THE ACCESSIBLE USER INTERACTION FRAMEWORK FOR ANDROID APPLICATIONS

A Thesis

presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Xiaoxi Luo

June 2013
COMMITTEE MEMBERSHIP

TITLE: The Accessible User Interaction Framework for Android Applications

AUTHOR: Xiaoxi Luo

DATE SUBMITTED: June 2013

COMMITTEE CHAIR: Chris Lupo, Assistant Professor, Ph.D., Computer Science Department

COMMITTEE MEMBER: Franz Kurfess, Professor, Ph.D., Computer Science Department

COMMITTEE MEMBER: David Janzen, Associate Professor, Ph.D., Computer Science Department
Abstract

The Accessible User Interaction Framework for Android Applications

Xiaoxi Luo

As mobile technologies advance, accessibility features of smartphones are continuously improving and becoming more prevalent. It is essential that the user interaction design of mobile applications provides equal access and equal service to all our community, especially for people with disabilities. However, there are no general mobile accessibility design guidelines publicly available. This paper reviews the web accessibility development guidelines as a reference, and the current accessibility features provided by modern mobile operating systems.

It is proposed in this paper to use interactive spoken dialogues as the user interaction framework for accessible mobile applications. The idea is realized by the design and implementation of an Android application, AccessibleToolKit. AccessibleToolKit is developed to help visually impaired users to dial a phone number, manage alarms and the calendar.

A usability study of AccessibleToolKit is conducted to evaluate the user experience. Based on the feedback from the study, the design of AccessibleToolKit is refined to provide a more user-friendly experience. The performance of AccessibleToolKit is measured and compared with the performance of native apps. The comparison shows that AccessibleToolKit allow a user to perform commonly used tasks more efficiently. This study shows that using interactive spoken dialogues as the user interaction framework is a promising approach to develop accessible mobile applications.
Acknowledgements

My foremost thanks goes to my thesis advisor, Dr. Chris Lupo, for his feedback, support, and patience on the thesis.

I would like to thank Dr. Franz Kurfess, for his guidance on the usability study.

I would like to thank Dr. David Jazen, for his help on Android development.

I would like to thank my husband and parents, for their unconditional love, encouragement and belief all through the work. Finally, I owe my loving thanks to my son, Matthew, who has been the most wonderful gift in my life, and who gives me the courage to overcome setbacks.
Contents

List of Tables viii
List of Figures ix

1 Introduction 1

2 Background 3
  2.1 Mobile Accessibility 3
  2.2 Accessibility Services by Mobile Operating Systems 5
    2.2.1 Android 5
    2.2.2 iOS 11
    2.2.3 Windows Phone 13

3 Design Approach 15
  3.1 What is AccessibleToolKit 15
  3.2 Application Workflow - Main Menu 16
  3.3 Application Workflow - Phone 17
  3.4 Application Workflow - Alarm 18
  3.5 Application Workflow - Calendar 21

4 Implementation 25
  4.1 Implementation Details 25
  4.2 State Diagram Design - Main Menu 29
  4.3 State Diagram Design - Phone 29
  4.4 State Diagram Design - Alarm 32
  4.5 State Diagram Design - Calendar 35
5 Validation

5.1 Usability Evaluation Plan ............................................. 41
5.2 Usability Survey ...................................................... 45
5.3 Results and Analysis .................................................. 46
  5.3.1 AccessibleToolKit .............................................. 46
  5.3.2 Native Apps ..................................................... 53

6 Future Work and Conclusions ........................................ 56

  6.1 Future Work ....................................................... 56
  6.2 Conclusions ........................................................ 57

Bibliography ............................................................... 58

Appendix A HSC Review Documentation ............................. 61
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Android accessibility gestures [5, 6]</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Android shortcut gestures in TalkBack [6]</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Functionalities of Android Voice Actions</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Functionalities of Windows Phone Speech</td>
<td>14</td>
</tr>
<tr>
<td>5.1</td>
<td>Study group information</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Study Group 2 performance measurement results - AccessibleToolKit</td>
<td>53</td>
</tr>
<tr>
<td>5.3</td>
<td>Study Group 3 performance measurement results - AccessibleToolKit</td>
<td>53</td>
</tr>
<tr>
<td>5.4</td>
<td>Study Group 3 performance measurement results - native apps</td>
<td>55</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Voice Actions Dialogue Boxes ........................................ 10
3.1 Architecture overview on AccessibleToolKit .......................... 16
3.2 Main menu prototype .................................................. 17
3.3 Main menu flow control diagram .................................... 18
3.4 Phone prototype ....................................................... 19
3.5 Phone flow control diagram ........................................... 19
3.6 Alarm prototype ........................................................ 20
3.7 Alarm flow control diagram .......................................... 20
3.8 Calendar prototype: delete and add ................................... 21
3.9 Calendar prototype: check schedule and change calendar .......... 22
3.10 Calendar prototype: edit .............................................. 22
3.11 Calendar flow control diagram ..................................... 24
4.1 AccessibleToolKit screenshots: the main menu page (left), the alarm activity page (middle), the calendar activity page (right). The phone activity page is similar to the main menu page. .................. 28
4.2 State diagram - main menu activity .................................. 30
4.3 State diagram - phone activity ...................................... 31
4.4 State diagram - single state .......................................... 32
4.5 State diagram - alarm activity ..................................... 34
4.6 State diagram - calendar activity 1 ................................. 39
4.7 State diagram - calendar activity 2 ................................. 40
5.1 Usability evaluation screen recording setup . . . . . . . . . . . . . . . 43
5.2 Study Group 1 (fully-sighted users) survey results . . . . . . . . . . 49
5.3 Study Group 2 (visually impaired users) survey results . . . . . . . 51
5.4 Study Group 3 (fully-sighted users) survey results . . . . . . . . . . 52
5.5 Usability survey results . . . . . . . . . . . . . . . . . . . . . . . . 52
Chapter 1

Introduction

In recent years, the wide usage of smartphones has arguably altered our daily lives. Every day more than 1 million new Android devices are activated worldwide [1]. Though lacking official statistics, it is estimated that a total of 219 million iPhones have been sold over the past three generations since June 2010 [3]. The rapid development of smartphones has fundamentally changed the way the user interacts with his or her device as well. It is essential that the user interaction design of mobile applications provides equal access and equal service to all our community, especially for people with disabilities.

However, due to the fast development of mobile operating systems and technologies, there are no general guidelines on mobile accessibility, and the best practice on user interaction design in that context. The User Interaction (UI) design, which uses screen gestures to interact with UI components is not intuitive for visually impaired users, as it is challenging for them to locate UI components on the screen and read the description of the UI component. The motivation of this paper is to explore using the interactive spoken dialogue as the user interaction framework to develop accessible applications for visually impaired users.
The objectives of the study are to:

- Design and implement an accessible application, AccessibleToolKit, to realize the interactive spoken dialogue UI framework.

- Conduct usability study on AccessibleToolKit to evaluate the user experience. Utilize the study results to refine the app development.

- Evaluate the performance of AccessibleToolKit by performing a set of pre-defined tasks. The time measurement and error count are used as the evaluation criteria. The native apps are also evaluated using the same set of tasks as the baseline of the performance study.
Chapter 2

Background

Since no general mobile accessibility design guideline has been developed, the related resources on web accessibility are reviewed in this chapter. Furthermore, the current accessibility features provided by modern mobile operating systems are also reviewed. The mobile operating systems reviewed include Android, iOS, and Windows Phone.

2.1 Mobile Accessibility

It is recognized by the UN Convention on the Rights of Persons with Disabilities that access to information and communications technologies is a basic human right [16]. It is essential that the mobile devices provide equal access and equal service to people with diverse abilities. According to the World Wide Web Consortium (W3C), Mobile accessibility generally refers to making websites and applications more accessible to people with disabilities when they are using mobile phones [21]. The fast development of smartphone applications provide users with a lot of functionality which should be beneficial for users with disabilities as well.
Web Accessibility Initiative (WAI) is a subgroup of World Wide Web Consortium (W3C), and it is dedicated to help making the Web accessible to people with disabilities. WAI develops Web Content Accessibility Guidelines (WCAG) primarily for Web content developers, Web authoring tool developers, Web accessibility evaluation tool developers, and others who need a standard for web accessibility. WAI is actively working on developing more guidance on applying WCAG in the mobile context [21]. Even though there is no standard for mobile accessibility yet, the principles from WCAG could be referred to as guidelines. The four principles of WCAG are listed below [22]:

- **Perceivable** - Information and user interface components must be presentable to users in ways they can perceive. This means that the user must be able to perceive the information being presented. If the user is visually impaired, it is not sufficient to present the information solely on the screen, as it is not visible to their senses.

