HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

GSM Tester - Netzwerkqualität messen mit Android

Studienarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Herbstsemester 2010

Autor(en):	Stefan Schöb
	Reto Zigerlig
Betreuer:	Prof. Eduard Glatz
Projektpartner:	Network Communication and Security Lab GmbH
Experte:	Dr. Ulrich Fiedler
Gegenleser:	Prof. Dr. Peter Heinzmann

Inhaltsverzeichnis

1. Abstract

Abteilung	Informatik
Namen der Studierenden	Stefan Schöb | Reto Zigerlig
Studienjahr	HS 2010
Titel der Studienarbeit	GSM Tester - Netzwerkqualität messen mit Android
Examinatorin / Examinator	Prof. Eduard Glatz
Themengebiet	Internet-Technologien und -Anwendungen
Projektpartner	Network Communication and Security Lab GmbH

Mit dem GSM Tester wurde eine Android Applikation entwickelt, die dazu dient, Verbindungsprobleme und die Sprachqualität im Zusammenhang mit dem Mobilfunk zu ermitteln. Dabei werden zwei Varianten der Datengewinnung angewandt. Erstens werden technische Messungen durchgeführt, die folgende Parameter aufzeichnet:

- Lokationsdaten
- Start- und Endzeit von Gespräch
- Signalstärke während dem Gespräch
- Verbundene Cell Id
- Location Area Code
- Welcher Provider verwendet wurde
- Telefonnummer der Gesprächspartner (Anonymisierung der letzten vier Stellen)

Als zweite Möglichkeit der Datengewinnung wurde der Fokus auf die gefühlte Sprachqualität gelegt. Dazu wird dem Anwender nach einem Gespräch ein Bewertungsdialog angezeigt, mit dem er das getätigte Gespräch mit maximal drei Fragen bewerten kann. Die gewonnen Daten werden anschliessend an einen Server übertragen.

In der Analysephase des Projektes wurde ersichtlich, dass der Zugriff auf die GSM Daten unter Android sehr beschränkt ist. Diverse Parameter, die relevant für die Erkennung der Sprachqualität sind, befinden sich im Closed Source Part des Android Systems und sind nicht öffentlich verfügbar. Dies hat den Funktionsumfang der technischen Auswertung stark eingeschränkt. Aus diesem Grund wurde zu einem späteren Zeitpunkt die Aufzeichnung der Lokationsdaten implementiert, die eine genaue Lokalisierung von Verbindungsproblemen ermöglicht.

2. Aufgabenstellung

2.1. Ausgangssituation

Mobiltelefonbenutzer in der Schweiz werden öfters durch Verbindungsprobleme beeinträchtigt, wie z.B. nicht erreichbare Teilnehmer, mittendrin unterbrochene Telefonanrufe oder einfach schlechte Sprachqualität. Bis heute wurde jedoch noch keine systematische Untersuchung dieses Problems und diesbezüglicher Unterschiede zwischen Mobiltelefon-Dienstanbietern durchgeführt.

2.2. Aufgabe

GSM Tester ist eine Android Applikation, die hilft die gefühlte und technische Netzwerkqualität für Telefondienste verschiedener Provider zu messen. Einerseits zeichnet GSM Tester technische Daten zu Ereignissen wie „kein Empfang“, „Verbindungsabriss“ und „schlechte Sprachqualität“ auf, aus denen auf die technische Netzqualität rückgeschlossen werden kann. Andererseits befragt GSM Tester die Benutzer direkt nach dem Ereignis, wie sehr sie das Ereignis verärgert hat und erfasst so die gefühlte Netzqualität. Die Arbeit umfasst die Programmierung der GSM Tester App zur Datenerfassung für ein Android Smartphone und einer Ablage der erfassten Daten auf einem Server.

Die Arbeit umfasst folgende Teilaufgaben:

- Einarbeitung in Android
- Analyse des GSM Stacks auf mindestens einem ausgewählten Smartphone, welche technischen Daten einfach erfasst werden können.
- Ausarbeitung und Realisierung der GSM Tester App inklusive einer lokalen Speicherung von Resultaten
- Übertragung der lokal gespeicherten Resultate an einen Server via WLAN
- Funktionaler Test auf den Netzen verschiedener Provider

Die zu realisierende App soll u.a. folgende Funktionen unterstützen:

- Befragung des Benutzers nach einem Gespräch („wie war die Sprachqualität? Gut/mittel/schlecht“, „ wie stark sind Sie verärgert? Überhaupt nicht/ein wenig/sehr“)
- Ablegen der Befragungsergebnisse
- Transfer der gespeicherten Ergebnisse zum Server
- Soweit einfach möglich Aufzeichnen von Daten aus dem GSM Stack
- Optional kann die App auch für Datenverkehr erweitert werden

Referenzen:

- Foliensatz zu einem GSM-Messprojekt auf Anfrage
- Foliensatz aus dem Android Einführungskurs auf Anfrage, siehe http://www.ncslab.ch/leistungen/schulungen/index.html

3. Urheberschaftserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Ort, Datum:

Reto Zigerlig Stefan Schöb

4. Projektmanagement

4.1. Projektplan

Der ausführliche Projektplan ist im Anhang ersichtlich.

4.2. Arbeitsaufteilung

Wir haben zu Beginn des Projekts eine grobe Aufgabenaufteilung durchgeführt. Dies heisst nicht, dass die einzelnen Projektteilnehmer sich im Umfeld des anderen nicht auskennen, sondern soll lediglich Doppelspurigkeiten in der Entwicklung verhindern.

Wir unterscheiden bei der Aufteilung der Projekttasks die Analyse von der Implementation. Aufteilung Analysephase:

- Reto Zigerlig: Datenaufzeichnung / UI
- Stefan Schöb: Datenaufzeichnung / Datenupload

Da die Analyse der Datenaufzeichnung sehr viel Zeit in Anspruch genommen hat, haben wir uns hier beide damit beschäftigt.

Aufteilung Implementationsphase:

- Reto Zigerlig: UI
- Stefan Schöb: Datenaufzeichnung/Datenupload

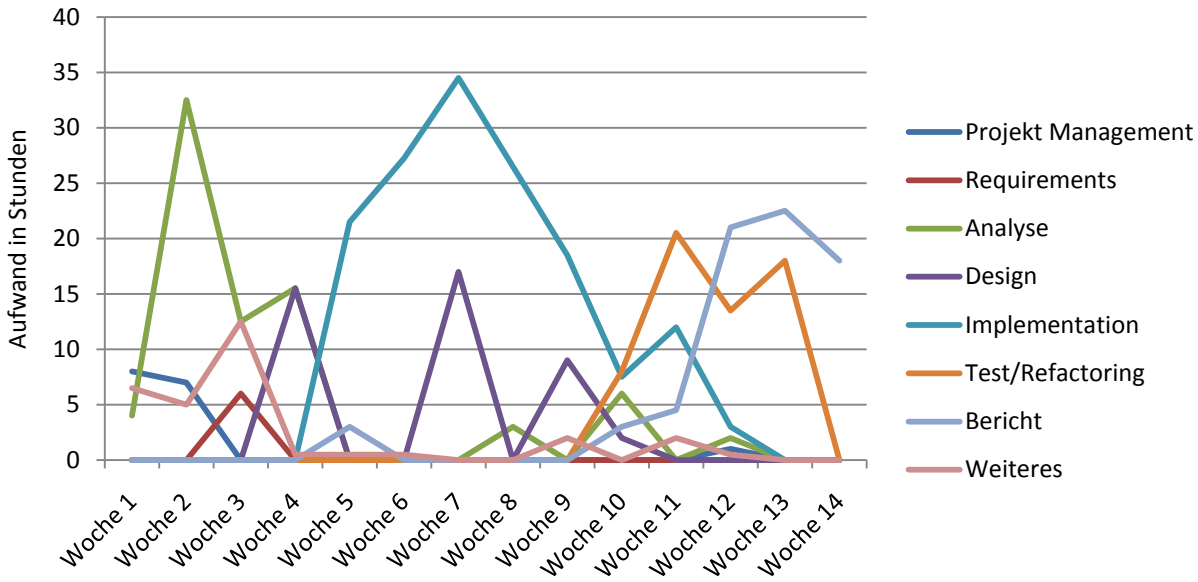
4.3. Arbeitsaufwand

Wir haben zu Beginn des Projekts einen Projektplan erstellt mit den durchzuführenden Tasks. Entsprechend wurden die 14 Wochen auf die Tasks aufgeteilt und eine grobe Zeitschätzung durchgeführt.

Der genaue Zeitplan kann dem Anhang entnommen werden

4.3.1. Ist pro Woche pro Task

In der folgenden Grafik kann man die Anzahl Stunden, welche wir für die Haupttasks pro Woche eingesetzt haben, sehen.

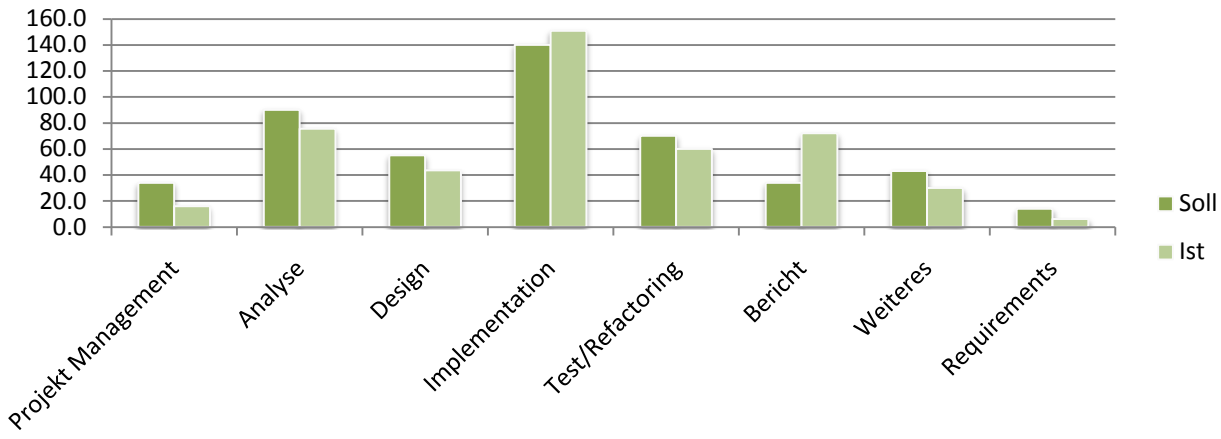


Man kann hier gut die einzelnen Peeks erkennen. Zu Beginn die Analyse, welche dann fliessend in das Design übergegangen ist. Nach diesen ersten beiden Haupttasks kommt der grosse Implementations-Block, welcher durch den Test/Refactoring-Block abgeschlossen wird. Zum Schluss wurde hauptsächlich am Bericht gearbeitet.

Während der Implementation haben wir bemerkt, dass unser erstes Datenbank-Design nicht funktionieren kann. Entsprechend haben wir hier einen weiteren Design-Peek, in dem wir ein neues Konzept ausgearbeitet haben.

4.3.2. Differenz Soll/Ist pro Task

Die folgende Grafik soll das Verhältnis zwischen Soll uns Ist-Stunden pro Haupttask zeigen.



Dieses Diagramm zeigt, wie viel Zeit wir für die einzelnen Tasks eingeplant haben und wie viel wir wirklich benötigt haben. Wir konnten dabei feststellen, dass wir bei den meisten Tasks die Zeit gut eingeschätzt haben. Zum Analyse-Task ist anzumerken, dass wir die 25 Stunden, welche wir in der Analyse des Datenuploads gespart haben, in die Analyse der GSM-Parameter einfliessen liessen.

Den Bericht haben wir im Umfang ein wenig unterschätzt. Jedoch haben wir viele Punkte, welche Dokumentiert wurden, schlussendlich unter diesen Task genommen, da die daraus entstandenen Dokumente direkt in den Bericht eingeflossen sind. Dazu zählt z.B. die Benutzeranleitung.

4.4. Persönliche Berichte

4.4.1. Stefan Schöb

Nach einer kurzen Phase der Einarbeitung in die Thematik, sowie einer ebenfalls kurzgehaltenen Anforderungsspezifikation, haben wir uns schnell an die Analyse der wichtigsten Punkte gemacht. Der anfängliche Optimismus wurde jedoch schnell durch Enttäuschung ersetzt. Wir kamen relativ schnell zum Schluss, dass Android zwar Klassen bietet, um gewisse GSM-Parameter auszulesen, diese aber fehler- bzw. lückenhaft sind. Wir haben uns dann in die Tiefen des Android-Source begeben, um dennoch einen Weg zu finden, an die nötigen Daten zu gelangen. Über diverse Infos, welche wir aus Foren erhalten haben, wurde uns mit der Zeit der gesamte Aufbau des Android Telephony-Stacks bekannt. Entsprechend konnten wir auch erkennen, dass es mit legalen Mitteln nicht möglich sein wird, an die Daten zu gelangen. Diese werden effektiv durch Android vor Applikationen von Drittherstellern geschützt.

Nach diesen Erkenntnissen und einer Besprechung betreffend weiterem Vorgehen, haben wir den Auftrag erhalten, das Beste aus der Situation zu machen und einfach die Daten aufzuzeichnen, welche uns zur Verfügung stehen. Nach einer kurzen Design-Phase, in welcher wir hauptsächlich überlegt haben, was wir benötigen, haben wir uns dann schnell an die Implementation gewagt. Die meisten Punkte waren schnell erledigt und konnten von der Liste gestrichen werden. Wir haben jedoch auch erkannt, dass unsere Planung im Datenbank-Bereich zu dürftig ausgefallen ist. Da wir hier immer wieder mit Problemen mit noch offener Datenbank usw. hatten, haben wir uns kurzerhand entschlossen, ein neues Datenbanksystem zu implementieren. Wie erwähnt, waren die Hauptfunktionalitäten zu diesem Zeitpunkt bereits in einer ersten Version implementiert. Wir legten nun unser Hauptaugenmerk auf die

Stabilisierung der gesamten Anwendung, sowie der Vereinfachung bzw. Verschönerung des User Interfaces. Der Upload der Daten stellte hier, wie immer, die Knacknuss im gesamten Projekt dar. Da diese Komponente von sehr vielen Faktoren (Verbindung über? Hintergrundtransfer erlaubt? Usw.) abhängig ist, zog sich der Stabilisierungsprozess enorm in die Länge.

Gut im Zeitplan erreichten wir einige Wochen vor Schluss eine Version, welche noch den einen oder anderen Bug aufwies, jedoch als erste Vorabversion mit Herr Fiedler besprochen werden konnte. Das Feedback konnten wir als durchaus positiv ansehen, wobei noch ein paar kleine Änderungen gemacht werden mussten.

Nachdem wir die Wünsche eingearbeitet hatten, ging Reto dazu über ein ausführliches Testszenario zu entwickeln, wobei ich gleichzeitig dem Code eine Schönheitskur verpasste. Die Systemtests zeigten dann auch noch einige Fehler auf, welche meist ohne grösseren Aufwand behoben werden konnten.

Die letzten beiden Wochen haben wir dann mit einer sauberen Dokumentation des Codes sowie der Architektur verbracht. Wir haben hier bewusst viel Zeit in eine saubere Architektur-Dokumentation gelegt, da wir von Herrn Fiedler darauf hingewiesen wurden, dass er die Applikation evtl. anpassen und weiterentwickeln will.

Aus der Arbeit konnte ich erneut viel über die Android-Plattform lernen. Jedoch muss ich eingestehen, dass wir durch den eher geringen Umfang der Applikation wohl nicht ganz so stark gefordert wurden wie erst angenommen. Dies wiederspiegelt sich sicherlich an der geleisteten Stundenanzahl, bei welcher wir die eigentlich für eine Studienarbeit geforderte Anzahl Stunden nicht ganz erreichen konnten.

4.4.2. Reto Zigerlig

Da wir letztes Semester für unser SE2-Projekt bereits eine Android Applikation entwickelt haben, und uns dies sehr gut gefallen hat, war für uns bereits während der Bewerbungsphase zu einer geeigneten Studienarbeit klar, dass wir die höchste Priorität auf die Projekte mit Android setzten. Da uns die Mobilkommunikation ebenfalls sehr interessierte, beschlossen wir, uns für die GSM Tester Arbeit mit der Priorität eins zu bewerben. Über den Zuschlag für die Arbeit freuten wir uns sehr und starteten somit top motiviert in die Studienarbeit.

