Bachelor-Thesis

Android-based questionnaire app to examine usage of Wikipedia Mobile in the field

Quality & Usability, Institut für Softwaretechnik und theoretische Informatik,
Fakultät IV

Autor:
Robert Schmidt
310330
Student of informatics BSc

Gutachter und Betreuer:
Prof. Dr.-Ing. Sebastian Möller
Dr. Robert Schleicher
Ina Wechsung

07/13/12

Chapter 4

Technical Background

This chapter gives a short introduction about the Android-System as well as an overview about some of it's main components and concepts. It describes the architecture, the concept of Activities and how User Interfaces are realized. With a view to the App *Fragebogen*, knowledge about types and operation of Android-Services is necessary and will be described in the App components-section 4.1.3.

4.1 Android

"Android is a software stack for mobile devices that includes an *operating system*, *middleware*, and *key applications*" [51] and the result of the Open Handset Alliance, which was founded in 2007. Meanwhile it has 84 members, inter alia Vodafone, Google, ARM and T-Mobile. [52] includes a categorized listing of all members.

Android is based on a customized *Linux Kernel* (version 2.6), though standard Linux Desktop-Appplications do not run on Android. This fact is caused by a lot of modifications in the Android-System, e.g. Androids own C-library called Bionic. It needs less space and is designed to work with slower CPU-clock speeds, as the standard GNU libc. (These and further information about Bionic can be found in /ndk/docs/system/libc/OVERVIEW.html inside the Andoid Native Development Kit (NDK) [53]).

Android has been developed to run on a variety devices. Therefore, each version has it's own Compatibility Definition Document (CDD), which describes the specifications and requirements that have to be followed if the manufacturer wants to access the play store [2] or use the name "Android". The CDD for Android Version 2.2 can be located at [54]. If the manufacturer followed the CDD, a test has to be passed to verify the compatibility of his device [55].

T-Mobile's G1 (also known as HTC Dream), produced by the Taiwanese manufacturer HTC has been released on 10/22/08 and was the first Android device. Since then, a lot of companies added Android-based smartphones to their portfolio and new versions have been developed. (In [56] manufacturer, version, and specifications of produced Android devices can be browsed.) The table 4.1 shows versions, codenames and distribution-data, collected during a 14-day period in May and June 2012 [57].

Version	Codename	API Level	Distribution
1.5	Cupcake	3	0.3%
1.6	Donut	4	0.6%
2.1	Eclair	7	5.2%
2.2	Froyo	8	19.1%
2.3 - 2.3.2	Gingerbread	9	0.4%
2.3.3 - 2.3.7		10	64.6%
3.1	Honeycomb	12	0.7%
3.2		13	2%
4.0 - 4.0.2	Ice Cream Sandwich	14	0.4%
4.0.3 - 4.0.4		15	6.7%

Figure 4.1: Overview of code names, versions and distribution from a 14 day period which ended on 06/01/12 [57]

Programmers can start to develop Android-Apps for free and without limitations. A Software Development Kit (*SDK*) is available for Windows, Mac OS or Linux and includes an Eclipse plug-in. Because of the well documented guides and beginner tutorials, the barriers to begin are low. In addition to the possibility of testing and debugging created Apps on own devices, debugging within various kinds of Android virtual devices (*AVD*) is supported. Figure 4.2 shows a screenshot of a running AVD with Android 2.3.3, a WVGA Display, 256 MB RAM and a 200 Megabytes SD-Card.

Figure 4.2: Screenshot of *Fragebogen* running on an AVD

Java has to be used as programming language, due to performance issues or in special situations, native C-Code can also be applied. Who want his App to be published in the official play store from Google, has to register himself as publisher by using a Google Checkout Merchant account,

which costs 25 USD [58]. A release process has to be passed, in which the developer can determine the price, country-distribution and has the possibility to use In-app Billing for which Google demands transaction processing fees. Besides that, there are various other ways to distribute an App, for example Amazon is also running an alternative Android-Appstore [59].

4.1.1 Architecture

In Google's architectural scheme of Android 4.3, the lowest level contains the adapted and customized Linux Kernel with it's special drivers. Green highlighted are included *Libraries*, for instance WebKit (which is required for browsing) and SQLite (provides Database and Storage-Management possibilities).
The *Application Framework* above ensures that developers can access also basic functionality of the device, which enables them to access the GPS coordinates through the Location Manager or make a call with the help of the Telephony Manager from within their App. On any Android device, there are a few standard *Applications* installed on top, including the Browser, a Phone- and a SMS-App.

An important aspect of Android's Development was error tolerance and robustness. This is why every App runs in it's separate virtual machine. It is called *Dalvik Virtual Machine* and is part of the *Android Runtime*, illustrated in yellow in the figure below. The Dalvik VM has been developed to match special requirements a mobile system has to deal with, i.a. a slow processor and a small RAM. For this reason, the Dalvik VM runs own bytecode, dex (Dalvik Excecutable Bytecode). [60] includes a video and handouts of the Google IO '08-presentation from Dan Bornstein, the main responsible developer for the Dalvik VM. Starting with a good introduction, he gives a profound insight in the Dalvik VM.

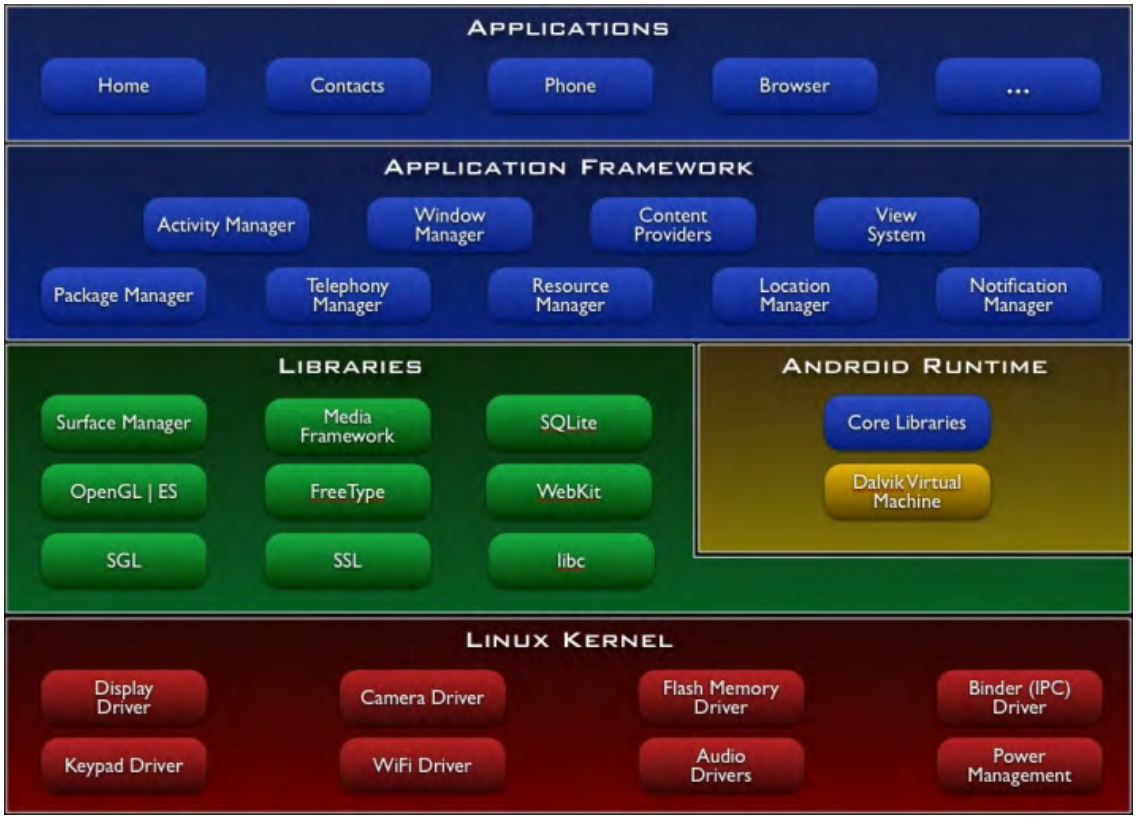

Figure 4.3: Architecture and components of the Android-System [61]

4.1.2 Build process

The following information has been extracted from the chapter "Build and running" of the Android Development Guide [62]. As mentioned in the previous section, Android-Apps are written in Java. Equal to Java Desktop applications, the code is translated into Java bytecode during the build process using the Java SDK compiler. After that and unlike in Java Desktop applications, an additional translation takes place, the tool dx creates the dex-file which contains the whole App and runs in the Dalvik VM. The Android Asset Packaging Tool (aapt) packs the dex file and additional resources like pictures, the Manifest and Meta-Information, as well as an App-certificate into the installable Android Package File (ending with .apk). A simplified process-graph of the build process is shown in 4.4.

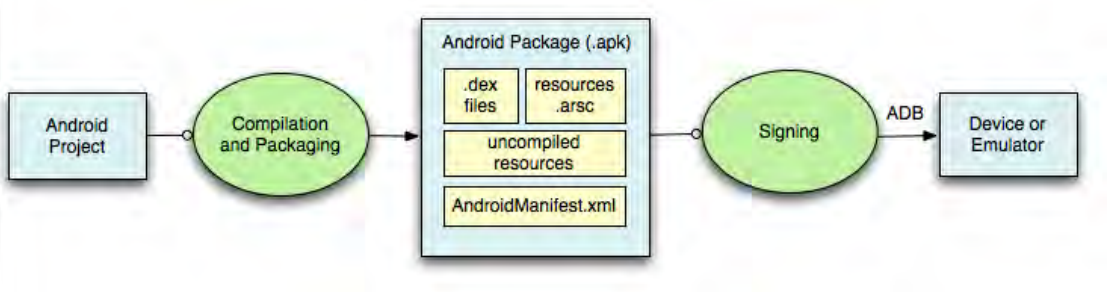

Figure 4.4: Simplified illustration of the Android build process [63]

4.1.3 App components

Due to the mobile orientation, Android Apps consist of different parts than for instance Java Desktop applications. The following section deals with these components and gives an insight about important concepts and their usage.

Context is an Abstract Class to access environmental functionality and one of the most important components of the Android-System. With the help of Context, e.g. phone, camera, USB-access or the WIFI-hardware can be operated. Therefore the getter-Method 4 - 1 of Context has to be called, it retrieves the suitable SystemService, for instance TelephonyManager for accessing the the devices phone-features. An example of accessing the vibration-hardware through Context is shown in 4 - 2.