- **Operable** - User interface components and navigation must be operable. The interface can not require interaction that a user cannot perform. For example, if an interaction requires the user to see a dialogue box to make a decision, it is not accessible for visually impaired users. In the context of mobile applications, operable means the user should be able to operate the mobile application with the interactions he or she can perform.

- **Understandable** - Information and the operation of user interface must be understandable. This means that the information provided by the app and the operation of the user interface should not be beyond the user’s understanding.

- **Robust** - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies. In the context of
mobile applications, robust means the mobile application content should remain accessible with the rapid evolving speed of mobile operating systems.

2.2 Accessibility Services by Mobile Operating Systems

In this section, the accessibility services provided by modern mobile operating systems are reviewed. This section also reviews some related features or applications, which are not featured as the accessibility services, but provide visually impaired users with convenience. The systems reviewed include: Android, iOS, and Windows Phone.

2.2.1 Android

The accessible features of Android OS include TalkBack, Explore by Touch, and Accessibility Settings [7].

TalkBack is a pre-installed screen reader service provided by Google [8]. It uses synthesized speech feedback to describe the results of actions, such as opening an app or clicking on a control, and events, such as notifications. It also lets the user set up these additional types of feedback [7]:

- Vibration feedback lets the user assign vibration patterns to different actions on the phone. (Also known as KickBack.)

- Sound feedback lets the user assign sounds to different actions on the phone. (Also known as SoundBack.)
<table>
<thead>
<tr>
<th><strong>Navigation gestures</strong></th>
<th><strong>Resulting action</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Drag one finger</td>
<td>Explore the screen and hear audible feedback for what is being touched.</td>
</tr>
<tr>
<td>Double tap</td>
<td>Opens or activates the item last touched.</td>
</tr>
<tr>
<td>Swipe up or down with two fingers</td>
<td>Scroll lists and pages</td>
</tr>
<tr>
<td>Swipe left or right with two fingers</td>
<td>Change pages and screens.</td>
</tr>
<tr>
<td>Swipe right (or down) using one finger</td>
<td>Move to the next item.</td>
</tr>
<tr>
<td>Swipe left (or up) using one finger</td>
<td>Move to the previous item.</td>
</tr>
<tr>
<td>Swipe down then up using a single motion</td>
<td>Transition to the next reading level when reading blocks of text, then swipe right to read forward or left to read backward.</td>
</tr>
<tr>
<td>Swipe up then down using a single motion</td>
<td>Transition to the previous reading level when reading blocks of text, then swipe right to read forward or left to read backward.</td>
</tr>
<tr>
<td>Swipe right then left using a single motion</td>
<td>Move to the next piece of information.</td>
</tr>
<tr>
<td>Swipe left then right using a single motion</td>
<td>Move to the previous piece of information.</td>
</tr>
</tbody>
</table>

Table 2.1: Android accessibility gestures [5, 6].

Explore by Touch is a system feature that works with Talkback [7], allowing the user to touch the phone’s screen and hear the content explored spoken aloud. This feature is especially helpful to users with no vision. Users can turn on these services in the Accessibility setting. Once these services are turned on, the gestures to navigate the device are changed. Table 2.1 lists the accessibility gestures.

Starting from Android 4.1 (Jelly Bean), a new set of accessibility shortcut gestures is provided for the user to navigate the phone. Such screen gestures work only when Explore by Touch and TalkBack are turned on. Table 2.2 lists some of the shortcut gestures.

Accessibility settings let the user modify the phone’s display and sound options,
such as turning on TalkBack, increasing the text size, changing the speed at which text is spoken, using the power button to end calls, Text-to-speech output, and more [8]. Android 4.2 allows triple-tap to magnify the entire screen, pan and zoom with two fingers. The user can adjust the speech rate of Text-to-speech to very slow, slow, normal, fast, or very fast. The user can also modify the following settings of TalkBack [10]: when to speak (including speech volume, ringer volume, using pitch changes, speak when screen is off, use proximity sensor, and speak caller ID); feedback settings (including vibration feedback, sound feedback, and sound volume). These settings allow some useful features: the user can have the spoken feedback turned on when the screen is off; read the caller’s information when receiving a call; silence spoken feedback when you put your phone to your ear or place your hand near its top.

In order for Explore by Touch to work with different User Interface (UI) design, UI components that do not have visible text should be labeled, especially ImageButton, ImageView, and EditText components. Developers could use the android:contentDescription XML layout attribute or setContentDescription() to provide this information for accessibility services [12, 13]. According to Android Developer documentation [13], the developer should also make sure that all user interface elements that can accept input (touches or typing) can be reached with a directional controller, such as a trackball, D-pad or navigation gestures. To assist users that are deaf or hard of hearing, the audio prompts should be accompanied by a visual prompt.

<table>
<thead>
<tr>
<th>Accessibility gestures</th>
<th>Resulted action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up then right</td>
<td>Open notifications</td>
</tr>
<tr>
<td>Up then left</td>
<td>Home button</td>
</tr>
<tr>
<td>Down then right</td>
<td>Recent apps button</td>
</tr>
<tr>
<td>Down then left</td>
<td>Back button</td>
</tr>
<tr>
<td>Rapidly stroking right than left</td>
<td>Scroll lists forward</td>
</tr>
</tbody>
</table>

Table 2.2: Android shortcut gestures in TalkBack [6].
or notification.

Starting from Android 2.2 (Froyo), Voice Actions are supported by the Android platform [18]. Voice Actions are a series of spoken commands that let the user control the phone using his/her voice. The user can use Voice Actions to perform common tasks, such as sending an email, calling a contact, or opening an app. To use Voice Actions, tap the microphone button on the Google search box on the home screen, or press down for a few seconds on the physical search button on the phone to activate the Speak Now screen. Starting from Android 4.1 (Jelly Bean), Voice Actions are further expanded and included as part of the Google Now application [9]. Voice Actions now can also be launched by touching the microphone button or saying “Google” in the Google Now app. Table 2.3 lists some of the recognized commands and associated functionalities that can be performed by Voice Actions [18].

Voice Actions allow users to launch different smartphone functionalities with voice. Even though it is not advertised as an accessibility feature, it does provide a convenient way for visually impaired users to launch applications, make phone calls, and more. However, due to the fact that Voice Actions is not designed as an accessibility feature, it lacks necessary feedback to the visually impaired users. For example, if the user tries to dial the number 805-756-1111, he or she says “call 8057561111”. Voice Action recognizes the phone number and performs dialing. During the process, Voice Actions pops up a dialogue box with a progress bar to tell the user the voice recognition result and the dialing is in progress, as shown in Figure 2.1. A full-sighted user can easily see and confirm the voice recognition result, and decide if he or she wants to end the dialing process. The user can end the dialing process by clicking the cross button next to the progress bar. However, for visually impaired users, the information presented by Voice Actions is not visible to their senses as the whole process is not audible.
<table>
<thead>
<tr>
<th>Say</th>
<th>Followed by</th>
<th>Examples</th>
</tr>
</thead>
</table>
| “Call”                                   | “phone number” or “[contact name] [phone type]” | “Call Mary Jackson home”  
|                                          |                                              | “Call 8057561111”                                                      |
| “Send text to”                           | “[recipient] [message]”                      | “Send text to Mary Jackson  
I will be home around 9”                                                   |
| “Directions to” or “Navigate to”         | Address, name, business name, type of business, or other destination | “Directions to 1299 Colusa Avenue, Berkeley, California”  
or “Navigate to Union Square, San Francisco.”                           |
| “Set alarm”                              | “Time” or “for” & time, such as “10:45 a.m.” or “20 minutes from now,” “Label” & name of alarm | “Set alarm for 7:45 p.m., label, switch the laundry”  
| “Create a calendar event”                | “Event description” & “day/date” & “time”   | “Create a calendar event:  
Dinner in San Francisco, Saturday at 7:00PM”                           |
| “Open”                                   | App name                                     | “Open Gmail”                                                           |
| “Map of”                                 | Address, name, business name, type of business, or other location | “Map of Golden Gate Park, San Francisco.”                               |
| “Note to self”                           | Message text                                 | “Note to self: remember the milk”                                     |
| “Listen to”                              | Play music in the Google Play Music app by speaking the name of a song, artist, or album. | “Listen to: Smells Like Teen Spirit”                                    |
| “Send email”                             | Say “send email to [recipient] [subject]* [body]” | “Send email to Hugh Briss,  
subject, new shoes, message, I cant wait to show you my new shoes, period.” |
| “Go to”                                  | Search string or URL                         | “Go to Google.com”                                                     |

**Table 2.3: Functionalities of Android Voice Actions**
Figure 2.1: Voice Actions Dialogue Boxes

Figure 2.1 also illustrates the dialogue boxes when the user tries to set an alarm or a calendar event. The user interaction pattern of setting alarms is similar to voice calling. However, when creating a calendar event, the user will have to confirm creating by clicking on the “Create event” button. The event creation does not occur automatically after the voice recognition results are returned. The inconsistency in the user interaction design is less possible to cause confusion for full sighted users. For visually impaired users, however, lack of sensible feedback easily causes confusion and results in an unfavorable user experience.

