Analysis of the Native Mobile Accessibility Landscape

AARON VONTELL, Decentralized Information Group, MIT CSAIL
WILLIAM CARUSO, Decentralized Information Group, MIT CSAIL
BYUNGKYU PARK, Decentralized Information Group, MIT CSAIL
LALANA KAGAL, Decentralized Information Group, MIT CSAIL

Mobile devices have become the primary method of communication and information retrieval for people around the world. With over one billion people living with a disability, it is essential that these devices and the software that they host be accessible to all. While standards exist for providing accessibility on the Web, rarely any have been widely adopted for native mobile applications, and those that do fall short. Currently, developers sacrifice accessibility for increased functionality geared toward non-disabled users – a tradeoff that should not exist, but is enabled by the current landscape of resources available to these developers. In this paper we detail the current accessibility guidelines and resources which help developers and designers implement accessible Web applications, discuss why these tools do not expand easily to native mobile apps, and conclude that there is a need for a unified, open-source collaborative accessibility development community.

ACM Reference format:
Aaron Vontell, William Caruso, Byungkyu Park, and Lalana Kagal. 2018. Analysis of the Native Mobile Accessibility Landscape. 1, 1, Article 1 (December 2018), 10 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

People with disabilities are limited in their interactions with computers. In order to provide a consistent user experience for a variety of abilities, devices and software must be "accessible." Software accessibility involves implementing specific features that assist people in using technology, often resulting in improved usability for all users, including those with a disability. According to the 2012 census, 56.7 million Americans (18.7% of the U.S. population) have some type of disability, with an estimated 38.3 million (12.6%) diagnosed with a severe disability [1]. To put this into perspective, a study performed by Google using data from the World Bank [2] and CDC [3] found that there are more hearing impaired users in the United States than the population of Spain, as well as more users who are blind and low-vision impaired than the entire population of Canada [4]. The most recent WebAim Screen Reader Survey (performed in 2015) showed that 69.2% of respondents used a screen reader on a mobile device, a 576% increase since the first survey in 2009 [5]. Considering the pervasiveness of mobile technologies and smartphones in today's society, one can conclude that accessibility on these devices is of paramount importance.

XXXX-XXXX/2018/12-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Developers currently follow the Web Content Accessibility Guidelines (WCAG) 2.0 from the World Wide Web Consortium (W3C) in order to provide a large range of accessible features within their software [6]. This standard makes implementing accessibility in applications clear and direct - assigning accessibility levels and success criteria to each recommendation that it provides. The recommendations in WCAG 2.0 have been adopted by governments and organizations around the world to help create Web sites that are accessible. However, as these guidelines were created for the Web, they do not sufficiently cover all aspects of native mobile applications. Working groups from the W3C have mapped WCAG 2.0 to the native mobile ecosystem and have proposed new recommendations for touch screens, but have not created a cohesive guideline such as WCAG 2.0 for native mobile. Thus, there is no agreed upon standard for creating native mobile applications accessible as there is for the web. Major mobile operating system companies have therefore taken accessibility into their own hands, and have taken various paths to ensure that their systems are accessible.

As we will present in this paper, we suggest that accessibility guidelines and third party resources need to be organized in an effective matter such that mobile application developers can use them. First, it will be argued that software accessibility has an element of legal obligation, and as such accessible development and design would be in the developer's best interest. Furthermore, in an attempt to address these considerations, work has been done by the W3C and related groups to adapt standards to mobile accessibility, but it will be shown that these standards do not map to the mobile landscape in an effective manner. As a result, a plethora of guidelines have been created, causing the native mobile accessibility landscape to be fragmented and difficult to navigate. Along with these guidelines, there exists a collection of tools and resources available for popular mobile operating systems which help developers implement and utilize accessibility within applications. However, these too are either incomplete or difficult to use. These realizations lead to the conclusion that the current resources for native mobile accessibility have too many barriers for developers, leaving disabled users behind as application creators move on to create future features for their platform. Both resources, tools, guidelines, and initiatives can and need to be created in order to remedy these issues.

2 BACKGROUND

When a company or individual builds an application to be used by consumers on mobile devices, there are typically two routes that they may take; they can build a website that will work on mobile interfaces, or they may build a native mobile application. A **native mobile application** is an application that has been developed for a specific mobile platform, most commonly with the tools provided

Table 1. Disabilities that Affect Mobile App Usage in USA [1]

Disability Type	Affected Users	Limitations & Needs
difficulty lifting or grasping	19.9 million (8.2%)	Holding a phone or performing gestures
cognitive, mental, or emotional impairment	15.2 million (6.3%)	Need simplified language, consistent color patterns and extra affordances
vision impairment	8.1 million (3.3%)	Rely on a screen magnifier or reader, voice assistant, color contrast
hearing impairment	7.6 million (3.1%)	Rely on captions for audio and video media, need alternate forms of audio feedback

by the creators of that platform. For example, the act of creating a native Android application is achieved through the use of Android Studio [7].

The creation of native mobile applications is important in that the development tools provided by the desired mobile platform are usually more suited to usage on that device than the tools available during, say, the development of a web-based application built for the phone using a service such as PhoneGap [8]. As more complicated features are added to the application, including accessibility features, it becomes increasingly difficult to implement software for the mobile platform using only Web technologies and universal mobile development platforms. Therefore, native mobile development, when compared to mobile Web or hybrid development, is prevalent in development choices [9].

Companies who develop mobile operating systems, such as Apple, Google, and Microsoft, have taken control of native mobile accessibility by building custom accessiblity features into their own mobile operating systems [10]. Although different on each platform, these built-in accessibility features have ultimately propelled the native mobile application ecosystem as a whole towards a more inclusive environment for those with disabilities. Each of these operating systems provide an API and tools for implementing accessible features. Steve Fischer, a developer on Apple's Accessibility team, stood in front of hundreds of developers at WWDC and requested "We need all of your apps to be accessible so we can create a great ecosystem for accessibility and empower these users."

