CHAPTER 8

UI Development Experiences of Programmers with Visual Impairments in Product Teams

Maulishree Pandey, University of Michigan, USA.*

Sharvari Bondre, University of Michigan, USA.

Vaishnav Kameswaran, University of Michigan, USA.

Hrishikesh Rao, University of Michigan, USA.

Sile O'Modhrain, University of Michigan, USA.

Steve Oney, University of Michigan, USA.

The tools and techniques that software engineers use to collaborate are critical in deciding who can contribute to software projects and the roles they can play within those teams. The consistent growth of UI developer job roles [7, 8, 9] has made many programmers seek UI engineering jobs. It is important to understand the accessibility of the profession and identify ways to make it more inclusive. We conducted two qualitative studies [10, 11] to better understand the strategies that mixed-ability teams – specifically teams where some team members identify as having a visual impairment and some do not – use to collaborate on user interface (UI) development. In this chapter,

Equity, Diversity, and Inclusion in Software Engineering, https://doi.org/10.1007/978-1-4842-9651-6_8

we summarize and synthesize the findings from our prior studies to highlight the challenges programmers with visual impairments encounter in collaborative UI programming. The chapter concludes with recommendations for building more inclusive software engineering teams by fostering communication and help-seeking interactions, which we hope product teams would find valuable. We also derive implications for UI frameworks that aim to support accessible application development. These implications can inform the engineering choices of product teams as well as inform the efforts of researchers and developers building these frameworks.

Background

Much of the existing empirical research has investigated the experiences of programmers with visual impairments as individual contributors. These prior studies offer insights into the challenges of creating websites from scratch using HTML (which is primarily responsible for defining a site's content), CSS (which is intended to specify how browsers should visually render the content), and JavaScript (which is primarily responsible for specifying the interactive aspects of the site) [3, 4, 6]. Programmers with visual impairments have shared that they feel less confident about modifying CSS rules on their own [2, 5]. Layout editors within IDEs and browser inspector tools interface poorly with desktop screen readers such as NVDA and JAWS and tend to not provide pixel positions, relative locations, and dimension information. Therefore, they often seek sighted assistance to spot-check the CSS edits and verify the layout of their design [4, 6, 11]. To foster more independent UI creation, Andy Borka developed the Developer Toolkit [1], an NVDA add-on that informs developers of pixel location and dimensions of UI elements. Another approach is to represent the HTML in nonvisual form, for instance, using tactile printouts [4] or tactile beads that can be organized on sensing boards, to teach students about UI development [13]. Existing research suggests that programmers with visual impairments find UI development for mobile platforms easier than developing for desktops because they can use the screen readers' gestures (e.g., single tap to explore the UI, double tap to select, swipe to move focus, etc.) to verify the size and position of UI elements [11, 12]. Programmers with visual impairments have leveraged the combination of touchscreens and cross-platform frameworks to develop UIs with relative independence for multiple platforms [10]. However, the aforementioned research does not reveal much about their collaboration with sighted developers and designers in the context of UI development. Prior research has mainly reported on challenges in collaboration over technical diagrams, presentations, and data visualizations [4, 5]. Given the importance of teamwork and collaboration in UI programming, this chapter describes ways software development teams can improve their work practices and tools to foster the participation of programmers with visual impairments.

Methodology

Our research was motivated by the program-l mailing list (program-l@freelists.org) – an active online discussion group for programmers with visual impairments to ask and share programming-related resources. We joined the mailing list in 2018 and regularly came across questions such as the following one, asking for accessible resources and tools to carry out UI development:

> *I want to learn to develop mobile apps for iOS and Android. I have both Windows 10 and MacBook Pro. I am totally blind so I'm dependent on JAWS and VoiceOver. The development stack is completely my choice. I am considering Xamarin and C#, Ionic and JavaScript, or React Native and JavaScript. My question is, what development stack are blind programmers having success in developing mobile apps?*

Posts such as these revealed that programmers with visual impairments had to identify accessible frameworks and development tools before they could dive into UI programming. Our first study broadly focused on understanding the accessibility challenges across various programming activities such as pair programming and code reviews, including participants' experiences learning and performing UI development. We conducted semi-structured interviews with 23 adult programmers with visual impairments (19 men, 4 women) between July 2019 and March 2020 [11]. Participants were between 21 and 73 years old. They hailed from the United States, India, China, and Europe and included software engineers, data analysts, freelancers, and researchers. The interviews elicited rich accounts about participants' collaboration with other developers and designers, including details on challenges and workarounds they identified to work in contexts primarily designed for sighted developers.

