CSE 491: Accessibility Usage in GooglePlay

Junyu Yan
Miami University
Oxford, OH
yanj10@miamioh.edu

***Abstract*—Accessibility design in Universal Design has a guiding role in improving user readability and comprehension. Especially for some people with disabilities, the lack of accessibility design can lead to operational limitations and inconvenience. A preliminary empirical study was recently published showing that open source Android applications hosted on GitHub have a poor use of Android's accessibility API. However, it is unclear whether the findings of this study apply to GooglePlay as well. Due to the lack of research on the accessibility of applications on the GooglePlay platform, this article mainly analyzes the applications published on GooglePlay to make up for this lack. This study implemented a tool to automate decompilation of a given APKs to extract the results of the call to the accessibility API and the provided results of the assistive content description. Descriptive statistics were then used to summarize the results.**

***Index Terms*—Empirical Studies, Mobile Accessibility, Universal Design**

I. Introduction

Mobile application development has become the most important aspect of the digital ecosystem as they design and develop the most popular and useful user applications. Gaming apps are expected to reach $105.2 billion in revenue by 2021 as usage surges [1]. With the tremendous success of this application economy, open-source developers are flocking to these markets. In 2018, Github had over 750K Android-related open-source libraries with a total of 37K contributors [1]. However, to survive in a competitive market, the design principle and user experience of the software are the decisive factors in this battle.

Competitive software should be developed in accordance with the terms of the universal design principles, in order to improve the use of the application satisfaction, so that users can be used in the process of convenient operation and comfort. The UD Principles state that software should be as open as possible to everyone (whether they have a disability or not). In the whole world, more than 1 billion people (15%) worldwide live with some form of disability [2]. In the United States, 26 percent of adults have some form of disability, according to the CDC's most recent statistics on disability and health [3]. Among these functionally disabled adults, 5.9 percent were deaf or had severe hearing impairment, and 4.6 percent had vision impairment, blindness or severe vision impairment even with glasses, the CDC reported [3]. In order for people with disabilities to use applications that operate like a healthy person, software developers should provide compatibility with technologies or devices that are used by people with limited senses to enhance the disabled experience.Accessibility design principles provide drivers and guidelines for designing services and products for different populations (children, the disabled, the elderly, etc.).

A preliminary empirical study [4] was recently published that looked at open-source Android apps hosted on GitHub, but the main of that work was a qualitative study on Stack Overflow. While that work demonstrated many apps did not provide assistive content descriptions for all of their GUI elements, it is not clear to what extent these are representative of commercial applications available on Google Play. The proposed research project bridges this gap by analyzing thousands of Android apps. The main outcome of this proposal is the development of a tool that given some apk will decompile it and extract the accessibility APIs and whether the developers provided assistive content descriptions. Subsequently, a high-level analysis (e.g., descriptive statistics like mean and median) of the prevalence of the accessibility features will also be produced.

In this study, we found that not all applications hosted or distributed on GooglePlay meet the UD. 10.1% have no GUI component assistive content at all. Of the remaining 89.9% of applications, 8.86% had no assistive content description. In terms of accessibility API imported, 4.04% of applications do not introduce any accessibility API. We also compared the data from this study with data from a preliminary study of Android apps on GitHub, and we found that the projects on GooglePlay are more focused on accessibility design, which we believe is related to Google governance.

II. Related Work

Previous empirical studies [4] explored the accessibility of android applications on GitHub: the researchers built a lightweight tool from a random selection of 13,817 unabandoned and unbranched android applications on GitHub to assist in static analysis of the APIs and assistive content descriptions of GUI components applied in the application source code. For the helper tools created in this study, we used the same ideas with this preliminary study [4]: 1). Identify the UI component. 2). Examine the assistive content description in the component. 3). Identify the dynamic markup of the assistive content description element.

This previous study [4] showed that in a dataset of 13,817 random projects, the accessibility API was imported at a fairly low rate, with only 2.08% of applications importing at least one accessibility API. These APIs have imported a total of 346 times. Of the imported APIs, the author [4] mentioned that android.view.accessibilit.AccessibilityEvent was called the most, 286 times in total. For the assistive content of GUI components, only 50.08% of android applications have assistive content at development time. However, in 46.25% of these applications with assistive content, there is no description of assistive content.

