

System Design and Software Engineering Challenges in Building an Android Application: a Case Study

Rawan Tarek Khalil and Ala' Fathi Khalifeh, German University in Cairo, Egypt
 rawan.khalil@student.guc.edu.eg, alaa.khalifeh@guc.edu.eg

Abstract— With the tremendous increase in the number of mobile phone users. The demand on having mobile applications that provide solutions to many of the people's daily life aspects increases too. However, designing these applications is different from the typical desktop applications and imposes many software engineering challenges that need to be taken into consideration. In this paper, we present the system design and the software engineering challenges encountered while implementing an Android mobile application that provides a voiced based text messaging functionalities for people who tends to text while driving their cars.

Index Terms— Voice recognition, Android mobile development, automatic text messaging, voice based software development, mobile phone software engineering.

I. INTRODUCTION

With the way that technology has revolutionized communication methods, mobile phones have become an indispensable part of keeping up with this fast paced world. Mobile phones have provided an on-the-go easy, fast and efficient method of communication and have opened up a lot of possibility for enhancing user experience with the digital world. However, this widespread adoption of these devices has stimulated software architects to create new applications and solutions that opened up a new horizon for the software engineering community and imposed new challenges and limitation that did not exist in the personal computer applications.

People are always looking for the easiest, most user friendly and most intuitive method of communicating through their devices and there is probably no easier way to achieve that than talking to your phone. Speech is one of the most intuitive ways of interaction since it is the natural method of communication between people. Thus, speech recognition over mobile phones is rapidly becoming a must have feature in any device. For people on the go, for example while driving or walking, it is quite difficult and not to mention very dangerous to use mobile phones without any kinds of aids such as wireless Bluetooth earpieces and other accessories that make it hands free. According to [1], a shocking 16,000 deaths were caused by texting while driving in the US between 2001 and 2007 and it is on the rise every year. A solution to a problem would be provided through the use of Automatic Speech Recognition (ASR). Through ASR software, drivers will never have to take their eyes off the road, their hands off the wheel or their minds off their safety. They can respond to text messages by simply speaking their reply. The main aim of this paper to present the software design methodology and steps needed to design a voiced based text messaging application over Android mobile phones that try to address the

mentioned problem, with some emphasis on the software and design challenges faced during the implementation, which is of particular interest to the Android mobile developers' community.

The rest of the paper is divided into three further sections. Section II describes the application developed and its architecture. Section III presents some of the software engineering challenges faced during the implementation and how they were addressed. Finally, Section IV summarizes the paper.

II. SYSTEM DESIGN AND ARCHITECTURE

In this section, we provide a detailed description of the application, its architecture, and all the modules and components of the application.

A. Application Description

The main aim of the application is to enable users, specifically drivers, to use their mobile phones on the road without having to resort to physical contact with the phone. This approach for interaction with the device will greatly reduce the risk of accidents caused by distracted driving. The application focuses on the feature of text messaging or Short Message Service (SMS). When a mobile phone user receives a text message while driving, he is either not going to reply until they stop or they will have to pick up the phone to read and/or reply to the message. To provide a solution to that problem, we created this application that combines the features of text messaging with those of speech recognition and speech synthesis. Instead of having the user to type in the reply using the keyboard, the user could use the speech recognition feature and reply which is then automatically translated by the recognition engine. The application is based on the idea of having a set of keywords that correspond to longer stored messages. This means that each user has a list of words stored on their phone and each keyword is a sort of abbreviation for a longer text message. For example, the user defined keyword "later" could stand for "I'm busy right now. I will call you later". The user creates his/her own set of custom keywords and messages depending on their preferences and the most common situations that could be encountered while driving. The keywords are set to be exactly one word long in order to make it easier and faster while driving to just say a keyword instead of a whole sentence. Moreover, this improves the chances of accurate recognition rather than using long complex sentences.

When the user receives a text message, by using speech synthesis the system will ask the user if they would like to reply. If the user replies with "Yes", they will be prompted for the keyword they would like to send. Once the keyword is spoken by the user, if the recognition is correct, the message associated with that keyword is retrieved and used as the

message body for the reply. The user will then confirm the system to send the message. This is the core part of the application that uses all the key components. There is also a component for viewing the inbox and a component for viewing and editing keywords.