Listing 4 - 1: Signature of Contexts important getter-Method:

```
public abstract Object getSystemService (String name)
```

Listing 4 - 2: Example how Context is used to gain access to the vibrator-hardware to give a 30 ms vibration feedback:

```
Vibrator vibra = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibra.vibrate(30);
```

String name	Functionality of the retrieved Object
ACCESSIBILITY_SERVICE	request Accessibility for UI-feedback
ACCOUNT_SERVICE	access registry of users online Accounts
ACTIVITY_SERVICE	retrieve ActivityManager
ALARM_SERVICE	access alarm
AUDIO_SERVICE	manage device-volume and ringer
CLIPBOARD_SERVICE	access clipboard
CONNECTIVITY_SERVICE	access network management
DEVICE_POLICY_SERVICE	work with the global device policy management
DOWNLOAD_SERVICE	request http-downloads
DROPBOX_SERVICE	access the global log-queue
INPUT_METHOD_SERVICE	access input methods
INPUT_SERVICE	interact with input devices
KEYGUARD_SERVICE	control the keyguard
LAYOUT_INFLATER_SERVICE	access layout-xml instantiation
LOCATION_SERVICE	get location updates
MEDIA_ROUTER_SERVICE	access media routes
NFC_SERVICE	use Near Field Communication-features
NOTIFICATION_SERVICE	request notification control
NSD_SERVICE	access network service discovery
POWER_SERVICE	control power management
SEARCH_SERVICE	access search functionality
SENSOR_SERVICE	request sensor data
STORAGE_SERVICE	handle system storage functions
TELEPHONY_SERVICE	access phone featues
TEXT_SERVICES_MANAGER_SERVICE	control text services
UI_MODE_SERVICE	access ui-modes
USB_SERVICE	act as USB host and control attached devices
VIBRATOR_SERVICE	control vibration hardware
WALLPAPER_SERVICE	access wallpapers
WIFI_P2P_SERVICE	control wireless peer-to-peer connections
WIFI_SERVICE	access WIFI
WINDOW_SERVICE	control window manager

Table: An overview of SystemServices, retrievable through Context

Windows to show App-content are called *Activities*. An Activity-Class has to extend Activity or one of its Subclasses. Every time, an App should display something an Activity has to be started, which takes the whole screen except the status bar. All Activities and Services, as well as their Intent Filters (which will be discussed later) are noted in the Manifest.xml. It also includes the required SDK-Version, the program icon, the App's package name and its permissions. An example of a uses-permission is shown in 4 - 3, the App is allowed to write on external storage.

Listing 4 - 3: Excerpt from Manifest.xml uses-permission

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Every App needs a StartActivity, a content which is shown after it was started by the Launcher. Listing 4 - 4 declares Menu as Starting-Activity.

Listing 4 - 4: Excerpt of the Manifest.xml- in which an Activity named "Menu" is described and will be shown after the App-Launcher:

```
<activity
  android:name=".Menu"
  android:label="@string/app_name"
  android:configChanges="orientation|keyboard|keyboardHidden">
```

```
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Activities can start other Activities, an application can contain one or several Activities. There are two ways to start an Activity from within another one, with or without a result (fire and forget). Both kinds need an Event (Intent). Example 4 - 5 creates and starts an Activity of the Activity-Class Service (the second parameter) from within the actual PackageContext Menu (first parameter).

Listing 4 - 5: The defined View "reportproblem" is explicitely placed in the window within the creation of an Activity.

```
Intent myIntent = new Intent(Menu.this, ServiceActivity.class);
startActivity(myIntent);
```

Intents are like purposes which can be executed. Besides the above usage, to start an Activity from within another one, they are also used to start Activities outside the own App. Therefore the other App has to register a suitable BroadcastReceiver for the Intent. Intents are also used to communicate with Services. A nice metaphor which helps to imagine what an Intent is, has been mentioned in the Intent reference from the Android Developers Guide [64] where it is called "the glue between activities". In the code 4 - 6 an E-Mail-Intent is created. Instead of calling an explicit Activity that should process it, a Chooser 4.5 is displayed and the user can pick the Activity he wants to start with the Intent. The shown Activities have been registered to this kind of Intent – with the help of a suitable Intent-Filter in the Activity-description of their Manifest.xml.

Figure 4.5: Screenshot of chooser with Apps for E-Mail Intents on Honeycomb HTC Flyer

Listing 4 - 6: Creating an E-Mail intent

```
Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND_MULTIPLE);
startActivity(Intent.createChooser(emailIntent, "Send mail..."));
```

Service is of high importance for *Fragebogen*, which is described in the subsection "Service" of the chapter "Realization" 6.1. Defined as "an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use" in the Service-Chapter of the Android-Reference [65], they can be started and stopped explicitly. Android Services run from within the process of the App, they are neither separate processes nor Threads. As illustrated in 4.6, a main difference to Java Threads is also their missing Pause-state, therefore they do not have to be resumed. It also reveals the two existing types, bounded and unbounded.

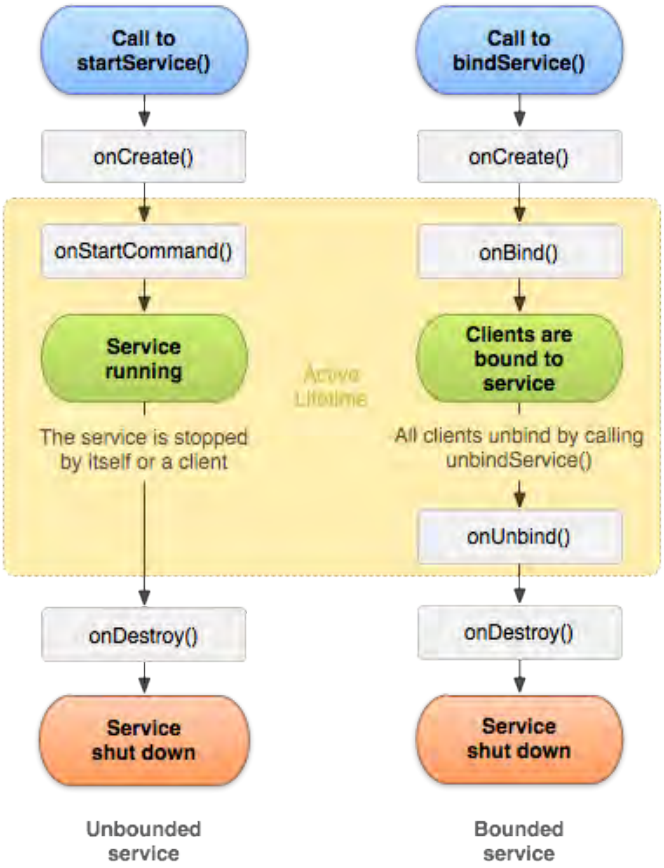

Figure 4.6: Life-Cycle of Android Services [66]. The right side shows a bounded Service which additionally enables the App to communicate with it.

Besides the two illustrated ways to kill a Service, (to stop or unbind it), there is another possibility how a Service is stopped: through low memory conditions. Two ways to prevent that are existing, firstly the Service can be started in mode START_STICKY, which makes it unlikely, the Service will be killed without an explicit call through *Context.stopService()*. The second method is to start the Service as a Foreground-Service, a special state with an additional notification in the Status bar. Android interprets these kinds of Services as foreground-activities and it is not allowed to kill them if more memory is needed. Further information about Foreground-Services and a code example of it's creation is described in chapter "Realization" 6.1.

4.1.4 Graphical User Interfaces and Layout

Although there are many similarities between Java Desktop applications and mobile Android Apps, the concept of Android User Interfaces (*UI*) differ in many aspects in comparison to SWT or SWING.
The whole UI is located as resource in the folder */res/* of the projects folder. Look and feel of the user interface, containing controls and layout are described and persisted in xml files, located in */res/layout/*. A main difference between Desktop-Applications and smartphones is the fact, that the display could be rotated and the user can switch between normal and landscape mode. Most devices have wide-screen displays and every rotation causes significant changes in terms of dimensions, positioning and layouting of the UI and its controls. For this issue, Andoid offers the possibility to define a separate layout for the landscape-mode in */res/layout-land/*.
Pictures and program icons are saved in */res/drawable/*. This folder contains several subfolders, high resolution pictures can be stored separately in */drawable-hdpi/*. Pictures with smaller resolution should be saved in the suitable folders */drawable-ldpi/* or */drawable-mdpi/*. It is also possible, to define separate layouts for different screen resolutions. Layouts for higher resolutions e.g. on tablets can be stored in *res/layout-xlarge/*. The concept behind this is to provide the opportunity to handle different kinds of displays and present an appropriate UI on many kinds of devices. Figure 4.7 has been taken from the screen-support section of the practices chapter from the Android Developers Guide [64] and gives an overview about categories, in which screens are divided.

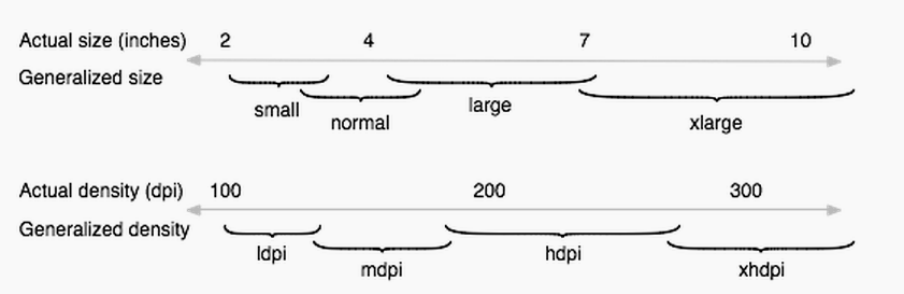

Figure 4.7: Mapping of densities and sizes [67]

Strings, for example label-, hint- and notification- texts should not be hard-coded in the layout-descriptions (xml-files) or in the program code. Instead, they are supposed to be stored in */res/-values/* and in the layout it should be referred to the string resource. There are three benefits of this concept: First sting-resources can be changed more easily because of the fact that they are located in a central file. Second, same texts does not have to be stored twice or more, this is also space-saving. And the third advantage is that translations can be added easily: Depending on the configured language of the device, Android chooses the correct translation if a suitable translation-file is present.
Once the interface and all needed resources are defined, the UI can be created within the code. Code example 4 - 7 shows the creation of an UI within the onCreate-Method of an Activity, therefore a *reportproblem.xml* has to be present.

Listing 4 - 7: The defined View "reportproblem" is explicitely placed in the window within the creation of an Activity.

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.reportproblem);
```

As shown in 4 - 8, the defined controls of the xml resources 4 - 9 can be accessed though their type and id.

