Design Systems and Component Packages as an Interface for Accessibility

Sofia Diniz
sdms@cin.ufpe.br
Universidade Federal de Pernambuco (UFPE)
Recife, Brazil

Kiev Gama
kiev@cin.ufpe.br
Universidade Federal de Pernambuco (UFPE)
Recife, Brazil

ABSTRACT

A design system is a collection of interrelated patterns and shared practices, systematically organized to fulfill the objectives of digital products. This research aims to understand how Design Systems can influence software developers in implementing accessibility guidelines. For this purpose, leading design systems were analyzed for their coverage of accessibility, and a component package seamlessly integrating accessibility concerns was created and tested by software developers. Results indicated that an accessible Design System can positively influence developers to incorporate accessibility, thus enhancing the user experience in the final product.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in accessibility**.

KEYWORDS

Accessibility, Design system, Swift, Inclusive design

1 INTRODUCTION

To drive innovation and profitability, many software companies are fostering diverse and inclusive teams [16, 17]. Despite the resource-intensive process, evidence shows that such diversity enhances team performance and innovation, though potential conflicts may arise [15, 26]. Recognizing and accommodating diverse groups such as women, LGBTQIA+, people of color, and individuals with disabilities is key for inclusion. In requirements engineering, understanding diverse users' needs is essential. For example, algorithmic bias in facial recognition software often stems from a lack of diversity awareness [30], and cultural misalignment in websites can exclude or offend users [23]. While gender-inclusivity in software engineering has been widely studied [31], recent research expands to other human aspects like age, disability, and neurodiversity. Developers, typically young, male, and of high socio-economic status, may inadvertently exclude underrepresented groups [14]. Thus, a "one-size" design approach is inadequate; it's key to employ techniques that address diverse user needs for inclusive software.

Our previous study [13] highlighted the significant gaps in developers' awareness and practices regarding user diversity. We observed that developers who do not belong to underrepresented groups exhibited a lower awareness of their users' diversity dimensions [27] such as race, ethnicity, gender, disability, neurodiversity, and age. Design systems emerged as a potential tool to circumvent this limited developer empathy around Diversity and inclusion (D&I) toward users. A design system is a collection of interrelated patterns and shared practices, systematically organized to fulfill the objectives of digital products [19]. The current study takes a step further by providing practical solutions through design systems that cover accessibility, thereby bridging the gap between awareness and implementation. The results of our previous study serve as a foundation, demonstrating that increasing developer awareness is fundamental but must be complemented by tangible tools and guidelines to achieve truly inclusive software. By connecting these two phases of research, we aim to contribute to a more inclusive digital environment, where both developers' awareness and the tools they use are aligned to support diverse user needs. This continuation not only validates our previous findings but also provides actionable insights to enhance the inclusivity of software products, ultimately benefiting a wider range of users.

Building on these findings, this research aims to further explore how design systems can influence software developers' implementation of D&I concerns toward users. To evaluate the effectiveness of such approach, we chose disability as the sole diversity dimension to focus, by focusing on accessibility guidelines. By analyzing leading design systems for their adherence to accessibility standards and testing a fully accessible version with software developers who not necessarily know details about accessibility, we seek to understand the impact of D&I aware design systems on promoting inclusive software development practices.

2 BACKGROUND AND RELATED WORK

2.1 Accessibility and Usability

In the context of Software Quality, accessibility is a sub-characteristic of of usability [10]. The concept of accessibility has evolved from making information available to people with disabilities, to ensuring it is accessible to all users, irrespective of their situation or capabilities [33]. This broader perspective acknowledges that anyone can experience temporary or permanent disabilities [2], such as a parent holding a child and being unable to use one arm [24]. This definition also includes various disabilities such as visual, hearing, cognitive, and motor impairments. [19]. To assess the accessibility of a mobile application, following established guidelines is essential. In response to the growing need for such content, the World Wide Web Consortium (W3C) developed the Web Content Accessibility Guidelines (WCAG). These guidelines have expanded to include mobile applications, ensuring they accommodate users with blindness, deafness, hearing loss, limited movement, and other impairments. The WCAG's recommendations are presented as success criteria —testable statements that are not technology-specific [4].

In a broader perspective, usability is a vital factor in developing high-quality user interfaces. In mobile application design, usability refers to how effectively, efficiently, and satisfyingly a user can achieve a specified goal within a certain context [33]. Therefore, good usability in a user interface allows users to quickly familiarize

themselves without much guidance and achieve their goals with minimal friction. When users return to a product, they should also easily remember how to navigate and perform familiar actions [19].

