, the first task was based on editing a string; the second task required editing code central to program execution; the third task required refactoring a specified set of lines into a function. The research team conducted multiple rounds of discussion to ensure that both task sets were of equivalent levels of difficulty.

The confederate *led* the code walkthrough and asked the participant to *follow* them during the tasks. They asked the participant to recommend changes and solutions to complete each task. Participants could explore, edit, or verify the actions of the confederate as they wished. This made the collaboration feel more natural.

All tasks were based on Hangman,[2] a common text-based game. The tasks were representative of software development activities and required participants to perform code reviews, bug fixing, and code refactoring. We downloaded publicly available source code for Hangman in C♯, Java, Python, and JavaScript so that participants could perform the tasks in their preferred programming language. For internal validity, we selected code samples with similar lengths and modified their source code to have similar file names, function names, variable names, and code structure. All code samples included (1) a main code file representing the game's logic, (2) a text file containing 851 words to play the game, and (3) a text file listing both sets of tasks in the order determined for the participant. The C♯ code sample included an additional file that represented the game's UI; this file was referenced for the string editing task. The code samples in JavaScript included HTML/CSS files which were not required for the study. Table 4 lists the order of conditions and task sets for each participant, along with the programming language used in their study session.

We conducted a pilot study with one BVI developer (not including in the main study) to ensure that each task set was possible to complete within 20 minutes and the total study time did not exceed 90 minutes. Based on their feedback, we found we needed only to improve two things: our instructions on how to connect remotely and the description of the extensions' features in both conditions.

5.3 Procedure

We conducted the studies remotely over Microsoft Teams or Zoom, as per the participant's preference. Participants were not required to turn on their cameras for the study. Before the main tasks, the study coordinator explained the key features of the IDE used in the study, the baseline condition, and CodeWalk. We asked participants to share their screen without including the system audio so that the confederate would not hear the participant's screen reader or CodeWalk audio output. We used the video conferencing tool's recording feature to capture the conversation between the participant and the study coordinator, which we referred to during our analysis.

To facilitate switching between study conditions, we created a Windows 10 virtual machine (VM) with two different versions of VS Code — one version with the baseline condition and another augmented with CodeWalk. Both versions had the same features, keyboard shortcuts, and UI settings. We installed JAWS (version 2020) and NVDA (version 2021) on the remote VM. We also set up Code Factory Eloquence [15], a popular text-to-speech (TTS) synthesizer used to customize screen reader voice and speech. Before each study session, we set NVDA as the default screen reader.

[2] https://en.wikipedia.org/wiki/Hangman_(game)

Participants connected to the VM using Microsoft Remote Desktop software. Upon login, we informed participants that they could modify the screen reader settings. Only P1 and P4 used JAWS; the others performed the tasks with modified settings for NVDA. We also turned on screen recording within the VM to record the screen reader speech. Due to technical glitches with screen recording software, we were unable to record the screen reader usage for P4 and missed a portion of the screen reader usage for P1 and P2.

Participants were instructed to switch to the IDE window for the first condition (see Table 4 for the order) and invite the study coordinator (referred to as the 'confederate' in this paragraph) to the collaboration session. The confederate was under strict instructions to hide the participant's shared screen to not look at their IDE contents. Participants could ask questions about the IDE features or share their comments about the baseline and CodeWalk during the study. We believed this approach allowed the collaboration and the conversation to proceed more naturally. After twenty minutes, the participant and the confederate switched to the other experimental condition to perform the next set of tasks.

After each condition, participants verbally responded to a 12 statement Likert-scale questionnaire (see Table 6). The questionnaire was adapted from existing scales [16, 64] and assessed participants' opinions regarding awareness and collaboration. Participants had to indicate on a five point scale whether they strongly disagreed (1) or strongly agreed (5) with the statements. The study concluded with an informal interview about participants' experience with CodeWalk and a short questionnaire about their personal and programming background.

5.4 Data Analysis

The confederate wrote analytic memos [59] after each study session to reflect on how each condition shaped their awareness and collaboration. One researcher reviewed all the video recordings and conversation transcripts to highlight the timestamps of sync operations, analyzed using a Poisson regression (see §6.1). Section 6.2 discusses how we adapted an existing list of codes from [16] to analyze the conversation between the confederate and the participants. Section 6.3 details our analysis of participants' responses to Likert-scale questionnaire . Lastly, two authors used descriptive coding [47] to analyze the interviews and organized the codes into themes around collaboration and feedback (see §6.3).

6 STUDY RESULTS

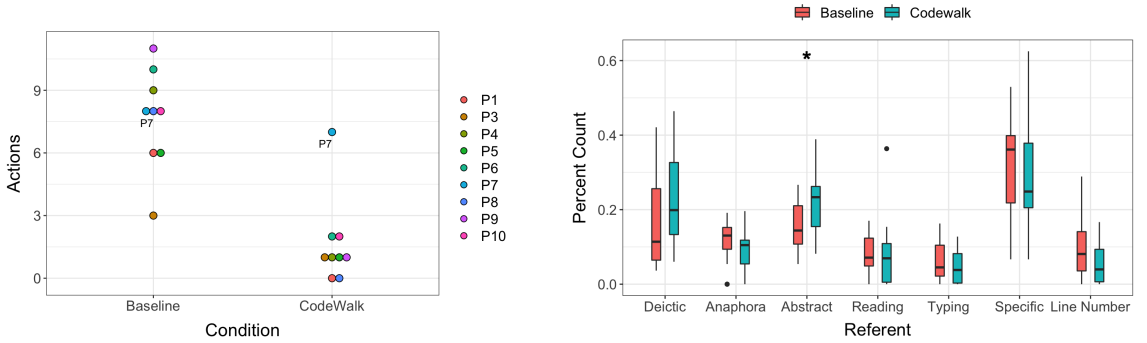
6.1 How well did CodeWalk improve coordination during collaboration?