These companies have been taking even greater initiatives lately in an attempt to bring accessibility to their platforms. For example, at Google I/O 2017, new accessibility features were added to the Android platform, discussions on these new features were recorded for later viewing online, and office hours for developers looking to make accessible applications were held [11]. These developments, with similar trends seen by Apple and Microsoft, mark a time in history when more developers are beginning to look into accessible application development.

3 LEGISLATION

Creating an accessible world requires legislation in place which promotes the creation of accessible devices and software. In the past fifteen years, both the United States and various international governments have developed legislation and taken action in order to ensure equal rights for people with disabilities, including equal access to electronic and information technology. In this section we will focus on the decisions and laws put into effect by the United States in the area of accessibility within software.

3.1 Department of Justice Intervention

In 2014, the Department of Justice (DOJ) entered into a landmark consent decree resolving its first lawsuit brought under the Americans with Disabilities Act, which is centered on the accessibility of corporate websites and mobile applications. Under the decree in National Federation of the Blind and United States v. HRB Digital LLC, H&R Block agreed to make its website, tax filing utility, and mobile applications conform to WCAG 2.0 at the Level AA Success Criteria.[1]

Later that year, the DOJ had entered another consent decree demanding that WCAG be applied to native mobile applications. Under a settlement between The United States of America and Ahold U.S.A. Inc. with Peapod LLC, an online grocery delivery service, Peapod was to make its web and mobile application software compliant with WCAG 2.0, hire an accessibility consusltant, and create automated tests for accessibility on its software platforms. [2]

In 2015, the DOJ entered yet another consent decree linking native mobile applications to WCAG 2.0. However, this instance included additional insights in the final ruling. EdX, a major platform for delivering online courses, was to make their mobile applications "conform with, at minimum, WCAG 2.0 AA ... [and] ... may rely upon UAAG 1.0, ATAG 2.0, and the Guidance on Applying WCAG 2.0 to Non-Web Information and Communications Technologies (WCAG2ICT), published by the W3C, as well as guidance published by the W3C's Mobile Accessibility Task Force." This was the first time that WCAG 2.0 was recognized as not being a perfect solution for native mobile apps, as it was noted that other guidelines and standards may need to be consulted in order to create an accessible mobile service. [3]

Consent decrees involving the Justice Department are highly influential, as they may be used to support future legal requests for mobile services to be made accessible. Mobile developers therefore face increasing pressure to develop mobile applications that conform to WCAG 2.0 and related standards, which are currently structured around the Web, not native mobile software.

[1] United States of America v. HRB Digital LLC and HRB Tax Group, inc. https://www.ada.gov/hrb-cd.htm
[2] The United States of America and Ahold U.S.A., inc. and Peapod, llc. https://www.justice.gov/file/163956/download.
[3] The United States of America and EdX inc. https://www.ada.gov/edx_sa.htm.

3.2 U.S.A Legislation

Mobile applications are generally covered by the same legislation that applies to non-mobile software and web applications. Applicable US laws include the Americans with Disabilities Act (ADA), Section 508 of the Rehabilitation Act, and the Twenty-First Century Communications and Video Accessibility Act (CVAA).

The **Americans with Disabilities Act**, often referred to as the ADA, gives civil rights protection to individuals with disabilities and guarantees equal opportunity in *public accommodations*, a limited number of facilities and services which are open to the public.**[CITE]** In 2012, the United States District Court of Massachusetts ruled that a website can be a public accommodation. This ruling may therefore be extended to mobile websites and native mobile applications.

Section 508 of the Rehabilitation Act (commonly referred to as Section 508) includes technical and functional requirements for Web information, software applications, and operating systems. The criteria discussing the availability of these services to people with disabilities can be applied to mobile content. While these standards do not include specific mobile application requirements, they do cover the general accessibility requirements that are relevant to mobile apps such as technical, function, and support requirements.

The **CVAA Rulemaking and Order** from October 2011 addresses guidelines on what performance objectives must be met for advanced communication devices and applications; this includes access without vision, with low vision, without hearing, without color perception, with limited manual dexterity, and without speech. The objectives also address availability of information without requiring vision, hearing, etc. These functional objectives are inline with the core principles of WCAG 2.0 guidelines and Section 508. Thus, WCAG 2.0 guidelines and Section 508 standards are a logical starting point for mobile app accessibility under CVAA.

These pieces of legislation create a foundation of protection for those with disabilities, by outlining the legal requirements for software in relation to accessibility. Promoting accessibility in native applications is not only a great idea for supporting disabled consumers, but it ensures that native applications in the mobile space conform to the rules and laws laid down by the United States government.

4 EXISTING ACCESSIBILITY STANDARDS

A great place to start when developing an accessible mobile application is finding a standard to evaluate the accessibility of your application. An accessibility standard or guideline assists the developer by explaining how to make certain components accessible to a user of a disabled group. Many standards and general guides exist today for making accessible software. In this section we will discuss WCAG 2.0, the main accessibility standard used on the web, a few other available guidelines, why these standards are difficult to apply to the mobile landscape, and current and future progress in this standards space.

4.1 WCAG 2.0

The W3C's Web Accessibility Initiative (WAI) founded in 1997 is an ongoing effort to improve accessibility on the World Wide Web for people with disabilities. They have been strong legal and social advocates for digital accessibility. Various working groups composed of top industry professionals and researchers create guidelines, technical reports, educational materials and other documents related to different aspects of web accessibility, such as the Web Content Accessibility Guidelines (WCAG 2.0). [CITE]

WCAG 2.0 is a document which aims to provide guidelines and advice in implementing accessible web content, whether that be images, text, video, and sound. It has 12 guidelines broken up into 4 principles (perceivable, operable, understandable, and robust), each with varying level of success criteria (A, AA, and AAA). Generally speaking, the more accessible you make each component of the application, the higher the level of success criteria. Due to the reference of WCAG 2.0 in many accessibility settings, it has become a widely-used standard across software.