2.2 Design methodologies and guidelines

The process of incorporating accessibility into a digital product is often related to design methodologies.Universal Design (UD) is a prominent example of an approach in Human-Computer Interaction (HCI) design that respects, values, and accommodates the widest range of human abilities and needs when designing computer-based products [28]. UD strives to design products that cater to the broadest possible user base while acknowledging that different solutions may be required for different contexts [33]. Another known approach is User-centered Design (UCD), a multidisciplinary activity that incorporates human factors and ergonomics techniques, focusing specifically on developing usable interactive systems [11]. This process is based on the understanding of users, tasks and environments and refined by user-centered iterative evaluations [33]. Additionally, the Universal Design Mobile Interface Guidelines (UDMIG) v.2.0 is an approach that provides guidelines for designing mobile interfaces for older adults [28]. Each set of guidelines includes the an element known as the "Person" component, responsible for addressing accommodation for various abilities. These guidelines are based on established strategies for desktop and mobile interfaces, and research on design for the aging population, integrating multiple strategies for mobile interface design [28]. The prominence and recognition of guidelines like UCD and UD indicates the design community's commitment to accessibility. It represents a positive outcome of increased exposure to accessibility in education and training, making these concerns relevant to the design process.

2.3 Software developers and accessibility

The development of an accessible mobile application is not solely the responsibility of designers. Software developers code user interfaces and directly impact the interaction between applications, users, and assistive technologies. Unfortunately, accessibility guidelines are often overlooked by developers, resulting in significant user exclusion [36]. This oversight can be attributed to factors such as lack of customer priority, constrained project timelines, and lack of emphasis on the topic within the software development company. Additionally, developers have demonstrated limited familiarity with accessibility guidelines, suggesting a need for increased training on this subject [5] and demonstrating a lack of awareness for how their indifference impacts disabled users [18].

2.4 Team dynamics

Considering the aforementioned approach taken by designers and developers to accessibility, significant discrepancy can easily be noticed. Consequently, the multidisciplinary collaboration between them in software organizations, while fundamental, can face challenges. For instance, a "culture of defensiveness" has been identified between these groups, leading to communication barriers [3]. The designers' perceptions of developers' empathy towards design work has also been examined. They found that developers' misunderstandings of design work often led to miscommunication, impacting the final product's quality [22]. It has also been discovered that breakdowns in designer-developer collaboration often happened when: (1) designers failed to communicate specific design details, (2) a particular case was not covered in the design, or (3) the design did not consider developers' technical constraints [21].

2.5 Design systems

Based on this conflicted scenario, design systems have been proposed as a medium for facilitating the collaboration between designers and developers [20]. A Design System (DS) is essentially composed of various elements such as layout, styles, components, regions, content, and usability [34]. Usability is where D&I can be integrated, as a typical DS often expands usability to encompass accessibility and internationalization, both of which are deeply connected to D&I dimensions (i.e., disability and culture, respectively). A DS provides meticulously detailed visual guidelines and recommendations that software teams must diligently follow. Design systems emerged during recent years in the industry as a tool aimed at the design and development of information and communication technologies [20]. The motivation for companies to adopt a DS usually originates from their design-related challenges, with common scenarios including issues in the consistency of their UI design, maintainability of design deliverables and the code related to them, and the collaboration between designers and developers [25]. When considering the perspectives of designers and developers towards accessibility, the purpose of a DS and the problems it can solve, it is reasonable to speculate whether a DS fully covering accessibility could facilitate developers adhering to these guidelines.

3 METHOD

The methodologies used in this study were designed to ensure a general evaluation of design systems and their impact on accessibility implementation. This mixed-methods approach combined qualitative analysis with practical testing to provide a broad understanding. First, a detailed, criterion-based qualitative evaluation of design systems was conducted. Followed by the practical implementation of an accessible library, which was then subjected to hands-on testing by developers. Feedback was collected and incorporated iteratively to refine the library and understand its practical application, ensuring an assessment of how design systems can influence software development practices toward accessibility.

3.1 Choice of Design Systems

In order to better comprehend the compliance of today's software with the WCAG 2.2 accessibility guidelines, the Forbes Top 100 Digital Companies ranking [12] was consulted. The top five companies were selected to have their Design Systems evaluated, which at the time of this research were Apple, Microsoft, Samsung, Alphabet, and AT&T. Out of the five companies, only AT&T [1], did not have any publicly available design system, leading to its disqualification from the study.