To analyze how well the participants could follow the confederate, we compared the number of times they attempted to *sync their location* with the confederate's location. We operationalized location syncing as attempts, including successful attempts, by participants to move their cursor to the confederate's location using one of the following: (1) moving from one file to another (2) going from one line to another (3) using the find tool to search for a specific word to navigate to its location (4) toggling the tether command if unsure of the tether status of cursors. Participants synced their locations to read the code that the confederate was referring to and to follow them during the collaboration. We hypothesized that participants would require fewer sync operations in CodeWalk because they would feel *less lost* compared to the baseline. We analyzed the screen share recording and participants' screen reader speech to calculate the total number of sync attempts.

The median value for the number of times participants tried syncing in CodeWalk was 1, compared to the median value of 8 in baseline, a huge drop. Figure 4a a visualizes the number of sync attempts for each participant in both conditions. As recommended for integral data with possibility of rare occurrences, we fit a Poisson regression [77]. We found that participants made *significantly fewer attempts* to sync locations in CodeWalk condition ($p = .000875$

Table 5. Reference Codes and Descriptions

Code	*Description*
Deictic	When a participant or the confederate uses a deictic reference such as this or here, e.g., *"Let's start with this task."*
Anaphora	When a participant or the confederate refers to a past action or location, e.g., *"Can you go back?"*
Abstract	When a participant or the confederate uses a broad category to refer to an object, e.g., *"We need to understand the function."*
Reading	When a participant or the confederate loudly reads a portion of the code, generally done when approximate location of the collaborator is known, e.g., *"Press Enter to leave the game!"*
Typing	When a participant or the confederate is referring to the text being typed, e.g., *"Let me confirm what you wrote."*
Specific	When a participant or the confederate uses a specific name to describe an object, e.g., *"Let's go to didGuessCorrect()."*
Line number	When a participant or the confederate uses a specific line number, e.g., *"I am on line 31."*



(a) Number of times participants attempted to sync locations with confederate in each condition.

(b) Percentage of referents of each type uttered in each condition. An * is shown above referent types that are significantly different across conditions

Fig. 4. Results from video and conversation analysis

< 0.01). The result indicates that CodeWalk enabled the participants and the confederate to stay closely coordinated during collaboration. Note, P7's outlier value in the CodeWalk condition. The followup interview and his screen share recording revealed that he had not realized that his cursor was tethered to the confederate's. He interpreted the sounds and speech in CodeWalk as locations he should move to, resulting in similar behavior across both conditions.

6.2 How did CodeWalk affect communication about the source code?

Since CodeWalk conveyed information on a co-editor's location and actions, we hypothesized that CodeWalk would enable the participants and the confederate to converse about code using more abstract and deictic references compared to the baseline. We also hypothesized that they would use more line numbers and specific names in the baseline condition compared to CodeWalk. Both our hypotheses were informed by D'Angelo and Begel [16].

To analyze how the confederate and the participant referred to locations in the code, we adapted and extended the list of *referents* from D'Angelo and Begel [16] by making two additions (anaphora and reading). Table 5 shows the codes of all 7 referents along with their definitions and examples. Each time a participant or the confederate made a reference to the code, we recorded the referent and its category. We calculated the total number of referents uttered by the confederate and the participant in each study session. Thus, we ended with 14 total referent counts (7 for the

confederate; 7 for the participant) in each condition per session. We normalized the data in each condition by calculating the percentage of referents in each category. We did not include P2's data because he experienced significant lags in screen reader speech causing the confederate and P2 to verbalize and read code aloud for a large portion of the study (an unlikely scenario for collaboration in outside the study). We visualize the fraction of referents in each category for each condition in Figure 4b. The figure shows that *specific* referents were used most heavily during the tasks in both conditions. We also note a greater usage of deictic and abstract referents in the CodeWalk condition compared to the baseline. We carried out a one-way ANOVA to compare the percentage of referents in both conditions. The analysis revealed no significant difference in the percentage values of any category except the abstract referents ($p = .037 < 0.05$), which were greater in the CodeWalk condition.

6.3 How did participants perceive their collaboration experience with CodeWalk?

6.3.1 Responses to Likert-Scale Questionnaire. Table 6 lists the statements in the Likert questionnaire along with the p value of participants' responses in both conditions. All statements had an equal or higher median value in the CodeWalk condition. A higher median indicates more agreement among the participants, implying a better overall experience with CodeWalk.

A one-tailed Wilcoxon signed-rank test indicated that the value for responses to ten out of twelve statements were significantly higher in the CodeWalk condition ($p < 0.05$). Their statement codes are followed by an asterisk in Table 6. The significantly different values confirm that participants *felt* more aware of the confederate's locations and actions with CodeWalk. Furthermore, participants felt that the shared awareness was reciprocated by the confederate when using CodeWalk *i.e.* the participants felt that the confederate was also aware of their actions (S4 in Table 6). This indicates greater shared intentionality with CodeWalk.