Nevertheless, Voice Actions provide a convenient way to launch commonly used functionalities for visually impaired users, which is a good start for developing fully accessible voice launchers.

Speech input can fundamentally change the mobile experience [19]. With the recent progress of speech recognition in mobile development, speech is becoming the new user interaction interface. Speech recognition could be beneficial for users with a disability to free them from having to use the transitional touch screen user interaction. Instead of physical touch, the user can have vocal communication with the
smartphone. This is one of the enabling technologies that are employed in the development of AccessibleToolKit. Started from Android 4.1 (Jelly Bean), two voice input services are provided. One is Voice Recognition, the other is Voice Typing. Voice Recognition API has been available since Android API 3. The application can potentially make use of any speech recognition service on the device that’s registered to receive a `RecognizerIntent` [19]. Google’s Google Now application, responds to a `RecognizerIntent` by displaying the “Speak now” dialog and streaming audio to Google’s servers - the same servers used when a user taps the microphone button on the search widget [19].

Voice Typing is a new feature introduced since Android 4.0 (Ice Cream Sandwich) [20], which makes the voice dictation system work even when no data connection is available. When the soft keyboard is brought up, the microphone button in the soft keyboard initiates the voice typing functionality. The recognition results appear in the text box while the user is still speaking. However, currently this feature is primarily used for Input Method Editor (IME), and is distributed by a library jar file. There is no API available to implement Voice Typing into other applications. Due to this reason, Voice Recognition API is used in the development of AccessibleToolKit.

2.2.2 iOS

iOS provides the following accessibility features:

- VoiceOver. VoiceOver [4] is a screen reader that is included as a standard feature on iPhone 3GS or later. Similar to Touch and Explore, VoiceOver allows the user to hear descriptions of items while the user’s finger interacts with items on screen. VoiceOver includes build-in voices that speak 36 languages. Once VoiceOver is turned on, the user needs to use a different set of gestures to control
Instead of tapping to activate a button, the user needs to tap the button first to hear the button description and then double-tap to activate it. Instead of using one finger to scroll, the user needs to use three fingers to scroll. VoiceOver features a special accessibility gesture-rotor. By rotating two fingers on the screen as turning an actual dial, the way VoiceOver moves through a document can be changed based on a setting the user chooses. The rotor can also be used to navigate web pages. When the user is viewing a web page, the rotor contains the names of UI items on the page. The user can move to the previous or next UI item on the page by flicking up and down.

When entering text, VoiceOver not only echoes the character the user entered, it can also read each completed word. VoiceOver can also speak the suggested word to help the user to speed up typing. When the suggested word is spoken, the user can choose to continue typing to ignore it, or to press the space key to enter the suggested word [4].

- Siri. Siri [4] is an interactive application to allow the user to perform some phone activities via voice communication with the application. It allows the user to use his or her voice to send messages, schedule meetings, place phone calls, etc. It is available for iPhone 5 and iPhone 4S. Siri is designed to process natural language and is a proactive application. Siri will also ask the user questions in order to collect the necessary information to perform the user’s request. Siri works with most of the built-in apps. VoiceOver can be used to read the text that is displayed on the screen as the responses from Siri. The user can touch and navigate the responses on the screen to have each item read aloud.

- Zoom. Zoom [4] allows the user to magnify the entire screen of any application by double tapping with three fingers.
• Invert Colors [4]. This feature allows users to change the display to white on black.

• Large Text [4]. The user can increase the font size to see up to 56-point text in alerts, Calendar, Contacts, Mail, Messages, and Notes.

2.2.3 Windows Phone

Windows Phone does not have a screen reading feature yet. Accessibility features on Windows phone include [14]:

• Change the text size: the user could choose from five different text sizes for phone, People, email, messaging, and on the lock screen.

• Use a high-contrast theme: change all of the icons and text to black and white and provide a solid background behind words that would otherwise be on top of pictures.

• Screen magnifier: allow the user to use gestures to enlarge the page by double tapping with two fingers.

• Speech. Started from Windows Phone 7, Speech is provided as a major accessibility feature of Windows Phone.

In Windows Phone 8, Speech accepts several command patterns, such as “Call”, “Text”, “Note”, “Find”, and “Open”. The user needs to say a command started with one of the five key words to perform certain desired functionality, such as make a call, send a text, or find certain content on the Internet. Table 2.4 lists the recognized commands and associated functionalities that can be performed by Speech [15].
<table>
<thead>
<tr>
<th><strong>To do</strong></th>
<th><strong>Say this</strong></th>
</tr>
</thead>
</table>
| Make a phone call         | “Call contact name”  
(where contact name is the name of someone in the user’s contact list)  
“Call phone number”  
(where phone number is any phone number, whether it’s assigned to a contact or not)  
“Call voicemail”     |
| Redial the last number    | “Redial”                                                                                                                                                    |
| Send a text message       | “Text contact name”  
(where contact name is the name of someone in the user’s contact list)  
This will start a text message to that person, and then the user can say the message. When the user finishes speaking, he or she can say “Send” to send it, “Add more” to add more info to your message, or “Try again” to say the message again. |
| Take a note               | “Note what you want to note”  
(where what you want to note is the text that the user wants to include in your note)                                                                                           |
| Open an app               | “Open app”  
(where app is the name of any app, such as “Calendar”  )                                                                                                  |
| Search the web            | “Find search term” or “Search for search term”  
(where search term is what you’re looking for)                                                                                                               |

**Table 2.4: Functionalities of Windows Phone Speech**
Chapter 3

Design Approach

This chapter describes the design of AccessibleToolKit. Section 3.1 gives a brief introduction of the functionalities and high-level architecture of AccessibleToolKit. The workflow and prototype of AccessibleToolKit is illustrated in the Section 3.2, Section 3.3, Section 3.4, and Section 3.5. The traditional UI prototype design is proposed to illustrate the functionalities of the application. Though many UI components described in the prototype are not implemented, the functionalities of these UI components are implemented through voice communication. The prototype provides an illustration on the features provided and the information that should be obtained from the user.

3.1 What is AccessibleToolKit

AccessibleToolKit is an Android voice launching application, which launches three tailored accessible functions: phone, alarm, and calendar. AccessibleToolKit is designed to provide equal access for visually impaired users to the commonly used smartphone functionalities. AccessibleToolKit uses interactive spoken dialogues in-
Figure 3.1 illustrates the high level architecture of AccessibleToolKit, which is similar to architecture of the standard Spoken Dialogue System [11]. The speech recognition process includes capturing user input, and communicating with Google voice recognition service to process user input. The linguistic analysis part is to map the voice recognition results to an operation command to the operation module. After performing corresponding operations, the text generation part provides human understandable text as the feedback to the user. Finally, the speech synthesis module reads the text to the user.

3.2 Application Workflow - Main Menu

The functionalities of the Main Menu include getting the user’s choice to launch the phone activity, the alarm activity, the calendar activity, the speech rate setting and a demo of the Push-To-Talk user interaction mechanism. The user can adjust the
text-to-speech speech rate to slow, normal, or fast. As discussed in Chapter 5, the usability study of AccessibleToolKit shows that there is a learning curve for users to grasp the Push-To-Talk mechanism. Therefore a demo feature is added to the Main Menu to help users understand how to interact with the app. The prototype of Main Menu is shown in Figure 3.2.

As illustrated in Figure 3.3, when the app is launched the first time, the demo module runs by default. The demo starts with an audible introduction of the app, and then prompts the user to tap the screen to hear the next instruction, which asks the user for his/her name. This instruction also instructs the user to tap the screen before speaking. After hearing the user’s input, the app greets the user and terminates the demo. The user can choose to try the demo again after it’s finished.

3.3 Application Workflow - Phone

As illustrated in Figure 3.4, the phone activity allows the user to dial with a phone number, or a contact name. The user might need to specify a phone type if the
Figure 3.3: Main menu flow control diagram

contact contains multiple phone types. The workflow for the phone activity is shown in Figure 3.5. After getting a phone number or a contact entry, such information is confirmed with the user before dialing. The user would have a chance to correct the misunderstood input. If there are multiple entries found for a contact name, the user would be prompted to choose the phone type to call.

