SWAT: Mobile System-Wide Assistive Technologies

André Rodrigues
LaSIGE, Department of Informatics,
University of Lisbon
Campo Grande, C6 Piso3, 1749-016,
Lisboa, Portugal
andrefprodrigues91@gmail.com

Tiago Guerreiro
LaSIGE, Department of Informatics,
University of Lisbon
Campo Grande, C6 Piso3, 1749-016,
Lisboa, Portugal
tjvg@di.fc.ul.pt

Mobile operating systems have evolved to provide increasing accessibility capabilities. However, mobile application developers are still restricted to deploy custom-made accessible applications or to extend limited and stereotyped accessibility services. In this paper, we present SWAT, an extensible framework that provides system-level content and event information to application developers. Its use is demonstrated in a multi-impairment case study achieving automatic row-column scanning with audio feedback. SWAT presents strengths usable in several other system-wide contexts that empower developers and users: adaptation, logging, testing and simulation.

Mobile Accessibility, Assistive Technologies, Extensible, Framework, I/O.

1. INTRODUCTION

Smartphones, powerful computers as they are, have long surpassed their desktop counterparts in usefulness and efficiency. They are not mere communication devices. They are our personal assistants, a handy way to connect to social media, a multimedia entertainment system and much more. Indeed, they are also, and increasingly more so, the prevalent communication artifact at one's disposal. Not being able to use a smartphone can be disadvantageous at several levels, with social exclusion being one of its greatest and most daunting consequences. Acquiring access to such devices can empower their users in so many ways going beyond the frontiers of the device (e.g., controlling other devices). Once with limited resources and restricted to their bulky physical interface, mobile devices are now coupled with a set of input, output and communication capabilities that potentially allow for the adaptation to their users' capabilities and needs. Conversely, the complexity of mobile user interfaces competes with the aforementioned potential damaging the accessibility of these platforms. Currently there are two major aproaches towards mobile accessibility. Creating a custom-made application or resorting to a system-wide solution.

1.1 Custom-made applications

In the quest for more accessible interfaces, there are applications that seek to replace all the look and feel of the device. This is the case of applications like Mobile Accessibility[1] or the one reported by Nicolau et al [7]. These type of solutions have the advantage of being tailored specifically for a target audience, which guarantees to a certain degree an improved user experience to those that fit the aimed stereotype. However, they lack the extensibility and adaptability to cover users with a different set of (dis)abilities. By creating a single or a set of applications specifically designed for a group of users we are consciously neglecting all other users. Developing applications to users with different abilities from scratch is a costly and time consuming process. Furthermore, in a time where new applications are available every day, creating custom-made access tools is too restrictive for the user, as we need to develop novel versions of the access tool every time a new functionality is desired.

1.2 System-Wide solutions

The advent of Apple's iPhone is a relevant mark in the road towards more accessible systems. Since then, the major mobile operating systems have started taking advantage of the device's potential and have been coupled with successful accessibility features (e.g., VoiceOver[2], Zoom[3], TalkBack[3],

[1] http://www.codefactory.es/en/products.asp?id=433, (Last Visited on 07/03/2014)
[2] https://www.apple.com/accessibility/ios/ , (Last Visited on 07/03/2014)
[3] http://developer.android.com/design/patterns/accessibi lity.html , (Last Visited on 07/03/2014)

Proceedings of BCS HCI 2014 - Sand, sea and Sky - Holiday HCI, Southport, UK

Switch Control[4]). They are characterized by providing different ways for people with disabilities to navigate and use their devices. As an example, VoiceOver allows a blind person to painlessly explore the screen by touching it. Selection is made by double-tapping or splitapping, or by releasing the finger from the screen in advanced mode. Switch Control, by turn, allows the parameterization of a scanning mechanism to be paired with external switches. TalkBack, on the other hand, is the default Android screen reader that works similarly to VoiceOver. Despite their success and possible parameterizations, these services are heavily stereotyped making assumptions about their possible users. They tend to focus on one kind of disability which leads them to neglect users with multiple disabilities.