Zu Beginn des Projektes wurde festgelegt, dass als erstes eine ausführliche Analyse durchgeführt werden muss. In dieser Phase wurden die Möglichkeiten, die Android bietet, um auf die GSM Parameter zugreifen zu können, analysiert und dokumentiert. Da die verfügbare Dokumentation in diesem Bereich sehr rar ist, mussten wir sehr viel Zeit in diese wichtige Phase investieren. Ebenfalls haben wir früh begonnen, die gewonnen Erkenntnisse in einem Prototyp zu implementieren. Konkret ermöglichte uns dies, bereits in einem frühen Stadium des Projektes die technischen Grundanforderungen wie das Erkennen eines Verbindungsabbruches ausführlich zu testen. Ebenfalls konnten wir dadurch untersuchen, welche GSM spezifischen Daten wir auslesen können. Diese Daten werden dazu benötigt, die Sprach- bzw. Verbindungsqualität abzuschätzen. Das Ergebnis der technischen Analyse war sehr ernüchternd. Wir stellten fest, dass wir lediglich an eine Hand voll Parameter rankommen und dass wir einen Verbindungsabbruch nicht zur benötigten Zeit klar erkennen können.

Diese Erkenntnis und unsere Schlussfolgerung daraus, dass nun unsere Arbeit keine Daseinsberechtigung habe, wirkten sich stark auf unsere Motivation aus, die sich zu diesem Zeitpunkt in einem Tiefpunkt befand. Als wir die Problematik unserem Betreuer mitteilten, wurde klar, dass wir nun die Aufgabe hatten, neue Möglichkeiten auszuarbeiten, mit der die Anwendung wieder mehr Funktionalität bietet. Diese fanden wir in den lokationsbezogenen Daten. Wir haben uns überlegt, dass es sehr interessant sein kann, wenn ein Verbindungsproblem auftritt und man dieses mittels Geo Lokationsdaten genau lokalisieren kann.

Mit neu gewonnener Motivation packten wir nun die anstehenden Aufgaben gerne an. Dazu gehörte auch die Gestaltung des Bewertungsdialoges, der dem Benutzer nach einem Gespräch angezeigt wird und der die gefühlte Sprachqualität ermittelt. Ebenfalls musste die Anwendung diverse Einstellungsmöglichkeiten bieten, mit denen der Benutzer die Applikation seinen Wünschen entsprechend anpassen kann. Dadurch, dass wir ein Grossteil der Funktionalität zu einem frühen Zeitpunkt implementieren konnten, waren wir in der Lage, die Applikation bei einer Besprechung mit Herrn Fiedler zu präsentieren und genau zu besprechen. Die gewonnen Erkenntnisse wurden anschliessend von uns direkt umgesetzt, um den Kundenwünschen gerecht zu werden.

Nachdem die Funktionalitäten der Applikation umgesetzt worden war, führten wir einen ausführlichen Systemtest durch, mit dem wir die Stabilität der Anwendung stark verbessern konnten. Gegen Ende des Projektes haben wir uns dann intensiv mit der Dokumentation der gesamten Anwendung beschäftigt. Auf Wunsch von Herr Fiedler erstellten wir sowohl eine ausführliche Code- und Architektur Dokumentation, als auch eine Benutzeranleitung, die dem User den Umgang mit GSM Tester vereinfachen soll.

Durch dieses Projekt konnte ich sowohl meine Fähigkeiten in der Erstellung von Android Anwendungen stark verbessern, als auch meine Projektmanagementskills verbessern. Der direkte Kundenkontakt machte dieses Projekt sehr realitätsgetreu und war ebenfalls ein sehr spannender Aspekt.

Abschliessend möchte ich mich bei meinem Projektpartner Stefan Schöb für die erfolgreiche Zusammenarbeit bedanken.

4.5. Projektreview

Die Aufgabe in diesem Projekt bestand daraus, eine Android Applikation zu entwickeln, die basierend auf technischen Werten und einer Userbefragung, Informationen für eine Messung der GSM Netzwerkqualität ermöglicht.

Da GSM Tester die verfügbaren technischen Parameter ausliest, der Benutzer zum getätigten Gespräch befragt wird und die Daten auf einen Server geladen werden, können wir rückblickend sagen, dass wir unser Ziel erreicht haben. Jedoch sind wir mit der Hoffnung in dieses Projekt gestartet, dass uns Android viel mehr technische Werte liefert, die uns genauen Aufschluss auf die Verbindungsqualität ergeben hätten.

In diesem Zusammenhang haben wir auch Alternativlösungen gesucht, die einen Zugriff auf die benötigten Parameter zulassen würden. Dies wäre durchaus ein Ansatz für weitere Tätigkeiten in diesem Bereich. Denn aus unserer Sicht wird sich die Situation auch in kommenden Android Versionen, betreffend RIL Zugriff, nicht erheblich ändern.

Der aktuelle Stand des GSM Testers ermöglicht es Herrn Fiedler, seinen Kunden eine lauffähige Version zu präsentieren und zu zeigen, was momentan möglich ist. Ebenfalls besteht die Möglichkeit, direkt auf Kundenwünsche zu reagieren und diese in der Applikation umzusetzen.

Durch die Arbeit in diesem Projekt stellten wir fest, dass Android durchaus noch einige Wünsche offen lässt, vor allem, wenn es um den erwähnten Zugriff auf technische Daten geht. Wir konnten unsere Fähigkeiten in der Programmierung von Android Applikation vertiefen und haben gelernt, wie man ein Projekt, das vom geplanten Projektverlauf abkommt, wieder in die richtige Richtung bringt.

4.6. Risikoanalyse

Risiko	Auswirkung	Massnahmen zur Vermeidung/Verminderung	max. Schaden[h]	Eintrittswahrscheinlichkeit	gewichteter Schaden [h]	Vorgehen bei Eintreffen
Direkter Zugriff auf GSM Informationen nicht möglich	Projekt muss auf bestehende Zugriffsmöglichkeiten beschränkt werden.	Ausführliche Analyse der Zugriffsmöglichkeiten	16	35 %	5.60	Für die technische Analyse der Verbindung werden die öffentlich verfügbaren Methoden verwendet
Erkennung der Unterbruchszenarien nicht möglich	Benutzer muss befragt werden, ob Verbindung unterbrochen wurde	Ausführliche Analyse der Möglichkeiten, um Verbindungsunterbrüche zu erkennen.	16	35 %	5.60	Dem Benutzer wird bei der Befragung eine zusätzliche Frage gestellt, ob die Verbindung unterbrochen wurde, oder ob sie bewusst getrennt wurde
Erkennung, ob verfügbares WLAN mit Internet verbunden ist, nicht möglich	User wird fälschlicherweise mit WLAN verbunden, das keinen Upload der Daten ermöglicht	Recherche der WLAN-Implementation von Android	12	20 %	2.40	Es wird lediglich erkannt, dass der Upload nicht funktioniert. Anschliessend kann dem User eine Meldung präsentiert werden, dieser kann sich somit mit einem anderen WLAN verbinden
Benachrichtigung eines getätigten Gespräches kann nicht abgefangen werden	Event eines Anrufes kann nicht erkannt werden	Eventhandling unter Android studieren	6	1 %	0.03	Projekt kann in diesem Rahmen nicht durchgeführt werden
			50	**91 %**	**13.63**	

4.7. Meilensteine

Hier befinden sich die von uns zu Beginn definierten Meilensteine, die im Projektplan angegeben sind. Ebenfalls wird eine kurze Beschreibung der einzelnen Meilensteine aufgeführt.

Meilenstein	Beschreibung	Semesterwoche
MS1	Projektplan	SW02 (01.10.10)
MS2	Anforderungen und Analyse	SW04 (08.10.10)
MS3	Architektur / Design (End of Elaboration)	SW06 (31.10.10)
MS4	Alpha Version GSMTester (Feature-Freeze)	SW10 (28.11.10)
MS5	Finale Version des GSMTester (Code-Freeze)	SW12 (12.12.10)
MS6	Abgabe	SW14 (23.12.10)

Projektplan:
Definition von Arbeitspaketen, Aufteilung auf Teammitglieder, Zeitplanung erstellt

Anforderungen und Analyse:
Spezifische Anforderungen, Usecases, technische Analyse

Architektur / Design:
Logischer Aufbau der Architektur, Designentscheidungen GUI/Datenhaltung/Backend

Alpha Version GSM Tester:
Funktionsfähige Version der Software, die alle Features enthält

Finale Version GSM Tester:
Stabile und überarbeitete Version

Abgabe:
Schlussabgabe der Software, inkl. Dokumentation

5. Requirements

5.1. Allgemeinde Beschreibung

5.1.1. Produkt Fokus

Heutzutage benutzt ein Grossteil der Schweizer Bevölkerung ein Mobiltelefon. Demzufolge ist fast jeder schon mal von einem Telefonunterbruch, kein Empfang Event oder sehr schlechter Sprachqualität betroffen gewesen. Diese Umstände sind noch nie flächendeckend über die verschiedenen Provider analysiert worden.

5.1.2. Produktfunktionen

GSM Tester ist eine Android Applikation, die es ermöglicht, Feedback zu getätigten Anrufen oder zu Netzwerkfehlern zu geben. Durch die Anwendung werden ebenfalls technische Daten zur Verbindung gesammelt und anschliessend gemeinsam mit den Feedbackdaten an einen Server geschickt.

5.1.3. Einschränkungen

Die Studienarbeit des GSM Testers beschränkt sich auf die Clientapplikation sowie eine rudimentäre Serverablage für den Dateiupload. Eine detaillierte Analyse der Daten ist nicht Teil dieser Arbeit.

GSM Tester wird für Android Smartphones programmiert und ist somit nur auf dieser Plattform lauffähig.

5.2. Spezifische Anforderungen

5.2.1. Bedienbarkeit

Es handelt sich um eine Befragungsapplikation. Der Benutzer ist prinzipiell abgeneigt, Zeit für so etwas zu investieren. Deshalb muss die Befragung möglichst kurz und schnell zu machen sein. Der Upload der Daten soll möglichst automatisch erfolgen, damit sich der Benutzer nicht darum kümmern muss.

Der GSM-Tester soll sich in der Befragung auf ein Maximum von vier Klicks durch den Benutzer beschränken. Der Upload wird standardmässig direkt nach der Befragung im Hintergrund ohne Benutzerinteraktion durchgeführt.

5.2.2. Verständlichkeit

Wenn der Benutzer nicht mit der Applikation zurechtkommt, wird er nie eine Befragung ausfüllen. Das heisst, der Benutzer darf in der Fragestellung sowie in den möglichen Antworten nicht überfordert werden. Der Benutzer soll die Fragen in seiner Sprache präsentiert bekommen.

Der GSM-Tester löst diese Anforderung, indem in den Befragungen keine für den „Standard-Android-User“ unbekannten Fachbegriffe verwendet werden. Technische Daten werden im Hintergrund aufgezeichnet und dem Benutzer nicht explizit gezeigt. Die Applikation ist zudem komplett Mehrsprachig implementiert. In einer ersten Version sind dabei English und Deutsch enthalten.

5.2.3. Performance

Der Benutzer möchte während einem Gespräch nicht gestört werden. Die Applikation muss deshalb so performant arbeiten, dass der Anruf dadurch nicht behindert wird.

Ein Anruf muss mit oder ohne Applikation genau gleich verhalten. In der Applikation wird ein aktives Pollen der Einstellungen verhindert und nur auf, durch das System gemeldete Änderungen, reagiert.

5.2.4. Konfigurierbarkeit

Die Applikation muss für den Benutzer anpassbar sein. Zum Beispiel möchte nicht jeder Benutzer die Daten automatisch auf den Server uploaden.

Der GSM-Tester bietet dafür eine Einstellungs-Seite, auf der sämtliche Einstellungen, die für den Benutzer interessant sind, konfiguriert werden können. Dazu gehören die folgenden Punkte: (Standardwerte Fett abgebildet)

- Automatischer Upload **ja**/nein
- Upload über Nur Wlan/**Beste Verbindung**
- GPS aktivieren → Link auf Aktivierungsseite von Android
- Häufigkeit Bereich zwischen 2 und 10 (**5**)
- Location aufzeichnen **ja**/nein

5.3. Use Case

5.3.1. Use Case Brief

5.3.1.1. Gespräch bewerten

Hans Bewertetgern möchte Informationen zu seinen Gesprächen an Drittpersonen zu Statistikzwecken weitergeben. Dafür hat er den GSM-Tester installiert. Die Applikation befragt den Benutzer nach einer zufälligen Anzahl von Anrufen über das gerade geführte Gespräch. Hans kann die simplen Fragen kurz und konkret beantworten und muss sich dann um keine weiteren Auswertungen mehr kümmern, da die Daten automatisch an einen Server übertragen werden.

5.3.1.2. Messdaten uploaden

Peter Misstrauisch bewertet seine Anrufe mit dem GSM-Tester. Er möchte jedoch nicht, dass seine Messdaten automatisch an Drittpersonen weitergegeben werden. Er hat deshalb die Einstellungen so verändert, dass die Messdaten nur noch direkt auf seinen Wunsch hin übertragen werden. Dazu kann er den GSM-Tester öffnen, die gewünschten Messberichte auswählen und direkt uploaden.

5.3.1.3. Daten aufzeichnen

Die Applikation soll ohne weitere Benutzerinteraktion GSM-Messwerte während eines Gesprächs aufzeichnen.

5.3.2. Use Case Fully Dressed

5.3.2.1. Gespräch bewerten

Umfang	Benutzer
Ebene	Anwenderziel
Primärakteur	Benutzer
Interesse & Absicht	Bewertung zu einem Gespräch abgeben
Häufigkeit	2x pro Woche (je nach Anzahl Gespräche)
Vorbedingung	• GSM-Tester läuft • Gespräch wurde erkannt • Gespräch wurde aufgezeichnet • Gespräch wurde als „zu bewerten“ eingestuft
Nachbedingung	• Befragungsdialog beendet • Befragungsergebnisse abgelegt

5.3.2.2. Messdaten uploaden

Umfang	Benutzer
Ebene	Anwenderziel
Primärakteur	Benutzer
Interesse & Absicht	Gespräche, die aufgezeichnet und noch nicht übertragen wurden, uploaden
Häufigkeit	1x pro Woche
Vorbedingung	• GSM-Tester läuft • Min. 1 Gespräch aufgezeichnet
Nachbedingung	• Gewählte Gespräche übertragen • Gespräche lokal als „bereits übertragen“ markiert

5.3.2.3. Daten aufzeichnen

Umfang	Applikation
Ebene	Anwenderziel
Primärakteur	Applikation
Interesse & Absicht	Während einem Gespräch werden die Daten im Hintergrund aufgezeichnet
Häufigkeit	Für jedes Gespräch
Vorbedingung	• GSM-Tester läuft
Nachbedingung	• Möglichkeit 1: Daten im System abgelegt • Möglichkeit 2: Daten an Befragungsscreen weitergegeben

6. Analyse

In diesem Abschnitt befinden sich die Resultate der technischen Analyse. In dieser Phase wurde analysiert, auf welche technischen Parameter wir Zugriff haben und wie unsere Anforderungen umgesetzt werden können. Dafür wurde die Android API (Version2.2) auf folgende Punkte untersucht:

- GSM-Parameter
- Verbindungserkennung
- WLAN-Verbindung

6.1. Analyse GSM-Parameter

Es wurde abgeklärt, welche Daten über die aktuelle GSM-Verbindung, uns die Android-Plattform zur Verfügung stellen kann.

6.1.1. Java API zum Android Telephony Stack

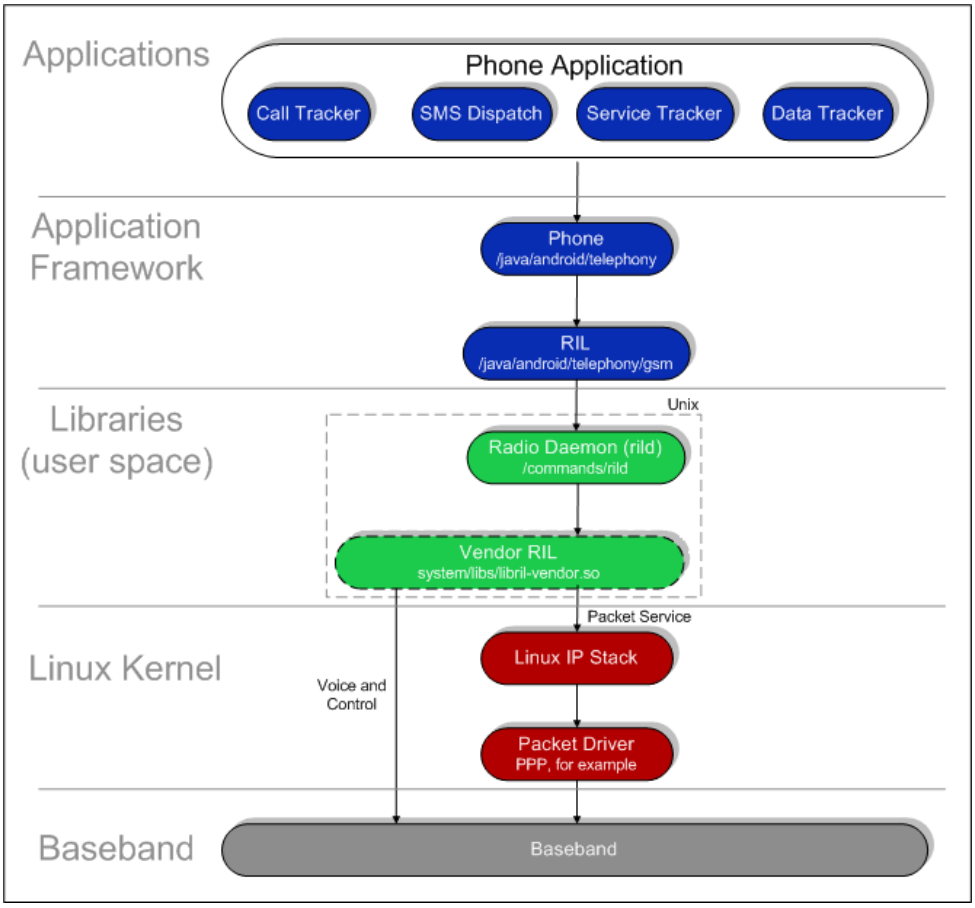


Abbildung 1: Android Komponenten des GSM-Prozesses

In diesem Bild werden die wichtigsten Komponenten dargestellt, welche im GSM-Prozess die Finger mit im Spiel haben. Bei den blauen Komponenten handelt es sich um in Java geschriebene Teilstücke. Grün kennzeichnet C++-Komponenten.

