#A11yDev: Understanding Contemporary Software Accessibility Practices from Twitter Conversations

Syed Fatiul Huq
fsyedhuq@uci.edu
University of California, Irvine
Irvine, California, USA

Abdulaziz Alshayban
aalshayb@uci.edu
University of California, Irvine
Irvine, California, USA

Ziyao He
ziyah5@uci.edu
University of California, Irvine
Irvine, California, USA

Sam Malek
malek@uci.edu
University of California, Irvine
Irvine, California, USA

ABSTRACT

It is crucial to make software, with its ever-growing influence on everyday lives, accessible to all, including people with disabilities. Despite promoting software accessibility through government regulations, development guidelines, tools and frameworks, investigations reveal a marketplace of inaccessible web and mobile applications. To better understand the limitations of contemporary software industry in adopting accessibility practices, it is necessary to construct a holistic view that combines the perspectives of software practitioners, stakeholders and end users. In this paper, we collect 637 conversations from Twitter to synthesize and qualitatively analyze discussions posted about software accessibility. Our findings observe an active community that provides feedback on inaccessible software, shares personal accounts of development practices and advocates for inclusivity. By perceiving software accessibility from process, profession and people viewpoints, we present current conventions, challenges and possible resolutions with four emergent themes: cost and incentives, awareness and advocacy, technology and resources, and integration and inclusion.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in accessibility**; • **Software and its engineering** → *Software usability*; • **Information systems** → *Social networks.*

KEYWORDS

software accessibility, human factors of software engineering, qualitative study

ACM Reference Format:
Syed Fatiul Huq, Abdulaziz Alshayban, Ziyao He, and Sam Malek. 2023. #A11yDev: Understanding Contemporary Software Accessibility Practices from Twitter Conversations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3544548.3581455

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581455

1 INTRODUCTION

As more people rely on software to perform their daily activities, it is essential to provide equal access to those with diverse abilities and needs, including people with disabilities. Digital accessibility is the practice of removing barriers that prevent interaction with or access to websites, mobile applications, and other online technologies. According to the World Health Organization [70], around 15% of the world's population, or an estimated 1 billion people, live with some form of disability. This number is expected to grow in the future due to several factors, including population growth and medical advances, which allow people to live longer. The aging population is more likely to experience disability due to age-related conditions such as arthritis, vision loss and hearing loss.

Over the years, various efforts for promoting accessibility have been undertaken, including government mandates, organization-driven initiatives and educational programs. Countries impose accessibility regulations, for instance, the Americans with Disabilities Act (ADA) [49], which mandates that all electronic information and technology, including web and mobile apps, must be accessible to those with disabilities. Assistive Technologies (AT), like screen readers and braille keyboards for people with blindness and low vision, external switch keyboards for people with motor disability, and accessible features, like auto-captioning for people who are deaf and hard of hearing, and alt tags in images, are being developed and incorporated into web browsers and mobile platforms. Organization-driven efforts are also advancing the industry, including standards such as the Web Content Accessibility Guidelines (WCAG) [91], the most recognizable standard for digital accessibility, and platform-specific accessibility guidelines from main players in the technology space such as Apple[27] and Google[48]. To assist software practitioners with accessibility development, semantic labeling [69], accessibility APIs [31, 68], auditing services and automated testing tools are promoted and distributed.

Despite these efforts in fostering accessibility in software development, recent studies [6, 20, 95, 98] have shown that the state of accessibility is still far from satisfactory. A 2022 report by WebAIM[95] found that 96.8% of all home pages of the top 1 million most-visited websites had accessibility issues. Other studies on mobile apps [6, 20] have also shown that many apps remain inaccessible for users with various disabilities. It is, therefore, imperative to understand the limitations of contemporary software development in adopting accessibility practices.

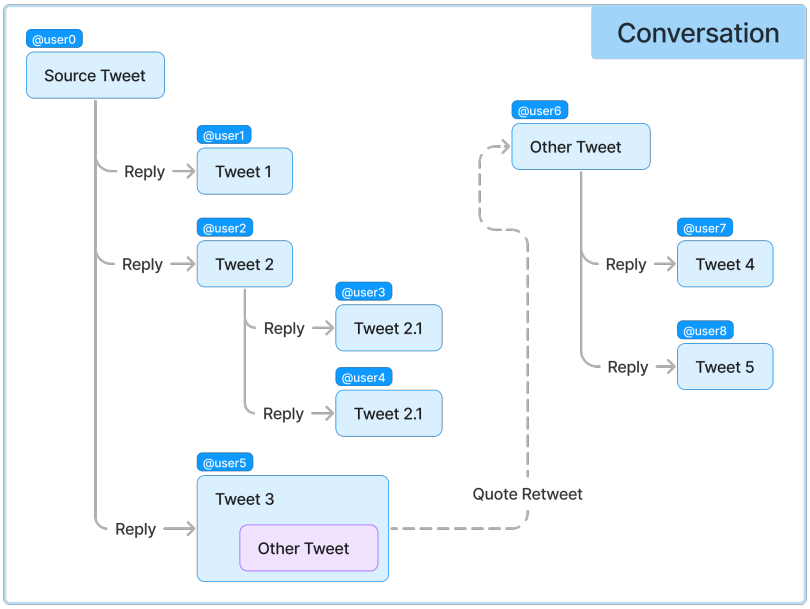

Figure 1: The structure and elements of a Tweet Conversation, used as the unit of analysis

Studies aimed at understanding accessibility development practices have surveyed software practitioners [6, 8, 14, 19, 36, 60, 66, 99] or investigated development environments like GitHub [13] and StackOverflow [21]. These gleaned insights from developers on the challenges behind accessibility development: lack of awareness, prioritization, tool support, affordability and managerial cooperation. Systematic literature studies on this domain [39, 71] have observed how research works have investigated different aspects of the software development life cycle, and proposed techniques or developed resources for practitioners to better implement accessible practices. These findings are important in locating points of interest in the development process that require improvement. However, there remains research gap in understanding how the different points interconnect, creating nuanced effects on accessibility development, and how factors beyond the development process, which the literature have not focused on, play a role. Moreover, the studies limit themselves in their exploration of perspectives, focusing on a specific demographic (e.g., developers in large software organizations or of a certain region) or a specific platform (e.g., web, mobile or Android). Accessibility development, on the other hand, is multifaceted, combining the viewpoints of different players — new practitioners, designers, developers, testers, consultants, leadership, organizations and users — and different operations — production, business and community.

To augment our overall understanding of accessible software development practices, it is therefore necessary to incorporate the insights of different groups and stakeholders, both individually and in collaboration. By inquiring on the limitations faced by the various players and operations, and their interrelations, we can detect further areas for improvement in this domain. We formulate the research question: *"From a holistic perspective, what is the current state of software accessibility, and its challenges and resolutions?"*.

To answer this question, we analyzed user conversations from Twitter [72], a popular social media platform, regarding software accessibility. Social media enables communication among different communities and demographics. Discourse, experiences and opinions posted by different users can provide a holistic view on a specific topic. Posts on social media are also candid expressions compared to interviews, one of the primary data collection methods used in the literature, which can suffer from biases from both the interviewers and participants [1, 5, 7, 42]. Prior studies have utilized Twitter in the software engineering context, to understand development practices [81, 83], software products [44, 45, 96] and security [50, 80]. Accessibility has also been studied using tweets, to investigate how social media is utilized by the disabled community [17, 41, 51]. Based on these, in this work, we explore Twitter to conduct a qualitative study on software accessibility at the intersection of user feedback, development experiences and accessibility advocacy.

We collected and analyzed conversations from Twitter, which are a series of tweets originating from a single source tweet and branching out through subsequent replies. Fig. 1 displays the structure of conversations and the element within its scope. Our collected dataset consists of 637 tweet conversations, containing more than

8500 tweets from 1800 unique users. The data introduces a multitude of accessibility issues in web and mobile applications, in the form of user feedback. It presents personal accounts of practitioners on their development practices and challenges, from scopes both individual and institutional. We observe community-wide advocacy for software accessibility, led by people with disabilities and allies to the cause, as both end users and practitioners.

We observe the emergence of three viewpoints, through which we can perceive software accessibility in a more detailed and nuanced manner. (1) Process: how accessibility is integrated into the software development life cycle — planning, design, development and testing. (2) Profession: how accessibility is treated in an organization, as a development skill, training material, regulation and business case. (3) People: how accessibility is learnt and mastered by practitioners, experienced by end users, and advocated by the accessibility community. Through the viewpoints, we observe four different themes: cost and incentive ⓒ, awareness and advocacy ⓐ, technology and resources ⓣ, and integration and inclusion ⓘ. Each theme synthesizes the challenges and existing solutions observed in the three viewpoints, along with possible measures for improvement.

Overall, the paper makes the following contributions:

- We identify three viewpoints pertaining to software accessibility by analyzing more than 8500 tweets from users, practitioners and software organizations.
- We detail the current challenges of software accessibility from a multifaceted perspective of the three viewpoints.
- We discover four cross-sectional themes and list recommendations and future work to improve the corresponding challenges.

2 RELATED WORK

Our study lies in the intersection between understanding development experiences of software accessibility and exploring Twitter for personal and professional insight. This section describes the related work in these two fields.

2.1 Software Accessibility Development Studies

Research on accessibility in the context of software engineering has been conducted from two exclusive perspectives: the users, through app reviews, and the software developers.

App reviews are considered a valuable source of information for users and developers, and a key element that contributes to an app's success. Several studies [34, 76] looked into how accessibility is discussed in user reviews posted on popular mobile app stores. For example, a recent study by Eler et al. [34] investigated accessibility feedback in user reviews posted on Android Play Store, and found that the number of accessibility reviews is very low (less than 1.24%), and most of the accessibility reviews are focused on a small number of apps, suggesting that accessibility feedback is scarce in app reviews. Another study by Arias et al. [76] suggests that accessibility-related app reviews can provide valuable feedback about user experience, and may also reveal accessibility issues that are not adequately covered in existing accessibility guidelines. Other researchers [3, 4, 86] in this area focused on supporting developers improve their apps by proposing automated techniques for the identification and classification of accessibility-related reviews. Such techniques can aid in identifying common accessibility issues and concerns reported by users.

Studies on user reviews not only lists, categorizes and details existing accessibility issues, but also indicates the need for user involvement in the software development process for better integration of various nuanced disability needs. Understanding how these issues are considered in the development process, however, requires studying the process itself and the people involved. Several researchers [6, 8, 13, 14, 19, 21, 36, 60, 66, 99] have assessed accessibility awareness among software practitioners, development practices, and perceptions of guidelines related to accessibility. A common theme identified by the studies is the lack of awareness and knowledge among developers regarding software accessibility, and how it can be implemented or tested. In cases where awareness was not an issue, other barriers such as lack of time and support from management hindered the inclusion of accessibility.

In trying to understand the role of accessibility within the context of software development, these studies explored developer perception and specific development practices. For instance, Vendome et al. [21] observed how accessibility guidelines are implemented in Android app projects. Bi et al. [13] looked into existing projects to understand the prevalence of accessibility issues, their reasons and the solutions implemented. Miranda et al. [66] investigated how projects that adopt an agile development methodology incorporate accessibility requirements in their systems. Others observed practitioner perception regarding multiple aspects: awareness of accessibility principles and technology [8, 19], relationship between accessibility, user experience and usability [99], current practices to adopt accessibility in the development life cycle [6, 14, 60]. Focus, therefore, had been within the scope of the development process. However, findings from some of these studies alluded to external factors, for instance, academic curriculum, organizational resource allocation or guidance from management. Research exploring these factors and how they, along with the individual concerns tackled by literature, relate to each other has not been conducted.