1.2.1 Android Accessibility Services

Towards higher flexibility, mechanisms for developers to create their own system-wide accessibility services are also available (e.g., in Android 4.0+ systems, one can implement an accessibility service and adapt the way in which the content is presented to the user. TalkBack is an example of an accessibility service).

Accessibility services can access the onscreen view hierarchy, accessibility events and perform system wide actions through the accessibility APIs (e.g. back, click an icon). They operate in between the application and system levels, and allows developers to be aware of system-wide context changes (i.e. changes that trigger accessibility events). As an example, every touch triggers an accessibility event; all the registered accessibility services will be notified of the event. Through this event, we are able to access all the view objects.

One service that takes advantage of these capabilities is *JustSpeak* [8]. It is an Android accessibility service that allows users to control the device via spoken commands. Using the information of the onscreen content, users are able to launch applications and activate onscreen controls system-wide. One interesting feature is the ability to queue multiple commands with a single sentence.

1.2.2 Android Input Methods

Input Methods can go beyond text-entry and provide control over screen navigation, as *TeclaAccess*[5] on Android. *TeclaAccess* allows a four way (i.e. up/down/left/right) system-wide navigation.

1.3 Limitations

One major issue with accessibility services is that they are still largely unexplored. This leads to a lack of support for the creation of technologies that take full advantage of these services' capabilities. Accessibility services are difficult to master and are mostly focused on the contents presented: the relationship between content and input control is overlooked.

TalkBack is an Accessibility Service with a lack of control over input. It is mostly a single touch interface with a handful of pre-conceived gestures that trigger special events. One problem is that the set of gestures we are able to detect through an accessibility service is predefined. As such, it restricts the developer's options when it comes to input. Developers are not able to create multi-touch accessibility services at will. As an example, why should not developers be able to create a multi-touch screen reader? A screen reader that could simultaneously read multiple selections. Or create a system-wide gesture recognizer? This limitation is derived from the lack of system-wide control over the system internal virtual devices (e.g. touch screen, gyroscope, keypad).

Although accessibility services provide developers with a deeper level of system-wide permissions it is still insufficient. As another example, although several adaptation models for touch screens have been motivated [5] and devised (e.g., [3, 6]) these limitations make it impossible for them to be deployed system-wide in a mobile device. We are not able to monitor any of the system internal devices on a system wide basis. Which means we cannot adapt events into these devices jeopardizing the aforementioned potential of personalization and adaptation.

Another limitation of the current accessibility APIs is the complexity when trying to develop any kind of assistive technology. For example, to access the current on screen content, the accessibility service created will receive an accessibility event. Through this event, we are able to get the source node, through this node we are able to get both its parent and its children. By navigating through these nodes we are able to create the full hierarchical view of the screen. This type of information should be easily attained through the APIs and not through a multiple step process.

Input Methods can be used to create system-wide navigation controllers but their usage is limited to navigating (4-way joypad) between individual items. They lack customization and adaptability and no information about the content being navigated is provided. To use input methods as a navigating mechanism (e.g. *TeclaAccess*) we need

[4] http://support.apple.com/kb/HT5886 , (Last Visited on 07/03/2014)
[5] http://komodoopenlab.com/tecla/ , (Last Visited on 07/01/2014)

applications to comply with accessibility norms, which is more often than not, not the case.

1.4 Overview

In this paper, we outline the limitations of current mobile operating systems regarding flexibility and extensibility to suit developer and user requirements. We then describe the pillars of our framework, SWAT, and how they tackle these limitations.

The amount of information acquired both at the presentation and input level along with the capability to simulate and inject events into the OS pave way for several other scenarios where SWAT can be useful for application developers and researchers. We proceed discussing the following promising applications of SWAT:

- Assistive Technology
- Assistive Macros;
- User Testing and Simulation;
- Interface Adaptation;
- Interaction Logging;

A preliminary validation of the potential of SWAT is presented through a case study: providing system-wide access in an Android device to a person who is blind, tetraplegic and faces severe speech impairments. With SWAT, this control was achieved with the creation of an alternative scanning-based navigation method; selection is performed with an external mouse pointer acting as a single switch and feedback is given through Text-to-Speech[6].