6.1.2. Radio Interface Layer RIL

Die sogenannte RIL bietet den Zugriff auf die Radio-Funktionalität des entsprechenden Smartphones. Dabei sind die folgenden drei Komponenten involviert:

6.1.2.1. RIL (Im Application Framework)

Hierbei handelt es sich um die Java-Schnittstelle in der Android-API, über welche mit den darunterliegenden Libraries kommuniziert wird. Sie befindet sich in folgendem Package:

com.android.internal.telephony

Wie der Name bereits sagt, handelt es sich um ein „Internal"-Package, welches von einer „normalen" Applikation nicht erreichbar ist. Ein kleiner Ausschnitt aus der Funktionalität, welche hier im gesamten Package zur Verfügung gestellt wird, kann eine Applikation über die öffentlichen Klassen im folgenden Package abfragen:

android.telephony

In diesem Package sind Klassen vorhanden, über welche die folgenden Parameter ermittelt werden können:

- Operator Name
- Operator Id
- Service State (In Service, Out of Service, Emergency Only, Power Off)
- Cell Id
- Location Area Id
- Bit Error Rate
- Signal Stärke
- Neighbor Cell Id

Genaue Klassendefinitionen sind unter folgendem Link ersichtlich:
http://developer.android.com/reference/android/telephony/package-summary.html

6.1.2.2. Analyse Framework API

Wir haben in diversen Tests die oben erwähnten Parameter überprüft. Die für die Gesprächsqualität informativen Parameter sind dabei:

- Bit Error Rate
- Signal Stärke

Die Bit Error Rate kann auf einem HTC-Desire mit Android 2.2 nicht ausgelesen werden. Es scheint hier ein Fehler auf Seiten von HTC/Android vorzuliegen. Die Methode liefert in jedem Fall den Wert -1 zurück.

Das entsprechende Verhalten wurde auch bereits im Bugtracking-System von Android eingetragen jedoch noch nicht approved:
http://code.google.com/p/android/issues/detail?id=9271

Die Signalstärke können wir mit Hilfe eines PhoneStateListener auslesen. Diesem können wir angeben, dass wir, sobald eine Änderung der Signalstärke auftritt, eine Benachrichtigung erhalten möchten. Der genaue Zeitpunkt des Versendens der Benachrichtigung kann nicht festgelegt werden, dies wird durch das Android System gesteuert.

Das Abfragen eines Verbindungsverlusts über die Signalstärke wird somit abhängig von den Benachrichtigungen des Systems. Dieses scheint jedoch eine Art Timeout zu implementieren, welcher erst abgewartet wird, bis ein SignalStrength-Update mit der Stärke 99 (kein Signal) gesendet wird. Dies macht es unmöglich, einen Verbindungsabbruch im Telefongespräch zu ermitteln, da dieser Timeout ca.

10 Sekunden beträgt, Android aber bereits nach ca. 2 Sekunden das Gespräch beendet. Zu diesem Zeitpunkt gehen wir noch davon aus, dass das Telefon mit der zuletzt gemeldeten Signalstärke verbunden ist.

6.1.2.3. RILd

Hierbei handelt es sich um den sogenannten RIL-Daemon. Dies ist ein kleines Programm, welches von Android beim Start des Betriebssystems mitgestartet wird. Es sorgt dafür, dass Events, welche in der Vendor-RIL auftreten, ins Framework weitergeleitet werden können. Weiter können über den Daemon Anfragen an die Vendor-RIL gestellt werden. Der Deamon ist aber so implementiert, dass er explizit nur dem Phone-Prozess erlaubt, direkt auf die darunter liegende Vendor RIL zuzugreifen. Diese Berechtigungsprüfung ist als ein einfacher String Vergleich von dem Namen des zugreifenden Prozesses mit einem Referenz-String implementiert.

6.1.2.4. Vendor RIL / Reference RIL

Die Vendor-RIL implementiert die eigentliche Kommunikation über die Radio-Schnittstelle (GSM). Android gibt dafür eine sogenannte „Reference-RIL“ mit. Über diese kann man erahnen, wie die einzelnen Hersteller (HTC, Motorolla, usw.) diese Vendor-RIL implementiert haben könnten. Die wirkliche Vendor-RIL ist jedoch closed-Source. D.h. es existiert momentan keine Dokumentation über die wirkliche Implementierung. Einzig die ril.h-Datei, welche die öffentliche Schnittstelle der Vendor-RIL definiert, kann einige Hinweise dazu geben.

6.1.3. Parameter

Für unsere Applikation sollen möglichst viele GSM-Parameter gesammelt werden. Wie bereits im Abschnitt 6.1.2.1. erläutert, stellt die Android-API eine kleine Anzahl von Parametern zur Verfügung. Für unsere Applikation währen jedoch noch weitere Parameter interessant. Während der Analyse sind wir auf folgenden Screenshot gestossen.

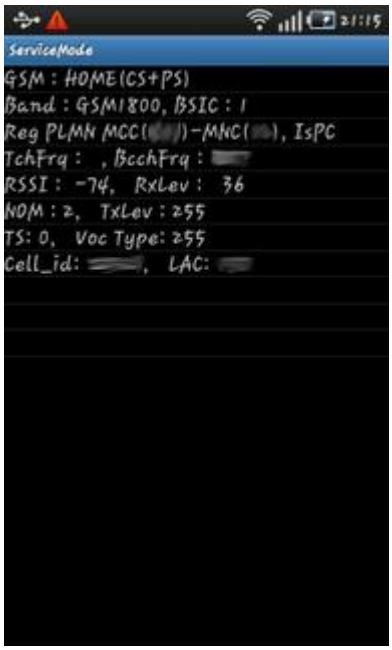


Abbildung 2: Samsung FieldTest-Applikation

In diesem Bild sieht man die Samsung FieldTest-Applikation welche weitere GSM-Parameter aus der aktuellen Verbindung auszulesen scheint. Dies funktioniert über den sogenannten Service-Mode. Dazu muss auf einem Samsung-Gerät die Nummer „*#*#197328640#*#*“ gewählt werden.

Diese Applikation zeigt, dass es möglich ist auf die GSM-Parameter zuzugreifen und diese abzufragen. Nachforschungen haben sich jedoch als sehr schwierig herausgestellt, da es in diesem Bereich überhaupt keine Dokumentation gibt und die Vendor-RIL nicht einsehbar ist.

6.1.3.1. OEM_RAW_REQUEST

Die FieldTest-Applikation verwendet zur Abfrage der entsprechenden Daten den sogenannten OEM_RAW_REQUEST. Sprich, es wird über den RIL-Deamon ein Request an die Vendor-RIL geschickt mit dem Code OEM_RAW_REQUEST(59). Dieser Request kann von jedem Gerätehersteller mit eigenen Informationen beantwortet werden. Recherchen haben ergeben, dass Samsung wie auch HTC hier je nach mitgegebenen Parametern, Informationen über die GSM-Parameter preisgeben. Diese

Informationen kommen in einem Byte-Array zurück, welches man entsprechend Konvertieren und auswerten könnte.

Das Problem liegt hier darin, dass solche Requests nur vom Phone-Prozess abgesetzt werden können. Jegliche Abfragen aus einem anderen Prozess führen zu einem „Permission denied“. Dies ist explizit so in der RIL programmiert und soll dafür sorgen, dass sich keine bösartige Drittsoftware in den Telefon-Prozess einklinkt.

6.1.3.2. Internal Android API

Die interne Telephony-API von Android bietet Zugriff auf erweiterte Parameter bzw. Methoden. Über diese wäre es möglich, die im Abschnitt 6.1.3.1. behandelten OEM_RAW_REQUESTS abzusetzen. Weiter könnten genaue Informationen über den letzten Trennungsgrund abgefragt werden. Auf diese API kann jedoch nicht zugegriffen werden. Bei unserer Recherche im Internet sind wir jedoch auf folgenden Workaround gestossen, um trotzdem an die Informationen zu gelangen:
http://groups.google.com/group/android-platform/browse_thread/thread/b55c8d3275ed7042/ad063c0a517a7a5f

(Siehe Post von chrisnew)

Dadurch wird es möglich, die eigene Applikation im phone-Prozess laufen zu lassen. Dies wiederspricht jedoch jeglichen Grundsatzentscheidungen der Architektur von Android und ist aus Sicherheitsgründen auch nicht so vorgesehen.

Als weiteren Workaround besteht folgende Möglichkeit: Um auf die API zugreifen zu können, muss man das Gerät rooten und eine eigene Firmware auf dem Android Gerät installieren. Dies bedeutet, dass man sich nur auf ein Gerät beschränken kann und somit die Applikation auch nur auf einem Gerät läuft.

Aus den oben genannten Gründen wurden diese Ansätze von uns nicht weiter in Betracht gezogen.

6.2. Gesprächserkennung

Die Applikation muss sämtliche Telefongespräche erkennen. D.h. sobald eine eingehende/ausgehende Verbindung aufgebaut wird, muss die Applikation die entsprechenden Daten auswerten. Weiter muss erkannt werden, wenn vom Gerät aus keine Verbindung infolge zu schlechtem Empfang hergestellt werden konnte.

Android bietet dafür in der Klasse TelephonyManager die Möglichkeit, sich an das Event „onPhoneStateChanged“ anzuhängen. Dieses wird ausgelöst, sobald ein Wechsel der Konnektivität stattfindet. Sprich, sobald der Benutzer eine Verbindung aufgebaut hat (einkommend oder ausgehend ist dabei egal)

6.2.1. PhoneState

Folgendes Diagramm illustriert, wie sich die Phone-States untereinander verhalten bzw., was nötigt ist, um von einem State in einen anderen zu gelangen:

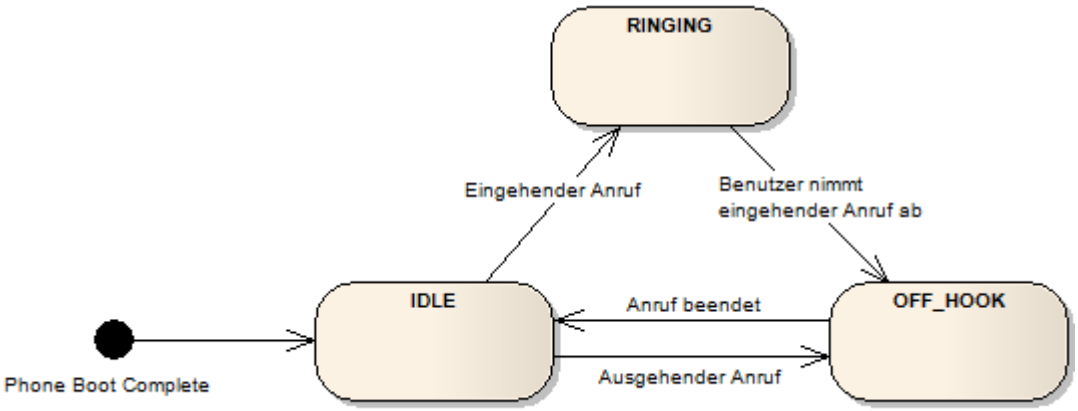


Abbildung 3: PhoneState Diagramm

6.3. Analyse WLAN-Verbindung

Android selbst handelt sämtliche WLAN-Aktivitäten. Sprich, die Auswahl eines WLANs muss nicht in der Applikation selbst implementiert werden.

6.4. Location-Parameter

Nach Anfrage von Herr Fiedler sind wir auf den Gedanken gekommen, dass allfällige Informationen zum aktuellen Standort des Users für eine Auswertung der Daten durchaus interessant wären. Aus diesem Grund haben wir uns mit den Möglichkeiten, lokationsbezogene Daten zu erlangen, beschäftigt.

Die Möglichkeit, die Lokation mittels GPS zu ermitteln, ist nicht immer vorhanden. Wenn man sich in Gebäuden aufhält, oder das verwendete Gerät kein GPS-Modul besitzt, muss man auf die alternative Lösung über die GSM-Zellinformationen und Wi-Fi Signale zurückgreifen. Diese Möglichkeiten werden durch Android zur Verfügung gestellt.

6.4.1. Location auslesen

Grundsätzlich ist das Auslesen der Location kein Problem. Dazu kann man sich am entsprechenden System-Broadcast registrieren, über welchen man dann über Positionswechsel informiert wird.

Weitere Informationen dazu können der Klasse LocationManager entnommen werden:
http://developer.android.com/reference/android/location/LocationManager.html

6.4.2. Unterscheidung GPS, GSM & WiFi

Ob die aktuelle Lokation nun mittels GPS oder GSM&Wi-Fi ermittelt wurde, muss grundsätzlich nicht unterschieden werden. Man muss sich lediglich bewusst sein, dass man sich für beide Szenarien bei den entsprechenden Komponenten registrieren muss. Zusätzlich ist die Genauigkeit, sowie die Geschwindigkeit der erhaltenen Daten unterschiedlich. Um via GPS an die Geo-Daten zu gelangen, benötigt das System im Schnitt ca. 10s. Mittels GSM&Wi-Fi verläuft dies in ca. 5s. Die Genauigkeit bei GPS liegt durchschnittlich zwischen 3-10 Metern. Mittels GSM&Wi-Fi kann sich die Genauigkeit auf bis zu 2km verschlechtern. Aus diesem Grund sind die Daten des GPS klar zu bevorzugen.

6.4.3. GPS aktivieren

Damit zu einem Telefongespräch die jeweiligen GPS-Parameter übertragen werden können, muss zuerst das GPS Modul im Smartphone aktiviert werden.

Dies ist programmtechnisch nicht möglich. Die einzige Möglichkeit, GPS zu aktivieren, ist das manuelle aktivieren durch den Benutzer.

Für ein Programm bietet sich die Möglichkeit den Settings-Dialog von Android aufzurufen, in welchem der Benutzer GPS aktivieren kann. Dazu kann folgende Action verwendet werden:

```
android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS
```

Warum das programmtechnische Aktivieren nicht möglich ist, wird durch folgendes Zitat erläutert:

"It is disabled for privacy reasons. If the user wants GPS off, the user should have GPS off, period"[1]

[1] http://stackoverflow.com/questions/1051649/how-to-programmatically-enable-gps-in-android-cupcake
Letzter Zugriff am: 22.12.10

Wichtig für den Benutzer, wäre hier zu erwähnen, dass GPS nur Strom verbraucht, wenn auch ein Prozess oder eine Activity GPS Daten benötigt. D.h. Android deaktiviert das GPS Modul solange niemand über den LocationManager einen LocationListener anhängt.

6.5. Analyse Internetverbindung

Um einen Upload durchführen zu können, muss der GSM Tester prüfen, ob eine Verbindung mit dem Internet besteht. Es könnte sonst der Fall auftreten, dass Android mit einem WLAN verbindet und angibt, eine funktionstüchtige Verbindung zu haben. Das verbundene WLAN ist jedoch nicht mit dem Internet verbunden.

Android bietet dafür die folgende Methode:

ConnectionManager.requestRouteToHost(ConnType, Host)

Diese prüft, ob direkt eine Verbindung zum angegeben Host aufgebaut werden kann. Der erste Parameter gibt an, ob man dies über das „Mobile“-Netz (also EDGE, 3G, usw.) machen möchte, oder über „WIFI“ was der WLAN-Verbindung entspricht.