Two statements (S9 and S12) were not significantly different across conditions. These focused on participants' perceptions of the confederate's communication style and effectiveness in collaboration during the tasks. Since the confederate remained unchanged in both conditions, participants may have felt that their communication style remained consistent across conditions. Participants' responses to S9 and S12 may have also been subject to *demand characteristics*, cues that shape participants' desire to form a positive impression on the experimenter [52]. Participants may have wanted to appear polite in their responses about the confederate's communication and collaboration abilities.

Table 6. Statements in the Likert-scale questionnaire along with their p values. All statements had equal or higher median value in the CodeWalk condition. * beside the statement code indicates $p < 0.05$.

#	Statement	p value
S1	I was keenly aware of everything in my environment.	.0009*
S2	I was conscious of what is going on around me.	.0035*
S3	I was aware of what my teammate did and how it happened.	.0029*
S4	I was aware that my teammate is aware of my actions.	.0197*
S5	I am aware of how well we performed together in the team.	.0118*
S6	I felt like my teammate and I were on the same page most of the time.	.0118*
S7	I could tell what my teammate was thinking about/looking at/talking about most of the time.	.0328*
S8	I felt like we shared common subgoals as we worked on the task.	.0294*
S9	My teammate communicated clearly during the task.	.1284
S10	I communicated clearly with my teammate during this task.	.0169*
S11	It was fun to work with my teammate on this task.	.0294*
S12	My teammate worked effectively with me to accomplish the task.	.1284

6.3.2 Interview Results. Video analysis revealed that the participants felt aware of being in the confederate's vicinity. They would highlight code or read code aloud to direct the confederate's attention. In addition, we observed that the confederate could easily keep track of the participant's cursor with CodeWalk's tethering feature. The participant's cursor was always visible in the confederate's viewport, and if the participant moved out of the viewport to read code, the confederate would scroll to keep track. On the other hand, participants reported that they *''leaned on the communication''* with the confederate *''pretty heavily''* (P7) in the baseline condition. They had to either wait for the confederate to verbally announce their location using a line number or function name or request the location information to sync cursors, also indicated by Figure 4a.

We noted instances where participants used CodeWalk's tethering feature to direct the confederate. For example, P5 asked the confederate to take her *''to the line again''* to revisit the source code. The confederate moved their cursors to the location P5 had specified; the sounds confirmed arrival for P5. Later on in the interview, P5 shared that the *''auto move [of cursors] was really useful''*. Similarly, P6 directed the confederate to move their cursor to various lines during the refactoring task. After each move, he would explore the code at the destination line, make recommendations for improving the code, and then instruct the confederate to take him to the next location.

Participants used CodeWalk's sound effects extensively to maintain awareness of the confederate's actions. For instance, after the confederate finished typing, P9 mentioned, *''Yeah, I can tell you are done 'cause the typing noise stopped''*, and then went on to verify the changes made by the confederate. Participants used the speech announcements to keep track of location changes. Many participants phrased this as being aware that *''things were happening''* (P8). Even on the occasions when the confederate moved quickly, leading to a succession of sounds, participants felt that *''at least [CodeWalk] conveyed a sense of movement''* (P4). The increased awareness seemed to positively shape the participants' feelings about collaboration and assuaged their worries about feeling lost: *''Because I could just snap to wherever you were, I wasn't worried about wandering off'' — P4.*

Furthermore, participants liked the design choice of primarily using audio cues to convey the confederate's actions and relying on speech sparingly. They shared that the audio cues *''packed a lot of info''* (P7) without seeming verbose. In addition, participants did not seem to mind when the audio cues played simultaneously with the screen reader speech, but they indicated a preference for shorter sounds. Most participants were able to quickly map the skeuomorphic audio cues to their awareness indicators. It took a few participants longer to associate the non-skeuomorphic audio cues with their intended meaning of direction changes. However, they acknowledged that they had not *''used it [CodeWalk] enough''* (P6) to remember the sounds and believed that *''some more sessions''* (P1) would enable them to map all the audio cues to their respective meanings.

Every participant told us that they would like to use CodeWalk to collaborate with their teammates. P5 mentioned that using CodeWalk in code reviews would enable her to be on the *''same page without lagging behind.''* P7 shared that CodeWalk would be *''absolutely instrumental''* in his pair programming assignments, and he would *''install it immediately''* if it were released. P9 felt that it would allow him to mentor junior developers by letting them *drive* collaboration sessions: *''When I'm collaborating, I'm the one driving and I share my screen and they look at it. It's just easier that way [...] I would be much more likely with an extension like this to let them drive more often.''* Participants also appreciated that CodeWalk was built for VS Code, a mainstream and accessible IDE that sighted *''people might have''* (P5). Upon its launch, they could use it without asking their colleagues to switch to a new IDE. These quotes suggest that CodeWalk can enable BVI developers to participate in collaborative activities without requiring them to manually manage the sessions on their own.

6.4 Threats to Validity

Our study employed a sighted research team member as a confederate for all study sessions. Employing a single confederate across all participants is common in HCI [11, 31, 51] and is recommended for maintaining the internal validity of the experiment [27]. Despite following the study protocol strictly, the confederate may have gained experience and improved as a communicator with each session. Thus, it is likely that later participants' collaboration experience may have been better than the former, resulting in fewer differences in metrics between conditions. We believe the within-subjects design choice would have addressed any learning effects on the part of the confederate.