Although the WCAG 2.2 guidelines encompass mobile applications, not all directives are fully applicable within a DS due to its limited scope. For example, WCAG's Success Criterion 3.3.7, which suggests auto-populating repeated information [35], cannot be enforced by a Design System. To address these limitations, a filter was applied to the WCAG 2.2 guidelines to assess their feasibility

for implementation in a Design System. After filtering, 46 out of 91 guidelines remained valid for the study.

3.2 Evaluation of Design Systems

For the evaluation of the Design Systems, a table was created with each row representing a Success Criterion and each column representing a Design System. Each Success Criterion was scored as follows: Complies (fully meets the guideline), Partial (partially meets the guideline), Does not comply (contradicts the guideline), No evidence (no mention of the guideline found.)

The evaluation of each of the 46 success criteria began by comprehending that criterion on WCAG's website. Their descriptions frequently included specific jargon with URLs that redirected to their definitions or mathematical formulas, therefore reading often expanded into comprehending related concepts and analyzing mathematical formulas. Afterwards, if the DS in question had an official Figma package, it was carefully evaluated, looking for signs of compliance in every single component that the current guideline could be applied to. Subsequently, the official documentation for said components was consulted, and any accessibility directives related to the criterion were accounted for. Then, if the DS had a page dedicated to general accessibility principles and guidelines that they followed, said page was consulted. Finally, a targeted research was done on the entire documentation using keywords present in or related to the success criterion at hand, aiming to find any other mentions of the criterion, which often led to the reading of other articles or pages of that documentation. These steps were rigorously applied in the evaluation of each criterion on each DS.

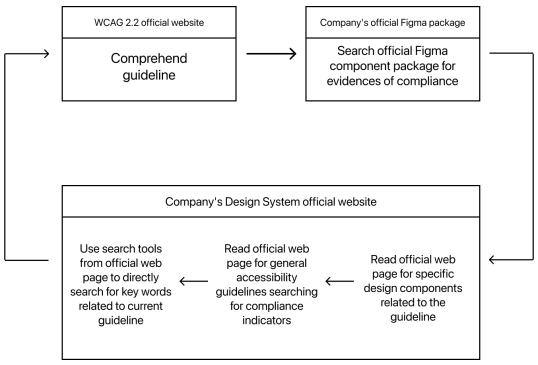

Figure 1: Steps taken in the evaluation of each component

3.3 Creation and testing of ALib

Following the evaluation, the DS with the most "Complies" score was chosen. This DS was then re-evaluated, with notes added to each Figma component to indicate necessary changes for full WCAG 2.2 compliance. With these requirements defined, an IDE, programming language, and UI framework were chosen based on their compatibility with the DS. For instance, Apple's HIG components are highly compatible with Swift frameworks like UIKit and SwiftUI, while Google's Material Design works well with frameworks such as React.JS. The improved components were developed and published as a code library, facilitating their incorporation by developers in their projects. After publishing the library, eight software developers with varying experience levels were invited to test it, provide feedback, and participate in a brief online interview. They answered questions about their previous experience with accessibility in software development, their thoughts on the topic, and their experience with the library. Given the limited sample size, a straightforward thematic approach was employed for the analysis, base on in vivo coding (using participants' own words) to identify recurring themes and patterns.

4 DESIGN SYSTEMS EVALUATION

Table 1: Design sytems' evaluation

	Complies	Partial	Does not comply	No evidence
Human Interface Guidelines (Apple)	39	4	1	2
Fluent 2 (Microsoft)	29	2	2	13
One UI (Samsung)	29	3	6	8
Material Design (Google/Alphabet)	21	2	6	17

One UI (Samsung) - One UI provided the best browsing experience for evaluation. Its instructions and guidelines are conveniently condensed into a single PDF, making navigation and information search simpler. At the end of the document, there is a checklist for designers to ensure their projects meet basic accessibility requirements. This is an invaluable tool, given that accessibility guides like WCAG often include several usability hurdles. This checklist also helped OneUI comply with most of the WCAG 2.2 guidelines since many of them are mentioned there. Among the four Design Systems, OneUI implemented the third-highest number of guidelines successfully. Although it didn't meet all the WCAG 2.2 requirements, it stood out for its concern for accessibility. However, it is worth noting that there is no mention or representation of diversity throughout the DS. Despite being the third-largest technology company globally and dominating markets in culturally, customarily, and ethnically diverse countries such as India, Argentina, and the Philippines [32], there is no mention of diversity or representation in the entire DS. This may be related to the company's cultural values [29], which do not mention diversity as their focus.