Tests haben ergeben, dass dies für den ConnType „Mobile“ gut funktioniert, doch leider für „WiFi“ immer false ergibt. Genauere Recherchen ergaben, dass diese Funktionalität auf Seiten von Android für das WiFi-Netz leider noch nicht implementiert ist und deshalb das Standardverhalten der Basisklasse ausgeführt wird (dies entspricht „return false“)

Als Workaround kann ein einfacher http-Request an eine Seite, welche sicherlich immer erreichbar ist (z.B. google.ch), abgesetzt werden. Falls dies funktioniert, existiert eine Internetverbindung.

6.6. Zusammenfassung

6.6.1. Was können wir

- NO-ACCESS kann erkannt werden.
  - Wenn versucht wird, einen Anruf zu starten, jedoch kein Netz vorhanden ist.
- Signal Stärke aufzeichnen
  - Die während dem Gespräch erhaltenen Signal Stärke-Updates können aufgezeichnet werden.
- Gesprächsdauer
  - Start- und Endzeit kann erkannt werden.
- Location kann ermittelt werden.
  - Die Lokation kann mittels GPS metergenau ermittelt werden.
  - Die Lokation kann mittels GSM/Wi-Fi auf ca. 2 km genau ermittelt werden.
- Internetverbindung prüfen

6.6.2. Was können wir nicht

- Grund warum die Verbindung beendet wurde
  - Wir können nicht erkennen, ob der Benutzer die Verbindung korrekt beendet hat, oder ob infolge eines Signalverlustes, die Verbindung verloren gegangen ist.
  - Unterscheidung zwischen local hangup und remote hangup
  - Aufzeichnung der Biterrorrate ist nicht möglich
  - GPS programmtechnisch aktivieren

7. Design

7.1. Domain Model

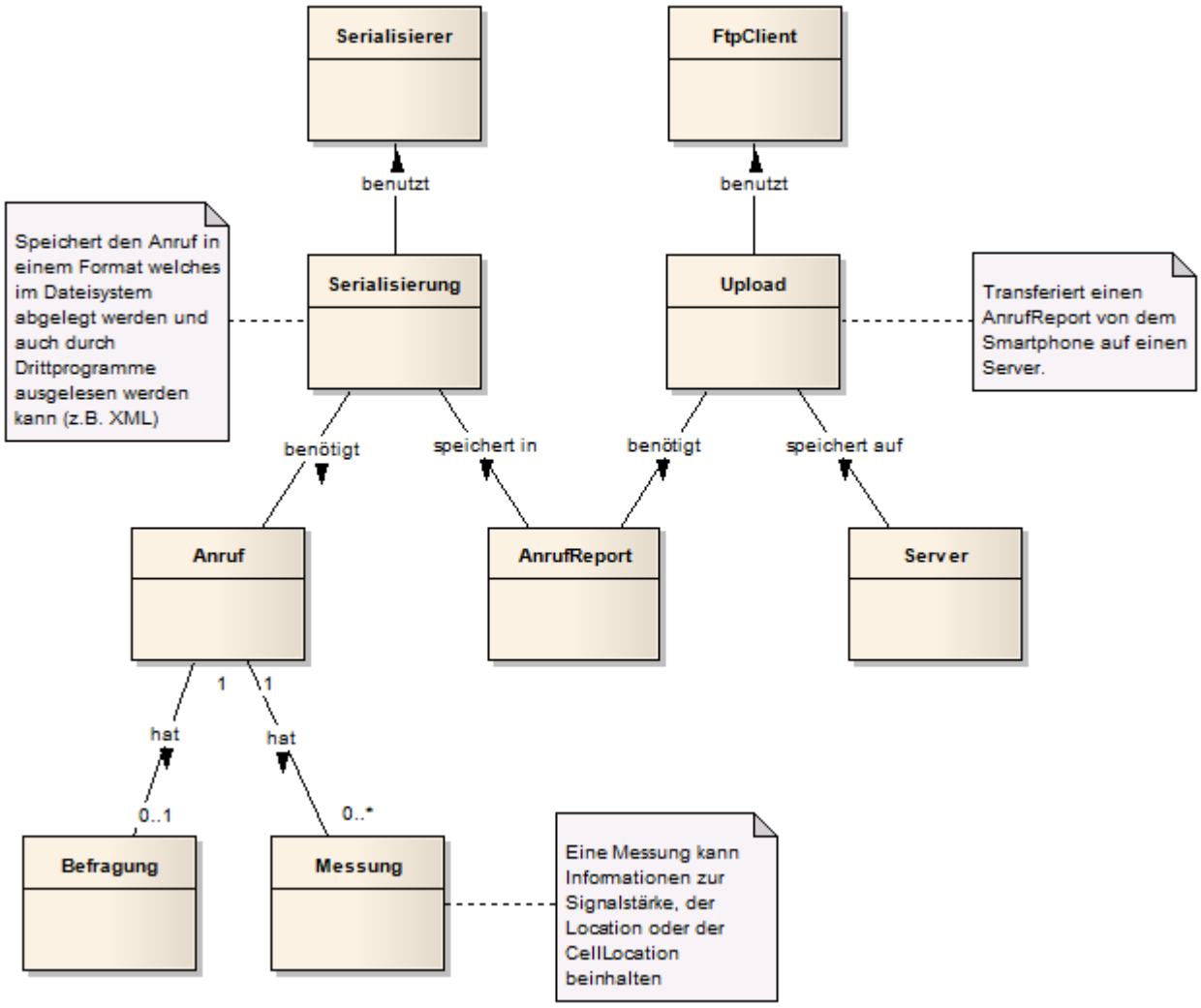


Abbildung 4: Domain Model

7.2. GUI Design

Hier befinden sich die Ergebnisse, die bei der Ausarbeitung von möglichen Varianten zur Gestaltung der Benutzerbefragung entstanden sind.

7.2.1. Variante 1

Die erste Variante der GUI Gestaltung basiert darauf, dass der User in einem Fluss durch die Anwendung navigiert. Dabei wird er automatisch durch die verschiedenen Views geführt. Auf jeder View befindet sich eine Frage mit den dazugehörigen Antwortmöglichkeiten. Sobald eine Antwort gewählt wurde, wird zur nächsten View gewechselt. Je nachdem, welche Antworten gewählt wurden, kann sich das auf die folgende Frage auswirken. Die Befragung kann zu jeder Zeit über den Button „Befragung abbrechen“ abgebrochen werden. Sobald die Letzte Frage beantwortet wurde, wird diese gespeichert und der User erhält eine kurze Meldung. Folgend wird ein Ablauf einer Gesprächsbeurteilung gezeigt:

Abbildung 5: GUI Design Variante 1

7.2.2. Variante 2

Bei der zweiten Variante wird dem User nur eine View zur Verfügung gestellt. Diese enthält zu Beginn alle Fragen. Unter den Fragen befinden sich jeweils Drop-Down Menüs. Beim Klick auf das jeweilige Menü öffnet sich ein Popup mit einer Radiogroup, die die möglichen Antworten enthält. Mit einem Klick auf die gewünschte Antwort kommt der User wieder zur Ausgangsview zurück. Um die Befragung zu speichern oder abzubrechen, befinden sich am unteren Rand zwei Buttons. Folgend wird ein Ablauf einer Gesprächsbeurteilung gezeigt:

Abbildung 6: GUI Design Variante 2

7.2.3. Entscheid

Die beschriebenen Varianten wurden dem Kunden für eine Beurteilung zugesandt. Ebenfalls haben wir unsere persönliche Meinung in Form des folgenden Vorschlages gemacht:

„Da wir eine Applikation erstellen möchten, die mit einer möglichst geringen Anzahl von Klicks zu bedienen sein soll, empfehlen wir die erste Variante des GUI-Designs zu wählen. Es ermöglicht eine einfache und intuitive Navigation durch die Applikation.

Darauf haben wir folgende Antwort vom Kunden erhalten:

„Bei den aktuellen Vorschlägen zum GUI bzw. zur Menuführung des GSM Tester kann ich Folgendes sagen.

- *Das Konzept hinter Vorschlag 1 halte ich für klar besser als das Konzept hinter Vorschlag 2*
- *Vorschlag 1 hat aber m.E. immer noch sehr viel Optimierungspotential*

Hierzu einige Gedanken:

- *Was halten Sie davon gleich auf ersten Bildschirmmaske oben einen Button zu machen: Gesprächsunterbruch o.ä. und darunter nach der Sprachqualität zu fragen?*
- *Darüber hinaus sind bei der Bewertung der Sprachqualität vor allem die Pole wichtig. Damit die User sich entscheiden, sollten MenuItems wie "mittel" weggelassen werden.*
- *Nach der Verärgerung sollte grundsätzlich nur gefragt werden, wenn etwas schief lief, also nicht nach Sprachqualität gut oder sehr gut.*
- *Das Feld unten in der Maske sollte eher "Kurz unterbrechen" als "Befragung abbrechen" heissen. Dann wird dem User Zeit gegeben um das Tram zu verlassen o.ä. und die Frage wird trotzdem beantwortet. Das halte ich bei max. 3 Klicks bis zum Befragungsende für zumutbar. Etwas anderes ist die Möglichkeit die Befragungsapplikation zu deinstallieren. Dies muss gegeben und klar dokumentiert sein. (ist nicht ihre Aufgabe innerhalb der Projektarbeit).“*[1]

Folgende Schlussfolgerungen haben sich daraus ergeben:

- Entscheid für Variante 1
- Fokus auf Extremwerte bei den Antworten
- Anpassung des Textes für den Unterbrechungsbutton

[1] Email von Ulrich Fiedler vom Do 28.10.2010 16:42

7.2.4. State Diagramm

Aus den Resultaten des GUI Designs entstand folgendes State Diagramm, das die verschiedenen States der Bewertung veranschaulicht:

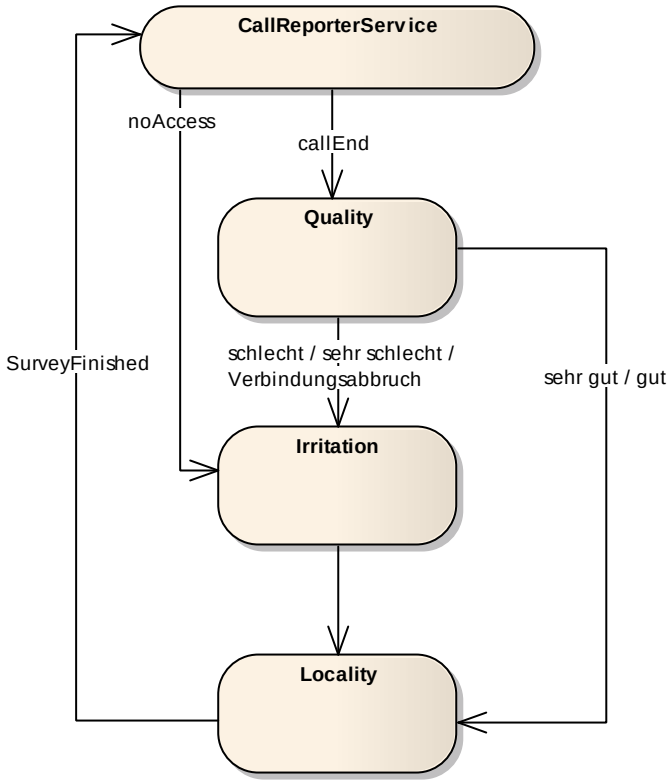


Abbildung 7: State Diagramm Bewertung

Sobald ein Anruf beendet wurde, erscheint die Frage nach der Sprachqualität. Wenn jedoch ein „no Access“ Szenario detektiert wird, wird direkt die Frage nach der Verärgerung gestellt. Wenn die Qualitätsfrage mit gut bzw. sehr gut beantwortet wurde, wird die Frage nach dem Ort, an dem sich der Benutzer während dem Gespräch befunden hat, gestellt. Falls jedoch die Qualitätsfrage mit schlecht bzw. sehr schlecht beantwortet wurde, wird auch hier die Frage nach der Verärgerung gestellt. Darauf folgt die Frage nach dem Ort. Sobald diese Frage beantwortet wurde, ist die Bewertung beendet.

7.2.5. GUI Gestaltung Verbindungsabbruch

Das Design und die GUI Navigation wurde ebenfalls an einer Besprechung mit Herrn Fiedler diskutiert. Daraus ist die Frage entstanden, wie die Eingabe des Users, ob es sich um einen Verbindungsabbruch oder um einen normal beendeten Anruf handelt, dargestellt werden soll. Anschliessend haben wir folgende Varianten erarbeitet und Herrn Fiedler zukommen lassen. Dieser wiederum hat die Varianten zur weiteren Analyse an einen Usability Psychologen weitergeleitet.

7.2.5.1. Variante 1: Zusätzliche Frage

Bei dieser Variante wird der Benutzer direkt gefragt, wie das Gespräch beendet worden ist. Wenn der User die Option „normal“ gewählt hat, wird die Frage nach der Sprachqualität gestellt. Falls er die Option „Abbruch“ gewählt hat, wird direkt nach der Verärgerung gefragt

Abbildung 8: Verbindungsabbruch Variante 1

7.2.5.2. Variante 2: Listelement in Sprachqualität

Bei der zweiten Variante wird nach jedem Call direkt nach der Sprachqualität gefragt. Dabei steht ein zusätzliches Element „Verbindungsabbruch“ zur Verfügung. Wenn der Benutzer „sehr gut“ oder „gut“ auswählt, wird gefragt, wo der User während dem Gespräch war. Falls er jedoch „schlecht“, „unverständlich“ oder „Verbindungsabbruch“ auswählt, wird zuerst die Frage nach der Verärgerung und anschliessend nach dem Ort während dem Gespräch gestellt.

Abbildung 9: Verbindungsabbruch Variante 2

7.2.5.3. Variante 3: Zusätzlicher Button in Sprachqualität

Auch bei dieser Variante wird nach jedem Gespräch nach der Sprachqualität gefragt. Nun wird dem User jedoch ein Button angezeigt, mit dem er den Verbindungsabbruch angeben kann. Der Ablauf ist dann derselbe, wie der bei der Variante 2.

Abbildung 10: Verbindungsabbruch Variante 3

7.2.6. Schlussfolgerung

Als Antwort erhielten wir ein ausführliches Statement, das im Anhang unter Kapitel 9.3. Kritik GUI Gestaltung Verbindungsabbruch betrachtet werden kann. Aus unserer Sicht wären die Variante 1 oder 2 die besten Lösungen gewesen, da in der Variante 3 dem Benutzer ein komplett neues Element zur Verfügung gestellt wird, das sonst so nirgends in der Applikation verwendet wird und somit eher verwirrt als Klarheit schafft. Nichts desto trotz sind wir dem Kundenwunsch sowie dem Statement des Usability Psychologen gefolgt und haben uns für die Variante 3 mit einem zusätzlichen Button entschieden.

8. Implementation

8.1. Einführung

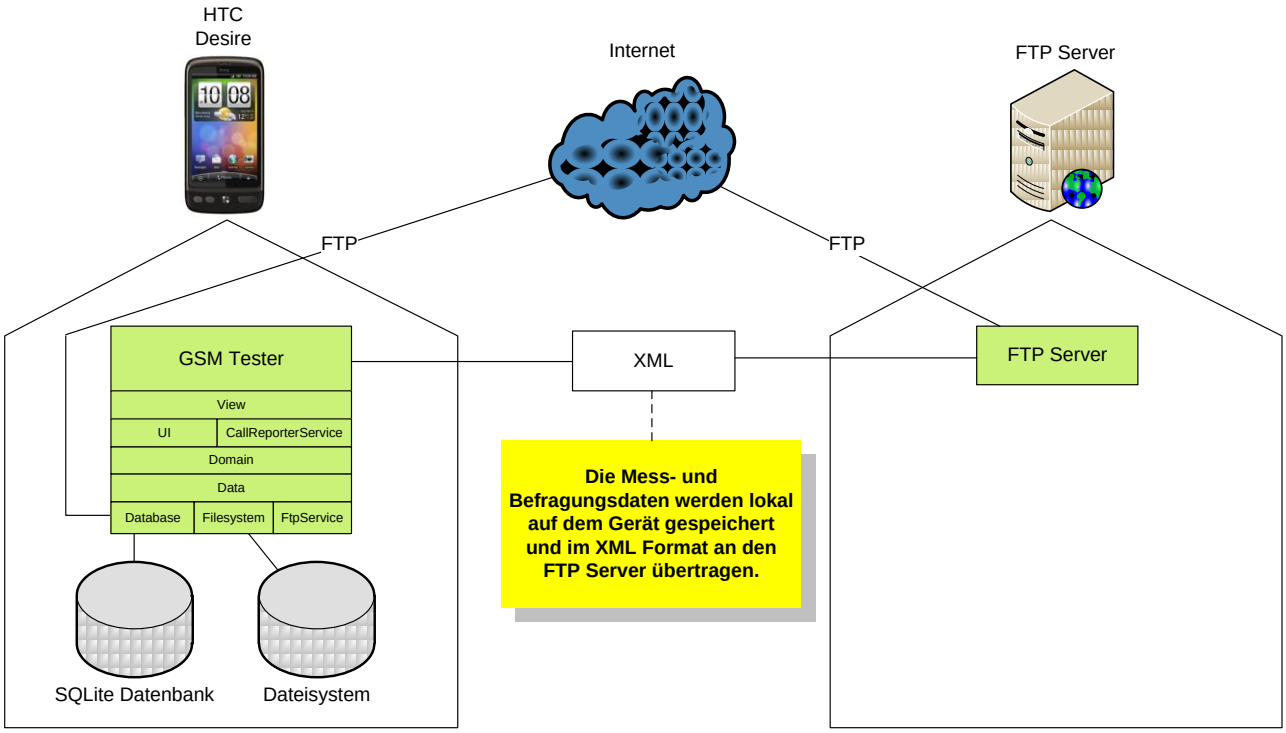


Abbildung 11: Architekturübersicht

Bei der GSM Tester Applikation handelt es sich um eine Clientapplikation. Diese wird auf einem Android Phone betreiben und beinhaltet die oben abgebildeten Elemente. GSM Tester zeichnet sowohl technische Daten als auch Befragungsdaten zur Mobilverbindung auf. Diese werden lokal gespeichert und anschliessend via FTP im XML Format an einen FTP Server über das Internet übertragen.