2. SYSTEM-WIDE ASSISTIVE TECHNOLOGY

Despite the efforts presented in the most recent systems, particularly on Android, developers are restricted either to develop at an application level, leading to custom-made applications that are hardly extensible, or to extend Accessibility Services or Input Methods.

2.1 SWAT design

SWAT was developed for the Android platform due to its open source nature and system wide capabilities. It was developed with a single goal in mind, to provide system-wide control over input and output systems in order to facilitate the creation of assistive technologies. In order to achieve our goal we focused in creating a Framework that would tackle the most severe limitations:

- Lack of control over system internal devices;
- Lack of adaptability (i.e. single user group focus);
- Expensive and restrictive (i.e. custom made solutions);
- Time consuming development;
- System-wide:
    - Content control;
    - Navigation control;

SWAT works as an Accessibility Service in order to have control over current onscreen content. It requires a rooted Android smartphone in order to access low-level events and functionalities (i.e. to have control over the system internal devices system wide). With this access, SWAT enables developers to monitor/block/inject low level events from/into the internal devices (e.g. touch screen, keypad) and provides an API where this usage is greatly facilitated.

With these two mechanisms, we created a framework that provides system-wide access to all content and events combined with the control over the system devices. This is the basis for the creation of system-wide solutions.

SWAT is designed in a way that can be used by extending it or by listening to its broadcasted events. This enables developers to develop based on the framework, extending its contributions, or to develop outside the framework, using it as a tool. We provide a couple of interfaces that ensure full and easy control of the system. By extending these interfaces developers are able to customize and adapt their solutions.

At its core it is an extensible framework that strives to allow complete control over input and content. With this goal in mind, SWAT is divided in two key modules.

- Activity Manager - handles all of the content information provided by the accessibility service (current screen content, notifications);
- IO Manager - handles all the smartphone's internal devices (e.g., touch screen, keypad, gyroscope, accelerometer); it is responsible for managing, monitoring, blocking, creating devices and for injecting events into them;

Using the Activity Manager, developers are always able to access the onscreen content. It provides information about each content node gathered through accessibility events. The monitored nodes are the ones that provide key information about the screen state and hierarchy (i.e. describable nodes, clickable nodes, lists).

The IO Manager provides information beyond what is achieved with a simple Accessibility Service. It

[6]http://developer.android.com/reference/android/speech/tts/TextToSpeech.html , (Last Visited on 28/03/2014)

gives developers the opportunity to manage low level events in a way that would not be possible otherwise.

3. APPLICATIONS

SWAT paves way for several different applications and usages:

3.1 Assistive Technology

The extensibility of SWAT interfaces enables developers to quickly create their own assistive technologies. We provide several key interfaces that make it possible. A control interface that handles all the navigation commands (e.g. navigate to next, focus node, click node, back). Three receiver interfaces to get content/input/notifications updates. The case study described in section 4 was developed by implementing the aforementioned interfaces.

The framework features an extensible keyboard that can be personalised to suit each of the user needs (in the case study described, an auto-navigated keyboard was created with a different key layout).

Several control interfaces were designed through the development of SWAT. One of them, the Touch Controller, enables the navigation of nodes on the device through two simple gestures (i.e. slide to navigate to next node, tap to select node). Wi-Fi Controller module that enables developers to send navigation and touch commands to their created control interfaces.

SWAT is meant to empower assistive technology developers with a framework that allows access to new features in an easy and structured way.

3.2 Assistive Macros

With SWAT, we developed a Macro recording application. Through the accessibility service information we are able to figure out in which clickable item the user clicked and record the sequence of navigation steps which we are later able to reproduce. Only using the accessibility service would not allow us to reproduce exact user interactions (i.e. swipes, touches, gestures). With the I/O monitoring capabilities we are able to record user interaction at low-level which allows us to reproduce exact interactions. Macro creation is a combination of both: we are able to record navigation steps (i.e. clicks) and we are able to record touch interactions simply by changing the current recording mode. After recording the Macro the respective icon appears on the Home screen.