Due to their research experience in accessibility, the confederate may better understand participants' awareness needs than sighted people unused to collaborating with BVI developers. Participants commented that the confederate was *"a very good communicator"* (P7), also confirmed by the lack of significant difference in responses to S9 on the Likert-scale questionnaire (see Table 6). The confederate's communication may have suppressed differences in referent counts between conditions. Therefore, in real-world conditions with more typical collaborators, CodeWalk may show even more improvement in communication metrics over the baseline.

We deployed CodeWalk on a cloud-based virtual machine (VM) to simplify the installation for our participants. Using screen readers through remote VM may have increased latency. Some participants reported lags in screen reader playback which may have impacted their experience with the extension and shaped their feedback. The latency issues are unlikely to occur in real-world conditions, since the extension would be installed on the user's own home system. Thus, we expect the experience of CodeWalk to be better upon its release.

7 DISCUSSION

Overall, we find CodeWalk successfully translates and conveys reference space gestures from sighted developers to their BVI colleagues, extending Buxton's model [13] for effective remote collaboration to mixed-ability collaborations. In this section, we summarize our findings, consider the role of interdependence in our design, and relate our results to the two projects that are most similar to ours. We then reflect on our own research practices and propose future work.

7.1 Summary of Findings

Our study results show that significantly fewer attempts were needed by our BVI participants to sync locations with CodeWalk than in the baseline condition. This suggests that the coordination burden (Design Criterion D3), which often requires explicit communication of awareness cues between collaborators, is reduced through CodeWalk's sound effects and speech. Automating the transmission of code location and navigation actions helps to ensure that the sighted colleague also benefits from a reduced coordination burden, since they need not remember to convey those actions verbally either. The participants' increased use of abstract referents to code locations showed a corresponding decrease in the number of more specific referents (using line numbers and function names). This suggests a reduction in cognitive load (Design Criterion D1) on the part of the BVI developer. It also shows an increased sense of shared awareness and shared intentionality between the participants, which ensures that the BVI developer had the capability to contribute equitably according to their ability rather than be sidelined by inaccessible collaboration tools.

From the Likert scale statements, we learned that participants felt that CodeWalk improved their awareness of their environment, their teammate, and their teammate's actions. They also felt that they were more likely to be on the same mental page most of the time and were able to work effectively together. BVI developers were more likely to highlight text on the screen using their keyboard, confident in the knowledge that their sighted colleagues would be able to

notice it and react to it. BVI participants also could tell from the sound effects when their colleagues were navigating or when they had stopped, concluding that they were now free to engage in conversation and explore the source code. Not only did this increase improved their communication, it also minimized the cognitive load (Design Criterion D1) of trying to intuit what their sighted colleague might be doing without any audible feedback.

CodeWalk's cursor tethering was designed to support tight coupling (criterion D4) between participants at the task level, so that when one navigated through the code or edited some text, the other would immediately be made aware and be able to respond. Some participants responded by directing the confederate to move to additional locations, showing increased agency (Design Criterion D2), looking around the code with their own screen reader, and then driving the confederate to the next code location.

CodeWalk's skeumorphic sounds (e.g., key clicks and scroll wheel sounds) were straightforward for the participants to understand with no training. However, some sound effects, e.g., rising tone and falling tone, used to convey directionality of movement, were not immediately obvious to the listeners. While participants got better at distinguishing these during their study session, others may need more time to get better at this.[3]

CodeWalk's spoken sentences were necessary to orient the BVI participants after their sighted colleagues navigated to new areas in the code. Sometimes, however, these spoken words collided with the participant's own screen reader speech. An early version of CodeWalk played its sounds and speech by extending the NVDA screen reader, which enabled us to detect overlapping speech utterances and cancel one of them. However, to ensure CodeWalk worked with multiple screen readers on multiple platforms (including Mac and Linux), we used Microsoft's Azure Cognitive Services to generate speech and platform-specific sound APIs to play it. It is possible to address the overlapping audio, however due to time constraints, we were unable to program CodeWalk to cancel our audio while the screen reader was talking. We encourage screen reader and operating system manufacturers to offer extensible platform-agnostic APIs for integrated systems like CodeWalk with screen readers.

One interesting form of spoken collision remains. Sighted colleagues receive no indicators when BVI users are listening to their screen readers, and thus do not realize to stop talking to the BVI user to avoid overlapping with the screen reader or CodeWalk. Participants expressed a need for avoiding *''double-speak''* (P8) between the collaborator and speech announcements by their screen readers and CodeWalk. We plan to explore designs of a visual indicator to non-screen reader users of CodeWalk to let them know when screen reader speech is active for any of their collaborators. This feature should require BVI users to opt-in before it is turned on because BVI users' opinions of whether to reveal their use of AT to colleagues varies by culture [39] and may have significant workplace consequences [5].

7.2 Accessible Co-Editing

Lee et al. [37]'s CollabAlly tool developed similar sound effect and speech-based feedback for BVI writers in common co-editing environments (e.g. Google Docs). CollabAlly found success in identifying collaborators' ongoing work and comments in the document, enabling collaborator awareness to avoid overwrites synchronously and asynchronously. Our work examined the awareness needs found in synchronous tasks of code walkthroughs and reviews and found the timeliness of push-based notifications vital to enable BVI collaborators to stay in sync with their sighted colleagues for extended periods of time. Many coding tasks fluidly switch between asynchronous and synchronous modes leaving unanswered how to best support users' cognitive load by conveying awareness information simultaneously using pull and push-based modalities.

