SAFETY WITH MOBILE TECHNOLOGIES: HANDLE ALL CALLS WITHOUT DISTURBING THE USER

Dr. Shantakumar B. Patil  
Professor and HOD  
Visvesvaraya Technological University Belgaum India  
Department of Computer Science & Engineering  
Nagarjuna College of Engineering & Technology, Bengaluru

Dr. Premjiyoti Patil  
Professor  
Gopinath A R  
Research Scholar

Abstract

Consumer adoption of mobile devices has skyrocketed, changing the way people behave in many environments, including on city streets. Much of the emphasis of governments, advocacy organizations, researchers and technologists concerned with the safety implications of mobile device use in. Now a day most of calls come to us when we are busy, it is an interrupt, which disturbs or sometimes accident may happen. To avoid this either we switch off the mobile or we put the mobile in silent mode. If this is so if any emergency call comes we cannot attend such type of calls. The other disadvantage of this is the calling person does not know when the called person is free. To overcome all these problems we presented paper, smart mobile which handles all the call when he/she is busy. If a person is driving, without disturbing him the calling person is going to receive voice message, called person is driving call after an hour or person is in meeting call after an hour and any emergency send an SMS as “EMERGENCY”. Software which filters this message and it gives an emergency alarm to called mobile. Before going to drive or meeting, the user has to set the time and message. The time set by the user varies dynamically.

I. EXISTING SYSTEM

Before presenting this paper we referred many papers. In the existing system either we switch of the mobile or drive the mobile into silent mode. If the mobile is in silent mode we miss the emergency calls and the messages. Where as in our paper instead of keeping the mobile in silent mode we enable the application which in turn give the voice message to the caller. If any emergency call the caller send the message as “EMERGENCY” the application filter the message and emergency alarm is given to the called mobile.

II. ARCHITECTURE

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

A. Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

B. Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

C. Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core
features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

D. Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

E. Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

F. Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact. There are following four main components that can be used within an Android application:

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities</td>
<td>They dictate the UI and handle the user interaction to the smart phone screen</td>
</tr>
<tr>
<td>Services</td>
<td>They handle background processing associated with an application.</td>
</tr>
<tr>
<td>Broadcast Receivers</td>
<td>They handle communication between Android OS and applications.</td>
</tr>
<tr>
<td>Content Providers</td>
<td>They handle data and database management issues.</td>
</tr>
</tbody>
</table>

G. Activities

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched. An activity is implemented as a subclass of Activity class as follows:

```java
public class MainActivity extends Activity { }
```

H. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. A service is implemented as a subclass of Service class as follows:

```java
public class MyService extends Service { }
```

I. Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcasted as an Intent object.

```java
public class MyReceiver extends BroadcastReceiver { }
```

J. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely. A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions. public class MyContentProvider extends ContentProvider { } We will go through these tags in detail while covering application components in individual chapters.

K. Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are:
<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragments</td>
<td>Represents a behavior or a portion of user interface in an Activity.</td>
</tr>
<tr>
<td>Views</td>
<td>UI elements that are drawn onscreen including buttons, lists forms etc.</td>
</tr>
<tr>
<td>Layouts</td>
<td>View hierarchies that control screen format and appearance of the views.</td>
</tr>
<tr>
<td>Intents</td>
<td>Messages wiring components together.</td>
</tr>
<tr>
<td>Resources</td>
<td>External elements, such as strings, constants and drawables pictures.</td>
</tr>
<tr>
<td>Manifest</td>
<td>Configuration file for the application.</td>
</tr>
</tbody>
</table>

### TABLE II

<table>
<thead>
<tr>
<th>Table II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Fragments</td>
</tr>
<tr>
<td>Views</td>
</tr>
<tr>
<td>Layouts</td>
</tr>
<tr>
<td>Intents</td>
</tr>
<tr>
<td>Resources</td>
</tr>
<tr>
<td>Manifest</td>
</tr>
</tbody>
</table>

### III. ACTIVITIES

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched. If you have worked with C, C++ or Java programming language then you must have seen that your program starts from `main()` function. Very similar way, Android system initiates its program with an `Activity` starting with a call on `onCreate()` callback method. There is a sequence of callback methods that start an activity and a sequence of callback methods that tear down an activity.