4.2 Other Standards and Guidelines

Besides WCAG 2.0, there are other standards and guidelines that provide tips and rules for creating accessible websites, software, and applications. For instance, he User Agent Accessibility Guidelines (UAAG) documents explain how to make user agents accessible. User agents include browsers, browser extensions, media players, readers and other applications - including native mobile apps - which render content. Another standard, ISO 9241 part 171, speaks about software accessibility, by using WCAG 2.0 as a base, but going into a lot more detail for non-Web content.

Outside of the US government and working groups, there are a few standards from other countries and private firms. The European accessibility standards, EN 301-549, provides accessibility guidelines that are separate for Web, Non-Web Documents and Non-Web Software (including native mobile apps). Although it borrows heavily from WCAG 2.0, EN 301-549 effectively requires mobile operating systems to have an accessibility API (requirements 11.3.2.1 and 11.3.2.2). It also encourages app developers to support that API (requirement 11.3.2.3)—otherwise, the developer must meet a long list of other specific accessibility features. Furthermore, private companies such as IBM and BBC [12] have their own accessibility standards for non-web software. [http://www-03.ibm.com/able/guidelines/ci162/accessibility_checklist.html#non-web].

4.3 Issues with Current Standards and Guidelines

Considering the success of applying WCAG 2.0 to accessibility on the web, many have tried to adapt its standards to non-web mobile applications. However, WCAG 2.0 does not specifically address native mobile applications and includes content that specifically refers to HTML, some of which do not apply to native mobile user interface components in any way. The W3C argues that "while mobile is viewed by some as separate from 'desktop/laptop', and thus perhaps requiring new and different accessibility guidance, in reality there is no absolute divide between the categories." This claim is backed with the fact that mobile and desktop are beginning to share of similar modalities of interaction (such as touchscreen, external controls, etc.) in which accessibility must be designed for the user's entire experience across devices and platforms.

However, the modalities of desktops and laptops still differs to those of mobile phones. Phones use the touchscreen as their main form of input, have motion data at their disposal, include a vibration motor for feedback, have much less screen real estate than your typical desktop or laptop, and are most often missing a persistent keyboard and mouse. The combination of these facts makes some of the WCAG 2.0 guidelines seem "outdated" or misaligned.

For example, take a look at guidelines 2.4.5 of WCAG 2.0:

> *More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process.*

There area a few problems with this guideline when applying to the native mobile space. For one, what is the native app equivalent of a web page? In Android, does this mean an Activity, Fragment, Application, Dialog, or all of the above? Additionally, within web pages this guideline is usually handled by adding a navigation bar to a website. However, mobile screens lack the real-estate for large navigation menus with many drop downs and breadcrumbs to follow; in-section navigation is possible, but across-app navigation is usually difficult to always include on the screen (for example, see figure 1).

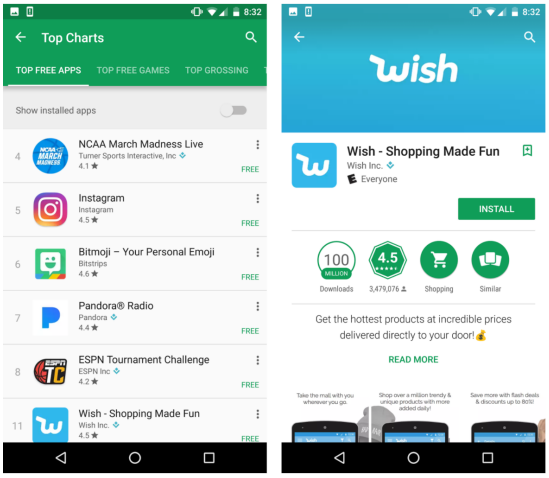

Fig. 1. The Google Play Store offers top level navigation for viewing different types of applications, but only a back button within specific pages. Once you enter an app page, you can only access the rest of the site by going back.

4.4 Current and Future Progress

Due to the issues found in adapting existing standards into the native mobile accessibility space, working groups have been formed and charged with creating new standards that modify or adapt existing guidelines into this new landscape. For instance, the W3C's Accessibility Guidelines Working Group (AGWG), which was created in December of 2016, has been drafting Silver, a generalized and extended version of WCAG 2.0. According to the AGWG, "the goal of the Silver project is to provide a major revision to the WCAG standard, and as part of the update to incorporate requirements for user agents and authoring tools as needed."

The Mobile Accessibility Task Force (Mobile A11y TF) is a part of the Accessibility Guidelines Working Group from WAI, and they are the premiere group on up-to date mobile accessibility. They recognize that WCAG 2.0 is not a perfect mapping native components and concluded that WCAG 2.0 should be extended to include a broader variety of digital interactions.

The primary effort of this Task Force was to develop documentation describing how to apply WCAG 2.0 and its principles, guidelines, and success criteria to non-Web Information and Communications Technologies (ICT). The basic conclusion of the Task Force was that only a subset of WCAG 2.0 could be applied directly to software in general. Instead, much of WCAG had to be remolded to meet the needs of the inherent differences between software and the Web. As a start, the task force has set out to extend WCAG 2.0 with mobile principles. The current working version of the extension proposes new guidelines, success criteria, and mobile techniques. Four new guidelines are being proposed including:

- Guideline 2.5: Touch and Pointer: Make it easier for users to operate touch and pointer functionality
- Guideline 2.6: Make it easier to use the physical features of the phone
- Guideline 2.7: Make it practical for speech input users to operate all functionality
- Guideline 3.4 Make content usable in device orientations

These additional guidelines, although not yet finished and accepted, will serve as a strong base for assuring WCAG 2.0 evolves to cover newer technologies, including native mobile apps.