In terms of data source, these studies established their findings by surveying software developers, through direct communication or observing online forums. Bi et al. [13] and Vendome et al [21] mined repositories dedicated to software projects and communication among developers, GitHub and StackOverflow respectively. Cao and Loiacono [19] surveyed students with experience of developing a web application. Others interviewed or conducted surveys with software developers in the industry. Some targeted certain platforms (e.g. web [8, 36], mobile [60] or Android [6]), or geographical area (e.g, the software industry in Brazil [8, 60]) to focus on a certain group of developers. In a field where developer perspective had been seldom observed, these studies played an important role in discovering new avenues for analysis. However, focusing exclusively on software developers, or certain groups of developers, also limits the insight from a diverse group of participants. Moreover, most participants had been accumulated through convenience or snowball sampling. The authors distributed their interview or survey resources through mailing lists, social media connections, acquainted practitioners and organizations. These limit the diversity of the participant pool. For instance, Bi et al. [14] interviewed practitioners from major software companies (Alibaba, Hengtian

and Microsoft), which excludes the insight from new practitioners and smaller organizations. Lastly, user perspective on industry practices has been completely absent in these studies. Observing insights from all groups participating in accessibility development therefore remains an important avenue for research.

The literature has primarily employed interviews and surveys as its method of data collection. However, these methods have been observed to contain some limitations. For instance, responses in an interview are influenced by participant perception of the questions, societal conventions and what the interviewer would approve or disapprove of [7]. Similarly, the questions and interpretation of answers are affected by the interviewers' own biases, ideology and theoretical standpoint, their expectations about what interviewees feel or know about a topic, and their appearances, age, abilities or backgrounds [1, 5, 42]. Responses regarding accessibility, a topic that is perceived as a civil rights issue, can therefore be affected, influencing participants to conform to expectations and norms. In comparison, data from online forums and social media are candid communications from a large variety of participants. What the users post about and how they interact with others are not influenced by the researcher or an imposed context.

These limitations in the literature necessitate a larger, more holistic inquiry into the accessibility practices that prevail in contemporary software development, from the unbiased perspectives of all those who build the software, finance it and use it.

2.2 Understanding Software Related Aspects Via Twitter

Twitter can provide valuable insights for researchers by giving them access to a vast amount of data about what people are saying on a given topic. It can also be used to identify experts on a particular domain, which can facilitate conducting interviews or gathering information.

In this section, we describe related studies on how Twitter data is used by researchers in various domains to understand software related aspects.

Prior research [16] has shown that the software engineering community makes extensive use of Twitter's ability to facilitate conversation and information sharing of various software engineering-related topics such as discussions of current development projects, or finding solutions to implementation issues. Sharma er al. [81] performed an exploratory study of trending topics in software engineering Twitter space, they found resource sharing, technical discussion, and software product updates to be the most popular topics discussed by developers. Singer et al. [83] conducted surveys and interviews with active developers on GitHub, and found that Twitter helped them keep up with the fast-paced development landscape. Developers used Twitter to stay informed on industry changes, for learning, and for building relationships and connections.

Other studies have looked into the feedback loop between developers and users on Twitter. Williams et al. [96] have investigated how Twitter data can be used as a source of software user feedback. The result indicates that tweets can contain useful technical information that can be translated into actionable bug reports and user requirements. Other studies [44, 45] looked into Twitter usage while communicating about software applications, and found that tweets contain relevant information for different stakeholder groups.

Additionally, Twitter data can be a valuable source for providing a broad understanding of various software related aspects such as security, privacy, or usability. Saura et al. [80] investigated concerns regarding security issues of IoT systems. As a result of their study, they identified 10 security and privacy issues for IoT users. Choudhury et al. [50] conducted a quantitative analysis of software vulnerability information on Twitter and other social media platforms to understand how vulnerability information is present on those platforms, and how that information affects the related software development activities.

3 RESEARCH DESIGN

To utilize the intersectional conversations that social media offers and fill the gap of a multi-perspective understanding of accessibility development practices in the software industry, we conduct a qualitative study on Twitter's software accessibility community. While Twitter currently offers a specific Community feature[1], we refer to non-formal communities — groups of users posting tweets on software accessibility, with relevant hashtags, or replying to and sharing such tweets. The use of hashtags is important as Twitter groups together posts with the same hashtag, enabling easy navigation of a certain topic and reaching those who follow it with a tweet. This community consists of users and developers with disabilities, advocates and practitioners with accessibility expertise, and software organizations.

We use 'conversations' as our unit of analysis, defined as a sequence of tweets, consisting of a source tweet and all its replies. Fig. 1 displays how a conversation branches out from the source tweet, through replies and quote retweets. Quote retweeting is a Twitter feature where a user can add their own text when sharing another tweet, as a way of building onto a conversation. In our analysis, we include conversations in quote retweets, when they are part of the target tweet or posted as a reply, as it provides necessary context to the tweet under analysis. We conduct our data collection and analysis in two phases: first, collecting general tweets related to web and mobile accessibility, denoting these as Development Conversations (DC), and second, tweets from accessibility advocates or Advocate Conversations (AC).

For our study, we adopt elements of Straussian Grounded Theory [24, 85] in referencing the literature to construct our research question, form search queries, determine coding methods and complement our primary findings. An overview of our methodology is demonstrated in Fig. 2.

3.1 DC (Development Conversations): Collection and Analysis

To collect tweet conversations, we used Twitter Academic API v2 [72], which provides researchers with elevated access to Twitter content. Researchers can use the API to conduct a variety of tasks, including searching for tweets, collecting tweets from a particular user, and extracting metadata about an individual tweet. We have

[1]Communities on Twitter. https://help.twitter.com/en/using-twitter/communities

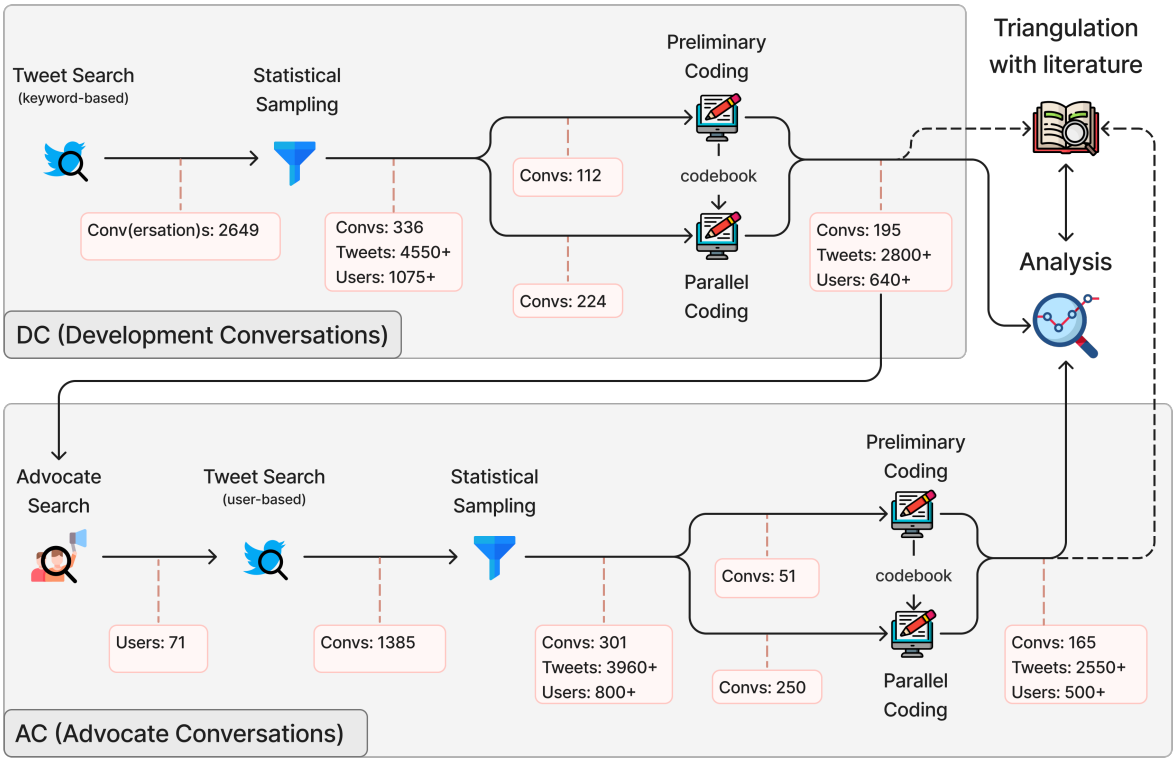

Figure 2: An overview of the research design

mainly utilized the full-archive search endpoint, which provides access to all public Tweets from the complete archive dating back to the first Tweet. This endpoint returns a set of tweets based on a specified search query, consisting of keywords and operators.

We iteratively constructed our search query for the API. First, we created an initial seed of keywords. We utilized the literature, specifically [13, 21], who mined StackOverflow and GitHub to mine accessibility-related discussions. Moreover, one of the authors had prior research experience mining Twitter for accessibility issue reviews from users. This initial seed evolved as we observed the resulting tweets and incorporated new keywords or discarded irrelevant ones. For instance, in our iterations, we found that users insert hashtags, such as "#a11y"[2], "#dev", "#A11yTwitter", "#BlindTwitter" and more, to direct their post towards a specific community. We ended our iterations of keywords once we perceived theoretical saturation — we observed that the addition or reduction of keywords from the query was generating similar or less relevant tweets. We defined relevance based on the literature, content that is situated within the context of web and mobile software accessibility.

Our final query is presented in Fig. 3. The first three sections, as grouped in the figure, filter a tweet's topic for accessibility, web and mobile application, and development concerns respectively. In our experimental runs, we found that a majority of tweets share external links, so we exclude such tweets with section 4, allowing URLs only as replies. Section 5 eliminates cryptocurrency related tweets, where the term "accessibility" is used in a different context. The remaining three keywords are used to exclude ads, exclude retweets (but include quote retweets) and only include tweets written in English respectively. Our search ranged from January 1, 2020 to April 30, 2022. Since our search provides a single tweet, from which we collect the entire conversation it is a part of, other tweets in the conversations can be posted outside of the specified date range.

This search generated a total of 2,649 conversations, among which we randomly sampled 336, with 95% confidence level and 5% margin of error for statistical soundness. The sample consisted of more than 4550 individual tweets and 1075 unique users. Each conversation had an average of 42 tweets, ranging from just one tweet to 4418, and an average of 36 unique users. The largest conversation (DC190) was a community post from Discord, where users discussed its features, including accessibility. The second largest

[2]a commonly used abbreviation of 'accessibility'

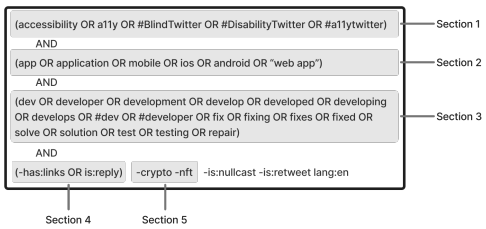

Figure 3: The final search query for Development Conversation search

(DC249), with 1748 tweets, was an educational video post showcasing how a user with visual impairment navigated on the iPhone.

Fig. 4 shows the distribution of user backgrounds in our data. For each conversation, we coded the personas of the target tweet's author, inferring their role based on their tweet and bio. The data shows a predominance of software practitioners personally engaging in software accessibility discourse. However, beyond the 12% end users detected in the target bio, many participated in the conversations from organization and practitioner posts, providing our analysis with multiple perspectives.