B. Application Architecture and Modules

Figure 1 shows the block diagram for the system architecture. There are three modules in the system; the keywords module, the messaging module and the dialogue module. The diagram shows the main Android packages used in each module. The boxes shaded in green correspond to Android packages [2] while the boxes shaded in blue correspond to custom classes. Below each box there is a note that contains all the classes used within each package. For the custom classes, the note boxes are also custom classes in the same module that are related to the main class. The boxes that exist outside any module like the Activity Manager are general items that were used in all modules.

1) Keywords Module: The keywords module is the component of the system that is responsible for managing the keywords. There are six main functionalities in this module. A user can display a list of all keywords, add, show, edit, delete or send a keyword. It is based on a simple Create, Render, Update and Delete (CRUD) system with a single two-column table database where one table entry corresponds to a keyword and its longer message. The user can see the list of keywords by selecting "Keywords" from the main screen. Once inside the keyword list, the user can then choose to view a specific keyword, add a new one or search for an existing keyword. If the user clicks on a certain keyword, he/she will be directed to the view or edit with the keyword and its message

2) Messaging Module: The messages module is a regular text messaging system. Using this module, users can view their inbox, send messages and reply to received messages. When users wish to send a message or reply to a message, they can choose from the list of stored keywords. Once they select the required keyword from the list, they will be directed to the show view of that keyword. They can then choose "Send" from the menu and will then be directed to the phone's text messaging application where they can choose a contact to send the message to. The message body is automatically filled out using the stored message from the chosen keyword and the user can edit it if he/she wishes to. In addition to the selection from the list, users can send messages by using speech recognition. When a message is received, they can reply to it by saying the keyword and the speech recognition will process the input speech and automatically send the message after confirmation from the user.

3) System-User Dialog Module: In order to make the application truly serve its purpose of eliminating physical and visual contact with the phone while driving, we combined the features of speech recognition and speech synthesis. Through these technologies, we were able to create a dialog between the user and the system on the event of receiving a message. This dialog is aimed at guiding the user to replying to the message

without the need to look at the phone. When a message is received, the system informs the user through speech synthesis that he/she has received a new message from certain contact. The system then asks the user if he/she would like to hear the message. The system will then start the speech recognition and wait for the user to respond. If there is no response from the user, the system will repeat the request two more times and if there is still no response, it will exit on the grounds that the user is unavailable. The system will also exit in the case that the user replies with "No". However, if the reply is "Yes", then the system will read out the received message. After that, the system asks the user if he/she would like to reply or hear the message again. If the user replies with "Again", then the message is replayed and the dialog sequence resets from that point. If the user says "No", then the system will exit only after confirmation from the user. If the user says "Yes", then the system will prompt the user for the keyword they would like to use as the reply to the message. If the keyword spoken by the user was found in the database, then the user is asked to confirm that the recognition was correct. If yes, then the system proceeds to send the message and exit. However, if the user says "No" or if the recognition returned no result, then the system will ask the user to say the keyword again. If the system fails to find the keyword three times, the system then exits. At any given point in the speech recognition process, if the user says the word "Quit" the system will automatically exit.

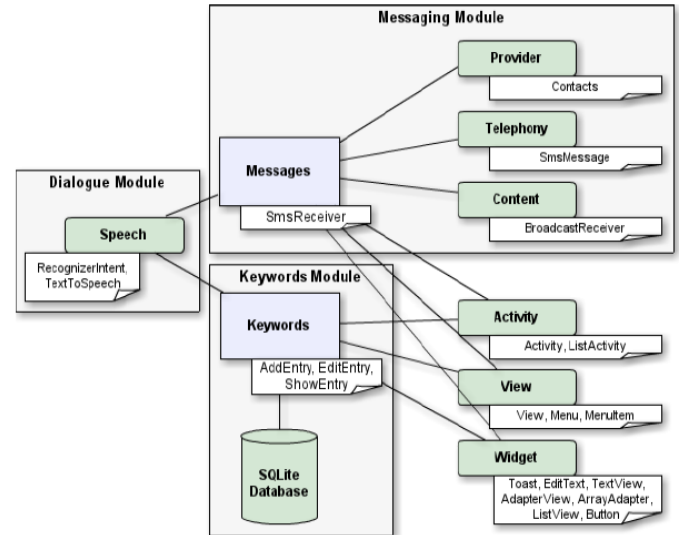


Figure 1. Application Architecture Block Diagram

III. SOFTWARE ENGINEERING CHALLENGES

In this section, we discuss some of the software challenges specific to the mobile device environment that we have faced during the system design and implementation.