The above motivational study [4] found that about half of the projects on Github provide assistive content, but the use of accessibility APIs is relatively low. The authors point out that developers do not use APIs for specific functional implementations as their primary goal in the development of project accessibility, but rather prefer to move toward this mechanism. The purpose of this study aims to complement the dataset of the above research to more fully explore developers' attitudes towards accessibility design and their experiences and lessons in the implementation of Universal Design Principles.

III. Empirical Study

The purpose of this study is to make up for the missing data of the application project on GooglePlay on the basis of the previous research and to provide a data conclusion. At the same time, we wanted to see if the open-source projects on GitHub would be more focused on developing projects that are accessible to everyone, especially users with disabilities, than projects under Google's supervision. This study USES the same ideas and framework as the previous study in the static analysis of applications, and focuses on the following two aspects:

RQ$_1$ What is the degree to which an application published on Google play is accessible?

RQ$_2$ How does GooglePlay's research differ from open source projects?

In order to answer the research questions, we created a lightweight tool to assist us in producing research data, as did in previous research. Then, we utilized a dataset of Android APKs [5]. This dataset consists of the top 240 applications across 39 categories. From this dataset, we randomly sampled 100 apps as the context for this study. Of these apps, the tool was unable to fully decompile one of them and so we excluded it from our study. After creating the resulting data set, we will compare the data from previous experiments with the data from this experiment and analyze the differences between open-source projects and those on GooglePlay.

IV. Design of Approach

In order to answer **RQ**$_1$, we extended the lightweight tool from the prior work [4] to analyze APKs. Figure 1 is a flowchart of the runtime for this lightweight tool. First, we built a component that reiies on Apktool [6] to reverse engineering the Android apps. From this process, the component is able to extract the relevant resouce files from *res/layout*.

It then iterates over each node in each XML files, searching for nodes with the names of the nodes imageButton and imageView. If a node exists, the tool extract the node's ID and contentDescription value. Addtionally, we count the number of total nodes and target nodes in all XML files for the current APK We utilize this information to understand the proportion of GUI elements that are utilizing assistive content descriptions. The next component extracts the dex files from APKs. It utilizes dex2jar [7]to convert these dex files into Java archives containing class files, which maintains the original layout of the source code. Subsequently, the tool extracts the class files from these Java archives. Wth the class files, the tool utilizes Java Decompiler (JAD) [8] to decompile each class file generating the corresponding Java source code file. The decompliation stage of this component preserves the overall structure of the application.

With the source code extracted from the APK, the final component relies on the Eclipse JDT [9] to statically analyze the source code. By analyzing the Abstract Syntax Tree, the tool is able to identify method invokations related to setting assistive content descriptions, and it can identify the various Android Accessibility APIs that were imported. Specifically, we extracted the *ImportDeclaration* and *MethodInvocation* nodes. In order to determine whether an Android Accessibility API was used, it checked the *ImportDeclaration* nodes against the APIs for the android.accessibilityservice and android.view.accessibility packages. For the *MethodInvocation* nodes, the tool identified the usage of *setHint()* and *setContentDescription()*, which are programmatic ways to set assistive text for GUI elements.

Lastly, the tool generates two reports: i) a report for the APK withe percentage of assistive descriptions for GUI elements and the imported Accessibilitiy APIs, and ii) a report aggregating the data for all of the APKs in the dataset (evaluating the descriptive statistics).

V. Results

A. Accessibility in GooglePlay Apps

After analyzing the 99 randomly selected applications on the GooglePlay, we found that 89 applications (89.90%) had assistive content (*imageView, imageButton*) for at least one GUI element. As shown in figure 4 ,of these 89 applications, 79 (88.76%) had assistive content descriptions that were not all empty, and seven (8.86%) had more than 50% coverage of assistive content descriptions. While there is no application had assistive text for all GUI elements and there is only one application had non-empty assistive text description for all assistive text.