Some interactions with mobile devices can be really complex (e.g., multi-touch gestures) and composed of multiple actions and screens. People with impaired dexterity can face problems in coping with the requirements of these interactions. Assistive macros enable other people to record these actions and create shortcuts that the person can easily select. Several steps actions and multi-touch gestures can be reduced to a single step, selecting the shortcut. We envision this to be an especially important contribution to caregivers as they can create one step Macros of complex actions that are usually out of reach for the main users like the case of Miguel which is described in section 4.

3.3 User Testing and Simulation

The popularity of mobile technologies and interactions has led to an increase of app development and HCI research for these devices. Despite the prevalent nature of these technologies, the platforms are lacking in guidance and tools to support testing and simulation of user interactions. SWAT can capture the low level interactions, and replay them through the interface (i.e. macros of touch interaction) which would allow the creation of similar systems to WebAnyWhere ABD [1]. This system permits the recording of accessibility problems at the time the user experiences them. This, in conjunction with the ability to reproduce the problem, can provide a powerful tool for testing and simulation.

3.4 Interface Adaptation

Through SWAT, we are able to monitor events while blocking their respective action; in conjunction with a virtual adaptation driver we are be able to perform on the fly event adaptation. Examples are using touch models system wide or adapting primitive recognition requirements to each user (e.g. touch timeouts).

The ability to access on-screen content allows us to gather crucial information about its objects and properties. We can acquire all the required information in order to adapt the display resorting to overlays. As an example, this enables adaptation of colour, item size or location.

With a refined control over input and a structured control over content we are able to create filters that allow dynamic customization of screen presentation. One simple example is adapting item order according to usage statistics which can be a useful feature for slower scanning method users.

Supple [3] is a system developed with the purpose to adapt UI controls on runtime in order to provide a personalised view of the contents. The system allows users with motor impairments to have their interface adapted accordingly with their specific needs. With SWAT, we will be able to provide a similar solution on a mobile device which we believe can drastically improve user performance over the default non-customizable interfaces.

3.4 Interaction Logging

Being able to record interactions system wide is a key feature to provide researchers with the capabilities to do repeatable HCI evaluations. Input observer [2] is a system that quietly observes user interaction with a personal computer. With SWAT, we are able to provide a similar system that allows developers to focus on the development of their applications and delegate interaction logging to our system

4. A MULTI-IMPAIRMENT CASE STUDY

Accessibility in mobile phones is still an issue for people with disabilities. The challenge in doing so is exponentially higher when the target audience gathers multiple severe impairments. Accessibility features like TalkBack are successful for a large amount of blind people; however, they do very little for a blind and physically impaired one.

Miguel is a 35 year old person who challenges himself daily to remain socially active despite his condition. Due to an accident at the age of 21, Miguel has tetraplegia. He only shows residual arm and neck movement. The accident caused both blindness and a speech impairment that makes him stutter. Due to his condition, Miguel is unable to control his mobile phone. Currently available accessibility solutions are not adaptable enough to support his specific case. Blindness and the tetraplegia turn him unable to use solutions like TalkBack. He is not able to control a pointer or use a keyboard; his options are rather limited. Speech could be the solution but the stutter makes it difficult if not impossible. Miguel is able to press a large and sensitive switch when with arm support.

This was the case study of a previous paper [5], where the solution was to create a custom-made Android application (easyPhone) that would replace the device interface. The application was created with the goal to enable performing simple tasks (make calls, clock, battery, missed calls and messages received). To do this, it resorted to the usage of an automatic scanning technique through a set of predefined options and a switch (mouse). This study revealed several current mobile OS shortcomings in respect to “system-wide” accessibility options, at both a user and developer level.