In the second study, we focused on how the use of UI frameworks and libraries shaped the workflows of programmers with visual impairments [10]. We first scraped all emails from the program-l mailing list between 2018 and 2021. Next, we identified the emails that seemed related to UI development by going over the posts' subject lines. Finally, we randomly sampled 96 emails and their replies from the filtered set of emails. This was followed by semi-structured interviews with 18 programmers with visual impairments (17 men, 1 woman) between 19 and 46 years old. We recruited participants from the mailing list and r/Blind subreddit. The eligibility criteria included that programmers should have experience using UI frameworks such as React Native, Flutter, Angular, etc.

We refer to participants from study 1 and study 2 as P*-I (e.g., P1-I) and P*-II (e.g., P1-II), respectively. Quotes from the program-l mailing list are indexed as T* (e.g., T1). We obtained prior approval from the University of Michigan's Institutional Review Board for both studies. We presented $15 and $30 USD gift cards (or their equivalent in local currency) to participants in studies 1 and 2, respectively.

Analysis and Limitations

For both studies, we used open coding followed by inductive and deductive coding to organize the data into high-level themes pertaining to UI development and collaboration. The research team met weekly to discuss the emerging themes and wrote memos to identify the missing details in the data to refine the questions for subsequent interviews.

Despite our best efforts to have a balanced gender representation, our participants' sample was skewed toward men. This was due to the software engineering field and the online communities we recruited from being largely male-dominated. Another limitation of our studies is that our participants and the mailing list members reported different vision-related disabilities. Since any disability falls on a spectrum, we refrained from analyzing the data based on the onset and the nature of the visual impairment. Instead, we distinguish between screen readers and assistive technologies such as screen magnifiers. Our findings report on developers' experiences who primarily rely on screen readers and scope our recommendations to people interested in designing for the audio modality.

Findings

We first discuss the collaborative experiences of programmers with visual impairments with sighted designers and developers, followed by their efforts in sharing their contributions broadly in the workplace.

Collaborating with Sighted Designers

UI development in software development teams often includes design discussions where developers and designers arrive at a mutual understanding of the UI's form, functionality, and interactions. These discussions are centered around visual artifacts such as wireframes and design documents, which specify the design details for developers to utilize in UI construction. We found that the design specifications could range from detailed to high-level. A detailed set of specifications stated the colors, size, and placement of individual GUI elements, enabling our participants to plug the specifications directly into the UI code. However, it was essential to provide the documents in accessible file formats such as word documents or PDFs correctly tagged for screen readers.

A high-level design document lacked strict rules and guides, requiring the developer to approximate size and placement from the visuals provided. Our participants shared that they needed to seek sighted assistance more often with loosely defined design documents. They would reach out to sighted friends, colleagues, and family members to spot-check the interface they were developing. Specifically, they would ask sighted people to verify that all UI elements lay within screen margins, did not overlap, and were visible on the screen. Additionally, if tasked with making decisions regarding the visuals (color selection, font selection, etc.), they would ask sighted people to determine if the UI looked aesthetically pleasing.

Through our studies, we found that collaborating with designers entailed a lot of communication and discussion to understand the UI's design. Participants mentioned that designers often struggled to describe the interface and omitted essential details that could help the participants visualize the interface. They felt responsible for framing and asking the right questions as well as narrowing down their questions to elicit the macro and the granular details from designers progressively:

> *When I put the question very precise one [...], they answer and they are eager to answer. But if I ask, for example, can you give me an idea of the layout [...], they used to say maybe much more than I need or maybe they miss some parts. It's to me, just to try to at the beginning, to ask very, very precise questions [...]. It's not always easy to know which are the elements that they want to put on the page, so I try and to refine step by step.*
>
> —P13-I (quoted from [11])

The discussions were more extensive when the UI in question had to be built from scratch using HTML and CSS. The conversations had to crystallize details regarding size, appearance, interaction, and the relative placement of widgets. If the team utilized existing components from a UI framework (e.g., Bootstrap, React Native, etc.), the communication became simpler – designers could cite the component to be used and state the expected modifications, and our participants could import them into the source code.