8.2. Architektur

8.2.1. Übersicht

Die Clientarchitektur besteht aus folgenden fünf Layern:

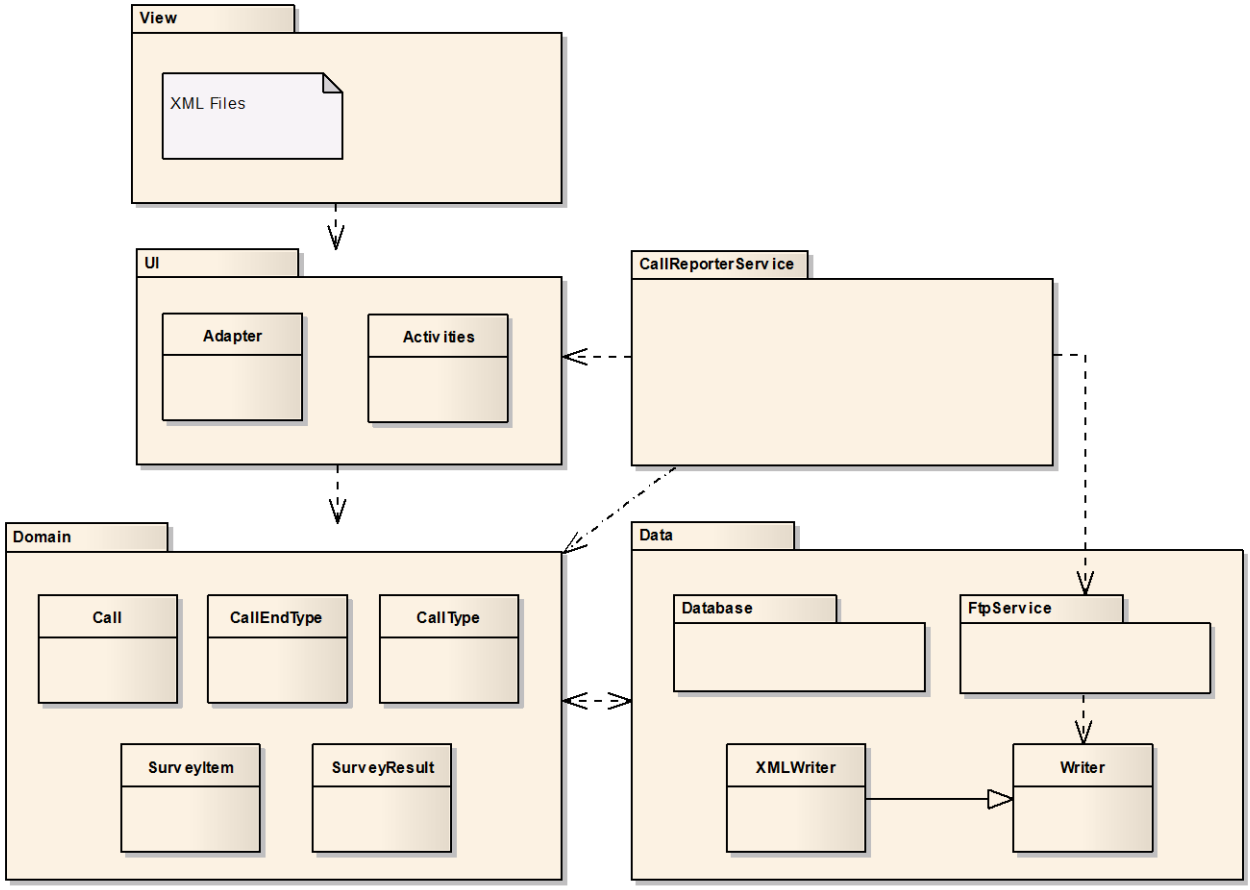


Abbildung 12: Client Architektur

8.2.1.1. View Layer

Der **View Layer** besteht lediglich aus den statischen XML-Dateien, die als UI von Android interpretiert werden. In dieser Schicht befindet sich kein Java Code.

8.2.1.2. UI Layer

Im **UI Layer** befinden sich die Activities, die den Ablauf der Applikation steuern. Hier werden die statischen Views mit Daten gefüllt und dynamische Anpassungen am UI vorgenommen. Ebenfalls befinden sich die Adapter in dieser Schicht. Diese stellen relevante Daten für die Anzeige bzw. Weiterverarbeitung zur Verfügung. Die benötigten Daten werden aus dem Domain Layer eruiert.

8.2.1.3. CallReporterService Layer

Die Aufgabe des **CallReporterService Layers** besteht darin, das Verhalten der Applikation auf die Call Events zu steuern. Je nach Event greift dieser Layer auf den UI Layer zu, oder gibt den Call an den FTPService im Data Layer weiter. Um diese Entscheidungen treffen zu können, werden die Elemente aus dem Domain Layer benötigt.

8.2.1.4. Domain Layer

Der **Domain Layer** enthält die gesamte Business Logik. Diese wird den übrigen Layern zur Verfügung gestellt. Für die Speicherung der Daten werden diese an den Data Layer weitergegeben.

8.2.1.5. Data Layer

Im **Data Layer** befindet sich sowohl die Datenbank als auch der FTPService, der für die Serialisierung auf den Writer zugreift. Die Datenbank ist für die lokale Persistenz Sicherung verantwortlich. Über den FTPService werden die lokalen Daten an einen FTP Server übermittelt.

8.2.2. Activitydiagramm

Das Activity Diagramm zeigt die relevanten Elemente und möglichen Abläufe durch die verschiedenen Activities. Für die StartActivity, die SurveyActivity und die ConfigActivity ist ebenfalls ersichtlich, dass die UI Gestaltung mittels main.xml, survey.xml bzw. preferences.xml vorgenommen wurde. Die schwarzen Pfeile zeigen mögliche Navigationspfade an, die grünen Pfeile visualisieren den Weg aus einer Activity oder View, wenn der Backbutton gedrückt wird. Für die HelpActivity wurde eine HTML Datei (help.html) erstellt. Diese enthält den kompletten Hilfetext.

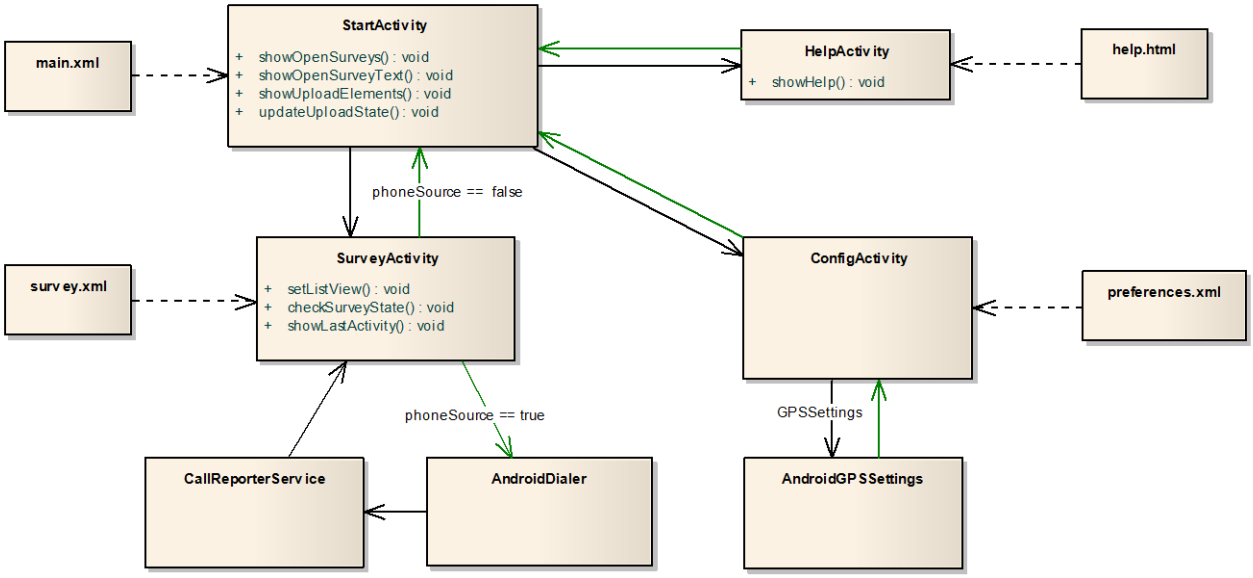


Abbildung 13: Activity Diagramm

8.2.3. Services

Android bietet für Hintergrundaktivitäten das Konzept von sogenannten Services an. Diese laufen, falls nicht anders markiert, im gleichen Prozess wie die Anwendung. Der Vorteil eines Service liegt darin, dass Android bei knappem Speicher zuerst andere Komponenten beendet um weitere Ressourcen zu erhalten. Der GSM Tester führt zwei verschiedene Hintergrundoperationen durch, welche in den folgenden Unterkapiteln genauer erläutert werden.

8.2.3.1. CallReporterService

Der CallReporterService sorgt dafür, dass ein Anruf erkannt wird und während einem Anruf die zur Verfügung stehenden Informationen aufgezeichnet werden.

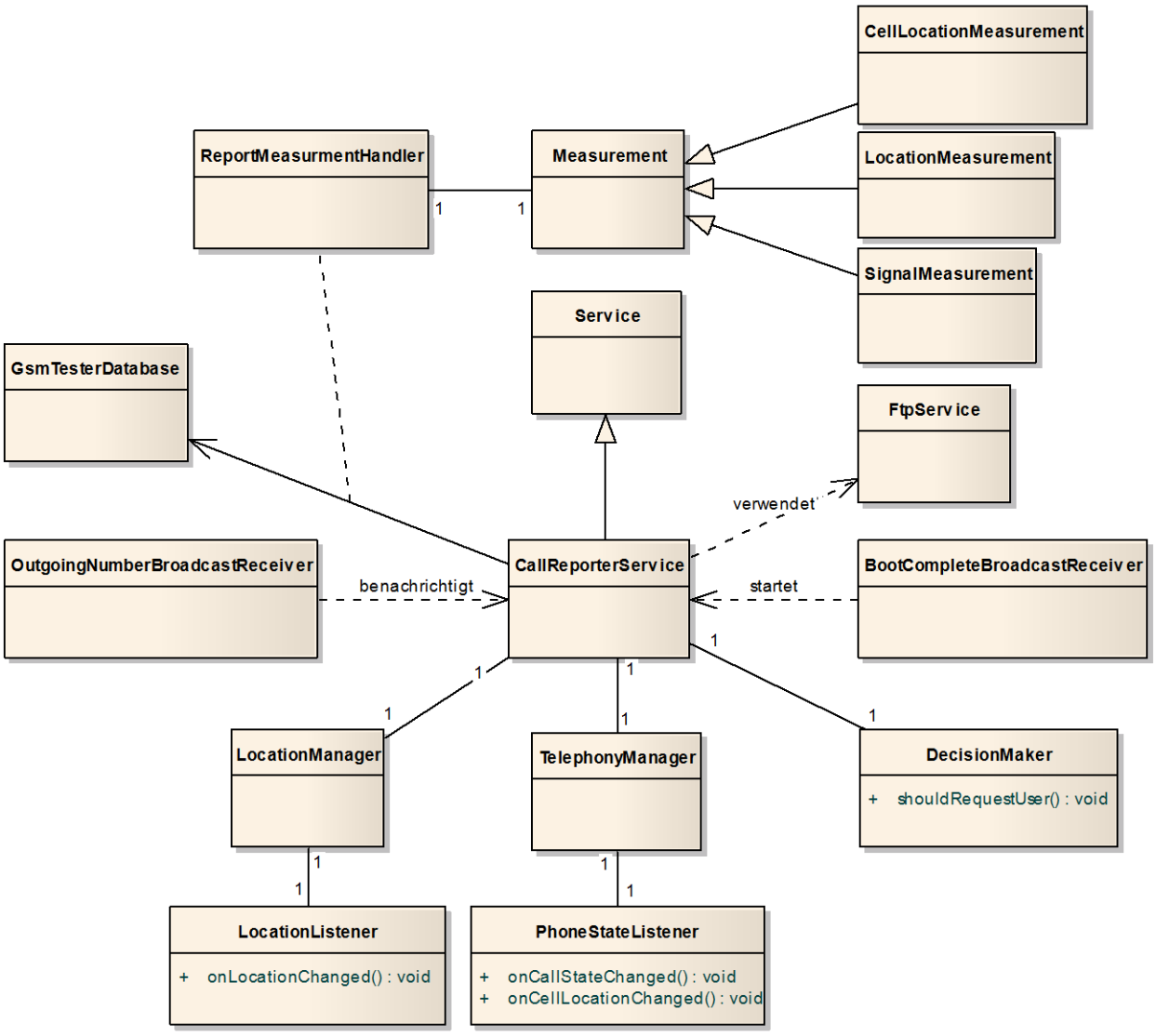


Abbildung 14: CallReporterService

Service

Der CallReporterService leitet von der Android-Klasse Service ab. Über diese erhält er die gesamte Funktionalität, um über die startService()-Methode des Android Context gestartet zu werden.

OutgoingNumberBroadcastReceiver

Wenn der Benutzer einen Anruf tätigt, ist die einzige Möglichkeit, die gewählte Nummer zu erhalten, sich am „NEW_OUTGOING_CALL“-Broadcast anzumelden. Dieser liefert in den Extras die gewählte Nummer welche vom broadcastReceiver an den CallReporterService gemeldet wird.

BootCompleteBroadcastReceiver

Der Service soll beim Start von Android automatisch gestartet werden falls der Service nicht in den Einstellungen explizit deaktiviert wurde. Die einzige Möglichkeit, eine Art „Autostart“-Service zu implementieren ist, den Service zu starten, sobald das System per „BOOT_COMPLETED“-Broadcast meldet, bereit zu sein.

Der BootCompleteBroadcastReceiver hängt sich genau an diesen Brodcast und ruft die startService()-Methode auf, falls die Einstellungen dies nicht verbieten.

LocationManager & LocationListener

Der LocationManager ist eine Klasse des Android Framework. Wir registrieren unseren eigenen LocationListener welcher vom LocationManager benachrichtigt wird, sobald neue Location-Daten vorhanden sind.

TelephonyManager & PhoneStateListener

Der TelephonyManager ist eine Klasse des Android Frameworks. Wir registrieren unseren PhoneStateListener, um folgende Ereignisse vom TelephonyManager zu erhalten:

- Wechsel des Phone-State (IDLE/RINGING/OFF_HOOK)
  - Über den PhoneState kann der CallReporterService den aktuellen Status eines Anrufes ermitteln. Sprich ob gerade ein Anruf läuft.
- Änderung der Signalstärke
  - Teilt dem CallReporterService mit, dass die Signalstärke geändert hat. Falls kein Anruf läuft, wird die letzte gemachte Messung zwischengespeichert. Falls die letzte Messung 99(entspricht kein Empfang) ist und ein PhoneState-Wechsel auf OFF_HOOK gemacht wird, können wir einen NO ACCESS ermitteln.
- Änderung der CellLocation
  - Wird nur während einem aktiven Gespräch als Messung in der Datenbank abgelegt → Siehe Measurement

DecisionMaker

Nach jedem Anruf soll entschieden werden, ob der Benutzer befragt werden soll oder nicht. Der DecisionMaker definiert hier die Strategie, nach welcher die Häufigkeit der Befragung durchgeführt wird.