[3]See Cat_ToBI (http://prosodia.upf.edu/cat_tobi/en/ear_training/listening.html) to practice distinguishing rising and falling tones from one another.

Das et al. [18, 19]'s Co11ab work supporting mixed ability co-editing stops short of handling collaborators editing near one another. Prior to CodeWalk, these kinds of close edits would preferentially disadvantage the BVI collaborator as their sighted colleagues could see the impending collisions and take their own steps to avoid them. CodeWalk's use of non-interruptible warning messages as colleagues get too close served to encourage all parties to communicate using alternate, more accessible, channels (e.g. a concurrent audio call) in order to appropriately synchronize their edits and avoid conflicts.

7.3 Interdependence

Bennett et al.'s reframing of the goals of assistive technology as *interdependence* instead of *independence* ring true in CodeWalk's scenarios [6]. Collaboration between colleagues of mixed abilities encourages each to play to their own strengths, while requiring that each cede some of their own power and control to cooperate effectively with others. Working together in a code walkthrough or code review, a BVI developer who might have special expertise in accessibility can disseminate that knowledge to sighted non-specialists *in situ* and create a better result for their customers. As shown by Pandey et al. [53], long-term mixed ability collaborators establish mutual reliance by learning how to work together by paying attention, responding to, and adapting to one another's task-related behaviors, habits, and needs. CodeWalk's sound effects and speech events make a colleague's navigation and edit work visible to BVI collaborators, enabling them to be used by a BVI collaborator as an essential assistive technology for remote collaborative work. Finally, CodeWalk challenges the established hierarchy of sighted participants controlling the task, enabling BVI developers to *lead* code walkthroughs and reviews instead of meekly defaulting to follow.

7.4 Researcher Reflections

This work improves our own practice to communicate accessibly and to advocate for our own accessibility when participating in collaborative software development activities. For example, the BVI member of the research team now always asks sighted collaborators to verbalize code locations. A sighted member realized that he needed to remember to stop speaking every so often to allow his BVI collaborator to ''read'' the code for themselves using their screen reader. The study coordinator recognized that each BVI developer's access and communication needs are different and that expressing these needs can be tricky when collaborating with someone for the first time. They have become mindful of attuning their communication to the preferences of their BVI collaborators. Finally, as we collaboratively author this paper using the Overleaf Latex editor, we yearn for it to make use of auditory feedback in order to fully include our BVI co-author in our writing efforts.

7.5 Future Work

In the future, we would like to explore how to combine the lessons learned from CodeWalk, CollabAlly, and Co11ab in supporting mixed ability remote collaboration, whether it be for document co-editing, software co-editing, or additional collaborative software development tasks. In particular, future designs should explore ways that BVI collaborators can most effectively and equitably lead interactions, in one-to-one and one-to-many scenarios, including collaborations involving two or more BVI developers.

Many participants said we should ensure that CodeWalk was accessible to deaf-blind programmers and usable with Braille displays. They felt that its reliance on the audio medium could exclude deaf-blind programmers. In future, we will extend our design to support communication of awareness information through tactile media.

We recommend that application design standards, such as ATAG (Authoring Tool Accessibility Guidelines) and WCAG (Web Content Accessibility Guidelines) be extended to support mixed ability teams and provide non-visual information about collaborators' location, navigation, and edit operations. This could increase the use of accessibility practices in the design of collaborative authoring tools.

8 CONCLUSION

Existing tools to facilitate tightly-coupled software development tasks rely on visual cues and create accessibility barriers to equitably collaboration for BVI developers. To address this accessibility gap, we designed, developed and evaluated CodeWalk, a set of features added to Microsoft's VS Code Live Share extension that makes a collaborator's location in a code file and their actions accessible through cursor tethering, as well as sound effects and speech. CodeWalk's features improved coordination between BVI developers and their sighted peers while reducing the explicit effort that BVI developers need to put to stay coordinated. We hope CodeWalk can serve as an exemplar for IDE manufacturers to make their environments more accessible to blind and visually impaired software developers.

REFERENCES

[1] Khaled Albusays and Stephanie Ludi. 2016. Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study. In *Proceedings of 9th International Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, Austin, TX, 82--85. https://doi.org/10.1145/2897586.2897616

[2] Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (Baltimore, Maryland, USA) *(ASSETS '17)*. Association for Computing Machinery, New York, NY, USA, 91–100. https://doi.org/10.1145/3132525.3132550

[3] Ameer Armaly, Paige Rodeghero, and Collin McMillan. 2018. Audiohighlight: Code skimming for blind programmers. In *Proceedings of 2018 IEEE International Conference on Software Maintenance and Evolution*. IEEE, Madrid, Spain, 206--216. https://doi.org/10.1109/ICSME.2018.00030

[4] Catherine M. Baker, Lauren R. Milne, and Richard E. Ladner. 2015. StructJumper: A Tool to Help Blind Programmers Navigate and Understand the Structure of Code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3043–3052. https://doi.org/10.1145/2702123.2702589

[5] Florian Beijers. 2019. How to Get a Developer Job When You're Blind: Advice From a Blind Developer Who Works Alongside a Sighted Team. https://www.freecodecamp.org/news/blind-developer-sighted-team/

[6] Cynthia L. Bennett, Erin Brady, and Stacy M. Branham. 2018. Interdependence as a Frame for Assistive Technology Research and Design. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility* (Galway, Ireland) *(ASSETS '18)*. Association for Computing Machinery, New York, NY, USA, 161–173. https://doi.org/10.1145/3234695.3236348

[7] Jacob T Biehl, Mary Czerwinski, Greg Smith, and George G Robertson. 2007. FASTDash: a visual dashboard for fostering awareness in software teams. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, San Jose, CA, 1313--1322.