Several posts on the mailing list and accounts from our participants suggested that collaboration with designers helped programmers with visual impairments delineate between design and development roles and made them more confident about their programming skills. They shared how they came to understand that the ability to see had little to do with the ability to do UI development; designers were responsible for making the interface user-friendly and visually appealing, and as developers they were responsible for implementing the designers' ideas:

> *My boss brought our company's graphic designer into my department to help. He has taken my super-simple UI and turned it into something my company could show off. So there definitely is a certain art to it and vision is not the issue.*
>
> —T15-II (quoted from [10])

Collaborating with Sighted Developers

When doing UI development, sighted developers often use GUI builders provided within IDEs such as Android Studio and Visual Studio. These are based on the WYSIWYG (What You See Is What You Get) paradigm. Users can drag and drop the widgets, modify their size, and adjust the relative placement to quickly create the UI with mouse-based interactions. However, mouse interactions are inaccessible to programmers with visual impairments, and GUI builders seldom support UI creation through keyboard shortcuts. Our participants reported typing the XML code by hand to create the UI. Accessible GUI builders were far and few in between, existing for select programming languages and IDEs. The different approaches to UI development led to some tensions during collaboration, which we describe in the following.

Our studies revealed that typing the UI code often meant dealing with verbose source code for programmers with visual impairments. It presented challenges in code readability and navigation with screen readers. The problem was exacerbated when modifying the UI, requiring them to recalculate the values for dimensions and positions and update the source file throughout. Their sighted colleagues could adjust these values by looking at the UI in the GUI builders. The XML code that represents the UI is often nested, which made it difficult to identify the location of visual parameters that needed to be updated:

> *I just found myself overwhelmed by the number of options and layouts with very little idea how to make sure they do what I want. I lose track once I am about two levels deep into the user interface element structure.*
>
> —T2-II (quoted from [10])

The use of GUI builders also led to generic variable names for UI controls, which further affected navigation. Since sighted programmers did not deal with raw code, they did not always realize how the poor variable names could cause confusion for their colleagues:

> *Putting 2 buttons on a WPF designer surface, then tabbing around, forces the screen reader to say "grid," "button," "button," "window." What button is what one? The (WPF) designerneedstoassign default names to controls dropped on the designer surface and expose them to screen readers.*
>
> —T56-II (quoted from [10])

One of our participants shared that he had given strict instructions to his team to modify the variable names to descriptively map to UI controls' functionality before sharing the source code with him.

Incorrect focus or tab order was another common issue due to the differences in approaches to UI development among our participants and sighted programmers. We found that sighted people would randomly drag and drop the UI components when creating the UI. While this did not alter the visual representation of the UI, it significantly affected the interaction for people with visual impairments by altering the focus order on screen readers (see Figure 8-1), thereby impacting the debugging and testing workflows for programmers with visual impairments.

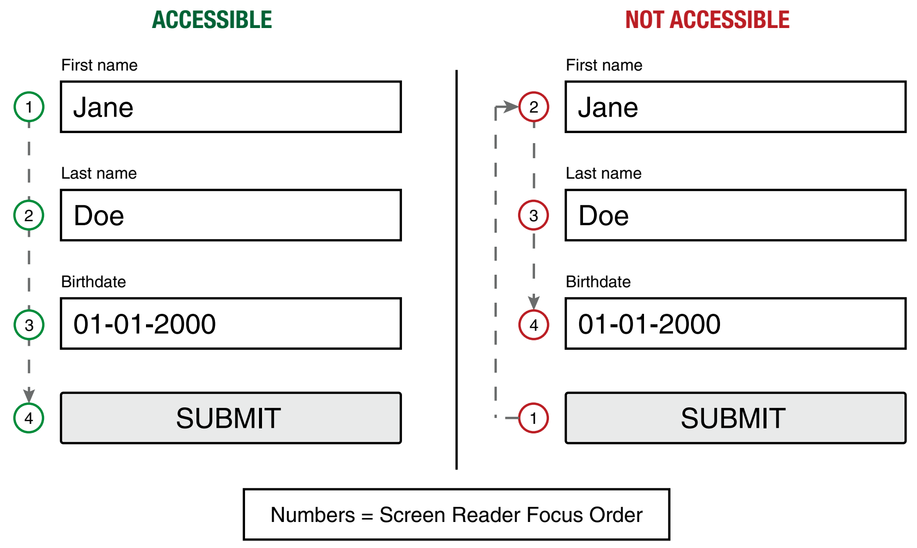

Figure 8-1. *Effect of dragging and dropping form elements in the wrong order on screen reader focus order*

Participants also shared that they often had to advocate for adoption of accessible UI frameworks and code editors. These decisions were often taken by developers on the team collectively. However, if the choice suffered from poor accessibility, it would impact the productivity of our participants and ultimately affect the collaboration workflows among developers. One participant shared that he convinced his team to program the Android application using Xamarin in Visual Studio instead of writing the code natively in Android Studio. The participant found the programming environment offered by Xamarin and Visual Studio more accessible and, therefore, a more productive option for him.