Der GSM Tester hat zwei Strategien implementiert. Standardmässig kann der Benutzer in den Einstellungen die Häufigkeit der Befragung definieren. Der DecisionMaker zählt dann jeden Anruf und sobald die Häufigkeit erreicht wurde wird die Befragung gestartet. In der zweiten Strategie, welche momentan nicht benutzt wird, ermittelt der GSM Tester selbst eine zufällige Häufigkeit.

GsmTesterDatabase & ReportMeasurementHandler

Die Datenbank-Architektur die der GSM Tester verwendet wird im Kapitel 8.2.5. Datenbank genau erläutert. Der ReportMeasurementHandler speichert die übergebene Measurment-Klasse in der Datenbank.

Measurement, SignalMeasurement, LocationMeasurement, CellLocationMeasurement

Jede Messung, welche durch den CallReporterService respektive durch den LocationListener oder den CallStateListener gemacht wird, wird in ein Measurement-Objekt gekapselt. Dies kann dann über den ReportMeasurementHandler in die Datenbank gespeichert werden.

FtpService

Sollte der Benutzer in den Einstellungen den automatischen Upload aktiviert haben, ruft der CallReporterService nach Beendigung des Anrufs den FtpService auf, damit der Anruf automatisch zum Server gesendet wird.

Falls der DecisionMaker angibt, die Befragung durchzuführen, kann der Upload hier nicht automatisch durchgeführt werden, da noch die Resultate aus der Befragung hinzugefügt werden müssen. Siehe dazu Kapitel 8.2.4.3. SurveyActivity.

8.2.3.2. FtpService

Der FtpService führt eine Transaktion von Callreports auf einen Server im Hintergrund durch. Dabei muss zuerst der Anruf aus der Datenbank gelesen, in das Dateisystem gespeichert und anschliessend per FTP auf den konfigurierten Server transferiert werden.

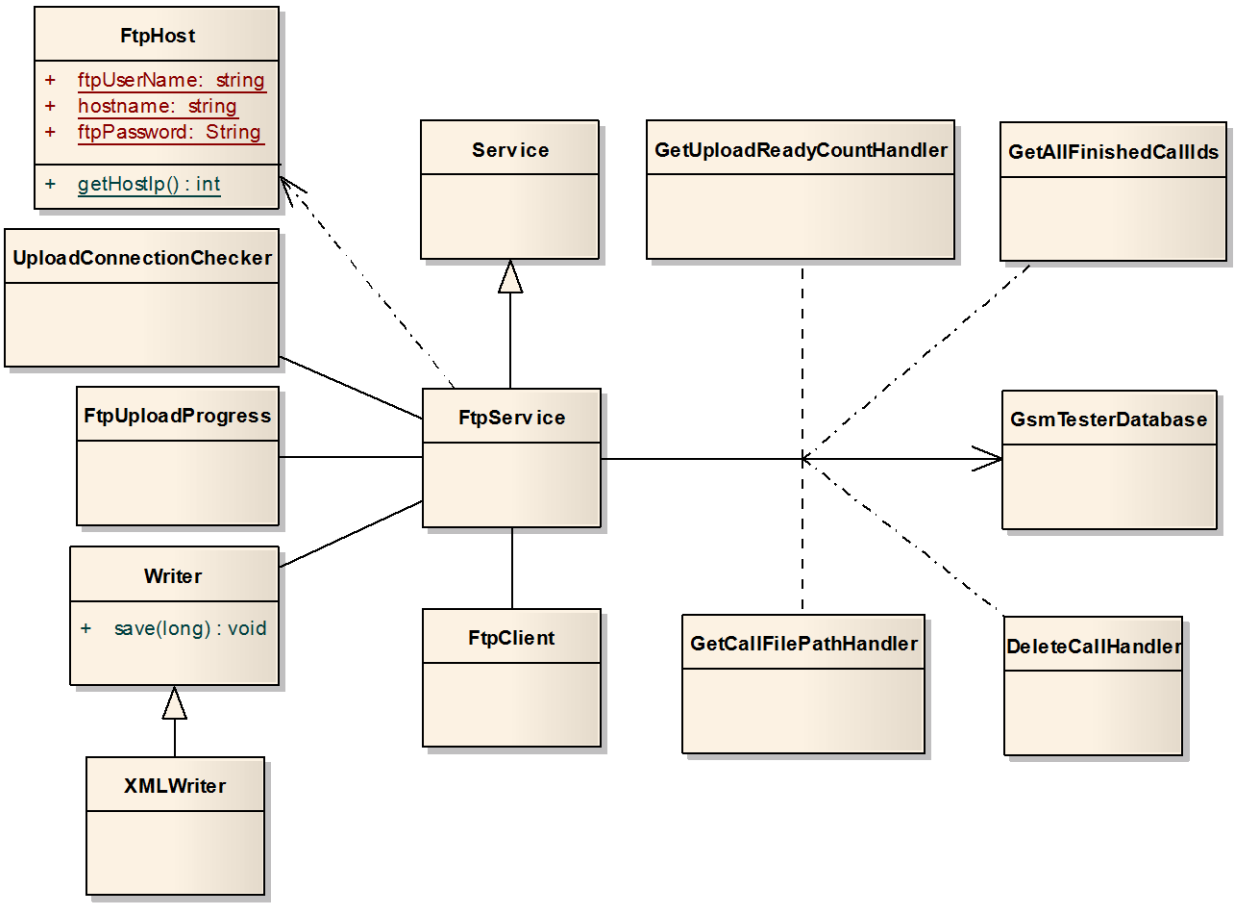


Abbildung 15: FtpService

Service

Der FtpService leitet von der Android-Klasse Service ab. Über diese erhält er die gesamte Funktionalität um über die startService()-Methode des Android Context gestartet zu werden.

FtpHost

Bietet Zugriff auf Zugangsdaten zu dem Ftp-Server sowie Hostname und IP des Servers.

UploadConnectionChecker

Stellt Informationen bereit, ob momentan ein Upload durchgeführt werden kann. Weitere Informationen dazu sind dem Kapitel 8.2.7. Verbindungsmanager zu entnehmen.

FtpUploadProgress

Der Fortschritt der Anrufreport-Generierung sowie des Dateiuploads wird vom FtpService im FtpUploadProgress gespeichert. Dieser stellt Methoden bereit, um den aktuellen Stand (Prozent) sowie die Uploadgeschwindigkeit abzufragen. Bei jeder Änderung des Fortschritts versendet der FtpService einen Broadcast, welcher von einer anderen Komponente (in unserem Fall der StartActivity) abgefangen werden kann, um den Fortschritt auf dem Bildschirm anzuzeigen.

Writer & XMLWriter

Um Anrufe, welche vom CallReporterService in der Datenbank gespeichert werden, in das Dateisystem abzulegen, hält der FtpService eine Instanz eines Writers. Weitere Informationen zum Writer sind dem Kapitel 8.2.6. Datenserialisierung zu entnehmen.

FtpClient

Eine Klasse aus dem externen ftp4J-Framework, welches wir für die FTP-Übertragung einsetzen. Diese handelt die ganzen FTP-Kommandos. Folgende FTP-Operationen werden ausgeführt:

- Verbindungsaufbau (Connect)
- Authentisierung (Login)
- Dateitransfer (Upload)

GsmTesterDatabase

Die Datenbank-Architektur, die der GSM Tester verwendet, wird im Punkt 8.2.5. Datenbank genau erläutert.

DeleteCallHandler

Nach einem erfolgreichen Upload muss der Anruf aus der Datenbank des GSM Tester gelöscht werden.

GetAllFinishedCallIds

Ermittelt vor dem Upload sämtliche Anruf-Id's, welche Datensätze in der Datenbank beschreiben, die zum Upload bereit sind.

GetUploadReadyCountHandler

Gibt die Anzahl zum Upload bereitstehender Datensätze zurück.

GetCallFilePathHandler

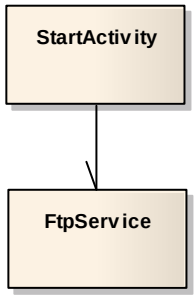
Gibt den Pfad zurück, an dem der Writer die Datei im Dateisystem gespeichert hat.

8.2.4. Activities

Der Zweck der Activities besteht darin, statisch definierte UI Elemente mit dynamischen Werten zu füllen und anzupassen. Die statische Definition findet in den dazugehörigen XML Dateien statt. Dies ermöglicht eine Trennung zu anderen Schichten und erhöht die Übersichtlichkeit immens.

8.2.4.1. StartActivity

Die StartActivity ist, wie der Name schon sagt, die erste Activity, die beim Programmstart aufgerufen wird. Das UI wird in der main.xml definiert. Die StartActivity ruft mit der Einstellung „Automatischer Upload" deaktiviert, den FtpService für den manuellen Upload auf.



8.2.4.2. HelpActivity

Die HelpActivity ist lediglich für die Darstellung der Hilfeseite verantwortlich. Sie wird nach einem Klick auf den Menübutton und das Hilfe Icon gestartet. Der Inhalt der Hilfeseite ist in der help.html Datei definiert.

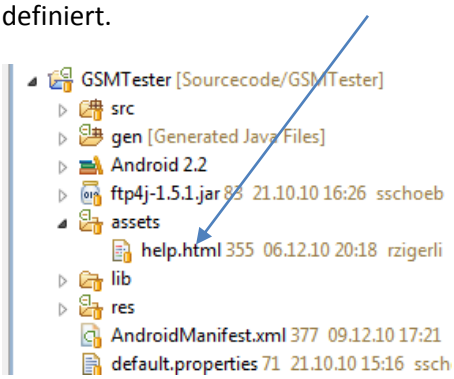
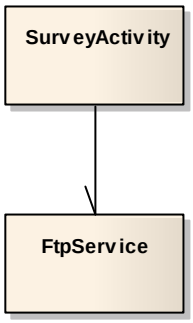


Abbildung 16: Ablageort help.html

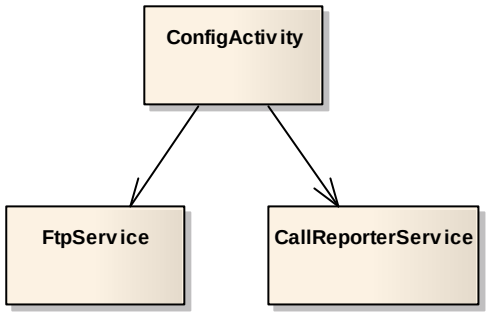
8.2.4.3. SurveyActivity

In der SurveyActivity befindet sich die Logik der Befragungsnavigation. Dies beinhaltet das Anzeigen der korrekten Frage und die dazugehörigen Antworten. Falls eine Befragung abgeschlossen wird, und die Einstellung „Automatischer Upload“ aktiviert ist, ruft die SurveyActivity den FtpService auf und übermittelt den Call inkl. Bewertungsdaten an den FTP Server.



8.2.4.4. ConfigActivity

Die ConfigActivity erlaubt es, die benutzerspezifischen Anpassungen der Applikation vorzunehmen. Sobald die Einstellung „Automatischer Upload“ aktiviert wird, ruft die ConfigActivity den FtpService auf. Wenn die Einstellung „GSM Tester Service aktiv“ geändert wird, wird der CallReportService mit dem jeweiligen Status angehalten oder gestartet.



8.2.5. Datenbank

Android bietet für jede Applikation die Möglichkeit einer eigenen Datenbank. Dabei handelt es sich um eine SQLite-Datenbank, welche im Data-Ordner der Applikation liegt. Diese Datenbank wird gelöscht falls die Applikation vom Gerät entfernt wird. Durch Ihren Speicherort wird sie aber vor Zugriff durch andere Programme automatisch geschützt.

8.2.5.1. SQL Datenbank

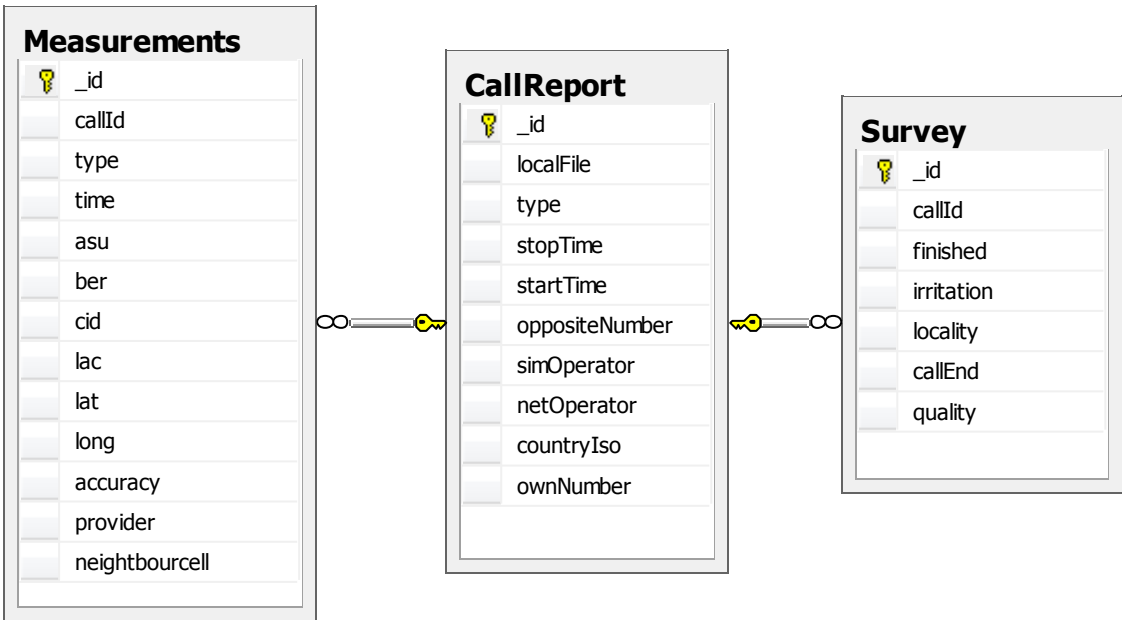


Abbildung 17: Datenbankschema

CallReport

Speichert allgemeine Informationen über einen aufgezeichneten Anruf. Für jeden Anruf der aufgezeichnet wird, existiert in dieser Tabelle ein Eintrag.

Measurements

Sämtliche Messungen zu einem Anruf werden in dieser Tabelle abgelegt. Je nach Typ der Messung (Signal, CellLocation, Location) werden nur die entsprechenden Spalten mit Daten abgefüllt.

Survey

Falls zu einem Anruf eine Befragung durchgeführt wird, werden die Resultate in dieser Tabelle gespeichert. Wir haben die Survey-Tabelle explizit von dem CallReport getrennt. Dies wäre aus technischen Gründen nicht nötig gewesen, jedoch wollten wir damit eine gewisse Unterscheidung zwischen diesen beiden Entities festigen.

8.2.5.2. Datenbankzugriff

Um den Zugriff auf die Datenbank aus dem ganzen System möglichst einfach zu gestalten, sowie die Datenintegrität und die Datenaktualität jederzeit zu gewährleisten, wurde die folgende Klassenarchitektur entwickelt:

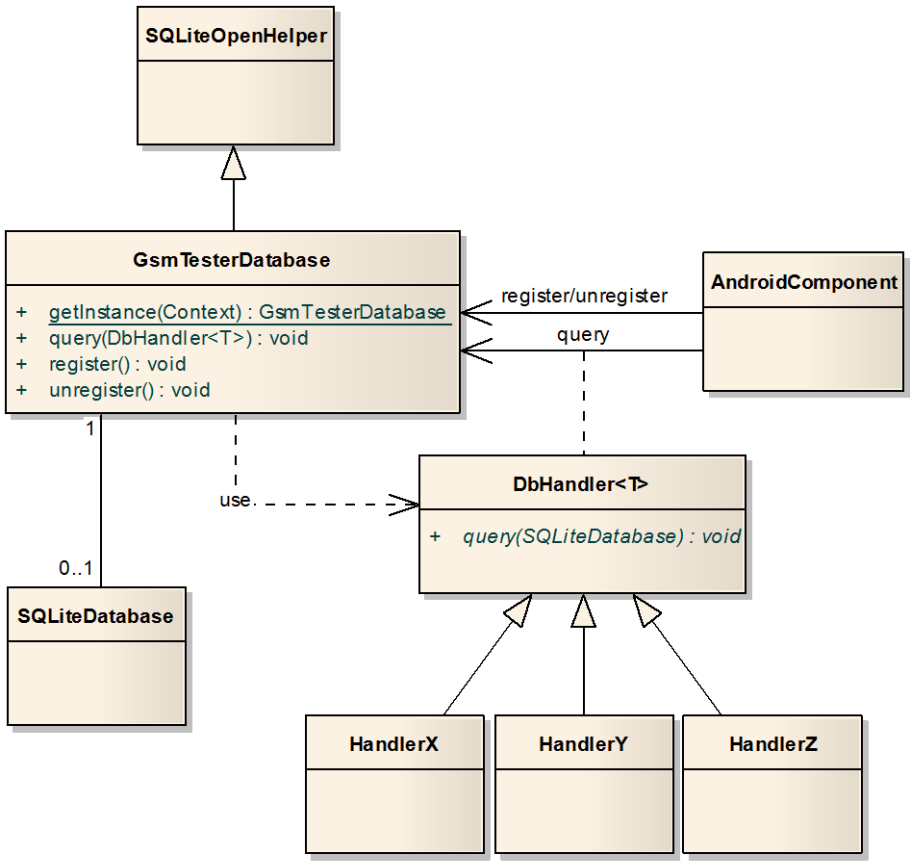


Abbildung 18: Datenbankzugriff

Das Diagramm wurde bewusst sehr allgemein gehalten, wiederspiegelt jedoch genau das Konzept, welches vom GSM Tester verwendet wird.