[8] Jeffrey P. Bigham, Anna C. Cavender, Jeremy T. Brudvik, Jacob O. Wobbrock, and Richard E. Ladner. 2007. WebinSitu: A Comparative Analysis of Blind and Sighted Browsing Behavior. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility* (Tempe, Arizona, USA) *(Assets '07)*. Association for Computing Machinery, New York, NY, USA, 51–58. https://doi.org/10.1145/1296843.1296854

[9] Syed Masum Billah, Vikas Ashok, Donald E Porter, and IV Ramakrishnan. 2017. Ubiquitous accessibility for people with visual impairments: Are we there yet?. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. ACM, Denver, CO, 5862--5868.

[10] Syed Masum Billah, Donald E Porter, and IV Ramakrishnan. 2016. Sinter: Low-bandwidth remote access for the visually-impaired. In *Proceedings of the Eleventh European Conference on Computer Systems*. ACM, London, UK, 1--16.

[11] Jeremy Birnholtz, Nanyi Bi, and Susan Fussell. 2012. Do you see that I see? Effects of perceived visibility on awareness checking behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin, TX, 1765--1774.

[12] Stacy M. Branham and Shaun K. Kane. 2015. The Invisible Work of Accessibility: How Blind Employees Manage Accessibility in Mixed-Ability Workplaces. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (Lisbon, Portugal) *(ASSETS '15)*. Association for Computing Machinery, New York, NY, USA, 163–171. https://doi.org/10.1145/2700648.2809864

[13] William Buxton. 2009. Mediaspace - Meaningspace - Meetingspace. In *Media Space: 20+ Years of Mediated Life*, S. Harrison (Ed.). Springer, London, UK, 217--231.

[14] William A. S. Buxton. 1992. Telepresence: Integrating Shared Task and Person Spaces. In *Proceedings of the Conference on Graphics Interface '92* (Vancouver, British Columbia, Canada). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 123–129.

[15] Code Factory. 2021. *Eloquence for Windows*. Code Factory. https://codefactoryglobal.com/app-store/eloquence-for-windows/

[16] Sarah D'Angelo and Andrew Begel. 2017. *Improving Communication Between Pair Programmers Using Shared Gaze Awareness.* Association for Computing Machinery, New York, NY, USA, 6245–6290. https://doi.org/10.1145/3025453.3025573

[17] Maitraye Das, Darren Gergle, and Anne Marie Piper. 2019. "It Doesn't Win You Friends": Understanding Accessibility in Collaborative Writing for People with Vision Impairments. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 191 (Nov. 2019), 26 pages. https://doi.org/10.1145/3359293

[18] Maitraye Das, Anne Marie Piper, and Darren Gergle. 2022. Design and Evaluation of Accessible Collaborative Writing Techniques for People with Vision Impairments. *ACM Transactions on Computer-Human Interaction* 29, 2 (2022), 1--42.

[19] Das, Maitraye and McHugh, Thomas B. and Piper, Anne Marie and Gergle, Darren. 2022. Co11ab: Augmenting Accessibility in Synchronous Collaborative Writing for People with Vision Impairments. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1--18.

[20] Paul Dourish and Victoria Bellotti. 1992. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. ACM, Toronto, ON, Canada, 107--114.

[21] Hongfei Fan, Chengzheng Sun, and Haifeng Shen. 2012. ATCoPE: Any-Time Collaborative Programming Environment for Seamless Integration of Real-Time and Non-Real-Time Teamwork in Software Development. In *Proceedings of the 17th ACM International Conference on Supporting Group Work* (Sanibel Island, Florida, USA) *(GROUP '12)*. Association for Computing Machinery, New York, NY, USA, 107–116. https://doi.org/10.1145/2389176.2389194

[22] Github. 2022. *Teletype for Atom.* Microsoft, Redmond, WA. https://teletype.atom.io/

[23] Max Goldman, Greg Little, and Robert C. Miller. 2011. Real-Time Collaborative Coding in a Web IDE. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 155–164. https://doi.org/10.1145/2047196.2047215

[24] Geoff Greer and Matt Kaniaris. 2020. *Floobits real-time collaboration plugin for Sublime Text 2 and 3.* Floobits. https://github.com/Floobits/floobits-sublime

[25] Carl Gutwin and Saul Greenberg. 2002. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3 (Sept. 2002), 411--446. https://doi.org/10.1023/A:1021271517844

[26] Lile Hattori and Michele Lanza. 2010. Syde: A tool for collaborative software development. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. ACM/IEEE, Cape Town, South Africa, 235--238.

[27] Scott Highhouse. 2009. Designing experiments that generalize. *Organizational Research Methods* 12, 3 (2009), 554--566.