Occasionally even the more accessible choice could be rendered inaccessible in part due to software updates. For instance, P17-I and his team used Visual Studio. However, upgrading to the 2019 version from the 2017 one impacted certain settings with the screen reader. P17-I had to work out of both versions to maintain an accessible workflow for himself as well as keep the project compatible with the rest of the team.

When programmers with visual impairments joined existing projects where the team had already made the decisions about the technical stack, they had to deal with legacy UI code. Participants shared how sighted developers were unlikely to have implemented accessibility for screen readers while developing the UI. The lack of accessibility kept them from interacting and experiencing the UI independently, preventing them from building a full context of the project. In addition, it could interrupt debugging and testing workflows. It was also difficult to add the relevant accessibility modifiers, such as ARIA labels[1] or APIs, to legacy code to make it accessible. The decision to do so could also require approval from senior management, who would base it on the time investment in making the code accessible vs. the impact on the developer's productivity without these changes. Plus, asking the team to improve legacy UI's accessibility as a new team member foregrounded the developer's disability and could lead them to form misperceptions about their ability as a programmer.

Sharing Contributions in the Workplace

We found that accessibility challenges would also hinder programmers with visual impairments from sharing their code contributions more broadly. The stakes differed significantly when they had to present internally to the team vs. when they had to demonstrate their work externally to stakeholders and clients. In the case of the former, inaccessible tools and work practices would prevent our participants from participating in meetings such as discussions of the system architecture. One participant (P16-I) mentioned that after his annual review, he and his manager mutually figured out a way for him to contribute to such team discussions, including creation of diagrams and visuals. However, not every participant had a similar positive experience. They had to often forgo participation in these activities as primary contributors and let other team members take the lead. This ultimately could impact their career trajectories within the organization.

When presenting to clients and stakeholders, participants expressed concerns about the UIs glitching or breaking down due to the accessibility issues in the technical stack. These could suggest poor quality of work by them and their team. One participant mentioned that he sometimes recorded the presentation ahead of time to avoid demoing the UI live to people. Another participant shared that he preferred handling the narration and had a sighted colleague operate the UI using mouse interactions. This avoided issues that might arise due to poorly operating the UI with screen readers:

> *If it's a demo for stakeholders or an audience outside of the team [...], I ask someone else from my team to drive the visuals and I do the technical narration [...]. It's hard when I'm the one dealing with the visuals.*
>
> —P3-II

[1] Accessible Rich Internet Applications (ARIA) is a set of roles and attributes that define ways to make web applications and content more accessible for users with disabilities.

Discussion

Our research studies were motivated by the limited reporting of collaboration within mixed-ability teams and the fast rise in UI development jobs, whose lack of accessibility can tilt the playing field against programmers with visual impairments. Our findings highlight the challenges that programmers with visual impairments face when collaborating with sighted developers and designers as they do UI programming. In this section, we discuss how our findings can inform the practices within mixed-ability software engineering teams as well as the design of UI frameworks.

It is not enough for the "tools" that most people associate with software and UI development – IDEs, code editors, browsers, etc. – to be accessible. Accessibility breakdowns can also occur as the result of collaborative practices or communication tools. For example, pair programming can limit the roles of programmers with visual impairments [11] unless accommodations are made to enable them to serve as the observer. In the case of UI programming, relying solely on inaccessible collaborative artifacts (like visual drawings and wireframes) or inaccessible tools for communication can restrict the roles in which programmers with visual impairments can serve. Providing nonvisual alternatives to these (e.g., text-based descriptions, accessible PDFs, tactile printouts, etc.) will enable more inclusive team discussions and collaboration.

Further, many tools can be inaccessible in subtle ways. For example, a code editor might be working fine, but updates may modify certain settings. Similarly, editor plugins, add-ons, and configuration tools might create accessibility problems [11]. With a heavy reliance on third-party APIs and frameworks, the scope of tools that are essential for development has also expanded. For example, programmers might rely on cloud services with advanced configuration tools. For some programmers, these tools are just as important for their work as their IDEs, and accessibility problems can represent significant barriers to progress.