Listing 4 - 8: gain access to the defined Radio-Button-control in the code (rbCriticalError is an instance of RadioButton.)

```
rbCriticalError = (RadioButton) findViewById(R.id.radioButtonError2);
```

Listing 4 - 9: Definition of a RadioButton-Control in the resources.xml-file:

```
<RadioButton
  android:id="@+id/radioButtonError2"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_weight="31.0"
  android:text="@string/lblHardError "/>
```

Android provides other controls than for instance SWING, or they differ at some points. This is due to its mobile orientation (an Android smartphone is also a music and video-player). For example a JSlider, known from SWING could not be found, instead a SeekBar-control, "an extension of ProgressBar that adds a draggable thumb" (SeekBar-reference of the Android Developers Guide [68] has to be used.
To arrange controls in the best way, Android provides different layout mechanisms. LinearLayout is one of the simplest and fits for numerous user interfaces with few content on it. All added Widgets (which are Views and should not be mixed up with the Home-screen Widgets) are arranged side by side vertically or horizontally (as set in the attribute *android:orientation*). The room for individual widgets can be enlarged by adjusting its attribute *android:layout_weight*.
Another layout is the RelativeLayout which is used to arrange containing views relatively to each other. Therefore the ID of the Views has to be specified.
Additionally there is ListView, which arranges its Views in a scrollable list and GridView. There are also specific layouts for instance TableLayout, which extends LinearLayout and organizes all child Views in a table whose rows and columns can be adjusted. These Views can also be spanned over several cells, it is possible to change its orientation and more.

Chapter 5

Requirements

In this chapter, the functional requirements and targeted additional features of the App *Fragebogen* are listed and described. Additional requirements, such as robustness and reliability are discussed in the chapter Realization 6.8. At the beginning of this work, a rudimentary version of *Fragebogen* was already present. Its features and limitations are shown in the table 5.1, as well as the required functionality.

	No.	Feature / functionality	Description
Current state	C1	Questionnaire shows after leaving a predefined App	After an undefined time interval (mostly within 3 hours), the questionnaire does not appear anymore
	C2	Create / change / remove questions	5-point Likert, smiley- and text-questions are possible
	C3	Start / exit App	the Application can be started and exited
	C4	Start / stop Service	The Service can be started and stopped
	C5	Logging	Questionnaire-data is logged, the format has to be improved (T9)
Targeted State	T1	File Atachments	test subjects should be able to add media files to a questionnaire and describe them
	T2	Log battery and network information	Additional data has to be retrieved and logged
	T3	Password protection	Fundamental functionality has to be protected from test subject's access
	T4	Reuse questionnaires	It should be possible to import and export questionaires, independent from the device
	T5	Create / edit questionnaires on a PC	It should be possible to create and change questionnaires with a PC
	T6	Reliability / Durability of the Service	compare to C1
	T7	Additional scales	Slider and 2-point Likert scales are required
	T8	Customizable scale-labels	Label-texts should be customizable from within the App
	T9	Formatted logfile	compare to C5, the logfile has to be more easy to import from spreadsheet-software and data analysis tools
	T10	Contact Administrator	Send an E-Mail with helpful attachments / information from within the App in order to report problems, errors and suggestions of improvements

Figure 5.1: Present and required functionality

Chapter 6

Realization

This chapter describes how the requirements and the targeted state have been reached as well as used methods and practices. The scheme 6.1 of the App's features and it's structure has been created from the requirements-definition and illustrates the targeted state.

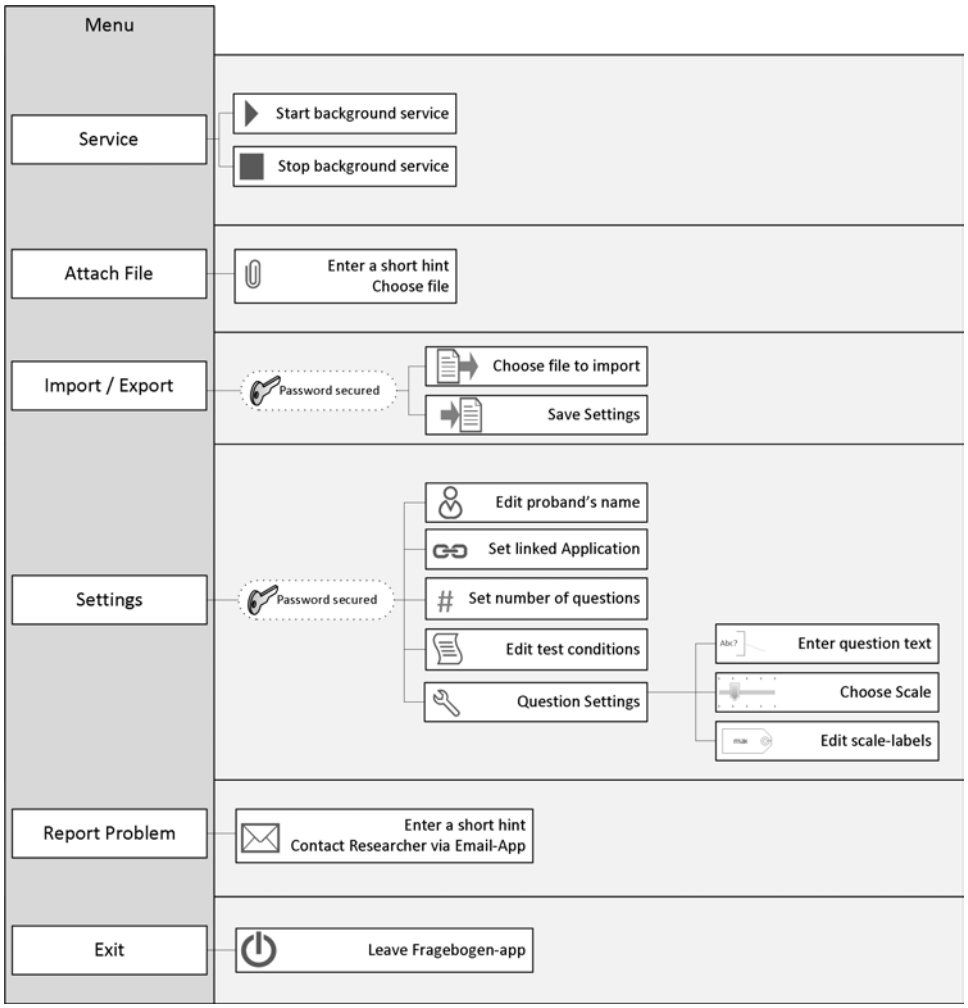

Figure 6.1: Overview of Fragebogen's features

6.1 Service

Fragebogen has to present a questionnaire, when another installed App was used. The observed Application does not know about the Service, that is why it cannot trigger the Activity, that starts the questionnaire, itself.
Best practices like the observer-pattern [69, Ch. 5, p. 326] can not be applied under these circumstances.
So *Fragebogen* has a Service *TaskListService* which checks the Application-states continuously with the help of a timer. The checking-process is illustrated in figure 6.2, a questionnaire is shown for one time, when the "observed" Application was in foreground-state and switched to background-state when the next check happens.
Because of the tickrate of 5000ms and callback removal of 1000ms, it is possible that extremely short App-usages are not recognized. For the test scenario with Wikipedia, an application-usage under 10 seconds is considered irrelevant.
If there is a need for recognition of shorter timer-ticks, they can be configured within *TaskList-Service.java*.

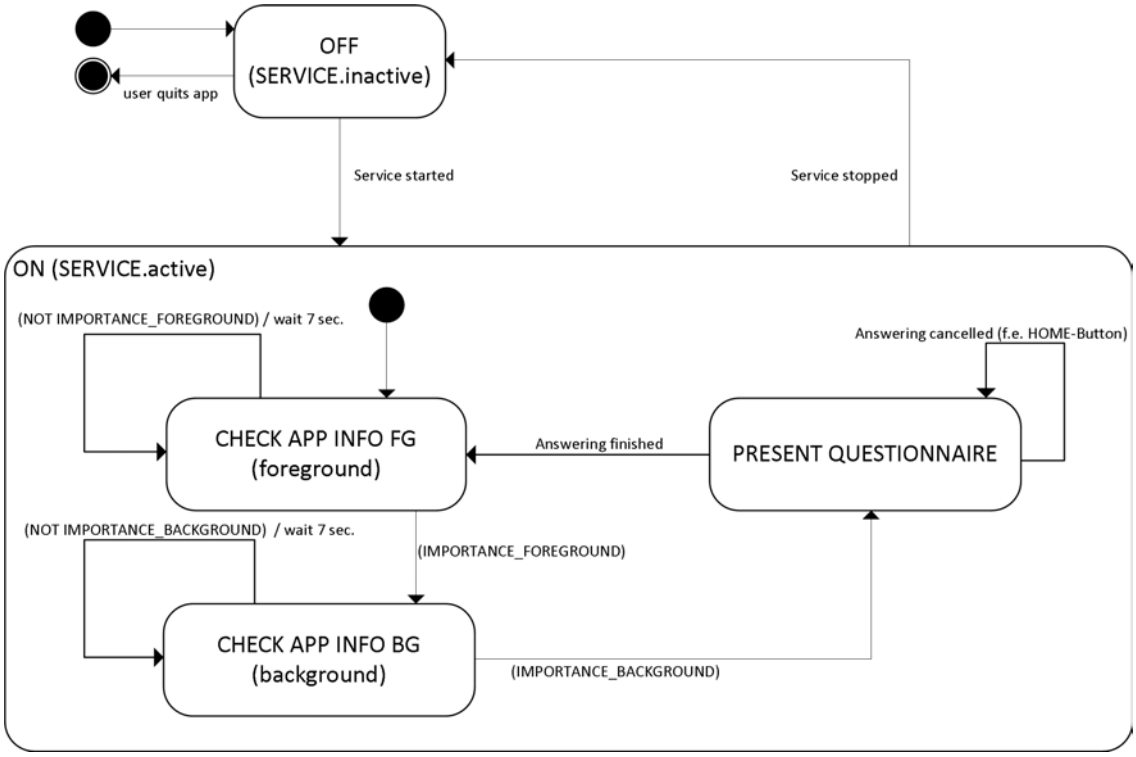

Figure 6.2: Timer and Appstate-check-cycle in statechart-semantics [70]

As mentioned in the chapter Background 4.1.3, Android kills Services under low memory conditions. Fragebogen should run it's Service until it is explicitly stopped by the user (requirement T6 5.1), to prevent earlier terminations, a foreground-flag can be set through method 6 - 1.

Listing 6 - 1: Defining a foreground-Service- an additional Notification is shown in the Status-bar:

```
public final void startForeground(int id, Notification notification)
```

In *Fragebogen*, the Notification should be non-functional and only indicate the running Service, but an Activity that starts after touching on the Service's icon (illustrated in 6.3) has to be specified in Android, therefore the empty Activity *DummyActivity.java* is used. Listing 6 - 2 shows, how

to achieve the foreground-state for the in Fragebogen's TaskListService with a non-functional dummy-Activity, which is listed in 6 - 3.