[28] Earl W. Huff, Kwajo Boateng, Makayla Moster, Paige Rodeghero, and Julian Brinkley. 2020. Examining The Work Experience of Programmers with Visual Impairments. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Online, 707--711. https://doi.org/10.1109/ICSME46990.2020.00077

[29] Earl W. Huff, Kwajo Boateng, Makayla Moster, Paige Rodeghero, and Julian Brinkley. 2021. Exploring the Perspectives of Teachers of the Visually Impaired Regarding Accessible K12 Computing Education. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 156–162. https://doi.org/10.1145/3408877.3432418

[30] Joe Hutchinson and Oussama Metatla. 2018. An Initial Investigation into Non-visual Code Structure Overview Through Speech, Non-speech and Spearcons. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1--6. https://doi.org/10.1145/3170427.3188696

[31] Jennifer Hyde, Sara Kiesler, Jessica K Hodgins, and Elizabeth J Carter. 2014. Conversing with children: Cartoon and video people elicit similar conversational behaviors. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, Toronto, ON, Canada, 1787--1796.

[32] Hiroshi Ishii and Naomi Miyake. 1991. Toward an open shared workspace: computer and video fusion approach of TeamWorkStation. *Commun. ACM* 34, 12 (1991), 37--50.

[33] JetBrains. 2020. *Meet Code With Me (EAP) --- a tool for collaborative development by JetBrains.* JetBrains, Prague. https://blog.jetbrains.com/blog/2020/09/28/code-with-me-eap/

[34] Sasa Junuzovic, Prasun Dewan, and Yong Rui. 2007. Read, write, and navigation awareness in realistic multi-view collaborations. In *2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007)*. IEEE, New York, NY, 494--503. https://doi.org/10.1109/COLCOM.2007.4553880

[35] Claire Kearney-Volpe and Amy Hurst. 2021. Accessible Web Development: Opportunities to Improve the Education and Practice of Web Development with a Screen Reader. *ACM Trans. Access. Comput.* 14, 2, Article 8 (July 2021), 32 pages. https://doi.org/10.1145/3458024

[36] Richard E Ladner and Kyle Rector. 2017. Making your presentation accessible. *Interactions* 24, 4 (2017), 56--59.

[37] Cheuk Yin Phipson Lee, Zhuohao Zhang, Jaylin Herskovitz, JooYoung Seo, and Anhong Guo. 2021. *CollabAlly: Accessible Collaboration Awareness in Document Editing.* Association for Computing Machinery, New York, NY, USA, 1--4. https://doi.org/10.1145/3441852.3476562

[38] Cheuk Yin Phipson Lee, Zhuohao Zhang, Jaylin Herskovitz, JooYoung Seo, and Anhong Guo. 2022. CollabAlly: Accessible Collaboration Awareness in Document Editing. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1--17.

[39] Franklin Mingzhe Li, Di Laura Chen, Mingming Fan, and Khai N. Truong. 2021. "I Choose Assistive Devices That Save My Face": A Study on Perceptions of Accessibility and Assistive Technology Use Conducted in China. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 372, 14 pages. https://doi.org/10.1145/3411764.3445321

[40] Stephanie Ludi, Jamie Simpson, and Wil Merchant. 2016. Exploration of the Use of Auditory Cues in Code Comprehension and Navigation for Individuals with Visual Impairments in a Visual Programming Environment. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility* (Reno, Nevada, USA) *(ASSETS '16)*. Association for Computing Machinery, New York, NY, USA, 279–280. https://doi.org/10.1145/2982142.2982206

[41] Stephanie Ludi and Mary Spencer. 2017. Design considerations to increase block-based language accessibility for blind programmers Via Blockly. *Journal of Visual Languages and Sentient Systems* 3, 1 (2017), 119--124.

[42] Sean Mealin and Emerson Murphy-Hill. 2012. An exploratory study of blind software developers. In *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Innsbruck, Austria, 71--74. https://doi.org/10.1109/VLHCC.2012.6344485

[43] Oussama Metatla, Nick Bryan-Kinns, and Tony Stockman. 2018. "I Hear You": Understanding Awareness Information Exchange in an Audio-Only Workspace. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174120

[44] Microsoft. 2020. *Accessibility in Visual Studio Code*. Microsoft. https://code.visualstudio.com/docs/editor/accessibility#_screen-readers

[45] Microsoft. 2020. *Microsoft/vscode-a11y - Gitter*. Microsoft. https://gitter.im/Microsoft/vscode-a11y

[46] Microsoft. 2022. *Visual Studio Live Share*. Microsoft, Redmond, WA. https://visualstudio.microsoft.com/services/live-share/

[47] Matthew Miles, A. Michael Huberman, and Michael Saldaña. 2013. *Qualitative Data Analysis: A Methods Sourcebook*. Sage Publications, Thousand Oaks, CA.

[48] Lauren R. Milne and Richard E. Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3173574.3173643

[49] Cecily Morrison, Nicolas Villar, Anja Thieme, Zahra Ashktorab, Eloise Taysom, Oscar Salandin, Daniel Cletheroe, Greg Saul, Alan F Blackwell, Darren Edge, Martin Grayson, and Haiyan Zhang. 2020. Torino: A tangible programming language inclusive of children with visual disabilities. *Human–Computer Interaction* 35, 3 (2020), 191--239.

[50] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. 2022. Addressing Accessibility Barriers in Programming for People with Visual Impairments: A Literature Review. *ACM Transactions on Accessible Computing (TACCESS)* 15, 1 (2022), 1--26.

[51] Katja Neureiter, Martin Murer, Verena Fuchsberger, and Manfred Tscheligi. 2013. Hand and eyes: how eye contact is linked to gestures in video conferencing. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, Paris, France, 127--132.

[52] Austin Lee Nichols and Jon K Maner. 2008. The good-subject effect: Investigating participant demand characteristics. *The Journal of general psychology* 135, 2 (2008), 151--166.