There are also private parties which have looked at this space of native mobile accessibility. For instance, the App Quality Alliance (AQuA) is a non-profit group headed by members from large and small technology companies and knowledge contributors since AQuA's inception, working with the industry to improve the quality of mobile apps [13]. AQuA has developed a set of accessibility testing criteria for native mobile applications for each mobile operating system. Their guidelines are intended to be used to check the application's accessibility for users with impairments in one or more categories of vision, color perception, hearing, speech, dexterity, and cognition. The testing criteria for iOS, Android and Windows are publicly released and free to download. Another group, Funka Nu, has created the Funka Nu Mobile Accessibility and Navigation Guidelines, which provide a set of criteria for accessible design and navigation within mobile apps [14].

5 CURRENT SUPPORT FOR NATIVE MOBILE ACCESSIBILITY

Native mobile applications are programmed using various languages and are extremely dependent upon the mobile frameworks provided by their host operating systems. Two major mobile operating systems, iOS and Android, both provide accessibility APIs which allow developers to access built-in accessibility features.[4] There are three types of accessibility support that a native mobile developer can provide in their app:

[4]While Windows Phone was originally a contender in the smartphone ecosystem, Microsoft has dropped official support of the mobile OS, and as such is not discussed within this analysis.[15]

(1) **Basic Accessibility Support** Screen Magnifiers, Larger Text, Grayscale and Media Captioning.
(2) **Advanced Accessibility Support** These are special features that mostly deal with on-screen navigation and content presentation, like Screen Readers and Color Correction.
(3) **Assistive Hardware** This includes devices such as refreshable brail displays, switches, TTY, and hearing devices.

Each operating system contains basic system-wide accessibility features that can be configured by the user, with a few advanced features also found in both iOS and Android. Instead of creating accessibility features from scratch, developers can write a few lines of code to make their customized components work with accessibility services and features within the mobile operating system. As long as developers properly utilize these APIs, their app will be 'accessible' to some degree, which can lead to an increase in active users. In addition, developers can create custom accessibility features for their applications. In fact, many apps are created specifically to assist and empower disabled users and are often features in specific parts of app stores[]. CITE?

Apple and Google have dedicated accessibility teams which are continuously developing features for their respective mobile platforms. The features that are included in the latest versions of these operating systems are discussed here, including a discussion regarding the ownership of these features. These features will paint a picture of the current progress towards handling accessibility guidelines and standards by including them directly in the mobile operating system.

These features covered by the Android and iOS operating systems are only considered up to API level 25 on Android (also known as Nougat 7.1) and up to iOS version 11 beta 5.

5.1 iOS

The original iPhone released in 2007 contained no accessibility features; blind people thought it would be impossible to use a completely touch screen device. Apple first introduced accessibility features in June 2009 with iOS 3 (then called iPhone OS 3) and the introduction of VoiceOver, the first gesture-based screen reader built on a borrowed platform from the Mac [16]. Since then, Apple has been working hard to provide accessible functionality within their software, and even won the "American Federation for the Blind's Helen Keller Achievement Award" in 2015 [17].

5.1.1 Accessibility Settings. iOS includes a rich selection of accessibility options in the **Settings→General→Accessibility** menu, as shown in Fig 2. There is no doubt that iOS offers the most broad and comprehensive selection of accessibility features and services, some of which have been adopted by other platforms. There are too many to talk about in detail, however Figure **??** includes complete details about each feature, including it's inital release date and a short desscription provided by Apple's iPhone User Guide [18]. The most famous feature, VoiceOver, has proven to be the favorite screen reader amongst the blind community for mobile devices.

In addition to the basic accessibility features, Apple has gone above and beyond to create innovative features that empower users. For example, instead of writing off the camera as a part of the device that a blind or low-vision person could not use, the built in Camera

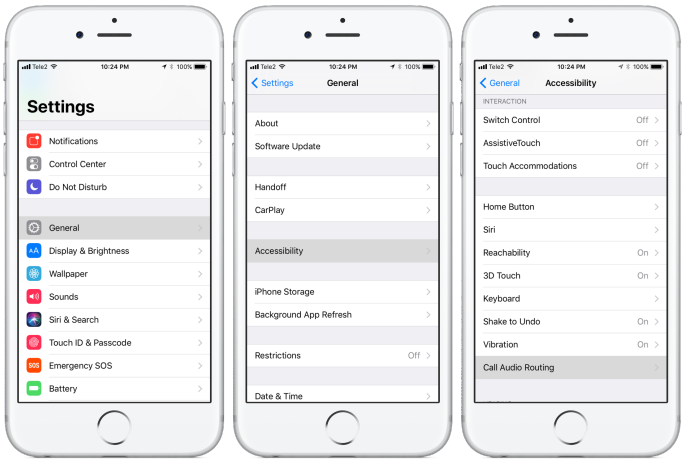

Fig. 2. Accessibility Settings menu in iOS 11.

app describes to the user the size, number and position of faces in the frame [19]. The camera will even let user's know when the image is in focus so a person who cannot see can take beautiful photographs. This is an excellent illustration of how Apple tries to deliver a consistent experience to all of its users.

Physical gestures that are inherent in the iPhone such as "Shake to Undo" or "Pinch to Zoom" were impossible for some users with motor skill impairments to perform until the release of Assitive Touch in iOS 5 [20]. Assistive Touch is a floating on-screen button that when touched opens up a pallete of phone control options including physical controls, swipe and physical gestures and quick shortcuts to settings and apps - including Siri. Best of all, Assistive Touch can be customized to provide users efficient means of navigating to the places they want on their phone and set their own gesture replacements.