3.4 Application Workflow - Alarm

Figure 3.6 illustrates the functionalities of the alarm activity. The user can add an alarm with repetition, edit an existing alarm, delete an existing alarm, and check all the alarms set. When trying to delete an alarm, the user can identify the alarm by either the alarm time w/o repetition, or the alarm index, for example, “alarm 1”. Before performing adding, editing, or deleting, the user would have a chance to correct the misunderstood input. The workflow of the alarm activity is illustrated in Figure 3.7.
Figure 3.4: Phone prototype

Figure 3.5: Phone flow control diagram
Figure 3.6: Alarm prototype

Figure 3.7: Alarm flow control diagram
3.5 Application Workflow - Calendar

The user can add an event or delete an event in the calendar, as illustrated in Figure 3.8. The user can also edit an existing event, as illustrated in Figure 3.10. The user can use the app to check his/her schedule on a certain day, or change the calendar AccessibleToolKit works with, as illustrated in Figure 3.9 and 3.10. The workflow of the calendar activity is illustrated in Figure 3.11.

To add an event, the user needs to at least specify the event title, the start time, and the end time. The start and end time includes both date and the time of a day. After providing this information, the event will be added to the calendar. After the event is added, the user may choose to add optional information to the event. Such information includes the event location, the event description, reminders, and repetition. The user may have multiple reminders for one event. The event may be set to be repeated one time only, daily, on weekdays, weekly, monthly, or yearly.

In order to identify the event to be deleted or edited, the user needs to specify the event title and start/end time. For the start/end time, the user needs to provide
Figure 3.9: Calendar prototype: check schedule and change calendar

Figure 3.10: Calendar prototype: edit
both the date and the time. From the feedback of the usability study, users found this to be a tedious process. In lots of cases, there is only one event with a certain tile on a certain day. Therefore, the app was redesigned to get the event title and start day first, and then try to find the event information. If no entries are found, such an event does not exist. If multiple event entries are found, the app will continue to get the start time, end date, and end time in order to identify an event. If one event is found, the app confirms with the user if it is the event he/she is looking for. For any event chosen, the user may edit the following fields: the event title, the start time (including both date and time), the end time (including both date and time), the event location, the event description, reminders, and repetition.

The user may have multiple calendars on the device. When the calendar activity is launched for the first time, the app reads all the calendar names and prompts the user to choose a calendar to work with. The user can also modify this setting later on.
Figure 3.11: Calendar flow control diagram
Chapter 4

Implementation

In this chapter, the implementation of the AccessibleToolKit application framework is discussed. Section 4.1 discusses the implementation details. Section 4.2 discusses the state diagram for the main menu activity. Section 4.3 discusses the state diagram and implementation of the phone activity. Section 4.4 discusses the state diagram and implementation of the alarm activity. Section 4.5 discusses the state diagram and implementation of the calendar activity.

4.1 Implementation Details

The two key technologies necessary for AccessibleToolKit are voice recognition and speech synthesis (Text-to-Speech). Voice recognition is implemented with Android RecognizerIntent API. The language model used is LANGUAGE_MODEL_FREE_FORM, which is based on free-form speech recognition. As discussed in detail in Section 4.2 - Section 4.5, AccessibleToolKit uses states to record the status of the dialogue between the user and the app. When launching the RecognizerIntent by startActivityForResult(), the current state is used as the request code. The same request code is returned
in `onActivityResult()` when the activity started by `RecognizerIntent` exists. This request code is then used as the flag on determining how to process the recognized voice input. Based on the request code, different processing functions are called to process the user input.

Speech synthesis is implemented with Android `TextToSpeech` API, which is provided since API 14 [2]. Since `TextToSpeech` is heavily used in all the activities, it is inefficient and expensive to create a `TextToSpeech` instance in each activity. As a solution, a Singleton class is created to wrap the `TextToSpeech` object, which is instantiated only once in the application. A general practice of using `TextToSpeech` is to call the `shutdown()` method to release the native resources used by the `TextToSpeech` engine when done using the `TextToSpeech` instance. However, this should only be done in the `onDestroy()` method of the main menu activity. Since the speech synthesis is an async task, if the `shutdown()` method is called in the `onStop()` method, the current utterance may be interrupted and discarded when switching between activities. Also the `shutdown()` method should not be called within the phone, alarm, or calendar activity. If the the `shutdown()` method is called within these activities, the `TextToSpeech` instance fails to bind to the `TextToSpeech` engine when returning to the main menu activity. This results in that the first utterance in the main activity is never read.

The `RecognizerIntent` returns an `ArrayList <String>` object, containing the recognition results. The key word/phrase to be matched can also be represented as an `ArrayList <String>` object. Algorithm 4.1 illustrates the fuzzy matching algorithm used to determine if the key word/phrase is found in the voice recognition results. Let `matches` contain the voice recognition results, and `targets` contain the key word or phrase to be matched.

This simple algorithm introduces a relaxation factor `MIN_MATCH_RATIO`, which
Algorithm 4.1 Fuzzy matching algorithm

1: $isMatched \leftarrow false$

2: $maxMatchCount \leftarrow 0$

3: for each str1 in matches do

4: \hspace{1em} $matchCount \leftarrow 0$

5: \hspace{1em} for each str2 in targets do

6: \hspace{2em} if str1 contains str2 then

7: \hspace{3em} $matchCount \leftarrow matchCount + 1$

8: \hspace{2em} end if

9: \hspace{1em} end for

10: if $matchCount > maxMatchCount$ then

11: \hspace{1em} $maxMatchCount \leftarrow matchCount$

12: end if

13: end for

14: $matchRatio \leftarrow maxMatchCount/targets.length$

15: if $matchRatio \geq MIN\_MATCH\_RATIO$ then

16: \hspace{1em} $isMatched \leftarrow true$

17: end if
Figure 4.1: AccessibleToolKit screenshots: the main menu page (left), the alarm activity page (middle), the calendar activity page (right). The phone activity page is similar to the main menu page.

determines whether the a given string should be considered as being matched with the target string. MIN\_MATCH\_RATIO being 1 means the given string has to be an exact match of the target string to be considered matched. In AccessibleToolKit MIN\_MATCH\_RATIO is set to be 0.6, which means the given string is considered a match if more than 60% of the words in the given string are matched with the words in the target string. This relaxation factor allows a fuzzy match, which reduces the voice recognition error rate to improve the user experience.

In order to help the visually impaired users to find and launch AccessibleToolKit, a large sized static widget is implemented by extending App\_Widget\_Provider class and overriding onUpdate() method. The functionality of the widget is just to launch AccessibleToolKit.

Figure 4.1 shows the user interface of AccessibleToolKit. The only interactive UI component is the push-to-talk button, whose size matches the size of the layout. The transcripts are shown at the bottom of each layout for debugging purposes.
4.2 State Diagram Design - Main Menu

Figure 4.2 shows the state diagram of the main menu activity. The status of each state is described as below.

- IDLE. Wait for user interaction to start.

- PROMPT_NAME. The start state of the demo feature. Prompt the user to tap screen to start a dialogue. Wait for taping.

- GET_NAME. The second state of the demo features. Echo the user’s taping and wait for the user to say his/her name.

- GET_SERVICE_TYPE. Wait for user’s command, such as “phone”, “alarm”, “calendar”, “settings”, “demo”, or “exit”.

- GET_SETTING. Wait for the user’s input on speech rate setting. Expected key words are “slow”, “normal”, and “fast”.

4.3 State Diagram Design - Phone

Figure 4.3 shows the state diagram of the phone activity. The status of each state is described as below.

- IDLE. Wait for user interaction to start.

- GET_NUM_CONTACT. Wait for the user’s input on phone number or the contact name. Expected input is in the form of “805-756-1111” or “Mary Jackson”.

- CONFIRM_NUM. Wait for the user’s confirmation on the phone number to dial. Expected input is “Yes” or “No”.

29
Figure 4.2: State diagram - main menu activity
Figure 4.3: State diagram - phone activity

- CONFIRM_TYPE. Enters this state if there are multiple entries found with same display name. Wait for the user’s input on the phone time, such as “Mobile”, “Home”, “Work”, etc.

- CONFIRM_TYPE_NUMBER. Enters this state if there are multiple entries with the same display name and phone type. Wait for the user’s input on which phone number to dial. Expected input is in the form of phone type followed by an index, such as “Mobile 1”.

To simplify the state diagrams, the transitions general to all states are discussed here and are therefore omitted in Figure 4.3, Figure 4.5, Figure 4.6, and Figure 4.7.
Figure 4.4 shows the general transitions of each state. For each state, the user can return to the main menu using the key word “Main menu”. The key word “Repeat” tells the app to repeat the previous instruction, and the current state remains unchanged. When invalid user input is received, the state remains unchanged. The key word “Phone/Alarm/Calendar home” can bring the state to the IDLE state.