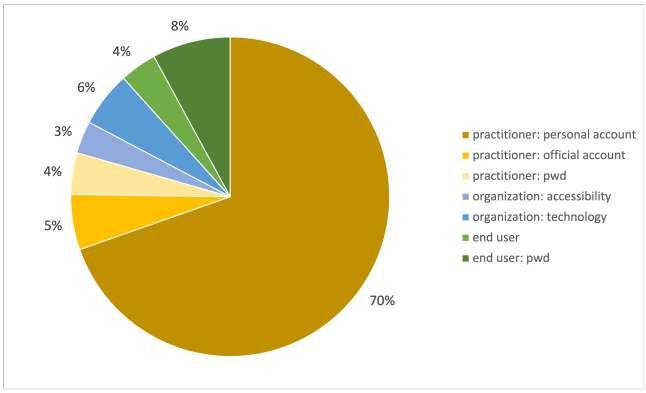

Figure 4: The inferred backgrounds of tweet writers. "Pwd" refers to "Person with disabilities"

We conducted inductive coding in two stages, as shown in Fig. 2. First, one author conducted preliminary coding of 112 conversations — one third of the sample. All three authors reviewed the preliminary codes to familiarize with the data and decide on a codebook. The codes characterized the persona of the tweet author (e.g., "user", "developer: personal", "developer: official"); the platform (e.g., "web", "mobile: iOS", "mobile: Android"), assistive technology (e.g., "screen reader", "alt text", "keyboard") or accessibility issue (e.g., "color", "sound", "text") the tweet mentions; the type of communication ("suggestion", "query", "practice: positive", "practice: negative") and the software accessibility-related concern (e.g., "career", "method: testing", "obstruction").

The remaining 224 conversations were distributed among the three authors such that each conversation was assigned to two authors. Authors were allowed to add new codes if existing ones did not satisfy, and communicate the update to others. After completion, all three authors discussed and resolved any conflicts. Authors had kept memos during coding that were utilized in the analysis.

3.2 AC (Advocate Conversations): Collection and Analysis

From analyzing DCs, we observed that some practitioners label themselves as accessibility advocates, and their discussions contain topics of interest and substantial community interaction. So we decided to complement our initial results with conversations from advocates.

As shown in Fig. 2, for collecting ACs, we first identified advocate users from DCs. We automatically located keywords such as "accessibility", "a11y" and "advocate" in the users' bios and manually inspected the results. We collected conversations from their profile on accessibility, from January 1, 2022 to July 25, 2022, totalling 1385 conversations. While, for DCs, our search ranged 2 year 4 months, we shortened the range for ACs. Advocates tweet primarily regarding our target topic, yielding significantly larger data compared to DCs. In order not to skew our analysis on a single perspective, that of advocates, and since we do not employ a temporal lens on the data, we do not collect older data for ACs.

Using similar statistical sampling, we randomly sampled 301 conversations, consisting of more than 3960 individual tweets and 800 unique users. The conversations averaged 30 individual tweets, ranging from one to 978, and 26 unique users. In the largest conversation (AC24), there was a discussion on what Amazon's internal worker chat app blocking words like "accessibility" insinuates.

ACs were collected using the user timeline endpoint on the Twitter API, which allows access to all the tweets posted by a specific Twitter user. The Twitter username of the user whose tweets were to be accessed was specified in the API request, and the results were sent back as a set of tweets.

With the authors familiarized with the data and process, we conducted a parallel preliminary coding of 51 conversations based on the previous codebook, discussed the codes and established a slightly modified codebook with emerging topics. The remaining 250 conversations were similarly distributed, coded and conflicts resolved.

3.3 Triangulation with Literature

Our goal in this step is to complement our data with relevant literature, providing context and interpretation for our findings within the existing body of research. To ensure a comprehensive set of relevant work, we identified literature using a range of strategies. Primarily, as indicated in Fig. 3, we searched for keywords that were mentioned in both DCs and ACs, along with alternative keywords with similar meanings. Additionally, we conducted forward and backward snowballing on that initial set to retrieve publications. The papers' fields spanned software accessibility and general software engineering concerns — security, collaboration, design —

along with education and business studies. A total of 160 papers were retrieved, 43 of which were deemed as relevant.

3.4 Threats to validity

Among the three threats to validity posed by qualitative research: researcher bias, reactivity and respondent bias [62], our work poses a threat for researcher bias. Since we do not interact with the participants (Twitter users) and only collect their candid communication, reactivity and respondent bias do not apply to our study.

To mitigate researcher bias, we considered multiple procedures: prolonged involvement, triangulation, peer debriefing, negative case analysis and keeping an audit trail [78]. The authors are not active members of the Twitter community the study analyzed, restricting involvement only to data collection and observation, thus counteracting *prolonged involvement*. We also decreased researcher bias with parallel coding among three authors. For *triangulation*, as described, we incorporated findings from literature in a variety of relevant fields. We conducted *peer debriefing* with researchers outside of our author group, to get opinions on data collection, coding, analysis and presentation. We performed negative case analysis, whenever we found experiences or opinions that contradicted that of the majority. In most cases, those views were refuted by others in the same conversation. We also triangulated with literature to find the presence of these views, and the reasoning or resolutions behind them. We *kept an audit trail* by storing any kind of research data, from the initial experimental tweet collection, to the different codes emerged from parallel coding and their final resolved states. Authors also kept notes during coding, used for understanding rationale during the analysis phase.

Our selection of keywords for searching tweets could pose a threat to validity, as no objective measurements have been used for its formation and stoppage. To mitigate the threat, we conducted multiple iterations on the keywords. For each iteration, the authors investigated a sample of the resulting tweets to assess relevance, review keywords used, detect emerging keywords and evolve the keyword set accordingly.

Presentation of the findings. From our analysis, we observed, and present as such, three viewpoints for discussing software accessibility practices: process, profession and people. Each viewpoint section presents claims, opinions and experiences from the conversations (DCs and ACs), cross-referenced with relevant literature. We provide our inferences of the findings in Section 7, discussing the four emergent themes — cost and incentive, awareness and advocacy, technology and resources, and integration and inclusion. We also incorporate these themes in the viewpoint sections, with the topics where they emerge, as their respective symbols: ⓒ, ⓐ, ⓣ, and ⓘ.

4 A11Y VIEWPOINT: PROCESS

The software development process spans multiple phases: requirements and project planning, system and UI design, coding or development, testing, and maintenance. The level of accessibility in the final software product is dependent on how well accessibility is integrated in these phases, either through inclusive practices or with the support of accessibility-oriented technologies. Our data sheds light on the contemporary practices and technologies that impede accessibility integration, and the changes to convention necessary for improvement.

4.1 Software Development Life Cycle (SDLC)

We begin by looking into development practices from the broader view of the whole development life cycle, and how accessibility is situated in it.

Prioritizing accessibility early and consistently in the project's life cycle (ⓘ, ⓒ). Practitioners opine that one of the biggest challenges regarding accessibility development is the lack of prioritization within the development lifecycle. Generally, accessibility is considered as an afterthought, due to legal concerns or negative user feedback. This aligns with past findings that companies still do not perceive accessibility requirements as a priority [6, 66]. Delaying accessibility for later stages or future updates adds on to maintenance and refactoring effort, accruing additional cost; also reported by prior research [89]. Therefore, it is advocated to *"shift accessibility to the left"* (DC161, AC7)[3], including it during project planning and requirements analysis. Doing so will ease accessibility implementation for developers and prevent the cost behind finding workarounds. It should also be required as a functional requirement instead of a *"nice to have"* (DC239) specification (DC27, DC37, DC188, AC65, AC297).

It is also advocated to prioritize accessibility consistently in all phases of the SDLC. Prior work [71] showed that design and testing has been the focus for accessibility research, excluding the rest. On the other hand, some do not consider accessibility as part of design, rather only a responsibility for the programmers. *"It's really important to bring accessibility to the main stage"* (AC28), which can be accomplished by making accessibility the foundation of these phases, and continuing to consider it during updates and maintenance (DC27, DC94, DC114, DC178, DC239, AC21, AC65, AC140, AC161, AC274, AC283).

Developing an "a11y strategy" (ⓘ, ⓣ). We observe the proposition of a tangible technique for accessibility prioritization named a11y strategy. It incorporates the knowledge and principles of inclusivity with the constraints of the different stages of development. It promotes cross-functional collaboration, with product teams and customer feedback. It prioritizes proactively reviewing designs for accessibility, as opposed to a reactive testing-centered process. *"You have to train people, embed accessibility using research into design systems, component documentation, code frameworks, QA … and fill in the gaps in WCAG … then evaluate how effective it is from the users' perspectives"* (AC161). Practitioners suggest incorporating a11y strategy as an aspect in a team's existing process model, for instance Agile methods, or adopt its components like Scrum meetings for daily checkups on accessibility (DC239, DC161, AC9, AC35, AC94).

4.2 Design

In the design phase, the software's UI structure is constructed, influencing the final product's accessibility. Conversations in Twitter discussed the current lack of accessibility consideration in design

[3]In the three Viewpoint sections, we cite the conversations that refer to the discussed topic. Conversations are labeled as DC(1-336) and AC(1-301) in our research artifacts.

efforts, its role in building accessible software, and the constraints designers face.

Design is integral for accessibility (ⓘ, ⓒ). In most project teams, practitioners report very little accessibility involvement in design. Designers often treat accessibility as a separate concern, beyond their project scope, leaving it for developers and testers. Designers, who come from *"a background in visual arts"*, are observed believing that *"a11y rules are stifling their creativity"* (AC20). Prior work also reports a false perception among designers that accessibility represents a restriction on creativity [75]. However, others say that the constraints imposed by accessibility can enable engaging and fun design efforts. It invokes new challenges in product design, instead of being *"just the same four patterns over and over again"* (AC29). We also observe that designers with an opposing outlook can be swayed once they are taught about the impact of bad design on disabled people (DC27, AC20, AC21, AC29).

Advocates report the necessity of convincing designers of their impact on accessibility. A recent study [10] reported that as companies became more experienced with accessibility, they tended to focus more on integrating accessibility earlier at the design phase. An accessible design is promoted for its comprehensive solutions, as it involves the visuals (color, shape, structure) as well as what is heard (alt tags), what is felt (braille keyboard), and how it is interacted with (click, keyboard navigation etc.). While a design where accessibility is built directly into the core experience provides a better design for everyone, a bad design directly affects a disabled user's experience: *"No-one is disabled until someone designs something that does not take that person's needs into consideration"* (DC27). Design is seen as critical in preserving accessibility of the product from the ground up. It holds the advantage of pre-reviewing accessibility for the product, as opposed to auditing and benchmarking in the development and testing phases, making it easier for early system-wide improvements. Practitioners experienced an accessible design to ease the effort to fix emergent accessibility issues in later phases, remediate issues that developers cannot effectively fix, and establish cross-team partnerships (DC27, DC288, AC3, AC26, AC27).

Challenges integrating accessibility in design (ⓣ, ⓐ, ⓒ). Firstly, designers report on a lack of design tools or frameworks that support accessibility. The few design resources that exist are either not comprehensive enough to provide a one-stop solution, or are catered to developers, requiring programming skills to operate. *"I design something and I can't work on accessibility until I've coded it in HTML"* (DC27). Design requires resources that are exploratory, experimenting with UI objects, rather than straight-to-production instruments. Prior work [47] showed how tools to automate and improve accessibility, contributed by research studies, do not serve the needs of designers. Designers suggest incorporating accessibility modules in UI mockup tools, a resource popular for creating experimental visual and interactive representation of the software.

Secondly, accessibility in academia, albeit in limited scale, is focused primarily on development practices rather than design. Accessibility expertise in design has only been possible through one's *"own (often extraordinary) effort"* (DC27) rather than formal or structured training. Without including accessibility in design education, practitioners fear design teams will remain unaware of accessible design and projects will resume excluding accessibility in design (AC27).

Lastly, our data shows that, in the professional setting, accessibility is excluded from a designer's skill requirement. While there are job postings for accessibility engineers or accessibility product managers, none such designation exists for accessibility designers. *"Everywhere that I have worked, accessibility has fallen to the developers"* (DC27). Such practices fail to incentivize designers to pursue learning accessibility.