Another great innovative feature is Touch Accomodations, which released in iOS 9. Touch Accomodations were created to help users who have tremors be able to use the touch screen effectively without causing too many errors and registering unwanted taps. When a user selects a component, a small cirlce appears under their finger with a timer. If the user holds down on the screen for the entire time period, the touch is registered. Otherwise, it is ignored.

5.1.2 Built-in Screen Reader. Apple's built-in screen reader - VoiceOver - has allows blind and low vision users to navigate and use iOS quickly and efficiently by speaking back the contents on the screen. VoiceOver can be activated through the Settings app and the playback voice, rate, and detail level can be customized. VoiceOver will understand all UIKit elements (the standard UI packing for iOS) but developers will need to include some essential information for VoiceOver to describe the purpose of the components.

Rotor

The Screen Curtain will turn off the display but keep the touch-sensing digitizer active, and will allow VoiceOver users to use their devices while the display is turned off. This is practicle for user's who cannot see the display and conserves battery life. Screen Curatin can

be enabled with a 3-finger double tap on the screen while VoiceOver is being used.

In a 2014 comparison of screen readers from iOS, Android and Windows - VoiceOver came out on top as it “has the most complete hints for interaction. All elements can be reached either by touch-explore or swiping." Whereas other software was harder to user and inconsistent [21].

5.1.3 Platform Provided Accessibility APIs. Prior to iOS 7, accessibility was simple: developers needed to add labels and hints to UIComponents as VoiceOver was the only supported assistive technology. However, since iOS 8, user interfaces have become much more gestural and dynamic. UIAccessibility is a lightweight API that helps an application provide all the information VoiceOver needs to describe the user interface aloud. Standard UIKit controls and views implement the UIAccessibility methods and are therefore accessible to assistive apps by default. All developers need to do is supply UIAccessibility with app-specific view details. When implementing custom views, developers must give their custom views properties that describe them.

Developers can provide information through the UIAccessibilityElement and UIAccessibilityContainer protocols. The UIAccessibilityElement contains traits for element labels, hints and containers - as well as custom accessibility actions. A container contains an array of AccessibilityElements, which can be bundled together as one whole AccessibilityElement. UIAccessibilityActions are used to provide users information about how to interact with a view, usally with a gesture such as selecting values in a range or scrolling through information on the screen. VoiceOver will pick up on the actions associated with a view automatically and speak them outloud when selected. In addition, assistive technology users can adjust the element using gestures specific to the assistive technology.

Switch Control allows user’s to navigate iOS with a single switch. As of iOS 8, developers can set custom paths for switch control users and can group elements together for more efficient usage. Also released in iOS 8 was the ability to identify device state (which accessibility features are enabled). This allows developers to customize their apps appearence and layout based on the user’s preferences. Developers can determine is Assistive Touch, VoiceOver and Switch Control are running in addition to whether shake to undo, closed captioning, darker system colors, grayscale, Guided Access, invert colors, mono audio, reduce motion, reduce transparency, speak screen and speak selection is enabled.

5.1.4 Accessibility Inspection and Assessment. Apple developes have a few tools to test the accessibility of their applications. Built into Xcode is the Accessibility Inspector, which makes testing accessibility interactive. The inspector is a desktop window that shows all of the properties and values, methods (actions that can occur from elements on the screen), and position of the object that’s currently being selected on the screen. Testers can perform actions through the inspector instead of having to perform a gesture with the mouse or an assitive technology. The Accessibility Inspector also has “Audit” feature that allows you to run an audit on your app and see what accessibility elements are missing [22].

5.1.5 Guidelines and Documentation. Apple has published various guides for developers on how to make thier iOS applications conform to UIAccessibility. In addition, Apple has been committed to enuring that users are both aware of Accessibility features and know how to use them.

Accessibility Programming Guide for iOS is the Apple guide for introducing developers to accessibility APIs [23]. The document is organized into two main parts: Understanding Accessibility on iOS and Making Your iOS App Accessible. The first part aims at introducing developers to VoiceOver, the benefits of making your app accessible, the UIAccessibility interface and Accessibility Traits. The other section focusses on technical implementation of UIAccessibility protocols and describes how to make custom UIViews accessible. In addition, this section describes how craft a helpful label. Unfortunately this guide has not been updated since 2012-02-16 and is missing a large amount of new accessibility features that were added in the past half decade.

Apple’s Human Interface Guidelines include an entire section dedicated to Accessibility [24]. The single page mentions concrete actionsf or develoeprs such as “Include closed captions and audio descriptions" but also contains broad statements such as “Respond to accessibility preferences." The UIAccessibility protocol is documented in Apple’s standard Cocoa Documentation [25]. There is also a guide to help developers verify their app accessibility on iOS: [26]. For iOS users, the iPhone User Guide goes detailed into accessibility settings for users [18]. The user guide is hosted as an accessible website on the Web. Lastly, iPhone Accessibility is displayed proudly on Apple’s website and contains videos of iPhone user’s describing how accessibility settings change their lives [27].

5.2 Android

Android, by Google, was widely adopted in 2009 (v1.5 Cupcake) and offered no accessibility features [28]. Since then, Android has come to host a solid base of accessibility features. Android Central, an Android developer community, explains the problem with accessibility on Android as such: “While things are getting better with each and every iteration of Android, there is still a long ways to go. That’s where third-party apps and the way they can integrate into the core system on Android comes in. There is plenty of room in Google Play for apps like keyboards or maps that can help those with special accessibility needs, and this is an area where we hope things can keep growing."[29]

5.2.1 Accessibility Settings. Android contains a dedicated section within the Settings called Accessibility, which allows users to control accessibility options for the operating system as a whole. As of Android v7.1.2 users are allowed to find, view, or hear information on the screen by enabling TalkBack, BrailleBack, contrast and color options, captions, magnification, and font or display size. Additionally, the built-in accessibility settings allows users to activate Voice Access or Switch Access, which allows the user to provide input using their voice or an external device [30]. These features, which are described more thoroughly in Table 2, provide a more pleasing and comforting experience to those with vision, hearing, and movement disabilities.