[53] Maulishree Pandey, Vaishnav Kameswaran, Hrishikesh V. Rao, Sile O'Modhrain, and Steve Oney. 2021. Understanding Accessibility and Collaboration in Programming for People with Visual Impairments. In *Proceedings of the CSCW Conference on Computer Supported Cooperative Work* (Virtual) *(CSCW '21)*. Association for Computing Machinery, New York, NY, USA, 30 pages.

[54] Yi-Hao Peng, JiWoong Jang, Jeffrey P Bigham, and Amy Pavel. 2021. Say It All: Feedback for Improving Non-Visual Presentation Accessibility. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 276, 12 pages. https://doi.org/10.1145/3411764.3445572

[55] Venkatesh Potluri, Tad Grindeland, Jon E. Froehlich, and Jennifer Mankoff. 2021. Examining Visual Semantic Understanding in Blind and Low-Vision Technology Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, 14 pages.

[56] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y. Vidya, Manohar Swaminathan, and Gopal Srinivasa. 2018. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3174192

[57] T. V. Raman. 1996. Emacspeak—Direct Speech Access. In *Proceedings of the Second Annual ACM Conference on Assistive Technologies* (Vancouver, British Columbia, Canada) *(Assets '96)*. Association for Computing Machinery, New York, NY, USA, 32–36. https://doi.org/10.1145/228347.228354

[58] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. 2021. Perceived diversity in software engineering: a systematic literature review. *Empirical Software Engineering* 26, 5 (2021), 1--38.

[59] Johnny Saldaña. 2016. *The coding manual for qualitative researchers* (3rd ed.). Sage, London, UK.

[60] Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk. 2010. Saros: an eclipse plug-in for distributed party programming. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*. ACM/IEEE, Cape Town, South Africa, 48--55.

[61] Harini Sampath, Alice Merrick, and Andrew MacVean. 2021. *Accessibility of Command Line Interfaces*. Association for Computing Machinery, New York, NY, USA, 1--10. https://doi.org/10.1145/3411764.3445544

[62] Emmanuel Schanzer, Sina Bahram, and Shriram Krishnamurthi. 2019. Accessible AST-Based Programming for Visually-Impaired Programmers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 773–779. https://doi.org/10.1145/3287324.3287499

[63] Freedom Scientific. 2022. Jaws Tandem Quick Start Guide. https://support.freedomscientific.com/JawsHQ/JawsTandemQuickStart

[64] Chirag Shah and Gary Marchionini. 2010. Awareness in collaborative information seeking. *Journal of the American Society for Information Science and Technology* 61, 10 (2010), 1970--1986.

[65] Kristen Shinohara and Jacob O Wobbrock. 2016. Self-conscious or self-confident? A diary study conceptualizing the social accessibility of assistive technology. *ACM Transactions on Accessible Computing (TACCESS)* 8, 2 (2016), 1--31.

[66] Alexa Siu, Gene S-H Kim, Sile O'Modhrain, and Sean Follmer. 2022. Supporting Accessible Data Visualization Through Audio Data Narratives. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 476, 19 pages. https://doi.org/10.1145/3491102.3517678

[67] Stack Overflow. 2021. Stack Overflow Developer Survey 2021. https://insights.stackoverflow.com/survey/2021#section-demographics-disability-status

[68] Andreas Stefik, Andrew Haywood, Shahzada Mansoor, Brock Dunda, and Daniel Garcia. 2009. Sodbeans. In *2009 IEEE 17th International Conference on Program Comprehension*. IEEE, Vancouver, BC, Canada, 293--294.

[69] Andreas Stefik and Richard Ladner. 2017. The Quorum Programming Language (Abstract Only). In *SIGCSE Technical Symposium* (Seattle, Washington, USA) *(SIGCSE '17)*. ACM, New York, NY, USA, 641.

[70] Andreas M. Stefik, Christopher Hundhausen, and Derrick Smith. 2011. On the Design of an Educational Infrastructure for the Blind and Visually Impaired in Computer Science. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 571–576. https://doi.org/10.1145/1953163.1953323

[71] Kevin M. Storer, Harini Sampath, and M. Alice Merrick. 2021. ''It's Just Everything Outside of the IDE that's the Problem'': Information Seeking by Software Developers with Visual Impairments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, 12 pages. https://doi.org/10.1145/3411764.3445090

[72] Sublime users. 2020. *A request for the implementation of accessibility. Issue #3392. sublimehq/sublime_text.* GitHub. https://github.com/sublimehq/sublime_text/issues/3392

[73] John Tang. 2021. Understanding the Telework Experience of People with Disabilities. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 30 (apr 2021), 27 pages. https://doi.org/10.1145/3449104

[74] Josh Tenenberg, Wolff-Michael Roth, and David Socha. 2016. From I-Awareness to We-Awareness in CSCW. *Computer Supported Cooperative Work (CSCW)* 25, 4 (Oct. 2016), 235--278. https://doi.org/10.1007/s10606-014-9215-0

[75] Christopher Toth and Tyler Spivey. 2018. Documentation NVDA Remote Access. https://nvdaremote.com/docs/

[76] Jeremy Warner and Philip J Guo. 2017. Codepilot: Scaffolding end-to-end collaborative software development for novice programmers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver, CO, 1136--1141.

[77] Jacob O Wobbrock and Matthew Kay. 2016. Nonparametric statistics in human--computer interaction. In *Modern Statistical Methods for HCI*. Springer, Berlin, Germany, 135--170.