The `Activity` class defines the following callbacks i.e. events. You don't need to implement all the callbacks methods. However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

### TABLE III

<table>
<thead>
<tr>
<th>Callback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onCreate()</td>
<td>This is the first callback and called when the activity is first created.</td>
</tr>
<tr>
<td>onStart()</td>
<td>This callback is called when the activity becomes visible to the user.</td>
</tr>
<tr>
<td>onResume()</td>
<td>This is called when the user starts interacting with the application.</td>
</tr>
<tr>
<td>onPause()</td>
<td>The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.</td>
</tr>
<tr>
<td>onStop()</td>
<td>This call is when the activity is no longer visible.</td>
</tr>
<tr>
<td>onDestroy()</td>
<td>This callback is called before the activity is destroyed by the system.</td>
</tr>
</tbody>
</table>

### IV. SERVICES

A service is a component that runs in the background to perform long-running operations without needing to interact with the user. A service is started when an application component, such as an activity, starts it by calling `startService()`. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

`onStartCommand()` To create a service, you create a Java class that extends the `Service` base class or one of its existing subclasses. The system calls this method when another component, such as an activity, requests that the service be started, by calling `startService()`. If you implement this method, it is your responsibility to stop the service when its work is done, by calling `stopSelf()` or `stopService()` methods. onCreate() The system calls this method when the service is first created using `onStartCommand()` or `onBind()`. This call is required to perform one-time setup.

### V. FILTER

A filter `<intent-filter>` may list more than one action but this list cannot be empty; a filter must contain at least one `<action>` element, otherwise it will block all intents. If more than one actions are mentioned then Android tries to match one of the mentioned actions before invoking the activity.

### VI. TEXT TO SPEECH

This is the so called cut and paste synthesis in which short segments of speech are selected from a pre-recorded database and joined one after another to produce the desired utterances. In theory, the use of real speech as the basis of synthetic speech brings about the potential for very high quality, but in practice there are serious limitations, mainly due to the memory capacity required by such a system. The longer the selected units are, the fewer problematic concatenation points will occur in the synthetic speech, but at the same time the memory requirements increase. Another limitation in concatenative synthesis is the strong dependency of the output speech on the chosen database. For example, the personality or the affective tone of the speech is hardly controllable. Despite the somewhat featureless nature, concatenative synthesis is well suited for certain limited applications.
VII CUSTOM COMPONENTS

Android offers a great list of pre-built widgets like Button, TextView, EditText, ListView, CheckBox, RadioButton, Gallery, Spinner, AutoCompleteTextView etc. which we can use directly in our Android application development, but there may be a situation when we are not satisfied with existing functionality of any of the available widgets. Android provides by means of creating our own custom components which we can customized to suit our needs. We need to make small adjustments to an existing widget or layout, we can Simply subclass the widget or layout and override its methods which will give us precise control over the appearance and function of a screen element.

VIII ALGORITHM

Enable software for the smart mobile
Add the required messages
Choose the services
Select the activity from the database
Apply the time for the enabled activity, this time will vary dynamically.

IX. RESULTS

Figure 2: Message Customizer

Figure 3: Start and Stop the Application

Figure 4: Adding and Retrieving Messages

REFERENCES


[2] Exploiting open functionality in SMS-capable cellular networks Patrick Traynor William Enck, Patrick McDaniel and Thomas La Porta Systems and Internet Infrastructure Security Laboratory,
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA


[5] Android text messaging application for visually impaired people Siddhesh R.Baravkar, Mohit Borde, Mahendra K. Nivangune Sinhgad Academy of Engineering, Pune, India