The implicit intent with the action of `ACTION_CALL` is used to initiate a phone call. After the call is hung up, the phone activity is brought up to the top of the activity stack by a `PhoneStateListener`.

4.4 State Diagram Design - Alarm

Figure 4.5 shows the state diagram of the alarm activity. The status of each state is described as below. Another flag variable, mode, is used in conjunction with state to specify the status of the dialogue. Three difference modes used are ADD_MODE,
DELETE_MODE, and EDIT_MODE.

- IDLE. Wait for user interaction to start.

- GET_COMMAND. Wait for the user’s command. Expected key words include “Add an alarm” (ADD_MODE), “Delete an alarm” (DELETE_MODE), “Edit an alarm” (EDIT_MODE), and “Tell me my alarms”.

- ALARM_TIME. Wait for the user’s input on alarm time. Expected input is in the form of “9:30 AM”.

- ALARM_ISREPEATED. Wait for the user’s input on whether the alarm is a repeating alarm. Expected input is in the form of “Yes” or “No”.

- ALARM_REPEAT. Wait for the user’s input on repetition. Expected input is “Everyday”, “Weekend”, “Weekdays”, and “Monday, Tuesday...”.

- ALARM_CONFIRM. Wait for the user’s confirmation on the alarm information (time and repetition). Transitions from the state are based on the current mode of the system.

- GET_ALARM. The first state entered if the user would like to delete/edit an existing alarm. Expected input is in the form of “Alarm 1” or “9:30 AM”.

- GET_EDIT_FIELD. Wait for the user’s input on which field to edit. Expected input include “Alarm time”, “Alarm repetition”, and “Finish editing”.

- GET_NEW_TIME. Wait for the user’s input on alarm time. Expected input is in the form of “9:30 AM”.

- GET_NEW_REPETITION. Wait for the user’s input on alarm repetition. Expected input is “Everyday”, “Weekend”, “Weekdays”, and “Monday, Tuesday...”.

33
Figure 4.5: State diagram - alarm activity
The current Android SDK does not provide an alarm API to access the system alarm provider. One can set an alarm by using an implicit intent with the action of `AlarmClock.ACTION_SET_ALARM`. However, there are no methods to delete, edit, or list the existing alarms.

In AccessibleToolKit, alarms are created and managed by the app itself. The alarm entries are managed with the SQLite database. The `AlarmManager` API is used to schedule a `PendingIntent` based on the alarm time and repetition. This `PendingIntent` object starts an activity, which creates a visible alarm indicator and plays an alarm tone. If the alarm is a one-time alarm, it will be removed from the database table after the user cancels it.

### 4.5 State Diagram Design - Calendar

The state diagram of the calendar activity is split into Figure 4.6 and Figure 4.7 due to its size. The three discontinued arcs in Figure 4.6 go to or come from the GET_ADDITIONAL_INFO state in Figure 4.7. The status of each state is described as below. Similar as the alarm activity, another flag variable, mode, is used specify the status of the dialogue. The three difference modes used are ADD_MODE, DELETE_MODE, and EDIT_MODE.

- **IDLE.** Wait for user interaction to start.

- **GET_COMMAND.** Wait for the user’s command. Expected key words include “Add an event” (ADD_MODE), “Delete an event” (DELETE_MODE), “Edit an event” (EDIT_MODE), “Change calendar”, and “Tell me my schedule”.

- **GET_EVENT_NAME.** Wait for the input of the event name.
- CONFIRM EVENT_NAME. Wait for the confirmation of the event name. Expected input includes “Yes” and “No”.

- GET START_DATE. Wait for the input of the start date. Expected input is in the form of “May 31” or “May 31, 2013”.

- GET START_TIME. Wait for the input of the start time. Expected input is in the form of “9:30 PM”.

- CONFIRM START_TIME. Wait for the confirmation of the start date and time. Expected input includes “Yes” and “No”.

- GET END_DATE. Wait for the input of the end date. Expected input is in the form of “May 31” or “May 31, 2013”.

- GET END_TIME. Wait for the input of the start time. Expected input is in the form of “9:30 PM”.

- CONFIRM END_TIME. Wait for the confirmation of the end date and time. Expected input includes “Yes” and “No”.

- CONFIRM EVENT. Wait for the confirmation of the event found. Expected input include “Yes” and “No”.

- GET CALENDARID. Wait for the input of calendar of choice. Expected input is in the form of “Calendar 1”.

- GET SCHEDULE_DATE. Wait for the input of the date. Expected input is in the form of “May 31” or “May 31, 2013”.

- GET ADDITIONAL_INFOR. Wait for the user’s input on which field to edit. Expected input include “Start time”, “End time”, “Event name”, “Repetition”, “Reminder”, “Description”, and “Location”.

36
• **GET_STARTDATE**. Wait for the input of the start date. Expected input is in the form of “May 31” or “May 31, 2013”.

• **GET_START_TIME**. Wait for the input of the start time. Expected input is in the form of “9:30 PM”.

• **CONFIRM_START_DATE**. Wait for the confirmation of the start date and time. Expected input includes “Yes” and “No”.

• **GET_ENDDATE**. Wait for the input of the end date. Expected input is in the form of “May 31” or “May 31, 2013”.

• **GET_END_TIME**. Wait for the input of the start time. Expected input is in the form of “9:30 PM”.

• **CONFIRM_ENDDATE**. Wait for the confirmation of the end date and time. Expected input includes “Yes” and “No”.

• **GET_NAME**. Wait for the input of the event name.

• **CONFIRM_NAME**. Wait for the confirmation of the event name. Expected input includes “Yes” and “No”.


• **CONFIRM_REPETITION**. Wait for the confirmation of the repetition setting. Expected input includes “Yes” and “No”.

• **GET_REMINDER**. Wait for the input of number of minutes of the reminder. Expected input includes “0”, “5”, “15”, “30”, “45”, and “60”.

37
• CONFIRM_REMINDER. Wait for the confirmation of the reminder. Expected input includes “Yes” and “No”.

• GET_DESCRIPTION. Wait for the input of the event description.

• CONFIRM_DESCRIPTION. Wait for the confirmation of the description. Expected input includes “Yes” and “No”.

• GET_LOCATION. Wait for the input of the event location.

• CONFIRM_LOCATION. Wait for the confirmation of the location. Expected input includes “Yes” and “No”.

Starting from API 14 (Ice Cream Sandwich), Android SDK provides a fully-worked-out set of API for accessing Calendar data, namely the CalendarContract API [2]. Calendar specific information is retrieved from the CalendarContract.Calendars table. The event specific information is retrieved from the CalendarContract.Events table. The reminders are inserted into the CalendarContract.Reminders table. To check the schedule on a certain day, the CalendarContract.Instances table is queried to get all the event instances on a specific day.

When first launching the calendar activity, the user is prompted to choose a calendar to work with. The user’s choice is stored in SharedPreferences. The user can also modify this preference later on.
Figure 4.6: State diagram - calendar activity 1
Figure 4.7: State diagram - calendar activity 2
Chapter 5

Validation

In this chapter, the validation framework of AccessibleToolKit will be discussed. AccessibleToolKit will be evaluated by a usability study. As a study benchmark, the native phone, alarm, and calendar apps are also evaluated. Section 5.1 describes the usability evaluation plan. Section 5.2 presents the content of the usability survey conducted. The usability study data is discussed and analyzed in Section 5.3.

5.1 Usability Evaluation Plan

AccessibleToolKit is developed to help visually impaired users to use smartphones more intuitively. The participants of the usability study include visually impaired users, as well as full sighted users. Since this study is human subjects research, the study procedures, the Informed Consent Form, and the Human Subjects Protocol Approval Form have been reviewed and approved by Cal Poly Human Subjects Committee (HSC). Full documentation is included in Appendix A.

The Disability Resource Center at Cal Poly, was contacted to recruit visually
impaired users. Fully sighted users are also recruited since the target demographic has a relatively small population. In order to minimize the bias as a fully sighted user, it is required that fully sighted participants perform the evaluation tasks without looking at the screen.

The procedures of the usability evaluation are described as below:

1. The phone screen is videotaped while the participant is interacting with the app.

2. The participant is asked to perform a set of predefined tasks using Accessible-ToolKit. The time the participant spends on each task and the errors that occur during each task are recorded.

3. The participant is also asked to perform the same tasks using the native apps with TalkBack turned on. The time the participant spends on each task and the errors that occur during each task are recorded.

4. The participant is asked to take a usability survey to collect the feedback on his/her experience with AccessibleToolKit. For the visually impaired users, the survey questions are read to them. Their answers are recorded by the researcher.