4.3 Development

Unlike design, development has seen the emergence of multiple tools, systems and frameworks that assist developers in incorporating accessibility. However, there are still limitations in the available resources and bad practices that pervade development processes, hindering accessibility in software.

Code level practices: ARIA and Semantic HTML (ⓣ). Two of the most popular instruments for accessibility in web development are ARIA [92] and Semantic HTML [69]. Accessible Rich Internet Applications (ARIA) is an accessibility-centric scripting tool that adds roles and attributes to HTML elements to render them meaning.

For instance, a progress bar can be written as,

```
<div id="update-progress"></div>
```

where `<div>` provides no meaning. Adding ARIA labels to this —

```
<div id="update-progress" role="progressbar"
    aria-valuenow="75 aria-valuemax="100">
</div>
```

characterizes this `<div>` as a progress bar, and enables assistive technologies to interact with it as such. Our data shows multiple instances of developers using or suggesting ARIA for bolstering accessibility. ARIA as a formalized practice can also instill consistency in frontend nomenclature, which would prevent different auditing services using different terms for the same component (DC312, AC128).

However, ARIA is built as a solution that tries to add meaning to meaningless HTML, whereas HTML already possesses a mechanism to provide meaning to components: with Semantic HTML. Instead of populating every HTML component in a page with `<div>`s or `<span>`s, Semantic HTML provides tags such as `<header>`, `<figure>`, `<nav>` and more to embed semantic meaning to that component. For our previous example, rewriting the component with `<progress>` instead of `<div>` —

```
<progress id="update-progress"
    value="75" max="100">
</progress>
```

also characterizes it as a progress bar. Practitioners in our results advise adopting a "Semantic-first" approach when writing HTML pages, as opposed to an "ARIA-first" one. Misusing ARIA can bring forth more issues than not using it. It is nevertheless warned by advocates from focusing solely on Semantic HTML for accessible software: "saying semantic html will give you accessibility for free is not the full story" (DC196, DC239, AC1, AC3, AC6).

Accessibility-oriented frameworks and design systems (ⓣ, ⓐ). Frameworks like React [61], Angular [37] and more have become industry standard as they help developers build their web applications faster. However, in doing so, they have abstracted away the HTML underneath, and this separation from the DOM has been seen as a barrier to accessibility practices. For instance, developers are not prompted to add Semantic tags, and adding them afterwords requires unraveling *"layers of JS spaghetti"* (DC288, AC37, AC53, AC82). Lack of accessibility support in modern JavaScript frameworks was also raised in the literature [63]. Furthermore, JS libraries are criticized for not considering accessibility, and bringing bigger payloads and slower loading times that disrupt AT use. There have been initiatives integrating accessibility into libraries, but practitioners found most lack proper documentation, perceiving these as marketing ploys that lack actual accessibility features (AC74, AC128, AC215).

Similar issues are brought up for design systems (e.g., Google's Material Design [26], Apple Human Interface Guidelines [43]), which are scalable collections of thematic guidelines and UI components for web and mobile applications. Developers adopt the styles and behavior offered by these design systems into their own software. Therefore, inaccessible design in these design systems, developed by people unaware of accessibility or with ableist biases, is feared to perpetuate systemic bias in industry-wide software development. It is difficult to fix inaccessible components set by design systems, without inadvertently breaking other components or features imported (DC30, DC31, AC37, AC128).

Practitioners prompt frameworks and design systems to provide AT support and incorporate accessible design for their base UI components. This can obviate developers from building accessible components from scratch and fixing repetitively reported design-level issues in audit reports. Removing accessibility issues from base components also ensures accessibility standards are maintained, even if developers are unaware of them, positively influencing industry-wide culture (DC40, DC122, DC288, DC313, AC128, AC139, AC215, AC297).

Accessibility beyond a developer's control (ⓣ). While most discourse regarding accessibility in software place responsibility on the developers, we observed multiple accounts on how practices and systems provided by hardware manufacturers, platforms and operating systems, and web browsers, which are beyond a developer's scope, are accountable as well.

Manufacturer: Different laptops and computers contain their unique drivers that interact with ATs. So despite a software being compatible with ATs in most devices, we observe accounts of some causing it to break, making the software inaccessible. In mobile, specifically Android phones, how accessibility services are implemented depends, to some extent, on the phone manufacturer (DC49, DC238).

Platforms and operating systems (OS): Similar to hardware, different platforms and OSes provide different levels of accessibility support. For instance, Twitter's alt tag integration with media, although available on web browsers, had not been integrated with iOS through the native app (AC119). Much has been discussed in our data on how accessibility is integrated differently in iOS and Android. The general consensus favors iOS, where accessibility is embedded more strictly, and app developers are forced to follow guidelines and integrate accessibility services. Android, on the other hand, provides more autonomy to app developers in how they use accessibility services, creating opportunities for avoidance or even misuse (DC208, DC238, DC247, DC278, DC288, DC327, AC56, AC119, AC156).

Browsers: Developers advocate for browsers to programmatically ensure web app accessibility: from providing APIs for accessibility services to not hosting inaccessible sites. *"I want to live in a world where if your component is not accessible, the browser won't render it."* (AC128, DC288)

The negative impact of accessibility overlays (ⓒ, ⓐ). Accessibility overlays are website add-ons that provide assistive widgets and/or automatically fix code to transform an inaccessible website into one that conforms to WCAG. While marketed as an accessibility solution, these products are strongly opposed by accessibility advocates[4]. Our findings list key criticisms against this technology, spanning technical and business malpractices. The widgets offered are at best redundant to disabled users who are using ATs and third party software, and at worst incompatible with the user's setup, disrupting their experience. The automated fixes are not reliable and can cause unwanted behavior. Conformance, as assured by overlay vendors, are often false promises, as they are unable to fix all the website's accessibility issues. However, their marketing capacity outshines the reported inadequacies, convincing clients, who cannot afford accessibility in their development processes, to incorporate an overlay as a cheap one-click solution. As such, overlays are deemed as an *"anti-accessibility product"* (AC84). Their inclusion on a website can render it unusable by people with disabilities. The false conformance puts well-meaning clients in risk of violating regulations. And their predominance in the market discourages development teams from integrating accessibility practices in their processes, helping accessibility issues to persist as a culture (AC14, AC84, AC89, AC174, AC175, AC177, AC183, AC203, AC233, AC232, AC236, AC248). Similarly, a recent study investigating accessibility overlays found that while overlays can help improve some aspects of websites accessibility, they are still too limited and could not achieve complete compliance with accessibility standards [32].

Better accessibility means better SEO (ⓒ). Search Engine Optimization (SEO) [30] is widely considered a success criterion for web applications. It is observed that an accessible website, that follows best practices like Semantic tags, score better for SEO. Consequently, practitioners say that SEO is used as an incentive for accessibility, a further indication of how technical standards set by system-wide platforms can bolster accessibility (DC137, DC292, AC240, AC284).

4.4 Testing

In current practices, accessibility is most commonly considered in the testing phase, usually to check for regulatory violations. Automated testing tools are used to streamline accessibility into testing. Manual and user testing is advocated for integrating the lived experiences of people with disabilities.

Automating accessibility checks (ⓣ, ⓘ). We observe a popularity of automation in the testing phase, conducted in two contexts:

[4]https://overlayfactsheet.com/

audits and developer testing. Auditing involves the use of an external tool or service that reports inaccessible components of a developed software. It is deemed as a quick mechanism for developers to detect which area of their code needs fixing, and for users to understand a product's accessibility. It can also be a resource for learning about accessibility (DC98, DC122, DC281, DC330, AC104, AC105, AC98, AC277, AC283). Depending on the raw audit reports, however, is noted as an inefficient technique that often demotivates developers from addressing the accessibility issues. Since audits generate repeating issues on UI components used in multiple pages, it is suggested to conduct a review process for grouping up and prioritizing the issues before handing them over to the developers. Component level reports are observed as more intuitive and manageable, as opposed to page-wise audits, which are reasonable for very small websites. This practice also leads us back to design level accessibility; if the UI is designed to be accessible, audits are easier to manage, or even redundant at best (DC31, DC40, DC122, DC169).

While audits are generally dependent on a service or team external to the core development team, our data suggests that automated tools are now being integrated into the development and testing processes. Developers utilize tools available on the browser, or create their custom setup. For instance, Android's Accessibility Scanner [84] can be integrated with Espresso[29] or RoboElectric [77] that provide automated testing features (DC322). Practitioners view automated tests to significantly reduce the overhead of evaluating accessibility. In particular, if integrated into the development and deployment pipeline using Continuous Integration (CI) technology, such tools can drastically streamline the process of evaluating accessibility. With automated testers, developers do not need to wait until the project is deployable and handed to auditing teams; they themselves can check the accessibility of each new feature introduced in real time (DC54, DC161, DC222, DC281, DC330, DC322, AC3, AC89, AC90). Nevertheless, current automated practices and tools are criticized for their limitations. They are often not comprehensive enough to detect all existing issues, their performance are dependent on test coverage, and tend to generate false positives. Furthermore, practitioners warn that no issues detected does not necessarily mean an accessible software (DC161, DC281, DC314, AC252).

Testing accessibility manually (ⓐ, ⓣ, ⓒ). With the limitations posed by automated tools, manual testing — navigating one's software using an AT to detect accessibility issues — is highly recommended by experts. Manual testing enables a broader and more context-specific assessment, as opposed to the WCAG-specific and generalized implementation of automated tools. Practitioners deem it as the nearest mechanism to simulate user experience without a disabled user on board. They also say that simulation was the biggest motivator for them to learn accessibility (DC49, DC118, DC161, DC281, AC1, AC3, AC18, AC33, AC43, AC243). Literature supports these observations, discussing the importance of manual testing and its ability to provide better coverage and discover the issues that automatic tools cannot. [46, 58, 90]

We observe that current practices for manual testing are conducted using ATs and alternative interaction methods. For instance, desktop and mobile screen readers, keyboard navigation with a blank screen, and with text or view enlarged. Tests are suggested to be conducted with multiple ATs, in different window sizes, on different browsers and devices (DC117, DC149, DC168, DC180, DC202, DC214, DC229, AC1, AC43, AC243). However, practitioners say that it is not readily adopted in testing processes as it requires testers to learn different ATs. For teams with financial limitations, some ATs may be too expensive. Testers complain that, unlike audit reports, no formalized documentation template exists for creating accessibility statements from manual testing, and unlike automated tests, no programmatic mechanism to integrate it into the development pipeline. And finally, despite best efforts, manual testing cannot fully replicate a disabled user's experience (DC27, DC92, DC147, DC243, AC128, AC250).

User tests cannot be excluded (ⓘ, ⓐ). Our results suggest that involving disabled users in the testing process provides the most insight on the accessibility of a product. User insight covers more factors and provide more nuanced feedback than accessibility checkers, which are limited by WCAG-driven heuristics. Developers and users alike state that testing with ATs is more efficient and effective by disabled users, who are knowledgeable of different AT features and workarounds, and more proficient than abled developers. For instance, screen reader announcements that developers may regard as too short or very detailed in a manual test, are reported by a disabled user as properly concise or too cumbersome respectively. "Center the lived experiences of people with disabilities, not that of non-disabled testers" (AC250). Therefore, the accessibility community advocates for user testing, despite the inclusion of automated and manual tests. However, despite companies starting to integrate the practice, we find that strategies for such testing are still not formalized or mainstream enough for easy adoption. Furthermore, Aizpurua et al. [2] have reported about potential personal biases in the process. Software teams must be trained and/or services can be introduced who will connect teams to professional testers (DC17, DC94, DC161, DC171, DC239, AC3, AC23, AC39, AC128, AC144, AC166)[11, 23, 65].