Fluent 2 (Microsoft) - Fluent 2 offers four distinct versions tailored to specific platforms (iOS, Android, Web, and Windows). Due to the study's focus on mobile software development and the limited availability of versions at the time of evaluation (iOS and Web only), the iOS version of Fluent 2 was chosen for further analysis. Fluent 2 prominently declares adherence to all WCAG 2.1 AA accessibility guidelines on its dedicated accessibility page. Evaluations using the updated WCAG 2.2 standard yielded similar results, with the primary discrepancy between Fluent 2's and Apple's scores attributable to a lack of explicit mention of certain WCAG guidelines within the Fluent 2 documentation. Fluent 2, Apple's DS and Material Design demonstrate awareness of user representation and inclusion. While not as broad as the efforts undertaken by Apple and Google, Fluent 2 incorporates a subtle yet diverse set of user profile picture examples throughout the DS. This approach aligns with their design principle, "One for all, all for one," which they define as "You want to be included. Your experiences should consider, learn, and reflect a range of perspectives and abilities for the benefit

of all" [6]. However, beyond accessibility features and the aforementioned user profile examples, no further explicit references to D&I were identified within the DS. This suggests that while Fluent 2 is on the right track, there is still room to improvem in this area.

Human Interface Guidelines (Apple) - Among the four Design Systems assessed, Apple's Human Interface Guidelines (HIG) emerged as the top evaluation. The assessment included Apple's official Figma file and its HIG website. The Figma file provided visual examples of components, while the HIG website gave detailed instructions for building iOS interfaces. The HIG website was invaluable, containing nearly all information about Apple's DS. It was easy to navigate, clear, and well-organized. Furthermore, Apple's HIG sets a commendable standard for D&I. In a dedicated chapter, they offer guidelines for iOS designers and developers on how to communicate with and represent users, with a focus on preventing discrimination, exclusion, or miscommunication. Their approach stems from the acknowledgment that biases and stereotypes often shape design decisions unconsciously [9]. To counter this, they provide a list of characteristics that are commonly subject to prejudice, encouraging designers and developers to reflect on and address their biases. Apple's dedication to inclusivity, diversity, and accessibility was reflected in its overall score in the DS evaluation, with a significant 10-point difference.

Material Design - Google's Material Design initially appeared to be and well-organized DS. However, it contained many incomplete pieces of information. For instance, despite mentioning in its documentation the WCAG success criterion 2.5.5 - Target Size (Enhanced), a significant number of clickable components available in their Figma file disregard it. Moreover, many ambiguous pieces of information were found, such as the instructions on adding descriptive texts to images. While WCAG 2.2 states that images with purely decorative function should not contain descriptive texts, Material Design chose a decorative image as its primary example of how to add alt texts to images. The correction of this error could only be found after targeted research throughout the documentation. Another issue is the omission of accessibility guidelines combined with their discreet usage. This is evident in the cases of instructions on the use of complex gestures, italicized text, and text with low visual weight. All of these artifacts are discouraged by WCAG 2.2, and Material Design navigates this by not using them in most of its components, while not mentioning them as bad practices. Similarly, during the analysis period, some components (e.g., Tooltip [7], Snackbar [8]) were available in the Figma file and featured on the Material Design website, yet when attempting to access their accessibility guidelines, a message titled "Accessibility guidelines coming soon" was encountered. Despite its shortcomings, Material Design offers a fresh view on diversity. Its avatars, designed to depict Google users, prioritize humanity, including individuals with various disabilities, ethinicities, and sexualities. This consideration is also evident in their team's blog posts.

5 ALIB - COMPONENTS LIBRARY

After assessing the design systems, the HIG components not complying with WCAG 2.2 were corrected and their updated versions created as a package of components named ALib.

5.1 Design

The design process initiated with another analysis of Apple's components on Figma. The purpose was to contrast each component against every valid WCAG 2.2 guideline once more. This analysis brought forth the shortcomings of individual components, which were then annotated within a Figma file, indicating the specific issues that needed to be addressed in the updated versions of the components. From the 27 total groups of components examined, it was found that 14 either did not comply with at least one WCAG 2.2 success criterion, or unintentionally encouraged designers not to adhere to the guidelines. The latter scenario represented a significant portion of the total components and could not be overlooked. This phenomenon was particularly noticeable in the case of buttons. The company managed to fully comply with the Target Size (Enhanced) criterion predominantly due to their exhaustive written documentation. The documentation reinforced this criterion and prescribed specific padding sizes for each button component, creating a cohesive set of instructions and avoiding contradictions. However, this scenario can be improved to reduce the chances of misleading designers and developers who incorporate these components into their projects. Out of the 14 components in the aforementioned categories, 7 violated Success Criterion 1.4.3, 12 provided misdirection that could cause designers to violate Success Criterion 2.5.5, and 6 were input methods that should comply with Success Criterion 4.1.2. This analysis made it clear which areas of the library needed addressing: color contrast, target size, and content description. Furthermore, the components were filtered based on the technical feasibility of correcting them through a code library. Consequently, out of 14 components, 8 were deemed suitable for development. Out of the 8 components, 6 were identified as collections of simpler parts. This indicated that fixing their problems required new versions of common building blocks that complied with accessibility standards. Finally, the components chosen for the Swift Package ALib were: Button, Text, Text Field, Toggle, List Item, and Image.