5.2.2 Built-in Screen Reader. Android's pre-installed built-in screen reader is TalkBack, which interfaces with layouts to provide meaningful information to the user regarding information on the screen. In order to utilize this feature, *content descriptions*, or tags on views and objects within Android user interface elements, can be added to user interface components during development or during runtime [31]. These content descriptions can then be read off by TalkBack using audio through the speakers or a headset. By providing these descriptions, TalkBack can convey information to the user about currently focused views, information within a piece of text, and navigation information.

In addition to simply speaking these tags out loud, content descriptions on images are also used by Android's *captions* feature, which places captions on images for extended explanation of the image [32]. Furthermote, developers can choose to expand on TalkBack and have the device speak generic pieces of text that may be relevant to the information on the screen through the use of the TextToSpeech API [33]. With so many features for screen reading, audio accessibility is one of the most important accessibility features available to Android developers.

5.2.3 Platform Provided Accessibility APIs. On top of TalkBack, BrailleBack, and similar features, another technique used by developers to provide accessibility within their applications is the creation of an `AccessibilityService` and `AccessibilityEvent`. These objects allow developers to create services which can listen for specific events on the device, and fire off actions in reaction to these events. For example, an event on the user interface might trigger an accessibility service to change the font size or color within an application.

However, while this API is useful in that it allows developers to create larger and more sophisticated accessibility services, applications that implement `AccessibilityService` contain user interface lag, which has been consistent across multiple releases. In addition to the standard UI, the device needs to relay a large amount of data to the `AccessibilityService` in order to properly react to user input [34].

In addition to these features, the recent release of Android O has debuted many exciting accessibility features for developers [**?**]. This version of Android will provide developers with APIs for accessible audio volume control, software-based accessibility buttons, fingerprint gestures, and enhanced screen-based gesture control.

5.2.4 Accessibility Inspection and Assessment. Developer's can test their Android apps for accessibility related issues with three different tools [35]. Lint, an Android Studio plug in, checks layout code for content descriptions and labels. Google Accessibility Scanner takes a screenshot of an application on a device and points out aesthetic related issues like touch target sizes and color contrast [36]. Accessibility Scanner comes preinstalled on Android devices running v7.1 or newer. With these tools, developers can debug their layouts to make sure that the order of the view hierarchy is correct, which ensures that services such as TalkBack can function properly. The Android platform also supports several testing frameworks, including Espresso and Robolectric, which allow developers to create and run automated tests that evaluate the accessibility of your app. However, developers will need to practice with these frameworks in order to write tests.

5.2.5 Guidelines and Documentation. The collection of guides, tutorials, and documentation on implementing accessibility into Android applications is quit extensive. Android has a site dedicated to these lessons [37], which spans from lessons in usability regarding Material Design [**?**] all the way to specific technical implementations of custom accessible views [38].

One of the most useful sections of this website is the Accessibility Developer Checklist, which provides a list of tasks and ideas to consider as a developer is designing and building their application [39]. These tasks include items such as including built-in accessible components, considering the delivery of information on the page, and avoiding complex controls. Links to documentation regarding these specific features are included in each section and topic, allowing the developer to easily dive in and begin building.

The resources provided on this main Android accessibility website are by no means complete, as they mostly discuss the implementation of hearing and vision features, with only a few lessons on disabilities such motor skills and combined disabilities (i.e. both vision impaired and hearing impaired). However, they serve as a great starting place for developers who wish to learn more and begin at least thinking about incorporating accessibility into their applications.

5.3 Patents and Native Mobile Accessibility

Designers and developers are awarded patents and copyrights for novel software and user interfaces, including interfaces created for increased accessibility within a companies mobile platform. For example, in March of 2014, Apple was awarded a patent for "Devices, methods, and graphical user interfaces for accessibility using a touch-sensitive surface." (CITATION???) This patent describes a three-finger zoom technique that many iOS users have learned to use and have possibly depended on.

Patenting these features are a great way to protect intellectual property and avoid spending company resources on a product that could potentially be used in an opponent's platform. However, patenting accessibility techiques is an interesting topic in that it often results in divided opinions [41].

One school of thought believes that patenting these features is morally wrong. For instance, patenting a feature such as the three finger zoom may be seen as an attempt to monetize the disabilities of some users. Another thought is that these patents may steer other mobile operating systems away from implementing that same feature, meaning that disabled users may lose the freedom to choose a mobile operating system based solely off their ability to use that operating system in an accessible way. If a user needs a specific accessible feature, but only iOS has this feature integrated, then this user will have no choice but to use iOS.

However, others may argue that the ability to patent these features is a positive for the disabled community. By patenting a specific accessible feature, other mobile operating system developers will be driven to ideate new and alternative solutions, which may ultimately lead to better products for disabled users overall [42]. Additionally,

Table 2. Existing Android Accessibility Features [40]

Title	Description
TalkBack (limited vision/blind)	Screen reader that describes your actions and the structure of the view and tells you about alerts and notifications.
Switch Access (limited mobility)	Switch Access is an alternative to using the touch screen. You can use a switch or keyboard to control your device.
Voice Access (motor impairments)	Voice Access app lets you control your device with spoken commands. Use your voice to open apps, navigate, and edit text hands-free. Currently in a limited beta release in English only.
BrailleBack (limited vision/blind)	Connect a refreshable braille display to your device via Bluetooth. Works with TalkBack for a combined speech and braille experience, allowing you to edit text and interact with your device.
Display size and font size (limited vision)	Change the size and appearance of items on your screen.
Magnification gestures (limited vision)	Temporarily zoom or magnify your screen.
Contrast and color options (color blind)	Adjust contrast or colors, use high-contrast text, color inversion or color correction.
Captions (hard of hearing/deaf)	Turn on captions for your device and specify options (language, text, and style) for closed captioning.

patents are drivers for innovation and funding within companies, and as such the application of patents to these features will drive these companies to build additional features.