5 A11Y VIEWPOINT: PROFESSION

Focusing on accessibility primarily from a development and technical perspective overlooks its important consideration in the professional setting. From discussions of practitioners' professional experience and the business aspects of software accessibility, we see that a lack of priority and the persistence of bad practices in an organizational capacity can undermine accessibility despite technological advancements and individual advocacy.

5.1 Personnel

Accessibility in software companies depends on those who develop the products. Therefore, how accessibility is treated as a development skill for employees and how leaders in an organization regard it determine how accessible the software product is.

Career opportunities and limitations in software accessibility (ⓘ, ⓒ). We observe accessibility being integrated as a career path, with designations such as accessibility engineer, engineering manager for accessibility and more. Such jobs are being offered by large software companies (like the FAANG companies), popularizing the concept. Accessibility is included as a specialist position in modern software projects, along with conventional specializations

like security, networking etc. Other than specialized positions, we observed discussions on accessibility in an aspirant developer's skillset being prioritized by employers. The data shows that jobs in software and digital accessibility also encourage inclusivity, where developers with disabilities are able to contribute with their lived experience, and disclosure of applicant's disability is observed as a non-issue, unlike other fields (DC1, DC260, AC2, AC13, AC211, AC216, AC285, AC293).

However, we find that accessibility related jobs are not universally common or prioritized, offered only in select states or countries where there is a perceived need or a growing culture of inclusivity. Job postings are also seen reflecting the inconsistent prioritization of accessibility in the SDLC, focusing on one phase over others. Accessibility developers are designated as junior engineers and the salary is perceived as *"insultingly low"* (DC27), indications of accessibility being undervalued. These bad practices dissuade developers from learning or specializing in accessibility (DC27, AC2, AC17, AC34, AC192).

Certifications for accessibility development (ⓐ, ⓣ). Employee certifications in the software engineering field provide aspirant developers with a proven specialization of skills and employers a benchmark for screening. Such advantages have seen the emergence of certifications, among many, for security, data management, cloud computing, and recently, accessibility. The two popular certifications are WAS (Web Accessibility Specialist) and CPACC (Certified Professional in Accessibility Core Competencies). These certifications are seen as a formalization of skills necessary for the development of accessible software, providing a prescribed trajectory for developers to learn accessibility as well. *"(Certifications) can ensure that even experienced practitioners focus clearly on a set of the best practices and demonstrate retention of that knowledge"* (AC86). Conversely, certifications are criticized for its quality and the field's excess dependence on it. Relevant study [35] echoes this complaint, showing how certification cannot fully predict competence and performance. However, with an industry of employers still not educated on accessibility, advocates opine that certifications cannot be excluded as an important component for practitioners (AC2, AC84, AC86, AC105) [56].

Influence from leadership (ⓘ). Depending on individual proficiency is not effective without the encouragement and reinforcement from leadership roles. Developers complain that despite best interests, their advocacy for accessibility is not heard and their interest in learning it is not given ample opportunity. *"This was never about me, the developer. I don't decide how much time I spend on a11y, my manager does. You don't need to convince me that a11y is important, I'm convinced. You need to convince my manager so I'm given time to learn a11y best practices and implement them"* (DC161). Prior work [6] similarly report a lack of support from management as an accessibility barrier. On the flipside, explicit statement from leadership about the presence or importance of accessibility has seen prompt impact on teams (DC27, AC20, AC64, AC298).

5.2 Organization

On the organizational level, accessibility is observed to be assigned to a singular team, or is propagated among different teams through onboarding efforts.

Dedicated accessibility teams (ⓘ, ⓒ). We observe that a common practice to integrate accessibility in an organization is assigning a dedicated team responsible for ensuring that the software is accessible. There are different levels of responsibilities for such a team: from focusing on the legal compliance of the end product to interacting with different teams to ensure accessibility practices are being adopted. Comprised of design, development and management personnel with accessibility expertise, these teams are seen helping train others or setting the culture of the organization. A recent study [10] also reported a similar observation, with accessibility teams including employees in various roles and taking on responsibilities for a wide range of tasks. However, practitioners state that housing such teams is expensive, which is why smaller companies rely on external services or teams for ensuring accessibility. We find that companies without in-house development teams depend on *"marketing and dev shops"* that do not prioritize accessibility (AC176). While employing a dedicated accessibility team is a sound strategy for those who can afford it, our data recommends developing a better organizational technique for an indiscriminate integration (DC161, DC288, DC326, DC239, AC123, AC124, AC176, AC179).

Organization-wide accessibility onboarding (ⓐ, ⓘ). Our data indicates that organizations also integrate accessibility by conducting training and onboarding throughout different teams: development, design, product, testing etc. This practice is observed to enable the ubiquitous prioritization of accessibility in the SDLC, placing it as a responsibility for everyone involved. Onboardings are commonly conducted by experts in the team or advocates working as consultants (DC146, DC161, AC92, AC128, AC140, AC277, DC288).

Experts or advocates who coordinated onboarding processes, inform about onboarding challenges and suggestions for best practices. They suggest training teams as early as possible, embedding accessibility into the organization's long term culture, which makes teams more receptive to accessibility practices. When training, it is recommended to focus on building empathy, through the use of ATs, rather than technical processes. *"I make sure they know WHY they're doing this, long before we touch on HOW to make things better"* (DC27). They should be informed on how accessibility is everyone's responsibility and the exact roles they play. Lack of time is noted as a large hurdle for team members being unable to train in or implement accessibility. In such cases, the learning curve should be eased down, helping them solve simple problems, and boosting the confidence and motivation. They can be assigned small batches of refactoring work, based on audits on their daily output. Gradually increasing the baseline of accessibility requirements is observed to be a sustainable and effective way to train busy developers (DC27, DC239, AC77, AC105, AC128, AC183, AC192, AC277).

5.3 Product

The software product brings with it business considerations, including legal and financial aspects, that affect and are affected by accessibility.

Legal compliance (ⓒ, ⓐ). Different countries have put in place different legal requirements for accessibility in digital products, which includes websites and mobile applications. We observe that these regulations, along with the fear of lawsuits they bring, largely

incentivize organizations today to develop accessible software. This mirrors findings from prior research [38, 52] suggesting that the primary motivation for increased adoption of accessibility practices was government laws and regulations. In companies and for stakeholders that resist investments in accessibility, legal compliance can be utilized for effective advocacy and integration (DC171, DC176, DC214, DC239, DC281, AC17, AC104, AC161, AC251). Despite it being a tangible incentive for accessibility, advocates warn of a worrying trend where the exclusive focus is on legal compliance rather than actual accessibility. Guidelines and regulations are generalized and non-comprehensive, enabling software to pass legal requirements, in spite of accessibility issues being present (DC27, DC239, DC275, DC281, AC17, AC252, AC287).

Accessibility requirement during procurement (ⓒ, ⓘ). Our data suggests procurement as medium for incentivizing accessibility. In the procurement process, buyers can add accessibility as a requirement, enforcing the client to integrate accessibility in the software they build. A consistent inclusion, conducted by government agencies, non-profits and other non-software entities, can foster an inclusive culture among small and medium enterprises. A relevant study [88] discussed how including mandatory accessibility criteria in the procurement process can have a transformative positive impact on improving accessibility levels for users. However, it is recommended that appropriate specifications for what constitutes as accessible must be in place, so that the software does not meet requirements without fixing all accessibility concerns (AC90, AC91, AC222)

Investment and profit concerning accessibility development (ⓒ). Our data shows that one of the major challenges of software accessibility is the investment required and the lack of motivation thereof. Speculators suspect whether a profit can be generated from this investment. Especially for smaller companies, such investments often seem beyond affordability (DC27, DC47, DC92, DC288, AC82, AC108, AC133, AC232). Conversely, advocates and users point out reasons to invest in accessibility development. Firstly, especially for government and non-profit software, accessibility should be viewed as a civil right and not a service. Until accessibility is integrated as an intrinsic component of the software, for profit seeking businesses, it needs to be built as a business case. *"The suits always get the 'you're losing money' talk"* (DC43). The UK popularized the concept of 'purple pound' [87], denoting the spending capabilities for disabled households, indicating profitability of technologies promoted towards them. Not only does an inaccessible software alienate people with disabilities, but it can also be unusable by abled users in various situations. We find in our data, software reported for their accessibility issues and companies openly undervaluing accessibility or making false promises are flagged by users and publicly denounced, making users, disabled or otherwise, switch to alternatives. These points have been reported to bolster the business case and aid in conversations to convince stakeholders to invest in accessibility (DC27, DC43, DC161, DC288, AC82, AC133, AC161, AC284, AC299).

6 A11Y VIEWPOINT: PEOPLE

Beyond the organization and the development practices within a project's scope, the individual, whether in the scope of a community or lone initiative, is an important perspective to consider. Based on our findings, there is a growing accessibility community consisting of new and old practitioners, advocates, and users and developers with disabilities, that aid one another in advancing best practices and spreading awareness.

6.1 Learners

Practitioners in the accessibility community begin as learners, and the ease of their path towards expertise determines the quality and priority of accessibility during software development.

Learning accessibility with guidelines and documentations (ⓐ, ⓣ). Accessibility guidelines, like WCAG, are regarded as the first step towards many developers' accessibility journey. Since most organizations refer to WCAG for developing software, learning via the guidelines is an effective strategy (DC116, DC27, AC105). However, our data indicates that software applications developed solely based on WCAG have not been fully accessible, questioning the effectiveness of guidelines as a complete resource for learning. Guidelines are also critiqued for being outdated and too technical. The specifications in the guidelines are noted as being behind the fast changing web and mobile implementations. WCAG specifically is regarded as a web-centric document, specifying very little on native mobile applications. *"There's a huge lack of resources for learning about native app accessibility compared to web accessibility"* (DC116). Developers have to create their own interpretations of native app guidelines, which challenges consistency and completeness of accessibility evaluations. Furthermore, learners review that the guidelines are written in a practical manner, efficient for seasoned engineers to consume, but assume a level of familiarity with technology that makes it difficult for new developers and designers to interpret (DC116, DC126, DC232).

Although not intended as a learning resource, accessibility documentations incorporated with frameworks, tools and libraries are recommended for learners. These not only provide general knowledge on best practices but also better detail nuanced problems and implementations with regards to that specific platform (DC135, DC203, DC214, DC288, AC105, AC128, AC215). However, documentations of development resources are often not well written or featured in the larger documentation library. Documentations are prompted to give more care to accessibility not only for practitioners to discover and learn from them, but for developers at large to better assess their capacity (DC288, AC128, AC215).

Community of shared learning (ⓐ, ⓣ, ⓘ). Our findings show a variety of sources through which new and practiced developers learn about accessibility. The most common occurrence is developers querying for specific development solutions or learning materials, and others sharing suggestions with their preferred resources, tools and implementations. *"Folks what are some essential plugins/tools you use to design and test for accessibility that you can't live without? #accessibility"* (AC279). This community aspect is extended to discourse on technology-agnostic best practices. Mentorship, even in the online capacity, is seen as key to spreading awareness. People with disabilities often share their lived experience on using web or mobile tools, which are received as important sources for understanding software accessibility. We observe this active community encouraging developers to build their own

accessibility-first frameworks or libraries, create online courses, forums and newsletters, publish blogs and podcast episodes, and host practice project development sessions dedicated to software accessibility. These resources, however, are decentralized, voluntarily moderated or catered to a specific group of practitioners. Learners outside the community or new to social media can be isolated from these resources (DC15, DC39, AC75, AC105 and 50 others).

Another resource often shared are talks from advocates and industry experts, covering specific or general topics on accessibility development. There are conferences dedicated to accessibility — Axecon [9] — or comprised of accessibility-focused sessions — Apple's WWDC [28] and CSUNACC [22] — that developers find impactful for disseminating knowledge and raising industry-wide awareness (DC63, AC19, AC28 and 15 others).