Table 2: Inadequate components

1.4.3 - Contrast (Minimum)	7
2.5.5 - Target Size (Enhanced)	12
4.1.2 - Name, Role, Value	6

5.2 Development

The choice of technology prioritized compatibility with the DS at hand. Therefore, Apple's developer ecosystem was chosen. This includes the IDE XCode, Swift as the programming language, together with SwiftUI and the Swift Package Manager.

To guarantee compliance with success criteria 2.5.3 and 4.1.2, alternative labels became a mandatory requirement for most of the components. For success criterion 2.5.5, a minimum height and width were set on all tappable components. Finally, to address success criterion 1.4.3, functions were created to calculate contrast using the W3C formula. This calculation compares the foreground and background colors defined by the developer. If the contrast ratio is less than 4.5, the foreground color is replaced with either

Table 3: ALib components

	2.5.3 - Label in Name	1.4.3 - Contrast (Minimum)	4.1.2 - Name, Role, Value	2.5.5 - Target Size (Enhanced)
AText	X			
AButton	X	X	X	X
AToggle			X	X
ATextField	X		X	X
AList	X	X	X	X
AVStack		X		
AZStack		X		
AHStack		X		

black or white, depending on which provides a higher contrast ratio with the background.

The ALib package also includes developer documentation on the Figma platform, aimed at guiding developers on how to effectively implement the ALib components within the SwiftUI framework. When combined with the other, already WCAG compliant, components of SwiftUI, ALib allows developers to experience working with a fully accessible DS.

6 DEVELOPER INTERVIEWS

For the concluding phase of this study, ALib was made available on GitHub. To gain more insights and assess the impact of a fully accessible DS, multiple developers from different experience levels were invited for an interview session, out of which eight accepted and gave their free and informed consent to participate. Initially, a profile of their experience and fluence in SwiftUI was established. Afterwards, developers were encouraged to share their thoughts and perspectives on accessibility in software development. They were also given an opportunity to recount their experiences with the implementation of accessibility features in their past or ongoing projects. Next, the developers were introduced to ALib and its documentation, and asked to perform a hands-on test. This test was designed to let them freely familiarize themselves with ALib, understand its capabilities, and evaluate its ease-of-use and functionality. Their feedback on this experience was collected, providing key data for comprehending the impact of an accessible DS on the adherence of software developers to accessibility. Finally, the developers were asked to comment on whether a DS built like ALib, which is geared towards enhancing accessibility, would make any significant difference in the software they develop.

Table 4: Participants profile

Participant	Gender	Experience
P1	F	2 years
P2	F	2 years
P3	M	1 1/2 years
P4	F	8 months
P5	M	3 years
P6	M	4 years
P7	M	5 years
P8	F	2 years

6.1 Developer profiles

The SwiftUI framework was released in 2019, five years prior to this research. Therefore, the maximum possible experience with the framework is five years. The interview sample was selected based on diverse experience levels to broadly understand developers' interactions with the library. Among the eight participants (Table 4), 2 had less than two years of SwiftUI experience, 4 had between two and three years, and 2 had over three years of experience. Participants were also asked to classify their programming knowledge and practical experience into one of four categories: intern, junior, mid-level, or senior. Since these roles lack clear definitions in the job market, participants provided classifications based on their current or most recent job positions. Consequently, 5 out of 8 participants' answers reflected their current roles, while the remaining 3 based their answers on their previous traditional software engineer positions and the experience gained since.

6.2 Accessibility within previous projects

In the third question, participants were asked about the accessibility in their past projects. The aim was to determine whether their experiences parallel the literature's portrayal of software developers' interaction with accessibility. This information could then be used to comprehend their initial position before using ALib.