Listing 6 - 2: Excerpt from TaskListService.java- starting a foreground-Activity

```
@Override
public void onStart(Intent i, int startId) {
...
  Notification note = new Notification(R.drawable.icon,
      "Fragebogen-Service",System.currentTimeMillis());
  PendingIntent.getActivity(this,0,i,0);
  note.flags |= Notification.FLAG_NO_CLEAR;
  Intent notificationIntent = new Intent(getBaseContext(),
      DummyActivity.class);
  PendingIntent contentIntent = PendingIntent.getActivity
      (getBaseContext(),0,notificationIntent,0);
  note.contentIntent = contentIntent;
  note.setLatestEventInfo(getBaseContext(),"Fragebogen is running...",
    null,contentIntent);
  startForeground(startId,note);
...
```

Listing 6 - 3: DummyActivity-class- shows nothing

```
public class DummyActivity extends Activity{
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.empty);
    finish();
  }
}
```

Figure 6.3: Fragebogen's Status-Bar-icon (left), indicating the running Service

Restarting smartphones, e.g through empty batteries, is always an issue in mobile Application development. This can happen everytime during field tests and should be considered in the design and functionality of *Fragebogen*.
In the case of a restarted Android-device, *Fragebogen* has to start or display a message to the user. In order not to patronize the test subject which in most cases is the owner of the phone, the second possibility was chosen.
Android signalizes a lot of events, even when the boot-process completed. An Activity that displays *ServiceActivity.java* is started after *BootReceiver.java* received the global signal (which is an Intent), the *onReceive*-method is listed in 6 - 4. All global signals, *BootReceiver.java* is registered to, have to be declared in the *Manifest.xml*, an excerpt from *Fragebogen*'s Manifest is listed in 6 - 5.

Listing 6 - 4: onReceive-method of BootReceiver- create and start a new Intent from ServiceActivity:

```
@Override
public void onReceive(Context context, Intent intent) {

  Intent myIntent = new Intent(context, ServiceActivity.class);
  myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
  context.startActivity(myIntent);
```

Listing 6 - 5: Excerpt from Manifest.xml- BootReceiver is registered to the boot-completed-intent:

```
<receiver android:name=".BootReceiver">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED" />
  </intent-filter>
</receiver>
```

6.2 Additional scales

Fragebogen is a universal tool which enables researchers to observe and review every Android-App that has a foreground-activity. To be more versatile regarding the questionnaire itself, more scales had to be added (requirement T7 5.1), because every scale has it's advantages and a set of questions that fit and do not. For example the question "Are you hungry?" - does not suit for a smiley-scale and "How old are you?" is difficult to be answered on a 2-point-likert-scale.

6.2.1 Slider

Slider-scales are versatile, intuitive and popular in computer-based questionings. Paper-based their disadvantage lies in the aditional expenditure of manually or technically measure the ratings. Sliders provide continuous ratings and are particularly suitable for 2-point-questions in which the the probands should not be forced to make a decision for one fix category.

The slider-view is implemented, using a SeekBar for the actual slider. It's value can be chosen from 0 on the left to 100 on the right with a default-value of 50 when it appears. *Fragebogen*'s "slider"-description is shown in 6 - 6, the screenshots 6.4(b) and 6.4(a) illustrate the slider-scale.

Listing 6 - 6: xml-description of the SeekBar-control

```
<SeekBar
   android:id="@+id/Slider"
   android:layout_width="300dp"
   android:layout_gravity="center"
   android:layout_marginLeft="2dp"
   android:layout_marginRight="2dp"
   android:gravity="center_horizontal"
   android:max="100"
   android:progress="50"
   android:secondaryProgress="100" />
```

To display it's current value on a TextView above the SeekBar, a Listener, listed in 6 - 7, that reacts when the SeekBar-progress changes, had to be specified:

Listing 6 - 7: Inline-implementation of the SeekBar-ProgressChanged-Listener:

```
SeekBar sb = (SeekBar) dialog.findViewById(R.id.Slider);
TextView tvProgress = (TextView) dialog.findViewById(R.id.SliderValue);
sb.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
...
   @Override
   public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
       tvProgress.setText(Integer.toString(progress) + " %");
   }
}
```

Figure 6.4: Screenshots of questions with slider-scale. (a): HTC Flyer (Android 3.2.1), (b): Samsung Galaxy Y (Android 2.3.3)

6.2.2 Binary likert

To add the ability to ask typical yes-or-no-questions, a binary scale had to be realized. The result is shown in 6.5.
This scale provides a binary choice, therefore it is not possible to choose both RadioButtons at the same time. A ButtonGroup-control provides this feature and in order to put one RadioButton exactly above the other, TableLayout was used. The Textview to display the question, TableLayout with it's two RadioButtons and the Button to enter the choice are placed vertically above each other and horizontally centered. 6 - 8 is a complete listing of the layout-xml.

Listing 6 - 8: Binary likert scale

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingBottom="20dp" />
    <TableLayout
        android:layout_width="310dp"
        android:layout_height="135dp"
        android:layout_gravity="center"
        android:paddingLeft="40dp"
        android:paddingRight="40dp"
        android:stretchColumns="1" >
        <TableRow>
            <RadioGroup
                android:id="@+id/btnGroup"
                android:layout_width="300dp"
                android:layout_height="130dp"
                android:layout_gravity="center"
                android:gravity="center"
                android:orientation="vertical" >
                <RadioButton
                    android:id="@+id/radioButtonBin1"
                    android:layout_width="300dp"
                    android:layout_height="50dp"
                    android:layout_gravity="left"
                    android:layout_margin="20dp"
                    android:text="" />
                <RadioButton
                    android:id="@+id/radioButtonBin2"
                    android:layout_width="300dp"
                    android:layout_height="50dp"
                    android:layout_gravity="left"
                    android:layout_margin="20dp"
                    android:text="" />
            </RadioGroup>
        </TableRow>
    </TableLayout>
    <Button
        android:id="@+id/btnEnterBinary"
        android:layout_width="100dp"
        android:layout_height="65dp"
        android:layout_gravity="center_horizontal"
        android:text="@string/enter"
        android:textSize="22dp" />
</LinearLayout>
```

Figure 6.5: Example of Fragebogen's binary likert scale

6.2.3 Customizable labels

Versatility has been mentioned in the previous two sections, the same thought led to requirement T8 5.1, to be able to customize questionnaires. Question-labels are TextView-controls on every layout, their values are stored as user-defined App-preferences (type:TextPreference) "LabelA" (left sided or minimum) and "LabelB" (right-sided or maximum), followed by an unserscore and the question-number which they belong to.

Listing 6 - 9 shows how the left-sided text is being read from the Preferences.

Listing 6 - 9: get preference from DefaultSharedPreferences

```
SharedPreferences app_prefs = PreferenceManager.getDefaultSharedPreferences(this);
LabelAPref[i] = app_prefs.getString("LabelA_" + i, "");
```

Writing a Preference is shown in 6 - 10:

Listing 6 - 10: Saving User-defined preferences

```
app_prefs.edit().putString("LabelA_" + i, text).commit();
```

6.3 Report problem

To provide a possibility to contact an administrative person during the field test (T10 5.1), *Fragebogen* has the Activity-Class *ReportProblem.java*. The interface for this contains:

- A Textfield for the message
- Two Radiobuttons to categorize the report (e.g. a serious program error or a simple suggestion of improvement)
- A Button to send the report

Textfields in Android are called *EditText*, the message from the test subject should be entered first, so this control needs to be focused (*requestFocus*) and it's text may be longer, therefore Android provides a special inputType, *textMultiLine*. 6 - 11 shows an example of the xml-description.

Listing 6 - 11: multi-line EditText

```
<EditText
            android:id="@+id/EditTextReportProblem"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:inputType="textMultiLine" >
            <requestFocus />
</EditText>
```

The Radiobuttons should not be checked both, therefore Android provides *RadioGroup*: within one group, only one Button can be checked. So both RadioButtons have to be inside the same group, as realized in the excerpt 6 - 12 from the layout-definition:

Listing 6 - 12: grouped RadioButtons

```
<RadioGroup
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >
            <RadioButton
                android:id="@+id/radioButtonSchwerwiegend"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="31.52"
                android:text="@string/errorGross" />
            <RadioButton
                android:id="@+id/radioButtonEingeschraenkt"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="31.52"
                android:text="@string/errorKlein" />
</RadioGroup>
```

In 6 - 13 the Button to send the report is defined.

Listing 6 - 13: Centered "send"-Button

```
<Button
            android:id="@+id/send"
            android:layout_width="100dp"
            android:layout_height="65dp"
            android:layout_gravity="center_horizontal"
            android:text="@string/sendEmail"
            android:textSize="22dp" />
```

All controls are arranged in a simple LinearLayout, as shown in both screenshots 6.6(a) and 6.6(b) for normal- and landscape-mode the same layout-file is used.

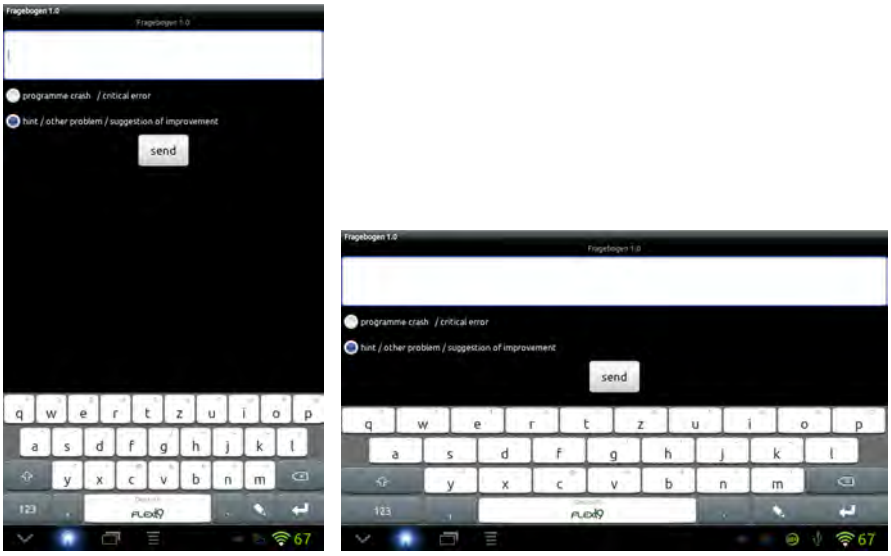

Figure 6.6: Screenshots of ReportProblem in normal and landscape-mode

The code behind the send-button creates an E-Mail-Intent with the information of from the interface. Additionally all files (the log and maybe error-files) should be attached to the E-Mail. Section Testing 6.8 describes usage and reasons for error-files.

The code from listing 6 - 14 iterates over every file in the *Fragebogen*'s folder and adds it's URI (Universal Resource Identifier) to an ArrayList (of type Uri) that is added to the E-Mail-Intent in the last line:

Listing 6 - 14: Putting file-attachments to an E-Mail-Intent.