Educational integration of software accessibility (ⓐ, ⓘ). Our data suggests that the challenges from a learning environment of decentralized and difficult to discover resources can be alleviated if accessibility is integrated into software education. The accessibility community has regularly voiced the need for such integration, in the undergraduate level, or even earlier. And not only in Computer Science courses that focus on programming solutions, but also in design curriculum. While there have been instances of curricular inclusion, resource sharing and bootcamps, in most cases, accessibility is found to be absent or minimally included in academia (DC27, DC161, DC288, DC332, AC20, AC105, AC192, AC219).

"Education in a11y is not just a slide that says a11y is important" (AC43). Our findings and related literature offer multiple best practices for teaching accessibility to students. The most important step is to make students empathize with accessibility, either through the use of ATs on inaccessible applications or through direct communication with disabled stakeholders. Students need to be provided with specific and tangible examples of inaccessible features and receive instructions on how to fix them (AC43) [33, 40, 64, 67, 79]. Accessibility can be integrated as its standalone course, or as a theme or additional material to existing courses like web development, requirements engineering and more. However, a longitudinal study [100] has found that addition of accessibility in a single course had not changed the development practices of those students after two years. Hence, accessibility can be integrated into every relevant course in the curriculum, so that students are consistently exposed to the different ways accessibility can be affected and fixed in the development cycle (DC225, AC192) [12, 18, 82, 93]. To effectively incorporate accessibility in the curriculum, the instructors must also be educated. Studies show that the biggest hurdles for instructors in teaching accessibility are the lack of specialized materials and formalized learning objectives. Administrative support is necessary for including additional materials in an already packed curriculum, and consistently planning out the integration throughout multiple courses [33, 57, 73, 82]. Lastly, demand in the industry must be established, otherwise courses offered regarding accessibility see poor enrollment [15, 33, 82].

6.2 Practitioners and advocates

Our results provide insight into how practitioners experience accessibility development in the workplace, their community interaction, and their advocacy efforts.

Acquiring expertise through collaboration and empathy (ⓐ). We observe expertise in accessibility not being characterized by the retention of various guidelines and implementation details, but in constant knowledge seeking and sharing. Team collaboration is advocated over depending on a single champion. If everyone in the team is aware of accessibility practices, it prevents teams from reverting back to old ways in the absence of this champion. It is also said to add more experienced eyes to new context-specific accessibility issues in the software, generating better solutions faster. It is suggested to collaborate among different teams, because it provides different perspectives on a concern, and in dealing with that, practitioners can learn and implement better accessibility (DC27, DC161, AC18, AC92, AC258, AC284, AC292). The idea of collaborative learning is well established in educational research, and has been shown to lead to increased engagement and trust among team members, and improved learning outcomes. [54, 55, 74]

Another important tool in cultivating accessibility expertise is empathy. Our results show multiple accounts of practitioners prioritizing accessibility more strongly once they realized the impact of their inaccessible software. Exposure to the lived experience of their disabled user base and experiencing accessibility issues through ATs helped them change their outlook. The need for empathy can be further motivated by observations from prior work on developers' accessibility practices [25], which suggest that developers may find it difficult to relate to some accessibility issues due to a lack of proper understanding of how people with disabilities use technology and software. Statistics, like the ratio of disabled users, is used as a motivator for accessibility development. While a tested argument to convince profit-seeking leadership, statistics is criticized as a reductive measure, reducing disability to a number. Some developers said that such statistics, on a personal level, do not motivate them. However, developers refusing to give accessibility a chance without comprehensive external incentives are criticized as taking a stance originating from ableism and privilege. *"Inclusion and accessibility work is the right thing to do, and it often makes everyone's lives better in a wide range of ways! #a11y is not zero sum!" (AC257).* The community promotes accessibility as a civil right rather than a business or charity case, and strives to help practitioners understand that and nurture empathy (DC13, DC27, DC194, DC239, DC288, DC314, AC3, AC105, AC158, AC174, AC272, AC284, AC292).

Activation online and at work (ⓐ, ⓘ). We find accessibility advocates utilizing both the online sphere and their professional influence at the workplace for accessibility activation. On Twitter, they promote accessibility integration and best practices. They encourage developers by sharing experiences with accessibility development or appropriate resources for easing implementation. Practitioners who have established themselves as accessibility advocates are referred by other developers as specialists in relevant discussions. Developers post about positive experiences learning about accessibility or coding accessibility updates on their own application. With a practiced eye for noticing problematic practices, they report accessibility issues, critique each other's work and hold software organizations accountable. Sometimes organizations themselves promote accessibility, further helping spread awareness. Advocates and practitioners band together against negative takes and promotions (DC4, DC7, AC75, AC125 and 45 others).

In the workplace, advocates talk about facing constant pushback on accessibility-related suggestions. They have to stay defensive, receiving backlash on their strictness and conviction. Some often feel their voice for advocacy is not being heard, resorting to solitary efforts within their teams. In environments where accessibility is being prioritized, however, advocates are provided space and importance. They lead onboarding efforts, run accessibility workshops and hold mentorship programs. Prior work [10] reports on their cross-team interaction to educate everyone about disability and accessibility. *"The most rewarding part of my job is when a team goes from not understanding the impact of their work on the accessibility of their app. To adding automated accessibility testing into their dev process and creating accessible content" (DC146).* Their role is necessitated because of the current state of resource availability and awareness (DC27, DC65, DC116, DC146, DC161, AC78, AC79, AC182, AC205, AC291).

6.3 Users

The group affected most directly by software accessibility are the users with disabilities. In our survey of tweets, we found that the disabled community plays an active role in spreading awareness to developers and organizations, and contributing in the inception of accessible software by sharing accounts of their lived experience.

Feedback on products and practices (ⓘ, ⓐ, ⓒ). Users utilize social media as a direct communication channel with developers and software corporations to provide their feedback on a product's accessibility. Twitter is proven as an effective feedback channel because of its extended reach [94] and shifting the balance of power between users and corporations [97]. They report on accessibility issues, sometimes detailing the use case that caused the issue, and promote for better accessibility testing. *"I have stayed for two hours on chat room but when I went to speak, I couldn't use the button to acess* [sic] *the moderator invitation cause was not acessible* [sic] *. The VoiceOver (screen reader from iOs) can't identify the notification on the top of the screen"* (DC178). They also acknowledge positive updates and subsequently promote the product for its good faith efforts. *"I'm soooo tardy on an update after the awesome devs with Todoist rolled out some #a11y fixes... which are fantastic fyi"* (DC167). Developers also acknowledge the user feedback and the help that insight provided. Such feedback results in fixing accessibility bugs, removing inaccessible services or, at the very least, notifying relevant teams about the issue. Developers themselves provide feedback on frameworks and tools, and appreciate improvements with regards to accessibility. On the contrary, sometimes user feedback is received poorly, characterizing it as slander or mob behaviour, disregarding it as a non-issue or simply ignoring the feedback. In cases where developers had promised for accessibility updates, users hold them accountable on later dates and report any false promises (DC16, DC34, DC124, AC44, AC59 and 40 others).

Beyond feedback on specific products or features, we observe that users are vocal about software accessibility practices at large. For instance, users have voiced their concerns regarding the affordability of new technologies. While popular ATs built into web and mobile platforms may be free, third party or specialized ATs, especially those that require external devices, need to be purchased. Moreover, since accessibility is implemented differently in various desktop and mobile devices, people with disabilities cannot afford high-end ones with the latest updates. *"...disabled people are more likely to fall into poverty and socio-economic injustices make the poorest, more likely to become disabled." "Socio-economic accessibility is a lens that all too often gets ignored"* (AC272). Related work [59] has also found that financial cost is a major barrier for disabled people who need to use assistive technology. Often, these individuals are forced to use less accessible technologies because they are more affordable.

Contribution towards accessible software development (ⓐ, ⓣ, ⓘ). On Twitter, we find that a common phenomenon for providing active feedback is voluntary testing. Disabled users conduct testing on software products in a variety of contexts, for instance: when a new update to an existing software is released, as an exploration of an inaccessible site, to check whether they want to use an app, to understand how ATs interact with the application, to find accessibility bugs and help developers fixing them, for testing the accessibility features for Beta apps, or without specified reason (DC48, AC89 and 18 others). Some users record or stream themselves using a product with their ATs as a way to spread awareness on how people with disabilities interact with applications, which also helps developers understand the state of their apps' accessibility (DC24, DC168, DC251, DC317).

However, freely assisting large corporations for accessibility is not favored by the community as a prolonged practice. It is deemed as a sign that the development organizations do not care about accessibility. *"I'm tired of giving them free labor by escalating issues a robust #a11y testing program should be catching" (DC79).* Incorporating the insight and feedback of disabled users, from the beginning of the project instead of after deployment, is necessary and demands compensation. We observe users trusting and promoting products, that incorporate the insight from disabled consultants in their design, as accessible (DC47, DC278, DC289, AC83).

7 DISCUSSION

The three viewpoints demonstrate a multitude of challenges present in the industry, the current countermeasures, and recommendations to mitigate the limitations. In analyzing these aspects with the combined understanding of the three viewpoints, we find four themes: 1. cost and incentive (ⓒ), 2. awareness and advocacy (ⓐ), 3. technology and resources (ⓣ), and 4. integration and inclusion (ⓘ). These themes emerge as cross-sectional concerns, showing how the viewpoints interrelate with each other. Understanding the interrelations provide a broader picture of the individual issues, their possible causes beyond the immediate context, and the external aspects that must be considered in trying to alleviate their impact. Furthermore, in describing the themes, we provide avenues for future work.

7.1 Cost and Incentives ⓒ

Literature has observed that the key factors behind the deprioritization of software accessibility is its cost of time, money and effort [60]. Bi et al. [14] indicated how these costs affect smaller organizations more, limited by their lack of expertise, management support and schedule flexibility. We expand on these findings, observing

cost factors ranging from maintenance efforts and expensive testing tools in the *process viewpoint*, to hiring specialized engineers, housing dedicated accessibility teams and conducting onboarding in the *profession viewpoint*.

While incentives like SEO improvement and legal regulations exist, these are deemed insufficient compared to the cost. They have also resulted in practices opposed by the accessibility community, for instance, accessibility overlays. This practice is indicative of two larger implications. One, it shows that reducing accessibility to only a regulatory conformance issue motivates companies to opt for the cheapest solution, even if it does not guarantee an accessible product. And two, it predicts a market saturated by similar solutions that aim to fix inaccessible software by technical means. While we cannot predict the technical effectiveness of these solutions, dependence on them prevents accessibility integration in the core development process and decreases the need for accessibility awareness for practitioners.

Yesilada et al. [99] found legislation and business cases to be the conventional motivators. Based on our findings, we propose focusing on three types of incentives:

(1) **ethical**: presenting accessibility as a civil rights concern rather than a technical or financial one, especially for leadership and organizations whose primary goal is not profitability; educating the reluctant on ableist privileges,
(2) **financial**: for businesses whose primary goal is profit, preparing business cases that focus on the loss of user base due to inaccessible products, by including statistics like the "purple pound" and demonstrating the negative reputation garnered online due to accessibility issues; presenting the advantages of early prioritization in its mitigation of maintenance cost,
(3) **institutional**: including accessibility as a skill requirement or certification during hiring, to motivate practitioners learning accessibility; labeling accessibility as a "must have" requirement during procurement, prompting companies to prioritize accessibility.