No matter which school of thought a consumer agrees with, the creation of these patents clearly indicates that companies such as Google, Apple, and Microsoft continue to consider accessibility a goal worthwhile pursuing.

6 DISCUSSION

6.1 Tools for Creating Accessible Apps

Through our analysis of the current landscape for creating accessible native mobile applications, we have found that the current tools available to developers are insufficient and can be improved. While the number of accessible features included on mobile operating system such as iOS, Android, and Windows are increasing, developers are still at a lost in providing accessible features within applications themselves.

The most useful tools available to developers are not software, but rather the tutorials and companies found online that can be used to guide the creation of an accessible application. However, we found that while these do provide a great start to creating accessible applications, they can also be expensive, exclusive, and incomplete. Additionally, the vast number of differing sources for these guides makes choosing the right tutorial difficult, resulting in a fragmented space of accessibility guidelines and suggestions.

6.2 Recommendations for Developers Creating Accessible Apps

Accessibility of mobile applications ensures that apps are operable, percieveable, and understandable by all users. In order to achieve this, developers today must find a balance between the functionality and aesthetic design of their mobile applications. In addition, applications must use the proper language to communicate with users. We suggest the following recommendations:

6.2.1 ***Leverage Existing Accessibility APIs.*** Accessibility functionality (operability) is specific to each mobile platform. Companies have taken it upon themself to build accessibility features into their mobile operating systems. They make these universal accessibility features available to all applications through Accessibility API's. Each company has resources detailing how to interface with their services, and we recommend that developers take the time to understand these services and how they can be incorporated into their applications.

6.2.2 ***Borrow Practically From WCAG 2.0.*** While WCAG 2.0 was not created with the intention of applying to native mobile software, it does contain key design and editorial information that can be used to create percievable and understandable applications. In addition, WCAG 2.0 provides insight into possible accessible functionalities that are not included in all mobile operating systems, but rather may be built out or adapted by the developer for the application's specific functionality.

6.3 Next Steps and Open Questions

It is quite evident that the barrier to entry for developing and designing native mobile applications is quite high. In order to lower this barrier, we propose that resources and tools must be available for developers to create accessible applications in a systematic way. These resources should allow developers to create accessible applications with less lines of code and less development time, all the while minimally changing the user experience for users who do not require the accessible functionality. Furthermore, these tools should help the developer to not only develop an accessible application from the start, but also to easily modify existing applications to become more accessible than before.

In addition to these resources, there are also many questions that we found difficult to answer during this analysis. For instance, how does an app record information about the disabilities that a user has for custom accessibility tailoring, without infringing on regulations such as HIPAA regarding protected health information? As discussed in section 6.2, is it justified to patent accessibility features within operating systems, or does it impact the user's freedom to choose an operating system? Can advancements in natural language processing and machine learning be used to assist disabled users, such as parsing key concepts from a paragraph of text to quickly display information to those with cognitive disabilities? The answers to these question may open a road to an improved environment for accessible software in general.

7 CONCLUSION

Throughout this paper, the current landscape of native mobile development with accessibility in mind has been discussed and presented. Accessibility for native mobile applications is an important topic, as it is both the morally and lawfully right thing to do. In the process of developing mobile applications that are accessible, developers currently face obstacles such as the inability to adapt pre-existing guidelines into their designs, using incomplete and insufficient libraries for providing features, and having a difficult time choosing which tutorials and guides to follow during development. Accessibility for native mobile applications has been addressed by many companies and groups, but has resulted in a plethora of guidelines that is overwhelming and still incomplete. Companies with mobile platforms have incoporated accessibility features into their operating systems, but more must be done in order to convince developers to incorporate accessibility into their applications.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Decentralized Information Group at CSAIL, as well as Judy Brewer, director of the W3C's Web Accessibility Initiative, for their insightful comments and suggestions. We would also like to thank the W3C, the Seth Teller Memorial Fund, BridgingApps, and the Accessibility and Usability Group at MIT for their support.