The camera recording setup is as shown in Figure 5.1. A Logitech BCC950 ConferenceCam is used to video record the phone screen while the user is interacting with the app. The video clips are analyzed later to evaluate the user interaction design. In order for the phone screen to be recorded, the participant needs to hold the phone in front the camera. Some participants found this setup to interfere with the way they interact with the phone and prefer not using the phone with the recording setup. In these situations, the participants are encouraged to use the phone in the way they prefer.
Figure 5.1: Usability evaluation screen recording setup
The participants are asked to perform the following tasks:

1. Task 1: dial by saying a phone number.

2. Task 2: try to call Mary Jackson’s mobile number. The testing device has a contact entry with the name Mary Jackson. This entry contains multiple phone types, such as work, home, and mobile.

3. Task 3: Set a repeating alarm at 9:30 am, repeated on workdays.

4. Task 4: Have the app read all the alarms.

5. Task 5: Disable the alarm you just set.

6. Task 6: Add an event to the calendar with the information as specified as below:
   
   Title: Movie time;
   Start time: May 16, 9:00 pm;
   End time: May 16, 10:30 pm;
   Location: Downtown San Luis Obispo.

7. Task 7: Check your schedule on May 16

8. Task 8: Delete the event you just added.

As indicated in the Informed Consent Form, the participants are not obligated to finish all the tasks. They may choose to discontinue any of the tasks. Actually some participants did choose not to finish all the tasks, often because they felt it was difficult to accomplish certain tasks with the application. These results are discussed in detail in Section 5.3.
5.2 Usability Survey

Nielsen’s [17] heuristic usability evaluation framework captures ten most general principles for interaction design. It is used as a baseline for developing the usability survey. The usability survey tailors the questions from Nielsen’s usability heuristic evaluation to fit into the context of AccessibleToolKit. The questions used in the survey are listed below.

1. Q1: Do you always know what is going on with the app? (Scale 1 - 10, with 1 being never, 10 being always)

2. Q2: Do you feel your user experience with AccessibleToolKit is similar to other interactive spoken dialogue systems, for example, the automated call center of the bank? (Scale 1 - 10, with 1 being least similar, 10 being most similar)

3. Q3: Do you feel the app is intuitive to use? (Scale 1 - 10, with 1 being most disagree, 10 being most agree)

4. Q4: Do you always know where you are in the app? For example, do you know how to go back to the main menu, navigate through the three sub functionalities, or exit the app? (Scale 1 - 10, with 1 being never, 10 being always)

5. Q5: Do navigation choices lead you to the expected task without confusion? (Scale 1 - 10, with 1 being very confusing, 10 being not confusing) If you feel any key words are confusing to use, please indicate which ones and briefly explain why.

6. Q6: Do you feel the instructions of the app are easy to follow, such that you do not have to memorize the navigation steps to use the app? (Scale 1 - 10, with 1 being hard, 10 being easy)
7. Q7: Did the app provide a lot of unnecessary information that did not aid in completing your tasks? (Scale 1 - 10, with 1 being much unnecessary information, 10 being little unnecessary information)

8. Q8: Are the error messages meaningful and helpful for you to correct the errors? (Scale 1 - 10, with 1 being confusing, 10 being meaningful/helpful)

9. Q9: How do you feel about the speed the app talks? (Scale 1 - 10, with 1 being too slow, 10 being too fast)

10. Q10: Any other features you would like the app to provide?

11. Q11: Any other comments?

5.3 Results and Analysis

In this section, the results of the usability study are analyzed. The performance of AccessibleToolKit and the native apps are measured by the time spent on each task and the error count for each task. For AccessibleToolKit, an error is counted if the participant says an unrecognizable key word or the application fails to recognize the participant’s input. For Android native applications, an error is counted if the participant clicked the wrong UI component, or entered wrong information.

5.3.1 AccessibleToolKit

Three study groups participated in the study, as shown in Table 5.1. The usability study of AccessibleToolKit is an iterative process. Study Group 1 includes four fully sighted participants and is conducted the earliest. No performance measurements are taken for Study Group 1. Based on the feedback from Study Group 1, modifications
<table>
<thead>
<tr>
<th>Study group</th>
<th>Study date</th>
<th>Number of participants</th>
<th>Participants</th>
<th>Performance Measurement</th>
<th>Usability Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>5/2/1013</td>
<td>4</td>
<td>Fully-sighted</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Group 2</td>
<td>5/6/2013</td>
<td>2</td>
<td>Visually-impaired</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Group 3</td>
<td>5/11/2013 5/12/2013</td>
<td>9</td>
<td>Fully-sighted</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 5.1: Study group information

are made to AccessibleToolKit. Study Group 2 and Study Group 3 use the newer version of the application for their evaluation. Study Group 2 includes two visually impaired participants. Study Group 3 includes nine fully-sighted participants. Performance measurements are taken for both Study Group 2 and 3.

Figure 5.2 illustrates the usability survey results of Study Group 1. Participants rated AccessibleToolKit 4.4 over 10 for Q1 (Do you always know what is going on with the app?). One major complaint participants have is the error message is too general and not very helpful to correct the error. For example, in the situations that the user’s input is not among the key words AccessibleToolKit can recognize, the error message read to the user is “Sorry, I did not understand your choice, please say again”. Participants are only aware of the error, however they are not provided instructions on how to correct the error. A better error message would be “Sorry, I did not understand your choice. Please tell me the alarm time”. Error messages are critical for the user to understand the current status of the app, and what they are supposed to do next. For accessible applications like AccessibleToolKit, this is especially important as the target users can mainly obtain information through audibles. This finding resulted in major modifications about the error messages in AccessibleToolKit. Other than indicating there is an error, the modified error messages are more specific in telling the user how to change behavior in case of error.
Participants rated AccessibleToolKit 5.8 over 10 for Q3 (Do you always know where you are in the app?). This result shows that the instruction and confirmation messages should better indicate the status of the app. For example, when the user returns to the main menu, the app should read a message like “back to main menu” to let the user know his/her choice is taken and the current status of the app. Another finding is that after a task is performed, the confirmation message should also include information on the status change of the app. For example, when an alarm has been added, the confirmation message could be “Alarm added. Return to the start of the alarm activity”. This message gives the user a better idea on what is happening with the app than simply saying “Alarm added”.

Other modifications include implementing a “repeat” feature, such that when the user does not understand the instruction or does not hear it well, he/she could easily replay the instruction to continue; and changing some of the navigation key words to better indicate the navigation choices.

Figure 5.3 illustrates the usability survey results of Study Group 2. Figure 5.4 illustrates the usability survey results of Study Group 3. Figure 5.5 compares the survey results for all the three study groups. It can be seen that the above modification results in noticeable improvement in user experience. The ratings for Q1 (Do you always know what is going on with the app?), Q3 (Do you feel the app is intuitive to use?), and Q4 (Do you always know where you are in the app/) have increased around 30%. Participants rated Q6 (Do you feel the instructions of the app are easy to follow, such that you don’t have to memorize the navigation steps to use the app?) on average 9 over 10, implying that the information flow in the app is intuitive.

The original design intention is to design the app such that the user experience is similar to other spoken dialogue systems. This may help users to relate to their previous experience with other spoken dialogue systems, such as automated call cen-
Figure 5.2: Study Group 1 (fully-sighted users) survey results
ters, or Google Now. However, all three studies show that users don’t quite feel it is a similar user experience. This can be seen from the score for Q2 (Do you feel your user experience with AccessibleToolKit is similar to other interactive spoken dialogue systems?). Some participants, who had experience with Google Now, or Siri, found the Push-To-Talk interaction mechanism easy to adapt. However, more than half of the participants found the Push-To-Talk mechanism not intuitive to use. They preferred to have the app to be always “listening”, which requires the app to be able to recognizing system-directed input without requiring a button press. This is difficult to achieve because the RecognizerIntent has a silence time length limit and can not be running all the time. As an alternative solution, a demo feature was added to the main menu activity to walk the user through the Push-To-Talk mechanism.

The answers to Q9 (How do you feel about the speed the app talks?) indicate that the visually impaired participants felt the speech rate to be a little fast. However, they also indicated that they might feel the speed is OK once they are used to the workflow of the app. To better compensate different user preference, a setting feature was added to the main menu activity to allow the user adjust the speech rate to be slow, normal, and fast.

Visually impaired participants indicated that one challenge they had when using apps was how to find and launch the app in their phone. To help solving this problem, a static widget with a relatively large size was created for AccessibleToolKit.