```
ArrayList<Uri> uris = new ArrayList<Uri>();
File dir = new File(Environment.getExternalStorageDirectory() + "/Fragebogen_Log");
File[] filelist = dir.listFiles();
  for (File file : filelist) {
        Uri u = Uri.fromFile(file);
        uris.add(Uri.parse(u.toString()));
  }
emailIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
```

Sending the E-Mail-Intent with the help of a Chooser-Activity is described and also illustrated with a screenshot in chapter Technical Background under App components 4.1.3.

6.4 Logged data

Reliable logging of the test subject's data is of fundamental importance in order to analyze it. All recordings have to be valid and complete, a corrupted file may be entirely unusable. In the worst case, whole surveys that lasted for weeks, can fail.
In *Fragebogen*, the log is stored as a textfile. The first two lines contain decriptions and general data, all following lines are recorded data and responses from one test subject. The detailed structure is shown below, all values are tab-separated:

- 1st headline (general data)
  - Proband's name
  - Test-conditions
  - Observed App
- 2nd headline (descriptions)
  - Date,Timestamp,AnswerNr,Type,AnswerValue)
  - Timestamp
  - Question-No.
  - Scale
  - Answer
- **1st** response
  - Date, Timestamp, Number of question, scale, answer to question **1**
  - Date, Timestamp, Number of question, scale, answer to question **2**
  - ...
  - Collected data
    - Battery-state
    - Network-information
- All following responses (same structure as the previous response)

Attached files are saved as a a single separate line per file within Logfile.txt and in the following order: *filename-description-date-timestamp*. This structure has been established after the field test, when requirement T9 5.1 was added- the earlier one stored every answer in a separate line, similar to the logfiles from the Experience Sampling Program 3.2. The actual version can contain extremely long lines, depending on the number of questions and may be difficult to read in a text-editor, but after importing it in spreadsheet-tools it is much easier to filter, order and analyze the data, because one response equals one line. Nevertheless the field test-logs were valid – it just took more time to restructure the data.

The recordings are stored under */Fragebogen_Log/* in a file named *Logfile.txt* on the external storage of the smartphone. This storage-directory can be discovered with the help of the code in listing 6 - 15.

Listing 6 - 15: Get path to the external storage

```
public final static String folder = Environment.getExternalStorageDirectory();
```

Issue: Most Android smartphones support USB-mounting, which provides an USB-storage device on a PC when the phone is connected via USB. In this case, the external storage directory can not be accessed from the smartphone itself. *Fragebogen* still works during this period of time, but recordings are not saved. *To solve this problem in further development, one possibility is to write the recordings in a temporary file on the internal storage, as long as the external storage is not mounted – it's actual state can be checked with the help of the android.os.Environment- method in listing 6 - 16. :*

Listing 6 - 16: Get state from external storage directory

```
public static String getExternalStorageState()
```

Before writing the data, it has to be checked if the temporary file is present and it's data has to be copied in Logfile.txt on the external storage, before appending the actual log-entry. After this, the temporary file can be deleted.

6.4.1 Mobile Connection

If and how a mobile device is connected to the internet is an important factor that influences it's usage. Corresponding to requirement T2 5.1 *Fragebogen* is logging the network information, Android devices provide through *Context*. Therefore a ConnectivityManager has to be instantiated to access it's status. Listing 6 - 17 is an example, how to get the required network-information, in *Fragebogen* it is surrounded by try-catch because a Runtime-Exception (ParseException) is thrown when the actual adress can not be parsed, which is part of the connection-info.

Listing 6 - 17: Reading network info

```
ConnectivityManager connectivityManager =
  (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
String info2Log = activeNetworkInfo.toString();
```

In case the device is connected, *info2Log* contains the type of connection, information about roaming, APN, IPv4 and IPv6-connectivity and more, otherwise *"NETWORK_ERROR"* is logged.

6.4.2 Battery state

There are two possibilities to access the state of an Android-device's battery. Firstly to read a system-file and secondly to create a BroadcastReceiver and register it to the broadcasted Intents. The method for the second option is equivalent to the realization of the BootReceiver-class in order to remember the test subject to keep *Fragebogen*'s Service alive 6.1. Due to that, the first option was realized, System's file and folder is shown in 6 - 18.

Listing 6 - 18: File that contains battery capacity

```
private static final String BATT = "/sys/class/power_supply/battery/capacity";
```

A *FileReader* is used to read the specified file char-wise and add it to a String which equals the actual battery-capacity (0-100) and is returned.
In case of an Exception, *"BATTERY_CAPACITY_ERROR"* is returned instead- this never happened during the tests or in the field, except on one device: the Sony Xperia Ray. It may be that the system-file was not present (and another file-structure has been used) or the permission to read the file was not granted, even though *Fragebogen* has the required permissions. Because of that, the solution for requirement T2 5.1 can still be improved.

6.5 File attachments

To fulfill requirement T1 5.1, an Activity *AttachmentManager* is used.
If a user wants to attach a file to provide additional information, e.g. a screenshot or a photo, he has to enter a description before choosing a file in order to attach it, therefore a filechooser-Activity [71] is started. Screenshots from Fragebogen's Attach-file feature are shown in 6.7(a) and 6.7(b).

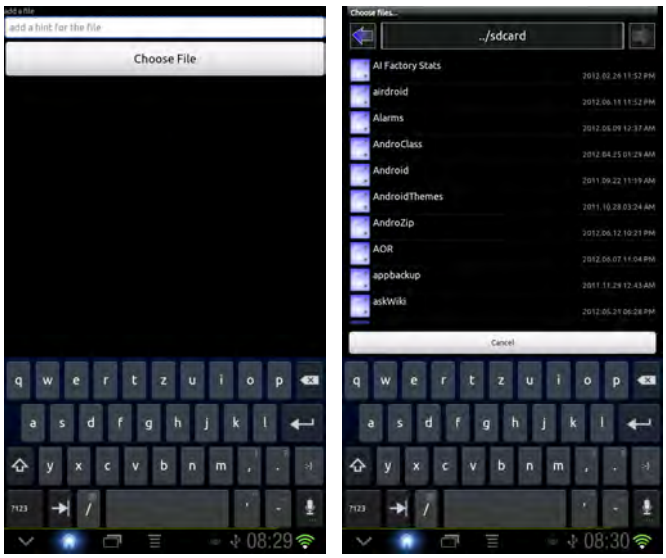

Figure 6.7: (a): AttachmentManager-View (b): file-chooser-Activity (touch & hold on a filename selects it)

The main thought behind using a foreign library for this task and to integrate this function in the App itself, is that some devices have no File-Explorer-App pre-installed and it takes time and a reliable internet-connection to download a suitable App from Google's play-store or to connect the smartphone to a PC and install an apk, especially when a survey is going to start and the researcher is setting up a lot of different smartphones.
Android-filechooser is open source, Apache 2.0 licenced and a small Activity which returns a result (the 2 kinds of Activities are explained in chapter Background 4.1.3), a list of *LocalFile*, a wrapper class for @link File. The first item of this list is the chosen file and the private helper-method *private void copy_rename(File f, String info)* is called with the file-reference and description as parameters. It copies the file into the *Fragebogen*-folder under a different name (a timestamp to prevent the files getting mixed up later, is appended). After the file has been copied, a logfile-entry is created that contains filename, time and the entered description. An informal program-sequence is shown in figure 6.8, listing 6 - 19 starts the chooser, awaiting it's result, which can be handled inside the method 6 - 20.

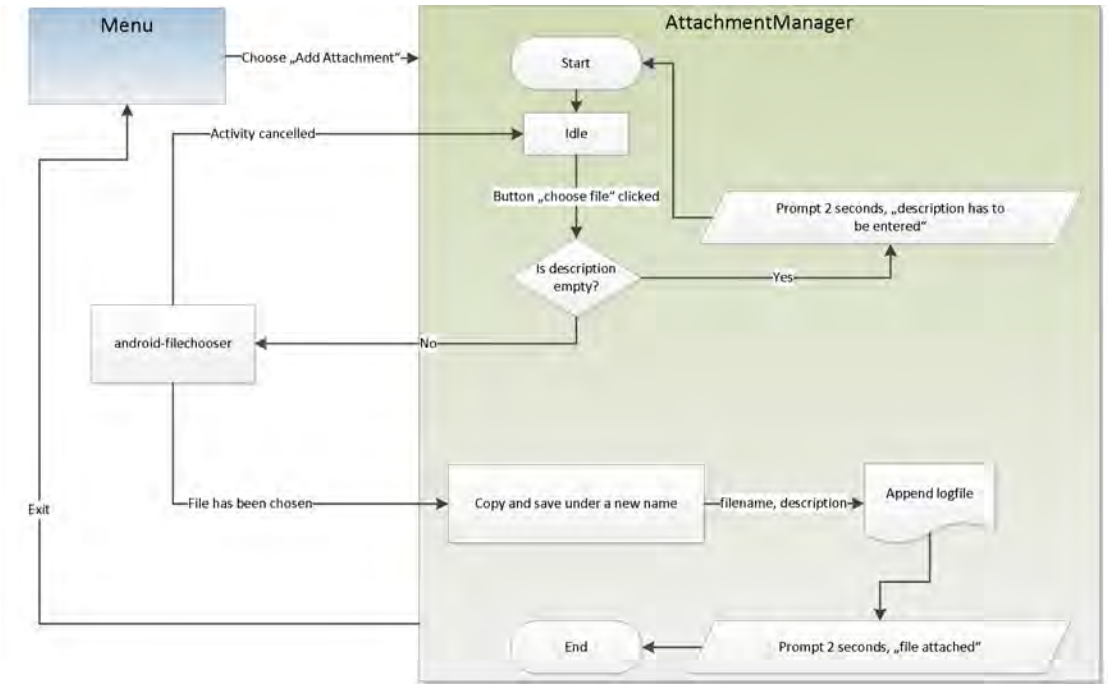

Figure 6.8: Attaching files to a questionnaire, program sequence-scheme

Listing 6 - 19: Starting the filechooser-Activity for a result:

```
Intent intent = new Intent(getBaseContext(), FileChooserActivity.class);
intent.putExtra(FileChooserActivity.Rootpath, new LocalFile(file_dir));
startActivityForResult(intent, ReqChooseFile);
```

Listing 6 - 20: Get and handle the additional data from the Activities-result:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
...
```

6.6 Simplification of questionnaire handling

In the previous state, researchers had to enter proband's name, test conditions, observed App, as well as every question and scale on every test subject's smartphone. Due to the huge variety of different devices [56], this was an unacceptable expenditure. To be capable of being used in larger-scaled studies, questionnaires had to be reusable and PC-configurable (requirements T4 and T5 5.1).