One of the first goals of SWAT was to be able to create a solution that provided the same features as easyPhone without resorting to the creation of a custom made application. SWAT allows Miguel to use the Android default applications (e.g. contacts) with the same easiness of usage of a custom-made one. To do so, a control interface was created for Miguel. To create it we used both content information from Activity Manager and the monitoring and blocking capabilities of the IO Manager. With this a simple row-column navigation over the content was developed (Figure 1). Control is achieved through an external mouse controller used as a single switch. With such adaptation, Miguel is able to navigate from the Android home screen to any other screen or application. To his benefit, the screens were populated with the applications, and just those, that he desired to use. Notifications and incoming events are also received and acted upon by Miguel.

One problem we face with SWAT is that we rely on the describable content of the screen to navigate through it. This means we navigate through all information that is describable. The problem is if crucial contents lack description or if the application is overloaded with describable contents.

SWAT is currently able to provide all the features of

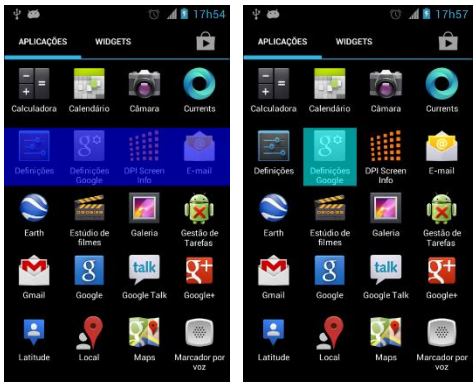

Figure 1. Row-Column Scanning over the Android Applications screen

easyPhone with the added potential of expanding its use to a system-wide navigation giving Miguel the opportunity to use any installed application or any other that he desires to install.

5. CONCLUSIONS

SWAT was motivated by the case of Miguel, a young man with severe multiple impairments. Current operating systems accessibility features are too restricted to deal with such complexity. His case lead us to the development of an extensible framework that allows for full system-wide control over content being presented and I/O events. This framework paves way to system-wide control in novel contexts like interface adaptation, logging and simulation.

6. ACKNOWLEDGMENTS

This work was supported by FCT through funding of LaSIGE Strategic Project, ref. PEst-OE/EEI/UI0408/2014. We would also like to thank Miguel and his caregivers for their participation.

7. REFERENCES

[1] Bigham, J. P., Brudvik, J. T., & Zhang, B. (2010). Accessibility by demonstration: enabling end users to guide developers to web accessibility solutions. Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility, 35–42.

[2] Evans, Abigail, and Jacob O. Wobbrock. "Input observer: measuring text entry and pointing performance from naturalistic everyday computer use." CHI'11 Extended Abstracts on Human Factors in Computing Systems. ACM, 2011.

[3] Findlater, L., & Wobbrock, J. (2012) Personalized input: improving ten-finger touchscreen typing through automatic adaptation.Proceedings of CHI 2012, 815-824.

[4] Gajos, Krzysztof, and Daniel S. Weld. "SUPPLE: automatically generating user interfaces." Proceedings of the 9th international conference on Intelligent user interfaces. ACM, 2004.

[5] Guerreiro, T., Nicolau, H., Jorge, J. and Gonçalves, D.. 2010. Assessing mobile touch interfaces for tetraplegics. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 31-34.

[6] Montague, K., Hanson, V. L., and Cobley, A.. 2012. Designing for individuals: usable touch-screen interaction through shared user models. In Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '12). ACM, New York, NY, USA, 151-158.

[7] Nicolau, H., Guerreiro, J., Guerreiro, T., Stressing the Boundaries of Mobile Accessibility, In CHI 2013 Third Mobile Accessibility Workshop, Paris, France (2013), arXiv preprint arXiv:1402.1001.

[8] Yu Zhong, T. V. Raman, Casey Burkhardt, Fadi Biadsy, and Jeffrey P. Bigham. 2014. JustSpeak: enabling universal voice control on Android. In Proceedings of the 11th Web for All Conference(W4A '14). ACM, New York, NY, USA