Some participants felt that after they became familiar with the workflow of the app, the instructions could be greatly simplified. For example, when prompting the user to choose a field of an event to edit, the instruction is “Please tell me which field to edit. You may say event name, start time, end time, location, description, repetition, reminder, or finish”. This instruction could potentially be simplified to “Please tell me which field to edit.” after the user is familiar with the editing choices.
Figure 5.3: Study Group 2 (visually impaired users) survey results

A future work could be provide different instruction sets for new users and experienced users.

Table 5.2 lists the performance measurement results (time and error count) for Study Group 2. Table 5.3 lists the performance measurement results for Study Group 3. The description of each task is explained in Section 5.1. Generally speaking, the time spent on each task is proportional to the error count and the complexity of the task. Shorter tasks, such as dialing and alarm management can be accomplished in around one minute. Event management tasks takes longer due to the fact that an event needs to get more information to create. This information includes event title, start date, start time, end date, and end time. One suggestion is to have the option of using duration in lieu of end date and end time to simply the input. The usability survey study also collects user input on what other features the visually impaired
Figure 5.4: Study Group 3 (fully-sighted users) survey results

Figure 5.5: Usability survey results
<table>
<thead>
<tr>
<th>Task</th>
<th>Time (second)</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>29</td>
<td>1</td>
</tr>
<tr>
<td>Task 2</td>
<td>63</td>
<td>1.5</td>
</tr>
<tr>
<td>Task 3</td>
<td>64</td>
<td>0.5</td>
</tr>
<tr>
<td>Task 4</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>Task 5</td>
<td>77</td>
<td>1.5</td>
</tr>
<tr>
<td>Task 6</td>
<td>169</td>
<td>3</td>
</tr>
<tr>
<td>Task 7</td>
<td>38</td>
<td>0.5</td>
</tr>
<tr>
<td>Task 8</td>
<td>110</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.2: Study Group 2 performance measurement results - Accessible-ToolKit

<table>
<thead>
<tr>
<th>Task</th>
<th>Time (second)</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>49</td>
<td>2</td>
</tr>
<tr>
<td>Task 2</td>
<td>36</td>
<td>0.6</td>
</tr>
<tr>
<td>Task 3</td>
<td>69</td>
<td>1.125</td>
</tr>
<tr>
<td>Task 4</td>
<td>22</td>
<td>0.125</td>
</tr>
<tr>
<td>Task 5</td>
<td>45</td>
<td>0.625</td>
</tr>
<tr>
<td>Task 6</td>
<td>108</td>
<td>2.125</td>
</tr>
<tr>
<td>Task 7</td>
<td>29</td>
<td>0.375</td>
</tr>
<tr>
<td>Task 8</td>
<td>89</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.3: Study Group 3 performance measurement results - Accessible-ToolKit

users would like to have. This is discussed in detail in Section 6.1.

5.3.2 Native Apps

As a baseline of the performance measurement data of AccessibleToolKit discussed in Section 5.3.1, the performance of native apps is also evaluated with the same set of tasks. The native Phone app is used to evaluate Tasks 1 and 2. The native Clock app is used to evaluate Tasks 3 - 5. The native Calendar app is used to evaluate Tasks 6 - 8.

Study Group 1 was not involved in the evaluation of native apps. For Study
Group 2, both visually impaired participants are not willing to participate in this part of the study. One participant tried to use the native apps before, but felt it was difficult to interact even with Talk Back turned on. Another participant tried to use the Phone app to dial a number, but gave up after multiple failures. According to the experience the participant shared, it was frustrating for them to perform these tasks, as there was not enough instruction from the native app on what they were supposed to do to interact with the app; also there was no error message to help them correct the error. For example, after trying to enter a phone number to dial, the participant found it is difficult to check the number entered and correct errors.

Luckily, some participants in Study Group 3 participated in the native app evaluation. Table 5.4 lists the performance measurement results of Study Group 3. The two dialing tasks take significantly longer time. This is due to the fact the user needs to explore the keypad to find the desired number key to enter each digit in the phone number, which requires more user interaction as compared to Accessible-ToolKit, where the user can input the phone number at once.

For the alarm management tasks, the time spent on each task is also noticeably longer than AccessibleToolKit. A significant amount of time was spent on exploring the screen to find the correct UI components to perform the task.

Participants were able to add an event to the calendar, with around 200% more time spent. However, participants failed to perform Task 7 (check the schedule on a certain day) and Task 8 (delete the added event). These failures are due to the same reason. There is no easy way for the user to tell the date of the calendar. In a calendar week view, when touching each day view, only the day of week, i.e. Monday, Tuesday, etc., is read. Participants had a hard time figure out the date of a day, which resulted in the failure of the two tasks, which requires operation on a certain date.
<table>
<thead>
<tr>
<th>Task</th>
<th>Time (second)</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>300</td>
<td>13</td>
</tr>
<tr>
<td>Task 2</td>
<td>289</td>
<td>8</td>
</tr>
<tr>
<td>Task 3</td>
<td>180</td>
<td>5</td>
</tr>
<tr>
<td>Task 4</td>
<td>109</td>
<td>3</td>
</tr>
<tr>
<td>Task 5</td>
<td>137</td>
<td>4</td>
</tr>
<tr>
<td>Task 6</td>
<td>355</td>
<td>8</td>
</tr>
<tr>
<td>Task 7</td>
<td>N/A (Task fails)</td>
<td>N/A (Task fails)</td>
</tr>
<tr>
<td>Task 8</td>
<td>N/A (Task fails)</td>
<td>N/A (Task fails)</td>
</tr>
</tbody>
</table>

Table 5.4: Study Group 3 performance measurement results - native apps

Since the participants are fully sighted users, they are used to screen gestures in normal mode. The accessible mode uses a different set of screen gestures. There is a learning curve for them to adapt to the new gestures. For example, to interact with the TimePicker, one needs to first hold two fingers on the screen. Scrolling up and down with the finger on the left would change the hour setting. Scrolling up and down with the right finger would change the minutes setting. Another time-consuming part is to enter the event title with the soft keyboard. The participant used voice typing to type. However, when the input is not correctly recognized, it is hard to edit the text. The participant had to delete all the input and type again.

The comparison of performance measurement results between AccessibleToolKit and native apps shows that using voice communication as the user interaction mechanism is more efficient than the Touch-and-Explore mechanism. Especially when the task needs more UI input, using the voice communication mechanism greatly saves input time.
Chapter 6

Future Work and Conclusions

In this chapter future work and conclusions are discussed. Section 6.1 talks about future works. Section 6.2 summarizes the conclusions.

6.1 Future Work

The usability survey of AccessibleToolKit shows that the visually impaired users feel that the interactive spoken dialogues has great potential to be used to as the UI framework to develop accessible apps. They would like to have more features, such as voice text, voice email, voice controlled music play, and reading feeds from news websites. Google Voice Typing is a good start to develop the voice text and voice email features. The users would like to be able to review and edit the input text, and perform commonly used email/text operations with spoken dialogues.

Other potential improvements include:

- Provide different instruction sets for new users and experienced users. Simplify the instruction set for experienced users.
• Support fuzzy match of date input, support phrases such as “next Tuesday”, “tomorrow”, “today”, etc.

• Support fuzzy match for calendar events. Support range-based retrieval for start and end time, and keyword-based retrieval for titles. For example, “meeting” can retrieve both “Meeting with Jack” and “Meeting with Mary”.

6.2 Conclusions

In this thesis, it is proposed to use interactive spoken dialogues as the user interaction framework for developing accessible Android applications. An Android application, AccessibleToolKit, is designed and implemented to realize the idea. AccessibleToolKit helps the user to dial a phone number, manage alarms and the calendar.

Usability study of AccessibleToolKit is conducted to evaluate the user experience. The study is done with both visually impaired users and fully sighted users. Based on the feedback from the study, AccessibleToolKit is modified to provide a more user-friendly experience. The performance of AccessibleToolKit is measured and compared with the performance of native apps. The comparison shows that AccessibleToolKit allows user to perform the commonly used tasks more efficiently and with less frustration. In summary, using interactive spoken dialogue as the user interaction framework is a promising approach to develop accessible mobile applications.
Bibliography


Appendix A

HSC Review Documentation

The following documentation is included in the appendix:

- Usability Study Procedures for Android App - Accessible ToolKit.
- Human Subjects Protocol Approval Form.
- Informed Consent Form.
Usability Study Procedures for Android App - AccessibleToolKit

Goals
The goal of this research is to evaluate the usability of an android application – AccessibleToolKit. AccessibleToolKit is developed to help visually impaired users to use smart phones more intuitively. AccessibleToolKit uses voice communication instead of scream gestures as the user interaction approach. The functionalities of AccessibleToolKit include helping the user dial a number, manage alarms and the calendar.