6.6.1 File structure

Every questionnaire contains following reusable information:

- Observed App
- Test conditions (e.g. a name for the survey)
- Number of questions
- For every question:
  - Question-text
  - Question-scale
  - Left scale-label (minimum)
  - Right scale-label (maximum)

A valid file is shown in listing 6 - 21.

Listing 6 - 21: Structure of questionnaire-file

```
org.wikipedia/VP/Feldtest/8/

Wie gefiel dir die Wikipedia-App dieses Mal?/1/ / /
Wonach hast du gesucht?/3/ / /
Hast du gefunden, was du gesucht hast?/2/nein/ja/
Wie dringend war deine Frage, bzw. das Thema?/2/nicht dringend/dringend/
Die Bedienung der App war.../2/schlecht/gut/
Die Geschwindigkeit der App war.../2/schlecht/gut/
Die Antwort fand ich durch.../4/das Bild/den Text/
Mit wievielen Personen hast du die App genutzt?/5/alleine/zusammen mit anderen/
```

The headline must contain the Application's name at first (e.g. *org.wikipedia*) – (it has to be the exact ApplicationInfo-name, otherwise the questionnaire will not appear), secondly the test subject's name or number (*VP*), followed by a text which describes the conditions of the survey or can be used as a survey-name (*Feldtest*). The last element of the headline is the number of questions which has to be equal with the number of questions specified in the following lines of the file (*8*). A slash ("/") is used as separator between the items, therefore it is not possible to use this symbol within a question – it will be replaced by *"-"* automatically.
If another symbol should be used as separator, e.g. when *"/"* is needed within a question, the final static String *delimiter* in the class *ImportExport* has to be changed. Listing 6 - 22 shows how the slashes are replaced.

Listing 6 - 22: Replacing all used separators with a minus-symbol:

```
FragePref[i] = app_prefs.getString("Frage_" + i,"")
    .replaceAll(ImportExport.delimiter, "-");
```

The second line of a questionnaire-file has to be empty. Every following line equals one question and consists of the same elements, separated by *ImportExport.delimiter*: Firstly the question, followed by it's scale (which is a number). The table below shows all number-scale-tupels:

No.	Scale
1	Smiley
2	5-point likert
3	Text
4	Slider
5	2-point likert

Table: numbers of scales

The two last items of the question-lines are left label-text (for the minimum-value) and the label text to the right, divided and followed by separators.

6.6.2 Export & import

The Activity *ImportExport* is responsible for writing questionnaire-settings to a file (Save) and read a questionnaire-file in order to overwrite the App-configuration (Load). Android provides *PreferenceManager*, for defining, setting and saving own Application-preferences as well as Preference-hierarchies.

The export-method reads all reusable configurations from an instance of PreferenceManager and writes it into a text-file, which is named "*Fragebogen_*" followed by a timestamp and the extension "*.txt*". SharedPreferences are user-defined key-value-pairs, reused questionnaire-information is shown in the following table:

Key	**Description of the preference**
ActivityName	Observed App
Name	Name of the test subject
Bedingungen	Survey-name or description
Anzahl_der_Versuche	Number of questions
Frage_	Question (followed by it's number)
Type_	Scale (followed by question number)
LabelA_	Left label (followed by question number)
LabelB_	Right label (followed by question number)

To access the SharedPreference, a suitable getter-method (depending on the type of the stored value) has to be called. The code-example below reads the observed App-preference and calls the helper-method *write2SaveHead(String, OutputStreamWriter)* which saves the export-file in the defined structure. An excerpt of saving the Preference "ActivityName" is shown in 6 - 23.

Listing 6 - 23: Exporting the observed-Application-preference stored as App-preference with the key "ActivityName"

```
final SharedPreferences app_prefs = PreferenceManager
    .getDefaultSharedPreferences(this);
String app = app_prefs.getString("ActivityName", "");
write2SaveHead(app, out);
```

Import works the other way around, reads the questionnaire-file line by line and uses the setter-methods equivalent to the used getters from PreferenceManager. In order to chose the import-file, a filechooser-activity (described in section File attachments) is started. The upper part of the Import-Export-View is shown in 6.9.

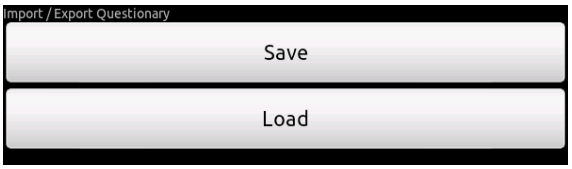

Figure 6.9: Screenshot of the two Buttons of the ImportExport-Activity

6.7 Password protection

Fragebogen is designed to be used on test subject's devices and should collect reliable and valid data over long-term surveys. To prevent intentional or unintentional preference-changes and affections by the test subjects (e.g. changing scales, questions or the observed App), a simple password-protection mechanism has been implemented (requirement T3 5.1).
Due to it's architecture (*Fragebogen* is a local App with no possibilitiy of being remotely-configured and -controlled), the password has been stored as a String-object in the code itself. It's value **"QULab"** is stored in *Preferences.java*, listing 6 - 24 shows the line. It is *final* to prevent changes during runtime and has the access-modifier *protected* to grant access from within the class itself, all subclasses and classes within the same package.

Listing 6 - 24: Definition and declaration of the password-object

```
final protected static String pw = "QULab";
```

This protection has been implemented as a prompt which compares the entered value with the stored String-object and enables or disables protected controls. To prevent that surrounding persons can read the entered password, the EditText is provided with an extra type of input (*InputType.TYPE_TEXT_VARIATION_PASSWORD*), shown in example 6 - 25. The password-prompt 6.10(a) and a disabled-preferences-view are shown in 6.10(b).

Listing 6 - 25: create a password-prompt with a masked EditText.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  AlertDialog.Builder alert = new AlertDialog.Builder(this);
  ...
  final EditText input = new EditText(this);
  input.setInputType(InputType.TYPE_CLASS_TEXT |
    InputType.TYPE_TEXT_VARIATION_PASSWORD);
  alert.setView(input);
```

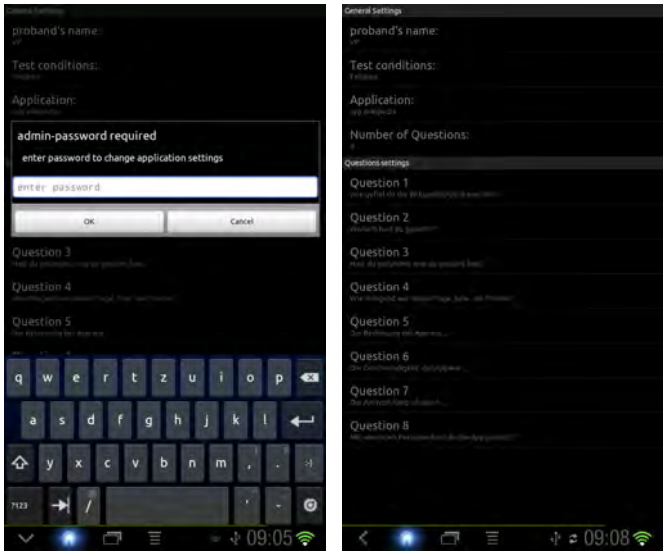

Figure 6.10: (a): Screenshot of *Fragebogen*'s password prompt, (b): Screenshot of disabled *Fragebogen*-preferences.

6.8 Testing

Android provides a range of functional mechanisms and tools to test self-developed Applications. *Fragebogen* has been tested at the following stages of this thesis:

- before the realization (to check the present state)
- during the devepopment (after adding / changing features)
- after the development

To debug on a device or an emulator, the following three preconditions have to be satisfied:

- The system is set up (e.g. Linux users have to add a udev-rule with the manufacturer's code (vendor-id), the exact steps are described in the tools-section from Androids developers page [72]
- Debugging is enabled on the smartphone (as shown in the screenshot 6.11)
- The developed Application is declared *debuggable* in the *Manifest.xml*

Figure 6.11: Screenshot- Settings->Application->Development (Settings->Developer options on Android 4 devices)

After these steps, an USB-attached device will show log-information in the LogCat-window and the developer can print suitable information from his code there, e.g. errors or program-sequence information. 6.12 is a screenshot of a log-cat-window inside the eclipse-ide.

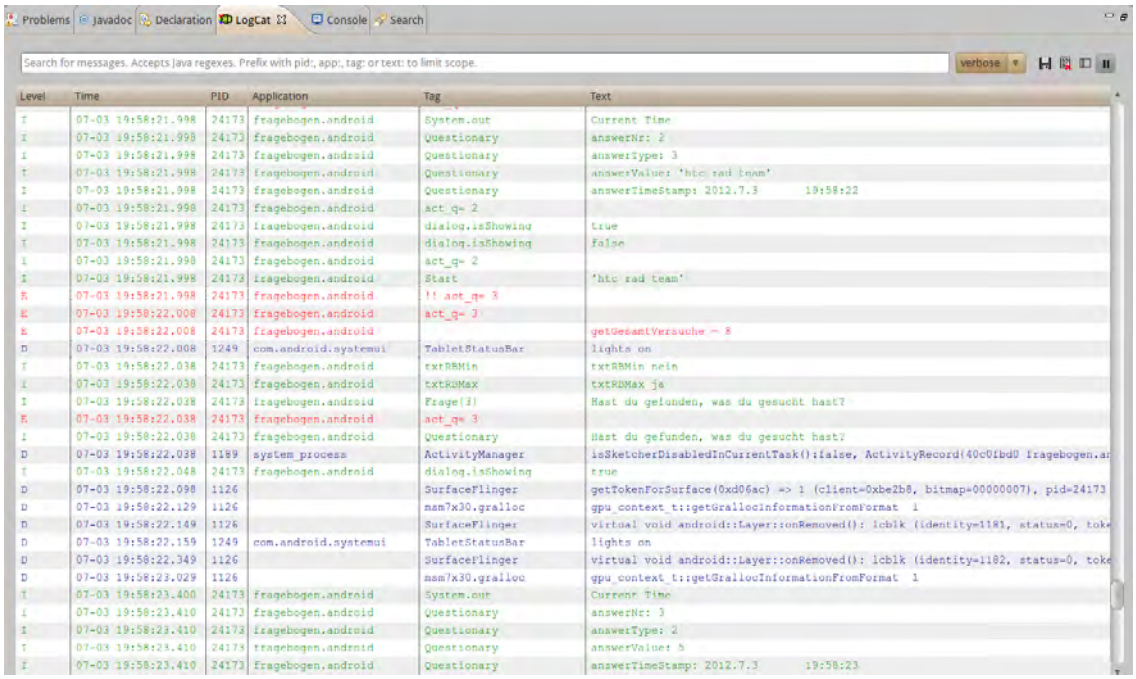

Figure 6.12: Screenshot from a logcat-window in the Eclipse IDE

Longterm tests should check if there are performance-, battery- or stability-issues in *Fragebogen*. Therefore the App had to be installed and used over a longer period of time- the longest test was 1 week without stopping the service or restarting the device. Because of energy-issues and the number of logCat-information that is produced in a few minutes, debugging an AVD or a device via USB could not be used for those kinds of tests. Therefore *Error2Log.java* has been implemented, it writes an error-message into the App-folder of *Fragebogen* and appends a timestamp in the filename. Listing 6 - 26 shows the head of it's most important method *write*.

Listing 6 - 26: method-head: error-logging to a file