7.2 Awareness and Advocacy ⓐ

Lack of awareness about accessibility has been repeatedly reported as a cause for inaccessible software, both in our findings and in the literature [6, 14, 19, 21, 52, 53, 60]. Our findings look into multiple factors hindering awareness. Accessibility is not commonly included in computer science, software engineering and design curricula. Practitioners are therefore dependent on scattered resources for learning accessibility, resulting in incomplete knowledge. Those who do not voluntarily learn remain unaware, and this lack of skill perpetuates their resistance to integrate accessibility later in their work. In the workplace, teams are dependent on experts or lone advocates for accessibility. While designations for accessibility specialists exist, it is not a required skill for design or development candidates. Perpetrated by the employers' own lack of awareness, this also, in turn, demotivates candidates from putting effort into learning accessibility and the academia from allocating accessibility-focused courses.

The chain can be broken institutionally, through the collaboration between academia and industry. The industry can prioritize accessibility in the development practice and promote it as a requirement for employment. Academia can produce practitioners with the required capabilities, convincing the industry of the worth of a developer learned in accessibility. Both can contribute to set standards on what constitutes a baseline of accessibility skills and compile comprehensive resources for teaching and learning.

To better increase practitioner awareness, Yesilada et al. [99] suggest WCAG training, while Leite et al. [60] mention social conscience. We suggest the inclusion of the following three components for effective accessibility learning:

(1) **empathy**: through simulation or interaction with people with disability, growing empathy in learners and practitioners is the tool for long-term personal inspiration and cultural shift in organizations.
(2) **collaboration**: cooperating with different teams and people provides insight into how accessibility affects each other's work, unearthing new insights and strengthening ones grasp on the matter.
(3) **gradual assimilation**: to prevent professionals from being intimidated by its steep learning curve, accessibility can be taught and trained as a progression of small manageable steps accompanied by regular audits.

Institutional changes of this nature require time, and until then, advocacy of accessibility in online communities — through resource creation and sharing, and observing user feedback — and in the workplace — through tutelage, cooperation and endorsing its consideration in every team and process — are valuable initiatives for raising awareness.

7.3 Technology and Resources ⓣ

Accessibility requires an array of technical, literary and institutional resources to be properly and efficiently integrated into software development. The literature pointed towards developers' dependence on existing tools [21], quality of accessibility evaluation tools [60] and outdated standards [14] as impediments for accessibility development. Our findings introduce multiple shortcomings of existing resources and instances where no resource exists.

(1) **Discoverability** is a key issue for technical resources, where the lack of documentation or the difficulty to locate available resources, like accessible UI component libraries, hinders their implementation by developers. It is also an issue for learning resources, which are vital for voluntary learning.
(2) **Quality** of literary resources like guidelines are also put in question, either because of their inability to update the text according to the evolving technology, or the writing style. These should be written both technically for easy translation to programming code and intuitively to be consumed by a broad non-technical audience.
(3) **Ease of integration** into the development process is posed as an issue for technological resources. Automated testing tools are good examples of tools satisfying that requirement, incorporating its results into a project's CI pipeline. Similar tools for other processes, like designing UI mock ups and manual testing, can be developed.
(4) On an institutional level, **human resources** like accessibility specialists (for onboarding), auditors (for assessing

individual work) and people with disabilities (for their insight) should be accommodated.

Focus should not solely be on the individuals or organization developing a software. Some technologies, that are beyond the purview of developers and which many are dependent on for developing their own software, can immensely improve software accessibility. Frameworks that construct web features, design systems that dictate the design principles of UI components, web browsers that host web pages, platforms that provide accessibility APIs, operating systems and devices that control how ATs function exemplify such technologies and their responsibility on accessibility.

7.4 Integration and Inclusion ⓘ

The most common theme present in our findings is the need for integrating accessibility practices into conventional software development, from individual practitioner and team tasks to organizations and the industry at large. The previous three themes included instances that can be utilized for integration, focusing on technical (e.g., automated tests connected to the CI pipeline) and logistical (e.g., certifications and dedicated teams) solutions. More importantly, however, integration should be complemented with inclusion; people with disabilities, for whom accessibility is most impactful, ought to be involved in these processes. Literature suggests user-centered design and user testing to better incorporate the diverse needs of different disabilities [6, 13, 21, 99]. Based on our findings, the three avenues their involvement can be established through are:

(1) **community**: utilizing the active community formed in online spaces to understand their challenges with general software techniques (e.g., navigation, color, labeling), feedback on accessibility issues in specific applications, and insight into logistical concerns (e.g., affordability of ATs),
(2) **organization**: incorporating them directly in the team, as stakeholders during procurement, professionals for user testing, and tutors during onboarding efforts,
(3) **consultancy**: consolidating their insight for specific practices, for instance, when constructing a curriculum and executing the lessons, planning an a11y strategy for a project, or designing UI components and features.

A thorough and effective integration of accessibility in software is essential to discard its perception as an extra consideration, instead forming a culture that establishes and normalizes the practices.

8 CONCLUSION

In order to understand accessibility in software products and the limitations behind its normalization, it is imperative to investigate how software is developed with accessibility in mind, from multiple relevant perspectives. Our work explores tweets from software practitioners, along with users and organizations, to bridge the gap of a holistic view of software accessibility development. We qualitatively analyze 637 tweet conversations, consisting of more than 8500 unique tweets from 1800 users, to discover three viewpoints: process, profession and people. These viewpoints group relevant practices, enabling a nuanced understanding of the existing conventions and challenges faced. Four themes emerged from the findings — cost and incentive, awareness and advocacy, technology and resources, and integration and inclusion — providing points to focus on for understanding software accessibility and better integrating it into contemporary development practices.

Our holistic investigation unearthed multiple facets of software accessibility, on which further research can be conducted, exploring individual aspects in more depth. A limitation of our study has been that we analyzed the tweets alone, leaving room for communicating with those who tweeted, or those active in the accessibility community, for a more thorough understanding of their concerns. A temporal analysis of the tweets, collected over a span of multiple years, can be conducted to understand how accessibility concerns and focus have evolved. New tools and resources for accessibility design, development, testing and learning can be developed, or old ones updated, as per our insights on integration, discoverability and ease of use. New perspectives emerged from our findings that have yet to be represented in the literature: accessibility advocates, certification organizations, management or hiring personnel, consultants and trainers. Future studies can delve into their insights and functions.

ACKNOWLEDGMENTS

This work has been supported, in part, by award numbers 2211790, 1823262, and 2106306 from the National Science Foundation. We are grateful for the detailed feedback from the anonymous reviewers of this paper, which helped improve this work.

REFERENCES

[1] Essa Adhabi and Christina Blash Anozie. 2017. Literature review for the type of interview in qualitative research. *International Journal of Education* 9, 3 (2017), 86–97.
[2] Amaia Aizpurua, Myriam Arrue, Simon Harper, and Markel Vigo. 2014. Are users the gold standard for accessibility evaluation?. In *Proceedings of the 11th Web for All Conference*. 1–4.
[3] Wajdi Aljedaani, Mohamed Wiem Mkaouer, Stephanie Ludi, and Yasir Javed. 2022. Automatic Classification of Accessibility User Reviews in Android Apps. In *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*. IEEE, 133–138.
[4] Eman Abdullah AlOmar, Wajdi Aljedaani, Murtaza Tamjeed, Mohamed Wiem Mkaouer, and Yasmine N El-Glaly. 2021. Finding the needle in a haystack: On the automatic identification of accessibility user reviews. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. 1–15.
[5] Ali Alsaawi. 2014. A critical review of qualitative interviews. *European Journal of Business and Social Sciences* 3, 4 (2014).
[6] Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. 2020. Accessibility issues in Android apps: state of affairs, sentiments, and ways forward. In *2020 IEEE/ACM 42nd International Conference on Software Engineering*. ICSE, Virtual, 1323–1334.
[7] Hamza Alshenqeeti. 2014. Interviewing as a data collection method: A critical review. *English linguistics research* 3, 1 (2014), 39–45.
[8] Humberto Lidio Antonelli, Sandra Souza Rodrigues, Willian Massami Watanabe, and Renata Pontin de Mattos Fortes. 2018. A survey on accessibility awareness of Brazilian web developers. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*. 71–79.
[9] axe-con Digital Accessibility Conference. 2022. https://www.deque.com/axe-con/.
[10] Shiri Azenkot, Margot J Hanley, and Catherine M Baker. 2021. How Accessibility Practitioners Promote the Creation of Accessible Products in Large Companies. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–27.
[11] Ibtehal S Baazeem and Hend S Al-Khalifa. 2015. Advancements in web accessibility evaluation methods: how far are we?. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*. 1–5.
[12] Catherine M Baker, Yasmine N El-Glaly, and Kristen Shinohara. 2020. A systematic analysis of accessibility in computing education research. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 107–113.

[13] Tingting Bi, Xin Xia, David Lo, and Aldeida Aleti. 2021. A First Look at Accessibility Issues in Popular GitHub Projects. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 390–401.

[14] Tingting Bi, Xin Xia, David Lo, John Grundy, Thomas Zimmermann, and Denae Ford. 2022. Accessibility in software practice: A practitioner's perspective. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 4 (2022), 1–26.

[15] Paul R Bohman. 2012. *Teaching accessibility and design-for-all in the information and communication technology curriculum: Three case studies of universities in the United States, England, and Austria.* Utah State University.

[16] Gargi Bougie, Jamie Starke, Margaret-Anne Storey, and Daniel M German. 2011. Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*. 31–36.

[17] Erin L Brady, Yu Zhong, Meredith Ringel Morris, and Jeffrey P Bigham. 2013. Investigating the appropriateness of social network question asking as a resource for blind users. In *Proceedings of the 2013 conference on Computer supported cooperative work*. 1225–1236.

[18] Julian Brinkley. 2020. Participation at What Cost? Teaching Accessibility Using Participatory Design: An Experience Report. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 114–120.

[19] Shiya Cao and Eleanor Loiacono. 2019. The state of the awareness of web accessibility guidelines of student website and app developers. In *International Conference on Human-Computer Interaction*. Springer, 32–42.

[20] Sen Chen, Chunyang Chen, Lingling Fan, Mingming Fan, Xian Zhan, and Yang Liu. 2021. Accessible or Not An Empirical Investigation of Android App Accessibility. *IEEE Transactions on Software Engineering* (2021).

[21] Santiago Liñán Christopher Vendome, Diana Solano and Mario Linares-Vásquez. 2019. Can everyone use my app? An Empirical Study on Accessibility in Android Apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE.

[22] CSUN Conference. 2022. https://www.csun.edu/cod/conference/sessions/.

[23] Joshue O Connor. 2008. User testing: How to involve users in technical web development cycles as a natural evolution in the creation of inclusive technologies and accessible content. In *International Conference on Computers for Handicapped Persons*. Springer, 258–263.

[24] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

[25] Michael Crabb, Michael Heron, Rhianne Jones, Mike Armstrong, Hayley Reid, and Amy Wilson. 2019. Developing accessible services: Understanding current knowledge and areas for future support. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[26] Material Design. 2022. https://material.io/.

[27] Accessibility Foundations Human Interface Guidelines Design Apple Developer. 2022. https://developer.apple.com/design/human-interface-guidelines/foundations/accessibility/.

[28] WWDC22 Apple Developer. 2022. https://developer.apple.com/wwdc22/.

[29] Espresso | Android Developers. 2022. https://developer.android.com/training/testing/espresso.

[30] SEO Starter Guide: The Basics | Google Search Central | Documentation | Google Developers. 2022. https://developers.google.com/search/docs/beginner/seo-starter-guide. (Accessed on 09/07/2022).

[31] UIAccessibility | Apple Developer Documentation. 2022. https://developer.apple.com/documentation/objectivec/nsobject/uiaccessibility.

[32] Niklas Egger, Gottfried Zimmermann, and Christophe Strobbe. 2022. Overlay tools as a support for accessible websites–possibilities and limitations. In *International Conference on Computers Helping People with Special Needs*. Springer, 6–17.

[33] Yasmine N El-Glaly. 2020. Teaching Accessibility to Software Engineering Students. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 121–127.