SQLiteOpenHelper

Klasse aus dem Android Framework. Diese untersützt die GsmTesterDatabase im Datenbankmanagement betreffend Zugriff auf die Datenbank, sowie durch Funktionen zum einfachen Upgraden bei Versionssprüngen.

SQLiteDatabase

Klasse aus dem Android-Framework, welche direkten Zugriff auf die der Applikation zur Verfügung stehenden Datenbank liefert. Um das Erstellen dieser Datenbank müssen wir uns nicht weiter kümmern, dies übernimmt der SQLiteOpenHelper.

AndroidComponent

Jede Android-Komponente (Service, Activity), welche Zugriff auf die Datenbank benötigt, muss sich in der onCreate-Methode beim Service registrieren, sowie bei der onDestroy abmelden. Somit kann die GsmTesterDatabase, die Datenbank, sobald diese nicht mehr benötigt wird, automatisch schliessen, bzw. falls sie benötigt wird, öffnen.

Die Komponente kann, sobald sie sich registriert hat, auf der Datenbank beliebige Operationen durchführen. Dazu muss die gewünschte Operation (ein DbHandler) an die GsmTesterDatabase übergeben werden.

GsmTesterDatabase

Die GsmTesterDatabase erbt von SQLiteOpenHelper. Dadurch erhält die Klasse die zwei Methoden onCreate() und onUpgrade(), welche vom SQLiteOpenHelper je nach aufgefundener Datenbank automatisch aufgerufen werden.

Weiter wird über die query()-Methode für sämtliche Android-Komponenten eine Schnittstelle zur Datenbank zur Verfügung gestellt, über welche die DbHandler ausgeführt werden können.

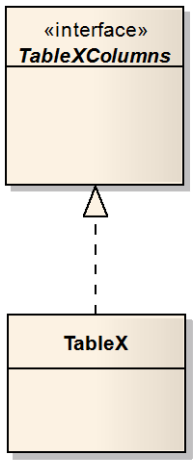
DbHandler<T>, HandlerX, HandlerY, HandlerZ

Jede Operation, die auf der Datenbank ausgeführt wird, ist in eine Handler-Klasse gekapselt. Vorteile dieser Lösung:

- Handler können wiederverwendet werden
- Trennung der einzelnen Datenbankoperationen
- Allgemeine Operationen (z.B. Resultat einer Abfrage in ein ContentValues-Objekt speichern) können in der Basisklasse allen Handlern zur Verfügung gestellt werden.

Der Handler wird von der Android-Komponente erstellt und der GsmTesterDatabase.query()-Methode übergeben. Die GsmTesterDatabase selbst ruft dann die query()-Methode des Handlers mit der aktuellen SQLiteDatabase-Instanz auf.

8.2.5.3. Datenbankabbildung



Wie in Kapitel 8.2.5.2. Datenbankzugriff bereits erwähnt wurde, handelt die GsmTesterDatabase-Klasse das Erstellen sowie Upgraden der Datenbank. Dazu werden SQL-Statements benötigt, welche je nach Tabelle anders aussehen. Weiter wird in den verschiedenen Handler jeweils auf Felder einer Tabelle zugegriffen. Damit diese Definitionen nicht in allen Handlern und Ressourcen verstreut werden, wurden für jede Tabelle die folgenden Komponenten erstellt:

Typ	Name	Zweck
Interface	TableXColumns	Beinhaltet sämtliche Spaltennamen als öffentliche Felder.
Klasse	TableX	Beinhaltet die für die Tabelle wichtigen SQL-Statements sowie den Tabellen-Namen.

8.2.6. Datenserialisierung

Anrufe, Messungen und Resultate der Befragung werden in der Datenbank gespeichert. Um diese auf einem Server abzulegen, müssen die Daten im Dateisystem abgelegt werden können.

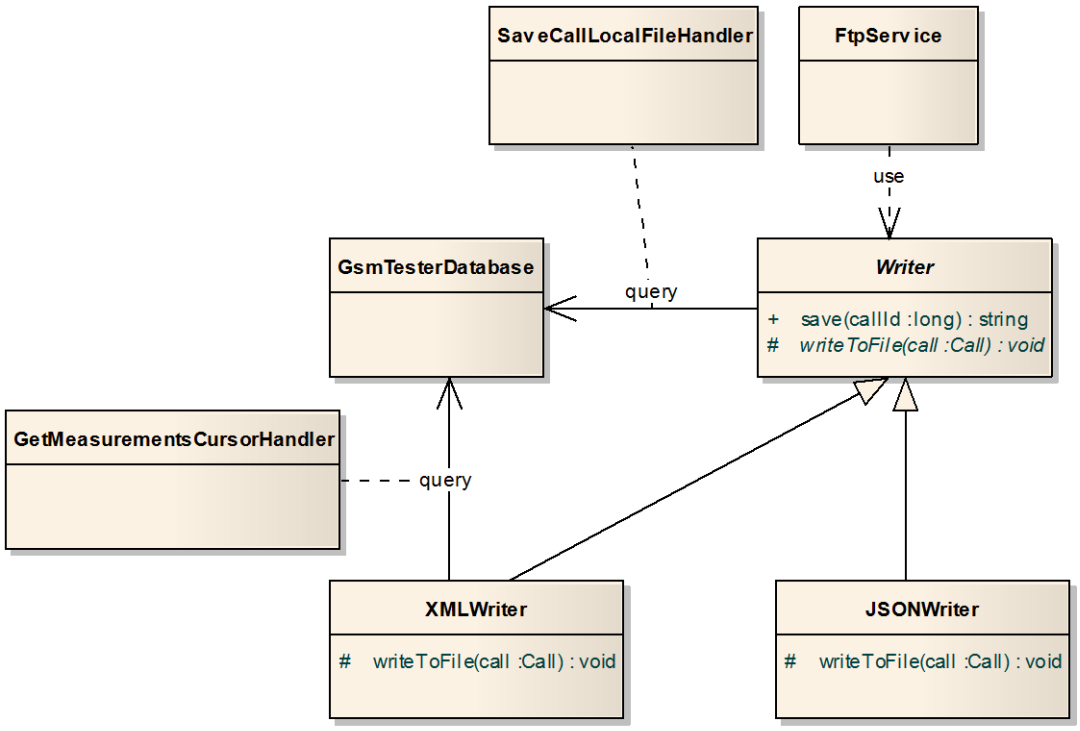


Abbildung 19: Datenserialisierung

Writer

Abstrakte Basisklasse für einen Writer. Der FtpService ruft die save()-Methode auf, welche die folgenden Schritte ausführt:

1. Suchen eines temporären Verzeichnisses zum Ablagen des Reports
   a. Wenn vorhanden und beschreibbar auf die externe Karte
   b. Ansonsten in das Cache-Verzeichnis
2. Abrufen der Anruf-Informationen (→ Call-Objekt erstellen)
3. writeToFile() Methode des konkreten Writer aufrufen, der die Daten des Anrufs im Format, welches durch den konkreten Writer definiert wird, ablegt.

XMLWriter

Konkreter Writer der den Anruf in einem XML-Format ablegt. Allgemeine Informationen zum Anruf erhält der Writer durch den übergebenen Call. Diese Messungen kann der Writer zu gegebener Zeit selbst über den DbHandler GetMeasuementsCursorHandler abfragen.

JSONWriter

Diese Klasse wird vom GSM Tester nicht implementiert. Soll aber hier als Beispiel dienen, wie man die Daten in ein anders Format abspeichern könnte, indem man den XMLWriter einfach durch eine andere Implementation ersetzt.

GsmTesterDatabase, GetMeasurementsCursorHandler, SaveCallLocalFileHandler

Die Datenbank-Architektur die der GSM Tester verwendet wird im Punkt 8.2.5. Datenbank genau erläutert.

Der GetMeasurementCursorHandler liefert einen Cursor, der sämtliche Messungen zu dem übergebenen Anruf bereitstellt, zurück.

Der SaveCallLocalFileHandler speichert den Namen der temporären Datei in der Datenbank. Dieser wird später für den Upload benötigt.

8.2.7. Verbindungsmanager

Der FtpService lädt Dateien von einem Smartphone auf den Server. Von einem Smartphone aus ist eine Internet-Verbindung nicht immer garantiert. Folgende drei Fälle verbieten es dem FtpService Daten auf den Server zu transferieren:

- Es besteht keine Verbindung (z.B. kein Empfang)
- Der Benutzer hat in den Einstellungen den Upload nur über WLAN erlaubt und es ist kein WLAN mit einer Internetverbindung verfügbar.
- Hintergrunddatentransfer wurde durch den Benutzer explizit deaktiviert

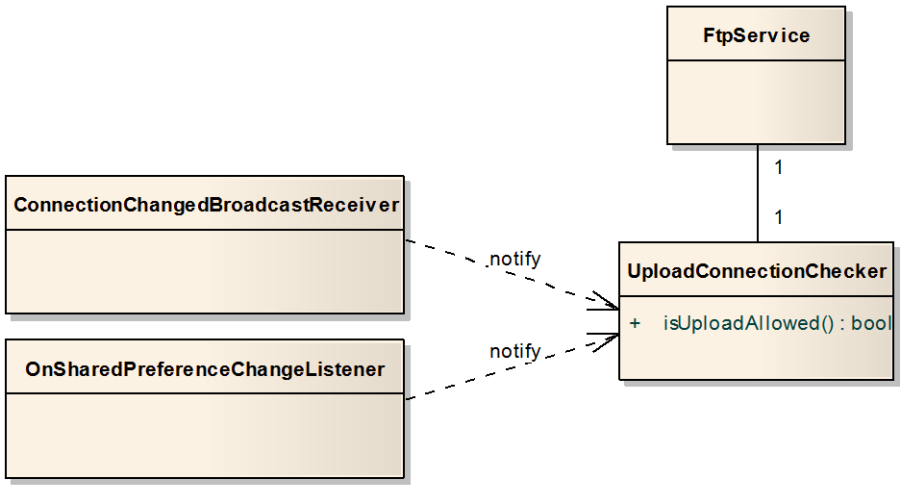


Abbildung 20: Verbindungsmanager

ConnectionChangedBroadcastReceiver

Fängt die folgenden, von Android versendeten Broadcasts, ab:

- android.net.conn.BACKGROUND_DATA_SETTING_CHANGED
  - Wird aufgerufen, wenn der Benutzer die Option “Hintergrunddaten” unter “Einstellungen → Konten & Synchronisierung” aktiviert bzw. deaktiviert.
- android.net.conn.CONNECTIVITY_CHANGE
  - Wird aufgerufen, wenn Android eine Änderung in der Konnektivität feststellt. Z.B. der Wechsel von einem Mobile-Netz (EDGE, …) zu einem WLAN.

Sobald der BroadcastReceiver eine Benachrichtigung erhalten hat, informiert er den UploadConnectionChecker über die Änderung.

OnSharedPreferencesChangeListener

Der Benutzer hat in den GSM Tester Einstellungen die Einstellung „Upload via“ geändert. Der Listener Informiert den UploadConnectionChecker über die Änderung.

FtpService

Führt den Datentransfer im Hintergrund aus. Mehr Informationen zum FtpService können dem Kapitel 8.2.3.2. FtpService entnommen werden.

Der FtpService hält einen UploadConnectionChecker, welcher er über die isUploadAllowed()-Methode befragen kann, ob er einen Upload durchführen darf.

UploadConnectionChecker

Erhält Änderungen vom OnSharedPreferencesChangeListener und ConnectionChangedBroadcastReceiver. Sobald eine Änderung eingetroffen ist, wird die aktuelle Konfiguration neu überprüft und je nach Status des FtpService, dieser gestoppt oder gestartet. Dies geschieht nur, wenn der Benutzer den automatischen Upload aktiviert hat.

Bei der Überprüfung wird, falls von den Einstellungen her alles in Ordnung ist, versucht eine Verbindung mit einem Host im Internet herzustellen. Somit kann garantiert werden, dass man nicht mit einem WLAN verbunden ist, dieses jedoch keine Internet-Konnektivität zur Verfügung stellt.

8.2.8. Logging

Android bietet über die Klasse Log bereits eine einfache Möglichkeit eines Loggings. Dabei werden die Daten je nach mitgegebenem String-TAG als Eintrag im Logcat-Output angezeigt.

Die Logger-Klasse ist eine Art Wrapper um die Android Log-Klasse und bietet entsprechend die folgenden Vorteile:

- Komplettes Logging kann an einem Ort ein- und ausgeschaltet werden
- File-Logging könnte hier eingebaut werden und wäre so überall verfügbar
- Erweitern der bestehenden Log-Klasse

8.2.9. Programmabläufe

8.2.9.1. Ablauf eines Anrufes

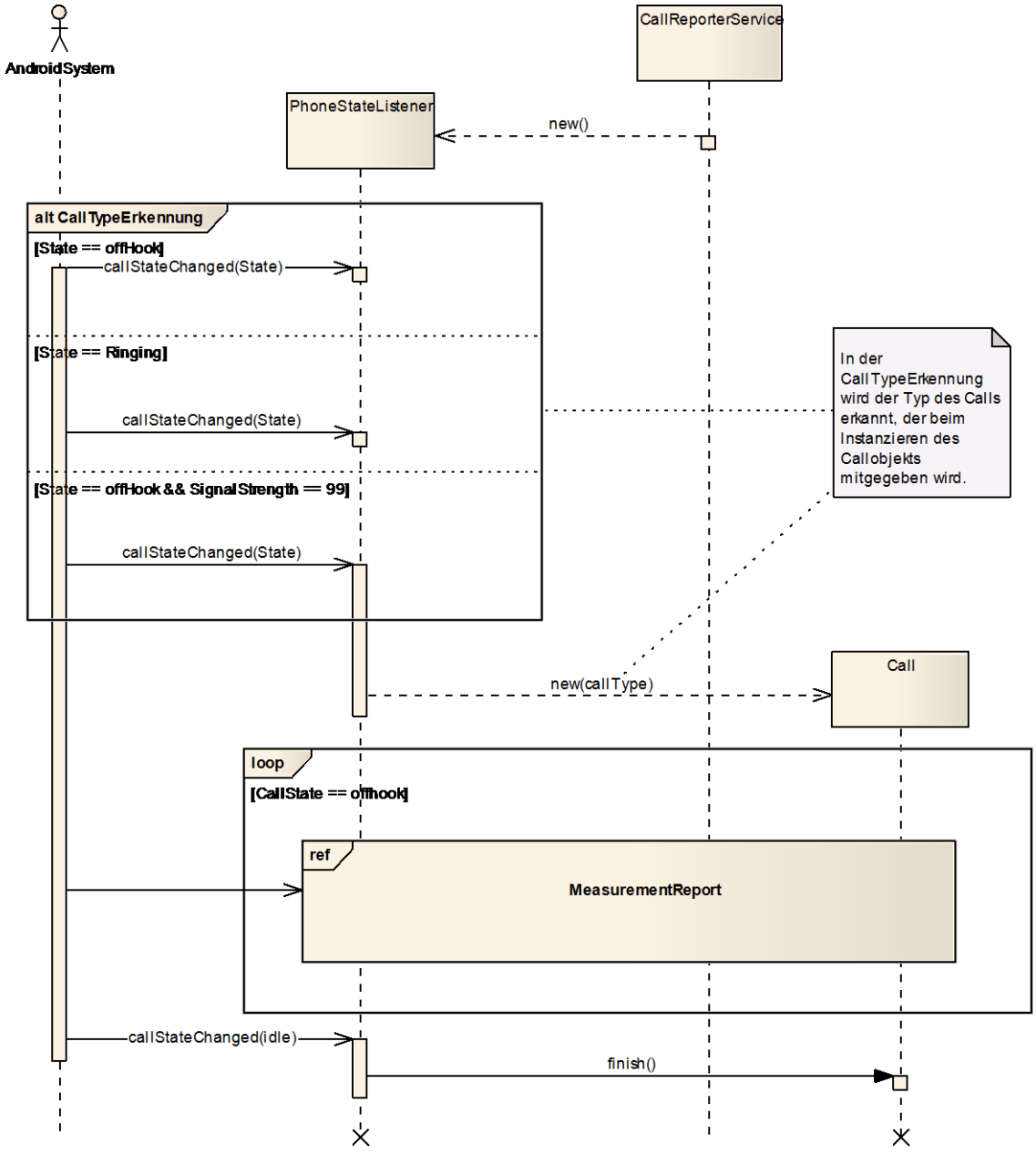


Abbildung 21: Ablauf Anruf

Das Android-System sendet Broadcasts welche vom PhoneStateListener abgefangen werden. Je nach übergebenem State wird der Anruf entsprechend instanziiert. Solange der Anruf nicht beendet wird, werden Messungen durchgeführt. In diesem Sequenzdiagramm ist das weitere Speichern und Uploaden des Anrufes nicht enthalten.