P1, who currently works as a junior software developer, explained that they simply follow the designer's instructions, and that they fully attribute the reasoning behind accessibility to the designers. Internally, they sometimes think about accessibility, but feel they don't have the power to share their opinions, as "Developers don't think they should give an opinion on an already finalized design". Curiously, when building independent projects, they think "a lot" about accessibility and utilize SwiftUI tools to implement it, such as supporting changes in font size by assistive technologies.

Moreover, P2, P3, P4, and P5 noted that accessibility was only a significant part of the development process when creating apps aimed at users with disabilities, such as a game for children with low vision. Additionally, P2 stated that in a developer-only team, they would likely be the most concerned about accessibility. However, if there was at least one designer, this role would typically fall to them. These participants also indicated that in projects aimed at the general public, accessibility was often viewed as "optional" or something to be added only if time permitted. Notably, two of the four participants used the concept of 'usability' in place of 'accessibility'. While there is some overlap between these concepts, and improved usability can enhance accessibility, they are not identical.

P6 stated that they prioritize accessibility, constantly implementing unit tests at work to simplify the application these requirements for other developers. However, P7 and P8 said they do not consider accessibility at all. These varying responses from eight individuals affirm the literature's suggestion that software developers may not consistently implement or emphasize accessibility enough.

6.3 Interest in accessibility

On the fourth question of the interview, participants were asked whether they had any intention to improve their knowledge on accessibility. This question was included to comprehend whether developers recognize that there is still room for improvement or not. Furthermore, if the answer was positive, they were asked to expand on some of their efforts. All of them declared some form of recognition agreement to this statement, but not in the same ways. P1 answered in terms of their experience, believing that

as they become more experienced, they'll gain more freedom to give opinions on accessibility. P4 also associated accessibility with work experience, but for the designers instead of their own. P2 and P6 demonstrated practical steps toward learning and applying accessibility-related techniques. One of them has been building projects to learn about accessibility tools, and the other proof of concepts to promote practices that could enhance their company's commitment to accessibility. Moreover, P3 said they would like to improve, but would only further study accessibility if a project required that. Otherwise, they will keep improving their coding skills, since good code practices can also partially influence accessibility. Finally, besides P8, all participants declared having studied accessibility at least once since they first began to use SwiftUI.

6.4 WCAG and W3C

When queried about their familiarity with WCAG or W3C, participants were mostly uninformed. P1 had come across and conducted research on it two years prior to the interview. P2, P3, P4, P7, and P8 had heard of these terms but lacked a clear understanding or never pursued further research. P5 had a very basic understanding of WCAG and W3C, while P6 hadn't heard of either. Interestingly, 7 out of 8 participants had received at least one lesson on accessibility during their university studies, but could barely remember any details. These responses suggest that developers may underestimate the importance of accessibility, even when they have access to relevant information. In the final segment of the interview, participants were asked to create a SwiftUI project in Xcode, integrate ALib into the project dependencies, and simulate interface construction using accessible components. To ensure all developers began with the same understanding of these components, everyone received a detailed documentation of the library and could consult it as they wished. During this session, participants were encouraged to voice their opinions, feedback, and general thoughts. Two participants suggested that the documentation should be directly available on XCode, alongside the code, to avoid the need to switch environments. P4 enjoyed their experience with ALib, particularly noting the autocomplete feature and the mandatory accessibility labels as positive aspects. P1 expressed their intention to use ALib on their projects due to the guidance it provides. P2 offered positive feedback, stating, "If accessibility features are right in front of you, you're going to use them anyway", and also praised the autocomplete feature of the library. P8 provided overall positive feedback on the library, but raised some interesting points. They mentioned that the library's behavior to correct contrast might pose a challenge to designers. However, they believed this would ultimately improve the original design. P8 also noted that while ALib seemed useful, a tight project deadline might deter developers from using it due to the extra parameters required to make components accessible. Finally, it was suggested that the tutorial on accessibility audits, currently located at the end of the library documentation, should be moved to the beginning. During this portion of the interview, it was noted that all of the participants demonstrated a lack of knowledge on proper accessibility label semantics. They were not aware of the recommendations made by Apple and WCAG 2.2 on how to interact with assistive technologies, suggesting that good practices on the subject would be a valuable addition to the documentation.

7 LIMITATIONS

As any research, this study has limitations. Concerning the evaluation of design systems, it was limited by selection bias towards top companies, potentially missing diversity in design systems across industries. The focus on WCAG 2.2 guidelines may have overlooked other accessibility aspects, and the methodology, lacking user testing with individuals with disabilities, might miss practical issues. Additionally, filtering guidelines by feasibility could exclude beneficial ones, suggesting the need for broader integration approaches.