```
public void write(StackTraceElement[] msg, String notice)
```

With the help of this method and class, special parts of the code can be observed for long-term-errors. In order to do this, the code with the possible exception has to be surrounded by try-catch and everytime an Exception happens in the try-block, it's stacktrace and a helpful notice can be written in an error-file through *Error2Log*. An example is shown in 6 - 27.

Listing 6 - 27: longer-observed code- surrounded by try-catch

```
private void testError() {
        try{
                /*observed code with possible errors,
                performance issues etc...*/
        }catch(Exception ule /* unknown longterm exception*/ ) {
                new Error2Log().write(ule.getStackTrace(),
                        "testError()-method:␣" + ule.toString());
        }
        /*finally
        {
                activities after catch AND try-block
        }*/
}
```

Longterm-tests included controlling the battery-stats, an App whose Service runs the whole time may drain the battery, but this could not be observed. Additionally, Services maybe cost performance and RAM, a reduction because of *Fragebogen* could also not be observed. (*Go Taskmanager* [73] was used to test and create the screenshots 6.13(a) and 6.13(b))

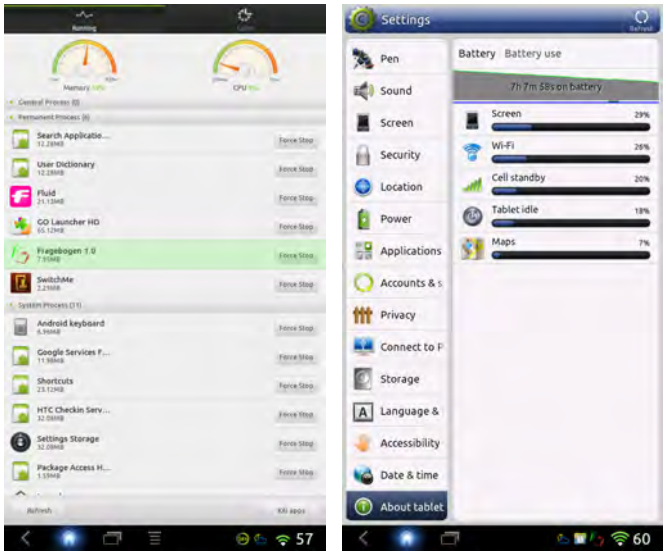

Figure 6.13: (a): screenshot of the Task-Analyzer (Fragebogen was highlighted in green), (b): screenshot of the battery-status-information

The first tests were done on a HTC Flyer with Android 3.2.1 and within an AVD, running 2.3.3. and undiscovered the issue with the Service as mentioned in requirements C1 and T6 5.1. During it's realization, *Fragebogen* was tested on the devices below:

Device	**Android-version**
HTC Wildfire	2
HTC Wildfire S	2
Sony Ericson Xperia Mini	2
Samsung Galaxy Y	2
HTC Flyer	3
Samsung Galaxy Nexus	4

All functions as well as the layout were tested on this devices, their created error-files were evaluated and with the help of USB-debugging the program-sequence has been checked. The screenshots 6.14(a), 6.14(b) and 6.14(c) are showing graphical layout-errors from an early state of the development.

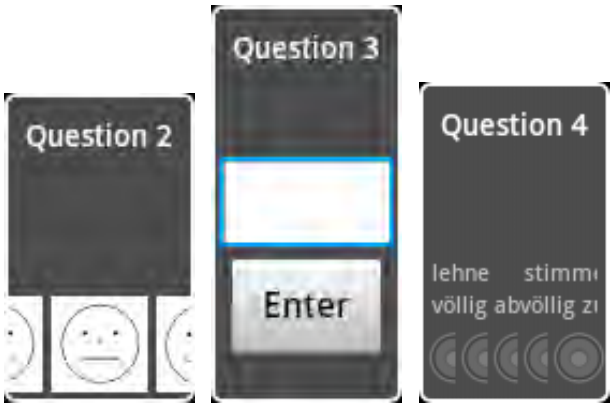

Figure 6.14: Screenshots of layout-problems, (a):smiley-scale too large for the view, (b): EditText too small for a normal anwser, (c):5-point likert scale is not correctly displayed

Android Virtual Devices do not support [74]:

- determining network connected state
- determining battery charge level and AC charging state
- determining SD card insert/eject

That is the main reason, why it has not been used frequently during the development- all those features are required in *Fragebogen*. Additionally, the number of test-devices seemed enough, the boot process took 1 minute and 33 seconds and the device itself lagged and was not very responsive.

Bibliography

[1] "Official android site." http://www.android.com, accessed: 2012-06-26.

[2] "Google play store." https://play.google.com/store, accessed: 2012-06-23.

[3] "Buxton collection IBM simon." http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/detail.aspx?id=40, accessed: 2012-06-30.

[4] "IBM Simon, IMG_0853.png (PNG image)." http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/a/m/IMG_0853.png, accessed: 2012-06-26.

[5] "Samsung Galaxy S3, de_GT-I9300MBDDBT_003_Right-Angle (JPEG image)." http://images.samsung.com/is/image/samsung/de_GT-I9300MBDDBT_003_Right-Angle?$Download-Source$, accessed: 2012-06-30.

[6] "Smartphone market hits All-Time quarterly high due to seasonal strength and wider variety of offerings, according to IDC - prUS23299912." http://www.idc.com/getdoc.jsp?containerId=prUS23299912, accessed: 2012-06-30.

[7] "Smartphone-Absatz steigt rasant (Presseinformationen) - BITKOM." http://www.bitkom.org/de/presse/8477_70921.aspx, accessed: 2012-06-30.

[8] "Cisco visual networking index: Global mobile data traffic forecast update, 2011–2016 [Visual networking index] - cisco systems." http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, accessed: 2012-06-30.

[9] "Apple - iPhone 4S - find over 500,000 apps on the app store.." http://www.apple.com/iphone/built-in-apps/app-store.html, accessed: 2012-06-27.

[10] "Apple wins injunction against samsung galaxy tab - the washington post." http://www.washingtonpost.com/business/technology/apple-wins-injunction-against-samsung-galaxy-tab/2012/06/27/gJQA4Ntk6V_story.html, accessed: 2012-06-30.

[11] "Nokia strikes out with patent suits - the washington post." http://www.washingtonpost.com/business/technology/nokia-strikes-out-with-patent-suits/2012/05/02/gIQAtGytwT_story.html, accessed: 2012-06-30.

[12] "Wikipedia:Meilensteine – wikipedia." http://de.wikipedia.org/wiki/Wikipedia:Meilensteine, accessed: 2012-06-30.

[13] "History of wikipedia - wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/History_of_Wikipedia, accessed: 2012-05-02.

[14] "Wikipedia.org site info." http://www.alexa.com/siteinfo/wikipedia.org, accessed: 2012-07-01.

[15] "App store - wikipedia mobile." http://itunes.apple.com/us/app/wikipedia-mobile/id324715238?mt=8, accessed: 2012-06-28.

[16] "Wikipedia mobil - apps in android market." https://market.android.com/details?id=org.wikipedia, acessed: 2012-05-20.

[17] "Help:Mobile access - wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Help:Mobile_access#Applications, accessed: 2012-06-20.

[18] *Mobile Projects/strategy - Meta.* https://meta.wikimedia.org/wiki/Mobile_Projects/strategy, accessed: 2012-06-30.

[19] "Wikipedia mobil - apps in android market." https://market.android.com/details?id=org.wikipedia, accessed: 2012-06-02.

[20] "Wikipedia beta - android apps on google play." https://play.google.com/store/apps/details?id=org.wikipedia.beta, accessed: 2012-06-30.

[21] J. M. Hektner, J. A. Schmidt, and M. Csikszentmihalyi, *Experience sampling method : measuring the quality of everyday life.* Thousand Oaks; London: SAGE, 2006.

[22] M. Hassenzahl and N. Sandweg, "From mental effort to perceived usability," in *MobileHCI '11 Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, p. 1283, ACM Press, 2004. http://portal.acm.org/citation.cfm?doid=985921.986044, accessed: 2012-07-01.

[23] C. N. Scollon, C. Kim-Prieto, and C. N. Scollon, "Experience sampling: Promises and pitfalls, strengths and weaknesses," *Journal of Happiness Studies*, vol. 4, no. 1, pp. 5–34, 2003. http://www.springerlink.com/openurl.asp?id=doi:10.1023/A:1023605205115, accessed: 2012-06-30.

[24] M. Csikszentmihalyi and J. Hunter, "Happiness in everyday life: The uses of experience sampling," *Journal of Happiness Studies*, vol. 4, pp. 185–199, June 2003. http://ideas.repec.org/a/spr/jhappi/v4y2003i2p185-199.html, accessed: 2012-07-01.

[25] J. Stiensmeier-Pelster and F. Rheinberg, *Diagnostik von Motivation und Selbstkonzept.* Göttingen [u.a.]: Hogrefe, 2003.

[26] L. Wheeler and H. T. Reis, "Self-Recording of everyday life events: Origins, types, and uses," *Journal of Personality*, vol. 59, pp. 339–354, Sept. 1991. http://doi.wiley.com/10.1111/j.1467-6494.1991.tb00252.x, accessed: 2012-06-20.

[27] A. Ahmad, J. Schneider, I. Nwadei, M. Darcy, R. Farr, R. Thal, and V. Arora, "A pilot study of Real-Time experience sampling method to evaluate student engagement in a global health rotation," *Education Research International*, vol. 2012, pp. 1–6, 2012. http://www.hindawi.com/journals/edu/2012/241874/, accessed: 2012-07-01.

[28] "Esp: The experience sampling program." http://www.experience-sampling.org, accessed: 2012-07-01.

[29] S. Möller, *Quality Engineering Qualität kommunikationstechnischer System.* Berlin: Springer, 2010. http://public.eblib.com/EBLPublic/PublicView.do?ptiID=571539, accessed: 2012-07-01.