[34] Marcelo Medeiros Eler, Leandro Orlandin, and Alberto Dumont Alves Oliveira. 2019. Do Android app users care about accessibility? an analysis of user reviews on the Google play store. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*. 1–11.

[35] Ali Dehghanpour Farashah, Janice Thomas, and Tomas Blomquist. 2019. Exploring the value of project management certification in selection and recruiting. *International Journal of Project Management* 37, 1 (2019), 14–26.

[36] Glen Farrelly. 2011. Practitioner barriers to diffusion and implementation of web accessibility. *Technology and disability* 23, 4 (2011), 223–232.

[37] Angular Framework. 2022. https://angular.io/.

[38] Jonathan Frank. 2008. Web accessibility for the blind: Corporate social responsibility or litigation avoidance?. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. IEEE, 284–284.

[39] Andre Pimenta Freire, Rudinei Goularte, and Renata Pontin de Mattos Fortes. 2007. Techniques for developing more accessible web applications: a survey towards a process classification. In *Proceedings of the 25th Annual ACM international Conference on Design of Communication*. 162–169.

[40] Greg Gay. 2021. Teaching accessibility awareness with games. In *Proceedings of the 18th International Web for All Conference*. 1–3.

[41] Cole Gleason, Stephanie Valencia, Lynn Kirabo, Jason Wu, Anhong Guo, Elizabeth Jeanne Carter, Jeffrey Bigham, Cynthia Bennett, and Amy Pavel. 2020. Disability and the COVID-19 pandemic: Using Twitter to understand accessibility during rapid societal transition. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–14.

[42] Thorsten Gruber, Isabelle Szmigin, Alexander E Reppel, and Roediger Voss. 2008. Designing and conducting online interviews to investigate interesting consumer phenomena. *Qualitative Market Research: An International Journal* (2008).

[43] Human Interface Guidelines. 2022. https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/.

[44] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. 2016. A needle in a haystack: What do twitter users say about software?. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 96–105.

[45] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. 2017. An exploratory study of twitter messages about software applications. *Requirements Engineering* 22, 3 (2017), 387–412.

[46] Roman Haas, Daniel Elsner, Elmar Juergens, Alexander Pretschner, and Sven Apel. 2021. How can manual testing processes be optimized? developer survey, optimization guidelines, and case studies. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1281–1291.

[47] Samine Hadadi. 2021. Adee: Bringing Accessibility Right Inside Design Tools. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–4.

[48] Google Accessibility Help. 2022. https://support.google.com/accessibility/?hl=en#topic=9071908. (Accessed on 08/24/2022).

[49] ADA.gov homepage. 2022. https://www.ada.gov/.

[50] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O. Hall, and Adriana Iamnitchi. 2019. Mentions of security vulnerabilities on reddit, twitter and github. In *IEEE/WIC/ACM International Conference on Web Intelligence*. 200–207.

[51] Shuichi Ichioka, Estelle Pouget, Takao Mimura, Jun Nakajima, and Toshihiro Yamauchi. 2020. Accessibility Service Utilization Rates in Android Applications Shared on Twitter. In *International Conference on Information Security Applications*. Springer, 101–111.

[52] Yavuz Inal, Frode Guribye, Dorina Rajanen, Mikko Rajanen, and Mattias Rost. 2020. Perspectives and practices of digital accessibility: A survey of user experience professionals in nordic countries. In *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*. 1–11.

[53] Yavuz Inal, Kerem Rızvanoğlu, and Yeliz Yesilada. 2019. Web accessibility in Turkey: awareness, understanding and practices of user experience professionals. *Universal Access in the Information Society* 18, 2 (2019), 387–398.

[54] David Johnson, Roger Johnson, and Karl Smith. 2007. The State of Cooperative Learning in Postsecondary And Professional Settings. *Educational Psychology Review* 19 (03 2007), 15–29. https://doi.org/10.1007/s10648-006-9038-8

[55] David W Johnson and Roger T Johnson. 2008. Social interdependence theory and cooperative learning: The teacher's role. In *The teacher's role in implementing cooperative learning in the classroom*. Springer, 9–37.

[56] Susan Schott Karr. 2001. IT certification pays off.(Education). *Financial executive* 17, 9 (2001), 60–62.

[57] Saba Kawas, Laura Vonessen, and Amy J Ko. 2019. Teaching accessibility: A design exploration of faculty professional development at scale. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 983–989.

[58] Karin Klooster. 2016. Applying a Security Testing Methodology: a Case Study.

[59] Ravi Kuber, Ama Hastings, Matthew Tretter, and Donal Fitzpatrick. 2012. Determining the Accessibility of Mobile Screen Readers for Blind Users. (06 2012). https://doi.org/10.2316/P.2012.772-003

[60] Manoel Victor Rodrigues Leite, Lilian Passos Scatalon, André Pimenta Freire, and Marcelo Medeiros Eler. 2021. Accessibility in the mobile development industry in Brazil: Awareness, knowledge, adoption, motivations and barriers. *Journal of Systems and Software* 177 (July 2021), 110942. https://doi.org/10.1016/j.jss.2021.110942

[61] React – A JavaScript library for building user interfaces. 2022. https://reactjs.org/. (Accessed on 08/29/2022).

[62] Yvonna S Lincoln and Egon G Guba. 1985. *Naturalistic inquiry*. sage.

[63] Michael Longley and Yasmine N Elglaly. 2021. Accessibility Support in Web Frameworks. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–4.

[64] Stephanie Ludi. 2007. Introducing accessibility requirements through external stakeholder utilization in an undergraduate requirements engineering course. In *29th International Conference on Software Engineering (ICSE'07)*. IEEE, 736–743.

[65] Sergio Luján-Mora and Firas Masri. 2012. Integration of web accessibility into agile methods. In *Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS 2012)*. 123–127.

[66] Darliane Miranda and João Araujo. 2022. Studying industry practices of accessibility requirements in agile development. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 1309–1317.

[67] & Research Nielsen Norman Group: UX Training, Consulting. 2022. https://www.nngroup.com/.

[68] Use of the AccessibilityService API. 2022. https://support.google.com/googleplay/android-developer/answer/10964491?hl=en.

[69] Semantics MDN Web Docs Glossary: Definitions of Web-related terms | MDN. 2022. https://developer.mozilla.org/en-US/docs/Glossary/Semantics.

[70] World Health Organization. 2019. World Report on Disability. https://www.who.int/teams/noncommunicable-diseases/sensory-functions-disability-and-rehabilitation/world-report-on-disability. (Accessed on 08/24/2022).

[71] Débora Maria Barroso Paiva, André Pimenta Freire, and Renata Pontin de Mattos Fortes. 2021. Accessibility and software engineering processes: A systematic literature review. *Journal of Systems and Software* 171 (2021), 110819.

[72] Twitter API Documentation | Docs | Twitter Developer Platform. 2022. https://developer.twitter.com/en/docs/twitter-api. (Accessed on 09/07/2022).

[73] Cynthia Putnam, Maria Dahman, Emma Rose, Jinghui Cheng, and Glenn Bradford. 2016. Best practices for teaching accessibility in university classrooms: cultivating awareness, understanding, and appreciation for diverse users. *ACM Transactions on Accessible Computing (TACCESS)* 8, 4 (2016), 1–26.

[74] Sharifah Nadiyah Razali, Helmi Noor, Mohd Ahmad, and Faaizah Shahbodin. 2017. Enhanced student soft skills through integrated online project based collaborative learning. *International Journal of ADVANCED AND APPLIED SCIENCES* 4 (03 2017), 59–67. https://doi.org/10.21833/ijaas.2017.03.010

[75] Bob Regan. 2004. Accessibility and design: A failure of the imagination. In *Proceedings of the 2004 international cross-disciplinary workshop on Web accessibility (W4A)*. 29–37.

[76] Jose E Reyes Arias, Kale Kurtzhall, Di Pham, Mohamed Wiem Mkaouer, and Yasmine N Elglaly. 2022. Accessibility Feedback in Mobile Application Reviews: A Dataset of Reviews and Accessibility Guidelines. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–7.

[77] Robolectric. 2022. http://robolectric.org/.

[78] Colin Robson. 2002. *Real world research: A resource for social scientists and practitioner-researchers*. Wiley-Blackwell.

[79] Brian J Rosmaita. 2006. Accessibility first! A new approach to web design. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*. 270–274.

[80] Jose Ramon Saura, Daniel Palacios-Marqués, and Domingo Ribeiro-Soriano. 2021. Using data mining techniques to explore security issues in smart living environments in Twitter. *Computer Communications* 179 (2021), 285–295.

[81] Abhishek Sharma, Yuan Tian, and David Lo. 2015. What's hot in software engineering Twitter space?. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 541–545.

[82] Kristen Shinohara, Saba Kawas, Amy J Ko, and Richard E Ladner. 2018. Who teaches accessibility? A survey of US computing faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 197–202.

[83] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. 2014. Software engineering at the speed of light: how developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering*. 211–221.

[84] Get started with Accessibility Scanner. 2022. p. https://support.google.com/accessibility/android/answer/6376570?hl=en.

[85] Anselm L Strauss and Juliet Corbin. 1994. Grounded Theory Methodology-An Overview. Handbook of Qualitative Research. NK Denzin and YS Lincoln.

[86] Murtaza Tamjeed. 2020. *Accessibility in User Reviews for Mobile Apps: An Automated Detection Approach*. Rochester Institute of Technology.

[87] Understanding the Purple Pound Market. 2022. https://wearepurple.org.uk/understanding-the-purple-pound-market/.

[88] William J Tibben and Gunela Astbrink. 2012. Accessible communications: Tapping the potential in public ICT procurement policy. (2012).

[89] Shari Trewin, Brian Cragun, Cal Swart, Jonathan Brezin, and John Richards. 2010. Accessibility challenges and tool features: an IBM Web developer perspective. In *Proceedings of the 2010 international cross disciplinary conference on web accessibility (W4A)*. 1–10.

[90] Aleksi Vuorjoki et al. 2021. A developer-friendly automated web GUI test strategy. (2021).

[91] WCAG 2 Overview | Web Accessibility Initiative (WAI) | W3C. 2019. https://www.w3.org/WAI/standards-guidelines/wcag/. (Accessed on 08/29/2022).

[92] WAI-ARIA Overview | Web Accessibility Initiative (WAI) | W3C. 2022. https://www.w3.org/WAI/standards-guidelines/aria/.

[93] Annalu Waller, Vicki L Hanson, and David Sloan. 2009. Including accessibility within and beyond undergraduate computing courses. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*. 155–162.

[94] James C Ward and Amy L Ostrom. 2006. Complaining to the masses: The role of protest framing in customer-created complaint web sites. *Journal of consumer Research* 33, 2 (2006), 220–230.

[95] WebAIM. 2022. WebAIM: The WebAIM Million - The 2022 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/. (Accessed on 08/24/2022).

[96] Grant Williams and Anas Mahmoud. 2017. Mining twitter feeds for software user requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 1–10.

[97] Lan Xia. 2013. Effects of companies' responses to consumer criticism in social media. *International Journal of Electronic Commerce* 17, 4 (2013), 73–100.

[98] Shunguo Yan and PG Ramachandran. 2019. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing (TACCESS)* 12, 1 (2019), 1–31.

[99] Yeliz Yesilada, Giorgio Brajnik, Markel Vigo, and Simon Harper. 2015. Exploring perceptions of web accessibility: a survey approach. *Behaviour & Information Technology* 34, 2 (2015), 119–134.

[100] Qiwen Zhao, Vaishnavi Mande, Paula Conn, Sedeeq Al-khazraji, Kristen Shinohara, Stephanie Ludi, and Matt Huenerfauth. 2020. Comparison of methods for teaching accessibility in university computing courses. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility*. 1–12.