In the context of the library study, including a small and diverse sample size of only eight developers limits the generalizability of the findings. Reliance on self-reported data may lead to biases, suggesting the need for triangulating findings with observational data or user testing. Additionally, the study is centered around the SwiftUI framework, making the findings particularly relevant to iOS development but potentially less applicable to other programming environments. Future research could address these limitations by including a larger, more diverse sample, standardized experience classifications, and a broader scope of accessibility dimensions.

8 CONCLUSION

Despite remarkable advancements in the field, it is evident that software developers and designers still grapple with the challenge of adequately integrating accessibility aspects into their projects. The reasons for this are manifold, ranging from a lack of awareness about accessibility standards to limitations of time and resources. However, the research points toward a promising strategy to address this issue. The implementation of a well-structured and fully accessible design system could prove instrumental in enhancing the accessibility of software projects. Such a system would ideally encompass a visual representation of components, providing a clear, tangible reference for designers and developers. Additionally, the design system would also include the general rules of usage for each component. This serves as a guiding tool for developers, ensuring that the components are used correctly and consistently, further enhancing the accessibility and usability of the final product. Lastly, it would feature the coded versions of these components. This would act as a ready-to-use toolkit for developers, eliminating the need to repeatedly code each component from scratch. This not only saves time but also ensures that the components are coded in a way that adheres to accessibility standards. In conclusion, this research suggests that the adoption of a well-structured and fully accessible design system, containing the visual representation of components, general rules of usage, and coded versions of said components, could be a valid and effective strategy to improve the current scenario on accessibility in software design and development. This approach could potentially pave the way for more inclusive digital environments that cater to the needs of all users, thereby bringing us a step closer to realizing the ideal of digital inclusion.

ARTIFACT AVAILABILITY

- Interview guide and full evaluation of the design systems: https://doi.org/10.6084/m9.figshare.26379760
- ALib code repository: https://anonymous.4open.science/r/ALib-442C/

ACKNOWLEDGMENTS

This work is partially supported by INES (www.ines.org.br), CNPq grant 465614/2014-0, FACEPE grants APQ-0399-1.03/17 and APQ/0388-1.03/14, CAPES grant 88887.136410/2017-00.