Figure 3 and Figure 2 shows the histogram and descripyive analysis of the percentage of assistive content in Android application. On average, the apps had assistive text for 5.88% of GUI elements and a median of 5.707. The maximum percentage of GUI components that have descriptive content is 22.08%.The minimum percentage of GUI components that have descriptive content is 0.00%.

In terms of accessibility API usage, we observed that out of 99 random apps, 95 android apps(95.96%) imported at least one accessibility API. Of the 95 applications, 1 app imported 1 accessibility APIs, 1 app imported 4 accessibility APIs, 4 apps imported 5 accessibility APIs, 21 apps imported 6 accessibility APIs, 8 apps imported 7 accessibility APIs, 47 apps imported 8 accessibility APIs, and 13 imported 9 accessibility APIs. The times of imported for each accessibillity

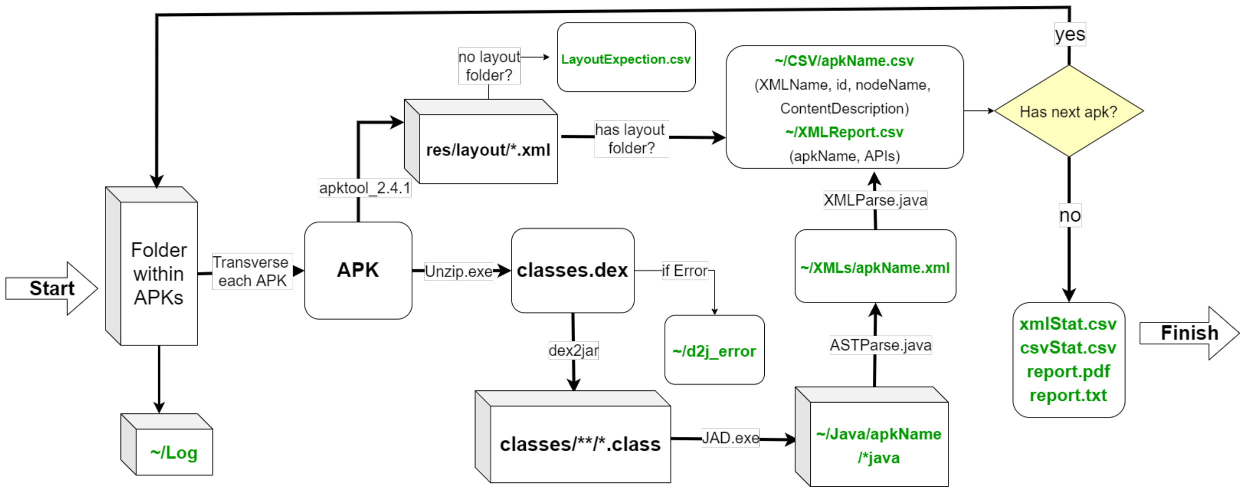

Fig. 1. Lightweight tool flow diagram

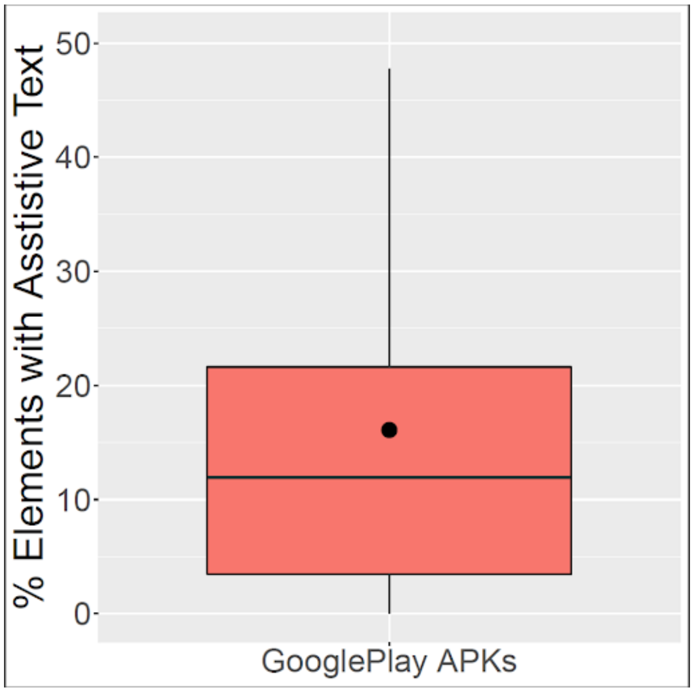

Fig. 2. A descriptive analysis boxplot of the percentage of assistive content in Android applications