Participants
The target users are users who are visually impaired and would like to use smart phone on a daily bases. We are planning on contacting the Disability Resource Center, Cal Poly, to recruit participants for the usability evaluation. If it is difficult to find enough participants for the study, we might also recruit students from Computer Science Department for the usability evaluation. In order to minimize the bias as a fully sighted user, we will require fully sighted participants to perform the evaluation tasks without looking at the screen.

Data Collection
- The phone screen will be video taped while the participant is interacting with the app. The participant will NOT be video taped.
- The participant will be asked to perform a set of tasks with the app, such as dial a phone number, set an alarm, set an event in the calendar, etc. The time the participant spends on each task will be recorded.
- The participant will also be asked to perform the same task with the native android app. The time spent on each task with the native android app will be recorded.
- We will also develop a survey to collect users’ feedback on their experience with the app.
- The above data will be collected anonymously.
- If the participant is visually impaired, we will read the Informed Consent form to them before starting the study. We will also read the survey questions to the participant and get their feedback.

Risks
The are minimal risks involved with the usability study of AccessibleToolKit.

Potential Benefits
It is essential that the user interaction design of mobile applications provides equal access and equal service to people with visual disabilities. This study will explore the feasibility and usability of using voice communication as the user interaction approach in android accessible applications, which will benefit both visually
impaired users and fully sighted users as the app allows the user interact with the application without looking at the screen.
All Cal Poly faculty, staff, and student research with human subjects, as well as other research involving human subjects that is conducted at Cal Poly, must be reviewed by the Cal Poly Human Subjects Committee for the protection of human subjects, the researchers, and the University. Human subjects research is defined as any systematic investigation of living human subjects that is designed to develop or contribute to generalizable knowledge. While the ethical guidelines for research are applicable to classroom activities, demonstrations, and assignments, the Human Subjects Committee does not review classroom activities unless data will be collected and used in a systematic investigation.

Researchers should complete all items on this approval form and submit three copies of it, along with three copies of a research protocol (containing the information detailed in Guidelines for Human Subjects Research Protocol), to the Office of Research and Industry Relations (Debbie Hart, Bldg. 38, Room 154). Please feel free to attach an additional page if your responses to any of the items require more space. Your answers to the items on this form, as well as the research protocol, should be typed. The Committee will make every effort to respond to your submission within two to four weeks. Committee approval should be received prior to contacting prospective subjects and collecting data. Please read carefully Cal Poly's Policy for the Use of Human Subjects in Research prior to completing this application.

If you require assistance in completing this form, contact the Office of Research and Industry Relations at (805) 756-1508.

1. Date: April 25, 2013

3. Type of Research:

☐ Senior project
☐ Master’s thesis
☐ Faculty research
☐ Other: please explain

2. Title of Research Project:

Usability Study of Android Application: AccessibleToolKit

4. Name(s) of Researcher(s)

Principal Investigator: Xiaoxi Luo
Department or other affiliation: Computer Science Department
Phone: 240-743-7054 Email: xluo01@calpoly.edu
Position: Faculty x Student
Other: please explain

Additional Researcher: Junxiao Liu
Department or other affiliation: Computer Science Department
Phone: 209-878-7873 Email: Jliu20@calpoly.edu
Position: Faculty x Student
Other: please explain

Any additional researchers involved in the project should be listed with the descriptive information requested above on a separate sheet.
5. **Faculty Advisor** (if applicable)
   
   **Name:** Dr. Chris Lupo  
   **Email:** clupo@calpoly.edu  
   **Department or other affiliation:** Computer Science Dept.  
   **Phone:** (805) 756-5659

   Other thesis committee members if the research is a thesis:
   
   **Name:** Dr. Franz Kurfess  
   **Email:** fkurfess@calpoly.edu  
   **Department or other affiliation:** Computer Science Dept.  
   **Phone:** (805) 756-7179

   **Name:** Dr. David Janzen  
   **Email:** djanzen@calpoly.edu  
   **Department or other affiliation:** Computer Science Dept.  
   **Phone:** (805) 756-2929

6. **Is there an external funding source for the project:**
   
   [ ] Yes, and the source is:  
   [x] No

7. **Is this a modification of a project previously reviewed by Cal Poly’s Human Subjects Committee?**
   
   [ ] Yes, and the approximate date of the last review was:  
   [x] No

8. **Estimated duration of the project:**
   
   Starting date: April 29, 2013  
   Completion date: May 31, 2013

9. **Describe any risks (physical, psychological, social, or economic) that may be involved.**
   
   See Specific Ethical Criterion #1 in Policy for the Use of Human Subjects in Research for a description of the types of risks.

   The goal of this research is to evaluate the usability of an android application – AccessibleToolKit. AccessibleToolKit is designed to help visually impaired users to use smart phones more intuitively. The functionalities of AccessibleToolKit include helping the user dial a number, manage alarms, and manage the calendar. Minimal risks are involved with the project.

10. **Indicate what measures will be taken to minimize risks.** See Specific Ethical Criterion #1 in Policy for the Use of Human Subjects in Research for a discussion of strategies for minimizing risks.
    
    N/A.

11. **Explain how subjects' confidentiality will be protected.** See Specific Ethical Criterion #5 in Policy for the Use of Human Subjects in Research for a discussion of strategies for minimizing risks.
    
    Anonymity of subjects’ responses will be maintained through the usability study.
12. Describe any incentives for participation that will be used. See Specific Ethical Criterion #2 in Policy for the Use of Human Subjects in Research for a discussion of the use of incentives in research.

No incentives will be provided.

13. Will deception of subjects be involved in the research procedures?

[ ] Yes*  
[ ] No

*If so, explain the deception and how it will be handled. See Specific Ethical Criterion #3 in Policy for the Use of Human Subjects in Research for a discussion of the use of deception in research:

14. Type of review requested:

[ ] Exempt from further review*  
[ ] Expedited review  
[ ] Full review

See Types of Review in Policy for the Use of Human Subjects in Research for a discussion of the criteria for exempt, expedited, and full reviews.

*The research protocol submitted for a project presumed to be exempt may be abbreviated but should contain sufficient information to support the conclusion that the project meets the criteria for exemption.

15. Signatures:

Your signature below indicates that the information presented in this application (the approval form and research protocol) is accurate and that you have read, understand, and agree to follow the Policy for the Use of Human Subjects in Research.

Name of Primary Researcher: Xiaoxi Luo

Signature: 

Cal Poly Faculty Advisor's Signature (Required if this is student research)

I have reviewed this research proposal which has been prepared by my advisee(s) in accordance with the Guidelines for Obtaining Human Subjects Approval.

Name of Faculty Advisor: Dr. Chris Lupo

Signature

Return to the Human Subjects Committee homepage.
Informed Consent Form

INFORMED CONSENT TO PARTICIPATE IN A RESEARCH PROJECT:

Usability Study of Android Application: Accessible ToolKit

Masters thesis research on the usability of the AccessibleToolKit is being conducted by Xiaoxi Luo and Junxiao Liu, graduate students in the Department of Computer Science at Cal Poly, San Luis Obispo, under the direct supervision of Dr. Chris Lupo.

You are being asked to take part in this study by using both the AccessibleToolKit and the native android app to dial phone numbers, and manage alarms and a calendar. Afterwards you will be asked to complete a usability survey. While you are interacting with the apps, the screen of the phone will be video recorded along with the time you required to complete each task. Your participation will take approximately 30 minutes. Please be aware that you are not required to participate in this research and you may discontinue your participation at any time without penalty. You may also omit any questions on the survey that you would prefer not to answer.

There are no risks anticipated with participation in this study. Your participation will be confidential and your name will not be used in any reports of this research without your permission. Potential benefits associated with the study include exploring the feasibility of using voice communication as the user interaction approach in android accessible applications, which allows visually impaired users to use smart phones in a more intuitive way.

If you have questions regarding this study or would like to be informed of the results when the study is completed, please feel free to contact Xiaoxi Luo, Junxiao Liu and/or Dr. Chris Lupo at (805) 756-5659. If you have concerns regarding the manner in which the study is conducted, you may contact Dr. Steve Davis, Chair of the Cal Poly Human Subjects Committee, at (805) 756-2754, sdavis@calpoly.edu, or Dr. Dean Wendt, Interim Dean of Research, at (805) 756-1508, dwendt@calpoly.edu.

If you agree to voluntarily participate in this research project as described, please indicate your agreement by signing below and completing the usability evaluation tasks and the survey. Please retain a copy of this consent form for your reference, and thank you for your participation in this research.

______________________________________________     _____________________________
Signature of Volunteer                                                      Date

______________________________________________     _____________________________
Signature of Researcher                                                      Date