REFERENCES

[1] 2024. ATT. https://www.att.com January 20, 2024.
[2] Apple. 2024. Human Interface Guidelines. https://developer.apple.com/design/human-interface-guidelines/accessibility February 10, 2024.
[3] Stephanie Chamberlain, Helen Sharp, and Neil Maiden. 2006. Towards a framework for integrating agile development and user-centred design. In *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, 143–153.
[4] World Wide Web Consortium. 2024. Web Content Accessibility Guidelines (WCAG) 2.2 - Abstract. https://www.w3.org/TR/WCAG22/#abstract February 05, 2024.
[5] Victor Leal de Almeida and Kiev Gama. 2021. Mobile accessibility guidelines adoption under the perspective of developers and designers. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 127–128.
[6] Fluent Microsoft Design. 2024. Design Principles. https://fluent2.microsoft.design/design-principles January 15, 2024.
[7] Material Design. 2023. Tooltips: Accessibility. https://m3.material.io/components/tooltips/accessibility November 18, 2023.
[8] Material Design. 2024. Snackbar: Accessibility. https://m3.material.io/components/snackbar/accessibility February 7, 2024.
[9] Apple Developer. 2023. Human Interface Guidelines: Accessibility. https://developer.apple.com/design/human-interface-guidelines/accessibility December 3, 2023.
[10] International Organization for Standardization. 2011. ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. https://www.iso.org/standard/35733.html.
[11] International Organization for Standardization (ISO). 1999. ISO 13407:1999 - Human-centred design processes for interactive systems. https://www.iso.org/obp/ui/#iso:std:iso:13407:ed-1:v1:en December 10, 2023.
[12] Forbes. 2019. Top Digital Companies. https://www.forbes.com/top-digital-companies/list/#tab:rank February 02, 2024.
[13] Kiev Gama, Ana Paula Chaves, Danilo Monteiro Ribeiro, Kezia Devathasan, and Daniela Damian. 2024. How much do you know about your users? A study of developer awareness about diverse users. In *2024 IEEE 9th International workshop on empirical requirements engineering (EmpiRE)*. IEEE.
[14] John Grundy, Tanjila Kanij, Jennifer McIntosh, Hourieh Khalajzadeh, and Ingo Mueller. 2022. Diverse End User Requirements. *preprint arXiv:2210.02543* (2022).
[15] Vivian Hunt, Sundiatu Dixon-Fyle, Sara Prince, and Kevin Dolan. 2020. Diversity wins: How inclusion matters. https://www.mckinsey.com/featured-insights/diversity-and-inclusion/diversity-wins-how-inclusion-matters
[16] Vivian Hunt, Sara Prince, Sundiatu Dixon-Fyle, and Lareina Yee. 2018. Delivering through diversity. *Mckinsey & Company*. 26 (2018), 2018.
[17] Stefanie K Johnson. 2017. What 11 CEOs Have Learned About Championing Diversity. *Harvard Business Review* (2017).
[18] Alenka Kavcic. 2005. Software accessibility: Recommendations and guidelines. In *EUROCON 2005-The International Conference on" Computer as a Tool"*, Vol. 2. IEEE, 1024–1027.
[19] Alla Kholmatova. 2017. *Design Systems: A practical guide to creating design languages for digital products*. Smashing Magazine.
[20] Yassine Lamine and Jinghui Cheng. 2022. Understanding and supporting the design systems practice. *Empirical Software Engineering* 27, 6 (2022), 146.
[21] Germán Leiva, Nolwenn Maudet, Wendy Mackay, and Michel Beaudouin-Lafon. 2019. Enact: Reducing designer–developer breakdowns when prototyping custom interactions. *ACM Transactions on Computer-Human Interaction (TOCHI)* 26, 3 (2019), 1–48.
[22] Malin Lundström, Johan Åberg, and Johan Blomkvist. 2015. Perceptions of software developers' empathy with designers. In *Proceedings of the 2015 British HCI Conference*. 239–246.
[23] Danaë Metaxa, Michelle A Gan, Su Goh, Jeff Hancock, and James A Landay. 2021. An image of society: Gender and racial representation and impact in image search results for occupations. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–23.
[24] Microsoft. 2024. Microsoft Inclusive Design. https://inclusive.microsoft.design January 20, 2024.
[25] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. 2002. Evolution patterns of open-source software systems and communities. In *Proceedings of the international workshop on Principles of software evolution*. 76–85.
[26] Lisa Hope Pelled, Kathleen M Eisenhardt, and Katherine R Xin. 1999. Exploring the black box: An analysis of work group diversity, conflict and performance. *Administrative science quarterly* 44, 1 (1999), 1–28.
[27] Gema Rodríguez-Pérez, Reza Nadri, and Meiyappan Nagappan. 2021. Perceived diversity in software engineering: a systematic literature review. *Empirical Software Engineering* 26, 5 (2021), 1–38.
[28] Ljilja Ruzic, Seunghyun Tina Lee, Yilin Elaine Liu, and Jon A Sanford. 2016. Development of universal design mobile interface guidelines (UDMIG) for aging population. In *Universal Access in Human-Computer Interaction. Methods, Techniques, and Best Practices: 10th International Conference, UAHCI 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings, Part I 10*. Springer, 98–108.
[29] Samsung. 2023. About Us - Company Information. https://www.samsung.com/latin_en/about-us/company-info/ March 6, 2023.
[30] Selena Silva and Martin Kenney. 2019. Algorithms, platforms, and ethnic bias. *Commun. ACM* 62, 11 (2019), 37–39.
[31] Karina Kohl Silveira and Rafael Prikladnicki. 2019. A systematic mapping study of diversity in software engineering: a perspective from the agile methodologies. In *2019 IEEE/ACM 12th Intl Workshop on Cooperative and Human Aspects of Software Eng. (CHASE)*. IEEE, 7–10.
[32] Statista. 2024. APAC: Samsung smartphone market share by country 2020. https://www.statista.com/statistics/1254665/apac-samsung-smartphone-market-share-by-country/ February 5, 2024.
[33] Constantine Stephanidis, Demosthenes Akoumianakis, Michael Sfyrakis, and Alexandros Paramythis. 1998. Universal accessibility in HCI: Process-oriented design guidelines and tool requirements. In *Proceedings of the 4th ERCIM Workshop on User Interfaces for all, Stockholm, Sweden*. 19–21.
[34] Sarrah Vesselov and Taurie Davis. 2019. *Building Design Systems*. Springer.
[35] World Wide Web Consortium (W3C). 2024. Web Content Accessibility Guidelines (WCAG) 2.2 - Redundant Entry. https://www.w3.org/TR/WCAG22/#redundant-entry February 10, 2024.
[36] Xiaoyi Zhang, Lilian De Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, et al. 2021. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.