Sighted team members often do not realize the impact of inaccessible UI frameworks, programming tools, and documentation on their colleagues with visual impairments. For instance, designers would share loosely defined or inaccessible design documents that required programmers with visual impairments to ask precise questions about the UI. Developers would select inaccessible UI frameworks and code editors for development, in which case programmers with visual impairments had to either convince their team to switch to more accessible alternatives or work with the choices made by their colleagues. These findings illustrate the additional communication and articulation that programmers with visual impairments must perform in mixed-ability teams. However, the workplace and the team have to offer an inclusive environment for programmers with visual impairments to communicate and advocate for accessible software and practices. It is also important to note that our participants were often the only team members advocating for accessible tools and educating team members [10, 11]. This work on advocacy and education can come with a social cost within their workspace and represent additional hidden work and can have implications for their future career prospects.

One way to inform sighted team members about accessibility is by improving the documentation of UI frameworks and programming tools. If the official documentation emphasizes compatibility with screen readers, much like how they emphasize compatibility with various operating systems, it can be the first step toward enabling informed software choices among teams. It would also prompt people behind programming tools to think more deeply about accessibility, and they describe it in the documentation. For example, UI frameworks would then be more mindful about calling their components as "out-of-the-box accessible" and use more accurate descriptions.

Large technical companies tend to have their own internal code authoring guidelines, which software engineers are expected to follow. Prioritizing the code writing preferences of programmers with visual impairments can be made part of these code styling guidelines. These include (1) using camel case for variable names to enable appropriate announcement on screen readers, (2) avoiding generic identifiers, (3) modifying the generic names assigned to UI elements by GUI builders, (4) including descriptive comments to facilitate quick search and navigation in source code, and (5) following the screen reader focus order when creating the UI. This would not only ensure accessible UI code from sighted engineers but also reduce the articulation that programmers with visual impairments have to do in the workplace.

Conclusion

In this chapter, we summarized and discussed findings specific to UI programming from our prior studies. Our reporting of the challenges and recommendations can be valuable to the software engineering teams, researchers, and creators of UI frameworks and tools. It can help them make the profession of UI development and the related collaborative activities more accessible to programmers with visual impairments.

Bibliography

[1] Andy Borka. *Developer Toolkit*. 2019.

[2] Claire Kearney-Volpe, Chancey Fleet, Keita Ohshiro, Veronica Alfaro Arias, and Amy Hurst. Making the elusive more tangible: remote tools & techniques for teaching web development to screen reader users. In *Proceedings of the 18th International Web for All Conference*, 1–14, 2021.

[3] Claire Kearney-Volpe and Amy Hurst. Accessible web development: Opportunities to improve the education and practice of web development with a screen reader. *ACM Transactions on Accessible Computing (TACCESS)*, 14(2):1–32, 2021.

[4] Jingyi Li, Son Kim, Joshua A. Miele, Maneesh Agrawala, and Sean Follmer. Editing spatial layouts through tactile templates for people with visual impairments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–11, 2019.

[5] Sean Mealin and Emerson Murphy-Hill. An exploratory study of blind software developers. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, 71–74, 2012.

[6] Kirk Norman, Yevgeniy Arber, and Ravi Kuber. How accessible is the process of web interface design? In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, 1–2, 2013.

[7] Stack Overflow. Stack overflow developer survey results. 2019.

[8] Stack Overflow. Stack overflow developer survey. 2020.

[9] Stack Overflow. Stack overflow developer survey. 2021.

[10] Maulishree Pandey, Sharvari Bondre, Sile OModhrain, and Steve Oney. Accessibility of ui frameworks and libraries for programmers with visual impairments. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–10, IEEE, 2022.

[11] Maulishree Pandey, Vaishnav Kameswaran, Hrishikesh V. Rao, Sile O'Modhrain, and Steve Oney. Understanding accessibility and collaboration in programming for people with visual impairments. *Proceedings of the ACM on Human-Computer Interaction*, 5(*CSCW1*):1–30, 2021.

[12] Venkatesh Potluri, Liang He, Christine Chen, Jon E. Froehlich, and Jennifer Mankoff. A multi-modal approach for blind and visually impaired developers to edit webpage designs. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, 612–614, 2019.

[13] Ashrith Shetty, Ebrima Jarjue, and Huaishu Peng. Tangible web layout design for blind and visually impaired people: An initial investigation. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*, 37–39, 2020.