Challenge 1: Mobile Hardware and Software Diversity: The diversity that now exists between software and hardware platforms for mobile devices is undeniable. It poses a great

challenge in choosing between them or trying to develop something that can work the same way across multiple platforms. Having to make this choice can be a difficult trade-off. Choosing a specific platform will limit the developer to the capabilities and features of this platform. Even within one platform, there could be several different versions that can make applications behave differently from one device to another. Developers must always try to make their applications work smoothly and as expected regardless of the device. Another significant problem is that it would limit the user base of the application to those that own supporting devices. However, sometimes limiting the user base is not necessarily a disadvantage. It depends on what the application needs and who it is targeted at. These problems can still exist with regular Software Engineering and not only Mobile Software Engineering. However, the problem is more pronounced and accounted for in mobile development since the scales of growth are now tipping into the direction of mobile devices over PCs. Moreover, most of the applications developed over PCs are usually web applications that are generally platform independent. In our application, we chose to develop on the Android platform. The reason for this selection is that the popularity of Android is increasing rapidly that it has now become the market leader. Furthermore, it is also open-source which gives the advantage of more developer collaboration and insight into the details. In addition, cross-platform adjustments required between Android devices is a simple process where the developer just needs to handle some design issues regarding the UI and defining the features that the application needs to run, so that if this feature is not present on a certain device, it would not install the application to avoid any crashes. In choosing Android, there are specific best practices recommended by developers that point others into the direction of what works better. This means that there are certain methodologies and concepts related to Android that distinguish the development process from other platforms. Hence, the software engineering procedure can not be generalized for other platforms.

Challenge 2: *Mobile OS Frequent Updates and Releases:* A challenge that may arise during mobile development is the ability of the application developed to be altered to match updates. Mobile platforms are very frequently updated and new versions come out quite often. Developed applications must be easily upgraded to fit the requirements and adapt to the changes of newer and better software versions. This is helped by the fact that Android SDK [2] comes in the form of several components that can be used to build an application. If an update occurs, then only the relevant components that are used in the application would need to be modified to fit the new version. For that reason, when developing for mobile software, it is better to separate different components from each other so that the application can be easily maintained and upgraded. Once it is altered, newer releases can be published to the market or app store.

Challenge 3: *Power, Processing and Storage Management:* Another significant challenge that comes with mobile

development is the lack of computational power required to perform complex operations. Mobile devices that exist so far have very limited computational power when compared to PCs. In our application, the use of voice recognition requires a huge amount of processing power in order to understand and analyze the speech input from the user. This is why Android voice recognition is done remotely on Google servers [3] rather than locally on the device. The amount of power required to perform speech recognition operations is too exhausting for a mobile device with the current hardware limitations. Furthermore, such heavy and complex operations will also drain the battery out too quickly. Developers must try to keep in mind the effect of the processing of the application and the services used by the application that will require a lot of battery power. However, one disadvantage of off-loading the mobile phone from the recognition functionality that it requires the mobile device to be connected to the Internet which may not be always possible. Another challenge is related to the limited storage space on mobile devices. In order to perform voice recognition, the application would require that there be a huge dictionary of words that the user may say to match them up. This will require a lot of valuable space. This space is not an issue on PCs, but for mobile devices, space consumption must be minimized to fit in all the other applications and processes required by the device. This is also helped by the recognition engine actually being on Google servers. The only space consumption required by the application is a very light and a basic database managed by SQLite engine that consists of only one table of keywords and messages.

IV. CONCLUSION

In this paper, we presented an Android mobile based voice application with an emphasis on the design aspects, and the software engineering practices related to the Android mobile platform. Utilizing our application, the driver can respond to the received text messages while driving by issuing a voice commands to his mobile phones. Some of the software engineering challenges encountered while building the application were the fact that designing a mobile application is to a big extent vendor and platform dependent, this is why we have chosen Android platform as it is implemented in a large number of mobile phones from different vendors. Another challenge was related to the frequent updates and releases of the mobile OS which was addressed by making the application modular and upgradable. Finally, one should bear in mind the limited computational and storage capabilities of the mobile phone which were taken into account in our application by off-loading the mobile device from the recognition process and instead, relying on the remote Google servers.

REFERENCES

- [1] F. Wilson and J. Stimpson, "Trends in Fatalities from Distracted Driving in the United States, 1999 to 2008." *Am J Public Health*, 100.11 (2010):2213-2219.
- [2] Android SDK: <http://developer.android.com/sdk/>
- [3] Google Mobile. <http://www.google.com/mobile/voice-actions/>.