APIs are shown in TABLE I. In total, the accessibility APIs are imported 700 times. Be imported most three accessibility API is *android.view.accessibility.AccessibilityEvent* (95 imports), *android.view.accessibility.AccessibilityNodeInfo* (94 imports) and *android.view.accessibility.AccessibilityRecord* (92 imports). All 95 Android applications which imported at least one accessibillity APIs in our data set are imported the *android.view.accessibility.AccessibilityEvent*.

B. Comparison to Open Source

By operating the above lightweight tool to analyze the sample dataset, we found that android applications on the GooglePlay platform invoke accessibility design more than open-source applications on GitHub. As for the assistive content of GUI components, only 50.08% of the android open-source applications on GitHub have assistive content, and a surprising 46.25% of the GUI components with assistive content do not have any descriptive information. However, nearly 90% of the project samples on Google play had assistive content, and about 90% of the projects with assistive content described above had non-empty assistive content descriptions. GooglePlay has significantly more accessibility API calls (95.96%) than GitHub's android application, where only 2.8% of the sample has at least one reference to one accessibility API.

TABLE I
NUMBER OF IMPORTED ACCESSIBILITY APIS

Accessibility APIs	Times
android.view.accessibility.AccessibilityEventSource	1
android.view.accessibility.CaptioningManager	28
android.view.accessibility.AccessibilityWindowInfo	53
android.accessibilityservice.AccessibilityServiceInfo	67
android.view.accessibility.AccessibilityManage	88
android.view.accessibility	91
android.view.accessibility.AccessibilityNodeProvider	91
android.view.accessibility.AccessibilityRecord	92
android.view.accessibility.AccessibilityNodeInfo	94
android.view.accessibility.AccessibilityEvent	95
total	700

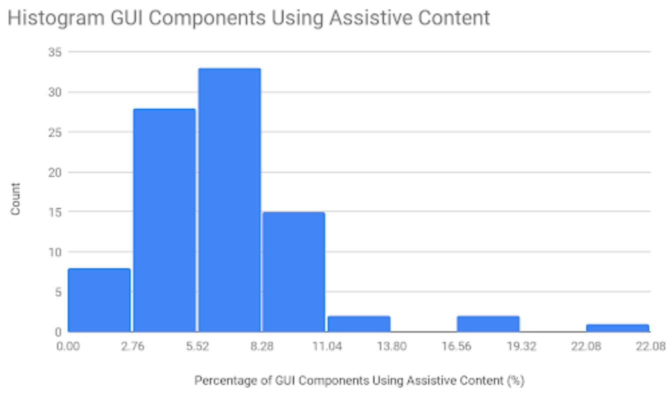

Fig. 3. Histogram of GUI Components Using Assistive Content

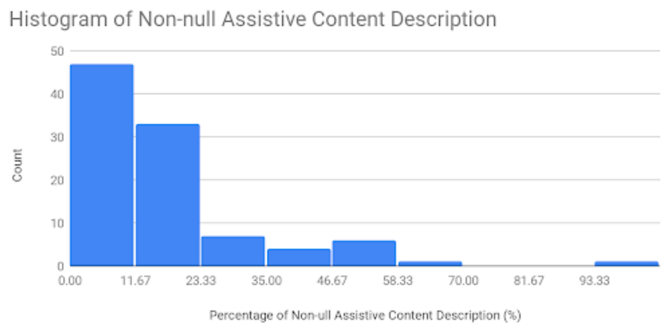

Fig. 4. Histogram of Non-null GUI Components Using Assistive Content

While we don't delve into whether developers are actually using accessibility APIs to achieve functionality that benefits users with disabilities, we can argue that applications published on GooglePlayare more conscious of considering and using accessibility APIs during development. While the degree of completion of the secondary content description is low and not all projects refer to the accessibility API, it is reasonable to think that the developers of GooglePlay are more concerned with the actual operation of the accessibility design than the open-source software on GitHub.