REFERENCES

[1] U. C. Bureau, "Anniversary of americans with disabilities act: July 26," July 2008. [Online]. Available: https://www.census.gov/newsroom/releases/archives/facts_for_features_special_editions/cb12-ff16.html
[2] T. W. Bank, "World development indicators 2008," 2008.
[3] U. C. Bureau, "Summary health statisticsfor u.s.adults: National health interview survey, 2008," December 2009. [Online]. Available: https://www.cdc.gov/nchs/data/series/sr_10/sr10_242.pdf
[4] Accessibility in google apps. Google. [Online]. Available: https://www.youtube.com/watch?v=Cc7EHIc2TUE
[5] "Screen reader user survey 6 results," Jul 2015. [Online]. Available: http://webaim.org/projects/screenreadersurvey6/
[6] B. Caldwell, M. Cooper, L. G. Reid, and G. Vanderheiden, "Web content accessibility guidelines (wcag) 2.0," *WWW Consortium (W3C)*, 2008.
[7] "Android studiothe official ide for android," Apr 2018. [Online]. Available: https://developer.android.com/studio/index.html
[8] C. Corp, "Native vs hybrid / phonegap app development comparison." [Online]. Available: http://www.comentum.com/phonegap-vs-native-app-development.html
[9] M. Asay, S. Mansuri, B. Solis, K. Williams, B. Anderson, M. Quoc, and Comcast, "You're using phonegap for all the wrong reasons," Jun 2015. [Online]. Available: https://readwrite.com/2015/06/24/phonegap-apache-cordova-cross-platform-tools/
[10] C. McMeeking, "An introduction to native mobile accessibility – featuring deque university for ios/android," Mar 2017. [Online]. Available: https://www.deque.com/blog/introduction-native-mobile-accessibility-featuring-deque-university-iosandroid/
[11] "Accessibility sessions at google i/o 2017," Nov 2017. [Online]. Available: https://www.novoda.com/blog/accessibility-sessions-at-google-io-2017/
[12] "BBC Mobile Accessibility Guidelines Prototype." [Online]. Available: http://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile
[13] "Testing criteria, docs and other info." [Online]. Available: http://www.appqualityalliance.org/resources
[14] "Mobile Navigation Guidelines." [Online]. Available: http://www.funka.com/en/our-assignments/research-projects/archive---research-projects/mobile-navigation-guidelines/
[15] L. Plummer, "Rip windows phone: Microsoft kills off support for ageing smartphone os," Jul 2017. [Online]. Available: http://www.wired.co.uk/article/windows-phone-dead
[16] M. Hansen, "5 years of voiceover: Look how far we've come," Jun 2014. [Online]. Available: https://www.applevis.com/blog/advocacy-apple-assistive-technology-ios-iphone-news/5-years-voiceover-look-how-far-weve-come
[17] B. Holton, "Apple receives afb's prestigious helen keller achievement award - accessworld® - june 2015," Jun 2015. [Online]. Available: http://www.afb.org/afbpress/pubnew.asp?DocID=aw160602
[18] 2017. [Online]. Available: http://help.apple.com/iphone/10/
[19] R. Ritchie, "Accessibility now," May 2017. [Online]. Available: https://www.imore.com/accessibility-now
[20] D. Pogue, "Apple's assistivetouch helps the disabled use a smartphone," Nov 2011. [Online]. Available: https://pogue.blogs.nytimes.com/2011/11/10/apples-assistivetouch-helps-the-disabled-use-a-smartphone/?nl=technology&emc=cta2&pagewanted=all
[21] "A comparison of speech output of voiceover, talkback and narrator." [Online]. Available: http://www.incobs.de/tests-english/items/a-comparison-of-output-of-voiceover-talkback-and-narrator.html
[22] E. Millado, "ios accessibility - the accessibility inspector (swift 3)," May 2017. [Online]. Available: https://medium.com/yay-its-erica/ios-accessibility-the-accessibility-inspector-voiceover-swift-3-5f8e2bc50b20
[23] "Accessibility programming guide for ios," Feb 2012. [Online]. Available: https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/iPhoneAccessibility/Accessibility_on_iPhone/Accessibility_on_iPhone.html#//apple_ref/doc/uid/TP40008785-CH100-SW1
[24] "ios human interface guidelines." [Online]. Available: https://developer.apple.com/ios/human-interface-guidelines/interaction/accessibility/
[25] "Uiaccessibility." [Online]. Available: https://developer.apple.com/documentation/uikit/accessibility/uiaccessibility
[26] "Verifying app accessibility on ios," Apr 2013. [Online]. Available: https://developer.apple.com/library/content/technotes/TestingAccessibilityOfiOSApps/TestingtheAccessibilityofiOSApps/TestingtheAccessibilityofiOSApps.html#//apple_ref/doc/uid/TP40012619
[27] "Accessibility - iphone." [Online]. Available: https://www.apple.com/accessibility/iphone/
[28] "A brief history of android accessibility," Aug 2013. [Online]. Available: https://accessibleandroid.wordpress.com/2013/08/02/a-brief-history-of-android-accessibility/

[29] “Accessibility,” Aug 2014. [Online]. Available: https://www.androidcentral.com/accessibility
[30] “Android accessibility overview - android accessibility help.” [Online]. Available: https://support.google.com/accessibility/android/answer/6006564?hl=en
[31] “Making apps more accessible,” Mar 2018. [Online]. Available: https://developer.android.com/training/accessibility/accessible-app.html#contentdesc
[32] “Captions - android accessibility help.” [Online]. Available: https://support.google.com/accessibility/android/answer/6006554?hl=en
[33] “android.speech.tts.texttospeech,” Apr 2018. [Online]. Available: https://developer.android.com/reference/android/speech/tts/TextToSpeech.html
[34] M. Rahman, ““working as intended" - an exploration into android’s accessibility lag,” Oct 2016. [Online]. Available: https://www.xda-developers.com/working-as-intended-an-exploration-into-androids-accessibility-lag/
[35] “Testing your app’s accessibility.” [Online]. Available: https://developer.android.com/training/accessibility/testing.html
[36] “Accessibility scanner - android apps on google play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&hl=en
[37] “Accessibility,” Feb 2018. [Online]. Available: https://developer.android.com/guide/topics/ui/accessibility/index.html
[38] “Building accessible custom views,” Mar 2018. [Online]. Available: https://developer.android.com/guide/topics/ui/accessibility/custom-views.html
[39] “Making apps more accessible,” Mar 2018. [Online]. Available: https://developer.android.com/guide/topics/ui/accessibility/checklist.html
[40] “Android accessibility overview - android accessibility help.” [Online]. Available: https://support.google.com/accessibility/android/answer/6006564?hl=en
[41] “Effects of gesture patents on accessible devices.” [Online]. Available: https://www.applevis.com/forum/general-chat/effects-gesture-patents-accessible-devices
[42] “Guide to the section 508 standards.” [Online]. Available: https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/guide-to-the-section-508-standards/software-applications-and-operating-systems-1194-21