[30] "Ergonomic requirements for office work with visual display terminals (VDTs) - part 11: Guidance on usability (ISO 9241-11:1998)," 1999.

[31] J. Nielsen, *Usability engineering.* Amsterdam: Morgan Kaufmann, 1993.

[32] M. Hassenzahl, *User Experience and Experience Design.* Aarhus, Denmark: The Interaction-Design.org Foundation, 2011. http://www.interaction-design.org/encyclopedia/user_experience_and_experience_design.html, accessed: 2012-07-01.

[33] "Ergonomics of human-system interaction – part 210: Human-centred design for interactive systems (ISO 9241-210)," 2010.

[34] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle," in *MobileHCI '11 Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, p. 47, ACM Press, 2011. http://dl.acm.org/citation.cfm?doid=2037373.2037383, accessed: 2012-05-02.

[35] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *IMC '11 Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, p. 329, ACM Press, 2011. http://dl.acm.org/citation.cfm?doid=2068816.2068847, accessed: 2012-07-04.

[36] S. Nylander, T. Lundquist, and A. Brännström, "At home and with computer access," in *CHI '09 Proceedings of the 27th international conference on Human factors in computing systems*, p. 1639, ACM Press, 2009. http://portal.acm.org/citation.cfm?doid=1518701.1518951, accessed: 2012-07-01.

[37] K. Church and N. Oliver, "Understanding mobile web and mobile search use in today's dynamic mobile landscape," in *MobileHCI '11 Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, p. 67, ACM Press, 2011. http://dl.acm.org/citation.cfm?doid=2037373.2037385, accessed: 2012-06-30.

[38] E. Karapanos, M. Hassenzahl, and J. Martens, "User experience over time," in *CHI EA '08 extended abstracts on Human factors in computing systems*, p. 3561, ACM Press, 2008.

[39] S. Kujala, V. Roto, K. Väänänen-Vainio-Mattila, and A. Sinnelä, "Identifying hedonic factors in Long-Term user experience," in *DPPI '11 Designing Pleasurable Products and Interfaces*, pp. 137–144, 2011.

[40] "Esp-screenshot, esp-question.gif (GIF image)." http://www.experience-sampling.org/pix/esp-question.gif, accessed: 2012-05-05.

[41] "Esp documentation, section: 7.2 what the data looks like." http://www.experience-sampling.org/esp.html#SEC89, accessed: 2012-07-01.

[42] "Pendragon software - palm OS database software home page." http://pendragonsoftware.com/, accessed: 2012-07-01.

[43] "RapidSMS / SMS application framework." http://www.rapidsms.org/, accessed: 2012-07-01.

[44] V. Khan, P. Markopoulos, B. Eggen, W. IJsselsteijn, and B. de Ruyter, "Reconexp: a way to reduce the data loss of the experiencing sampling method," in *MobileHCI '08 Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, p. 471, ACM Press, 2008. http://portal.acm.org/citation.cfm?doid=1409240.1409316, accessed: 2012-06-20.

[45] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, "MyExperience," in *MobiSys '07 Proceedings of the 5th international conference on Mobile systems, applications and services*, p. 57, ACM Press, 2007. http://portal.acm.org/citation.cfm?doid=1247660.1247670, accessed: 2012-06-30.

[46] "App store – „Mobile experience sampling"." http://itunes.apple.com/de/app/mobile-experience-sampling/id375547220?mt=8, accessed: 2012-07-01.

[47] "Strategy and organization [Prof. welpe]: iPhone." http://www.strategie.wi.tum.de/index.php?id=79&L=1, accessed: 2012-06-30.

[48] J. Hicks, N. Ramanathan, D. Kim, M. Monibi, J. Selsky, M. Hansen, and D. Estrin, "AndWellness," in *WH '10 Wireless Health 2010*, p. 34, ACM Press, 2010. http://portal.acm.org/citation.cfm?doid=1921081.1921087, accessed: 2012-04-20.

[49] K. Wac, S. Ickin, J. Hong, L. Janowski, M. Fiedler, and A. K. Dey, "Studying the experience of mobile applications used in different contexts of daily life," in *W-MUST '11 Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*, p. 7, ACM Press, 2011. http://dl.acm.org/citation.cfm?doid=2018602.2018605, accessed: 2012-07-01.

[50] D. Kahneman, "A survey method for characterizing daily life experience: The day reconstruction method," *Science*, vol. 306, pp. 1776–1780, Dec. 2004. http://www.sciencemag.org/cgi/doi/10.1126/science.1103572, accessed: 2012-07-01.

[51] "Google developers product page - android introduction." https://developers.google.com/android/, accessed: 2012-06-25.

[52] "Alliance members open handset alliance." http://www.openhandsetalliance.com, accessed: 2012-06-26.

[53] "Android ndk." http://developer.android.com/tools/sdk/ndk/index.html, accessed: 2012-06-26.

[54] "Android 2.2 compatibility definition," 2010. http://static.googleusercontent.com/external_content/untrusted_dlcp/source.android.com/de//compatibility/2.2/android-2.2-cdd.pdf, accessed: 2012-06-26.

[55] "Android compatibility android open source." http://source.android.com/compatibility/index.html, accessed 2012-06-26.

[56] "Android device gallery." http://www.android.com/devices/, accessed: 2012-06-26.

[57] "Dashboards | android developers." http://developer.android.com/about/dashboards/index.html, accessed: 2012-06-20.

[58] "Get started with publishing android developers." http://developer.android.com/distribute/googleplay/publish/register.html, accessed: 2012-06-23.

[59] "Amazon.com: Appstore for android." http://www.amazon.com/b?node=2350149011, accessed 2012-06-25.

[60] D. Bornstein, "Dalvik VM internals - 2008 google I/O session videos and slides." https://sites.google.com/site/io/dalvik-vm-internals, accessed: 2012-06-22.

[61] "Android system acchitecture, system-architecture.jpg (JPG image." http://androidteam.googlecode.com/files/system-architecture.jpg, accessed: 2012-07-06.

[62] "Building and running | android developers." http://developer.android.com/tools/building/index.html, accessed: 2012-06-30.

[63] "Android build, build-simplified.png (PNG image." http://developer.android.com/images/build-simplified.png, accessed: 2012-07-06.

[64] "Android developers guide." http://developer.android.com/guide/, accessed: 2012-06-25.

[65] "Services | android developers." http://developer.android.com/guide/components/services.html, accessed: 2012-07-02.

[66] "Lifecycle of android services, service_lifecycle.png (PNG image)." http://developer.android.com/images/service_lifecycle.png, accessed: 2012-06-07.

[67] "screens-ranges.png (PNG image)." http://developer.android.com/images/screens_support/screens-ranges.png, accessed: 2012-05-05.

[68] "SeekBar | android developers." http://developer.android.com/reference/android/widget/SeekBar.html, accesed: 2012-06-20.

[69] E. Gamma, *Design patterns : elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley, 1995.

[70] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, pp. 231–274, June 1987. http://linkinghub.elsevier.com/retrieve/pii/0167642387900359, accessed: 2012-06-13.

[71] "android-filechooser - [library] file chooser activity for android - google project hosting." http://code.google.com/p/android-filechooser/, accessed: 2012-06-30, used version: 3.4.

[72] "Using hardware devices | android developers." http://developer.android.com/tools/device.html#VendorIds, accessed: 2012-07-06.

[73] "GO TaskManager - android apps on google play." https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.taskmanager&hl=en, accessed: 2012-07-02, used version: 2.4.1 BETA to 2.5.2.

[74] "Using the android emulator | android developers." http://developer.android.com/tools/devices/emulator.html#limitations, accessed: 2012-07-07.

Attachment A

Questionnaire from the field test

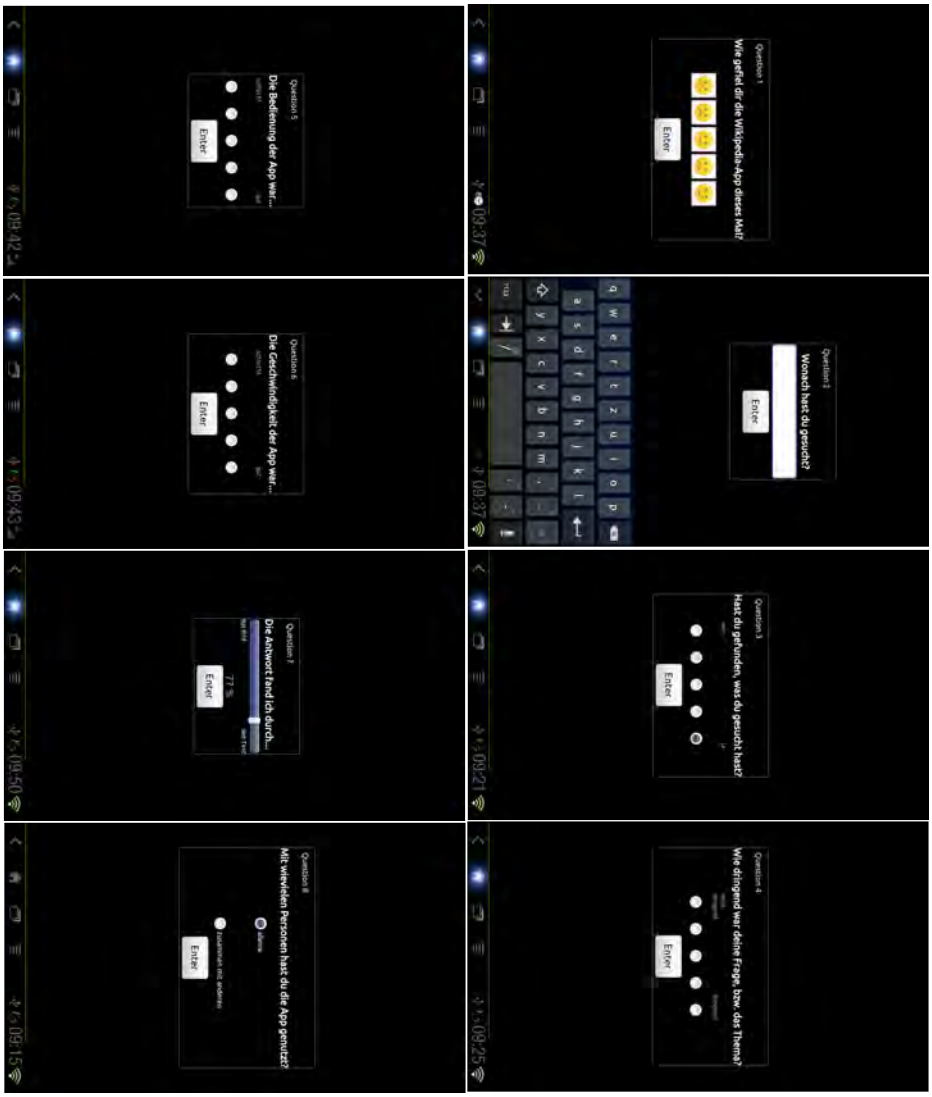

Figure A.1: questionnaire from the field test