As for the large difference in research conclusions between the two platforms, it may be related to Google's regulation of Android applications. Google encourages developers and publishers to consider accessibility when designing and building products and provides detailed accessibility guidelines for android developers [10]. Developers are implicitly being asked to develop products that take more account of the experience of people with disabilities.

Not only that, but Google also has strict regulations on the abuse of accessibility APIs. In a press release in 2017 [11], Google stated that applications that use Accessibility Services should use the system only if it directly benefits people with disabilities. Developers were told they needed to explain how using the service would benefit those users, and if they did not meet Google's requirements, their applications would be removed from the Play store [11]. Google's announcement highlights both the need for developers to be more careful about referencing accessibility APIs and the concern for people with functional disabilities. Therefore, we are in favor of thinking that Google's official oversight is one of the reasons why android apps on GooglePlay are more accessible.

VI. Threats to Validity

The sample size of this experiment is small because the sampling error will be larger when the previous studies are compared.

The tool applied to this study is based on a Java implementation, and we used the *getRuntime()* in the program interacts with the system command line. The tool decompiles and analyzes the files in the APK folder one by one. In this process, because the local system is available for the buffer pool provided by the standard input and output, the fast write to the standard output and the quick read from the standard input can cause the child process to lock or even deadlock.

The lightweight tool designed in this study does not ensure the complete recovery of some apk files during decompilation, which may lead to a certain deviation of experimental data conclusions. For example, in our study, we found that some apk did not have the res/layout folder that we needed to parse after the first decompile. At the same time, our tool decompresses dex files with dex2jar, and for some APKs, an error detail package is generated when decompiled, so some APKs may not be fully decompiled.

VII. Conclusion

The purpose of this study is to make up for the data missing from the previous study for applications on the GooglePlay platform. By building a lightweight tool to decompress, it analyzes the extent to which android programs on GooglePlay use accessibility design and makes descriptive statistics on the research data. Finally, it compares the extent to which open-source software on GitHub USES accessibility design. Our research data show that:

The motivational study of the 99 random Google play apps mentioned above, we found that almost 90% of the apps on Google play provided assistant content, and more than 95% of the projects imported accessibility APIs. Projects on Google play are more conscious of providing accessibility support in their projects than in the main research.

By comparison with the open-source applications on GitHub, we believe that the applications on GooglePlay make more reference to the accessibility API and implement the assistive content of GUI components. However, because the experimental sample of this study is relatively small, larger samples should be used to further demonstrate the experimental conclusion.

References

[1] J. Businge, M. Openja, D. Kavaler, E. Bainomugisha, F. Khomh, and V. Filkov, "Studying android app popularity by cross-linking github and google play store," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 287–297.
[2] "Disability." [Online]. Available: https://www.who.int/health-topics/disability#tab=tab_1
[3] "Disability impacts all of us infographic," Sep 2019. [Online]. Available: https://www.cdc.gov/ncbddd/disabilityandhealth/infographic-disability-impacts-all.html
[4] C. Vendome, D. Solano, S. Liñán, and M. Linares-Vásquez, "Can everyone use my app? an empirical study on accessibility in android apps," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 41–52.
[5] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2020.
[6] C. Tumbleson and R. Wiśniewski, "Apktool." [Online]. Available: https://bitbucket.org/iBotPeaches/apktool/downloads/
[7] B. Pan, "dex2jar." [Online]. Available: https://sourceforge.net/projects/dex2jar/
[8] P. Kouznetsov, "Jad." [Online]. Available: http://www.javadecompilers.com/jad
[9] E. Foundation, "Java development tools core." [Online]. Available: https://mvnrepository.com/artifact/org.eclipse.jdt/org.eclipse.jdt.core/3.17.0
[10] "Resources for developers and publishers." [Online]. Available: https://www.google.com/accessibility/for-developers/

[11] J. Maring, “Apps using accessibility services could be removed from the play store,” Nov 2017. [Online]. Available: https://www.androidcentral.com/apps-using-accessibility-services-could-be-removed-play-store