8.2.9.2. Measurment Report

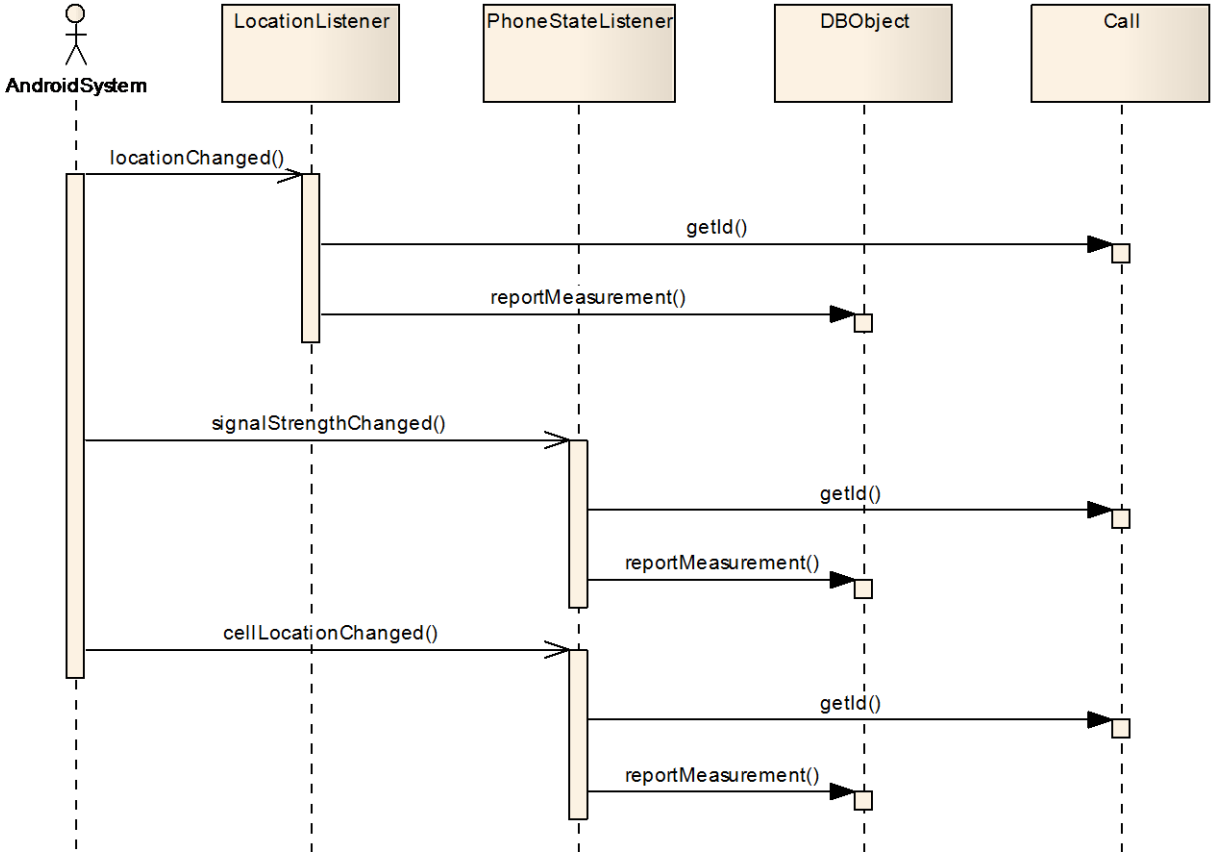


Abbildung 22: Ablauf Measurement Report

Sobald der Benutzer sich in einem Gespräch befindet, werden Messungen aufgezeichnet. Über den LocatinListener erhalten wir Änderungen, welche die Location des Benutzers (Longitude, Latitude) betreffen. Der PhoneStateListener erhält Änderung, welche die Signalstärke und die CellLocation (Cid, Lac) wiederspiegelt.

Zu sehen ist, dass der Listener zuerst die Id des aktuellen Anrufs abfragt und anschliessend die entsprechende Messung in der Datenbank speichert.

8.2.9.3. ManualUpload

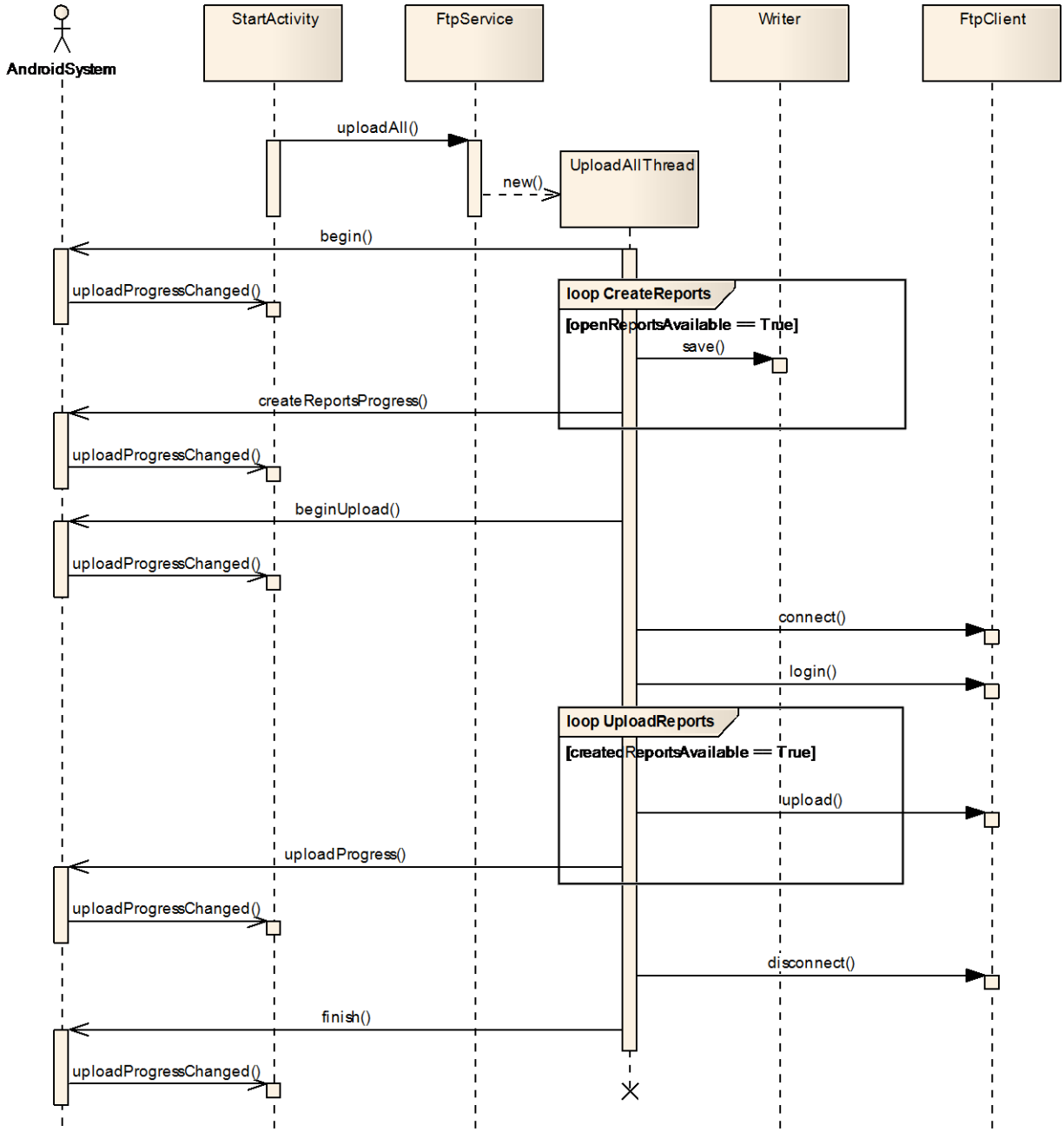


Abbildung 23: Ablauf manueller Upload

Der Manuelle Upload wird durchgeführt, wenn der Benutzer in der StartActivity auf „Upload“ klickt. Es werden dann sämtliche verfügbaren CallReports an den Sever gesendet.

Die Meldungen vom Hintergrund-Thread (UploadAllThread) werden per Broadcast an das System gemeldet, welches den Broadcast an die StartActivity weiterleitet. Diese aktualisiert dann die Oberfläche gemäss erhaltenem Broadcast.

Es werden zuerst sämtliche in der Datenbank gespeicherte CallReports im Dateisystem gespeichert. Dies wird folgenden Gründen so gehandhabt:

- Je nach Länge des Anrufs kann das Speichern im Dateisystem seine Zeit in Anspruch nehmen, welche in einem Timeout des FTP-Clients enden könnte. Ein neu-Verbinden vor jedem Upload ist keine Option, da z.B. bei 200 verfügbaren Uploads 200 Mal eine neue Verbindung aufgebaut werden müsste.
- Wir möchten dem Benutzer beim Upload einen Gesamtfortschritt anzeigen. Damit wir wissen, wie weit der Upload fortgeschritten ist, muss die totale Grösse ermittelt werden können. Dafür müssen die Dateien physikalisch vorhanden sein.

8.2.9.4. AutoUpload

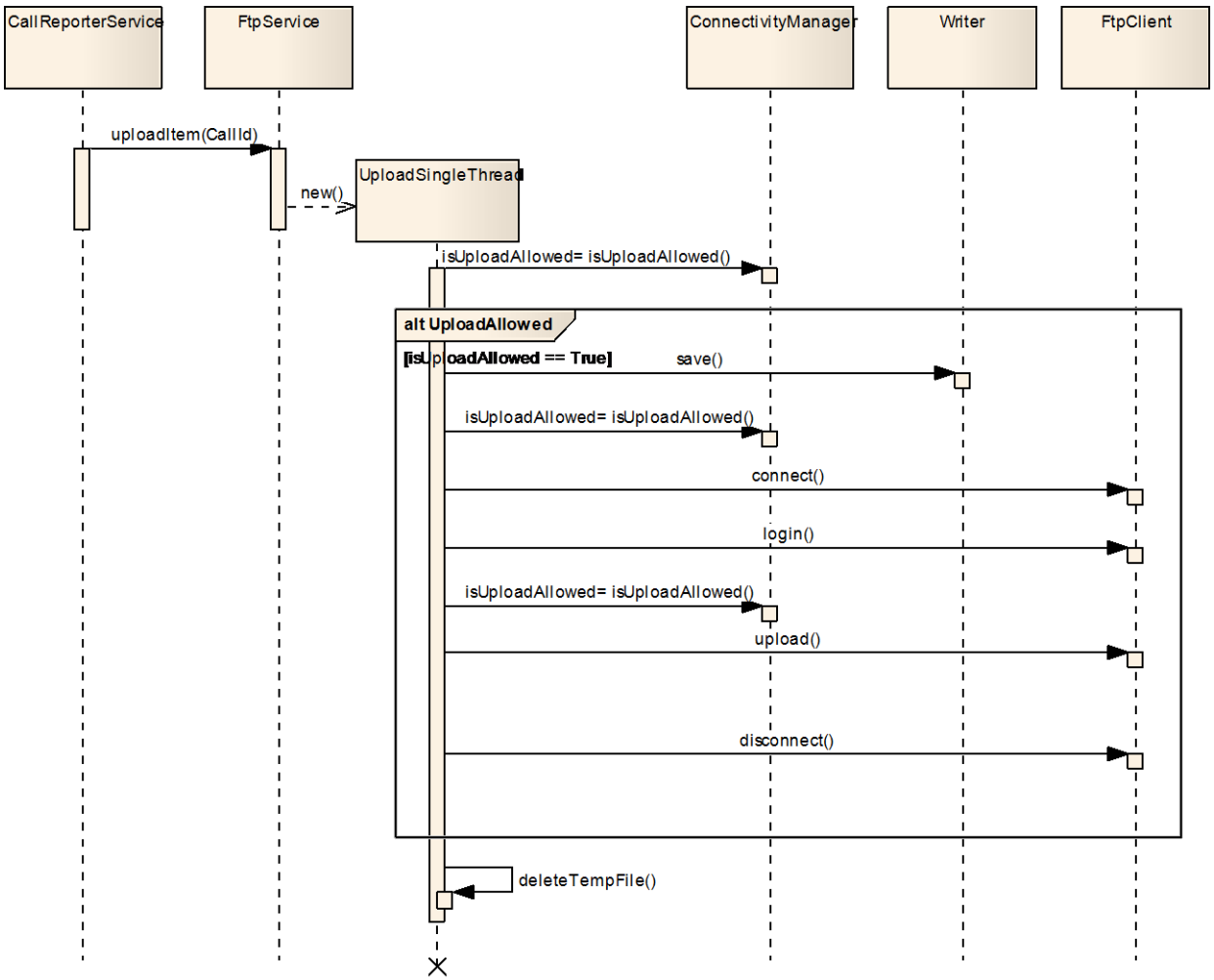


Abbildung 24: Ablauf automatischer Upload

Wird vom CallReporterService durchgeführt, wenn der automatische Upload aktiviert ist. Dabei wird dem FtpService der Anruf übergeben, welche an den Server übertragen werden soll. Da der Upload nicht direkt durch den Benutzer initiiert wurde muss zuerst abgeklärt werden, ob dieser überhaupt durchgeführt werden darf. Dies geschieht in folgenden Situationen:

- Vor dem Erstellen des Reports
- Vor dem Verbinden mit dem FTP-Server
- Vor dem direkten Upload der Datei

8.3. UI

8.3.1. StartActivity | main.xml

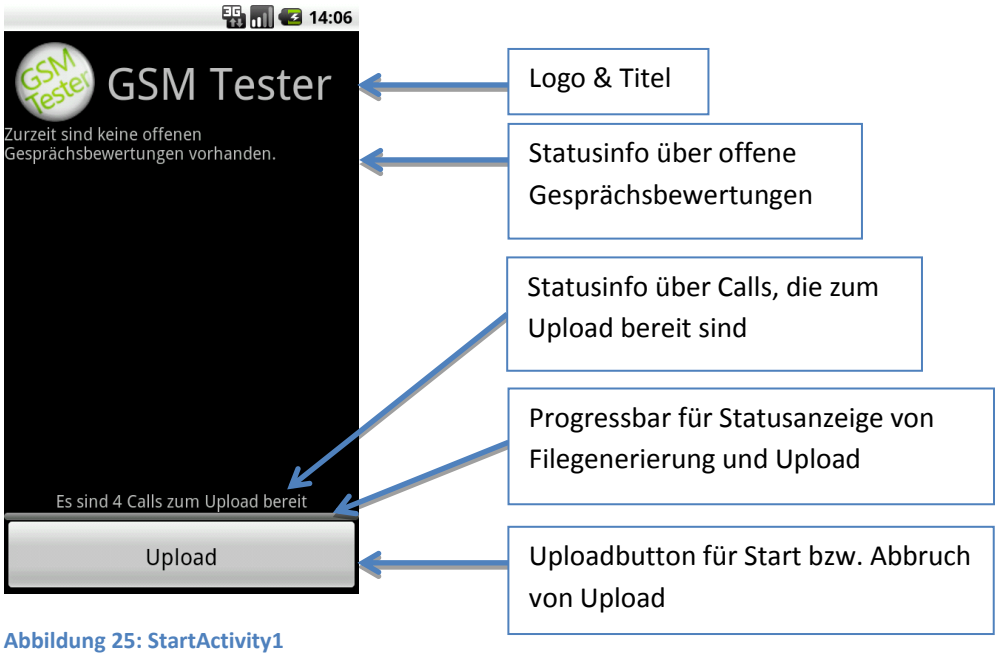


Abbildung 25: StartActivity1

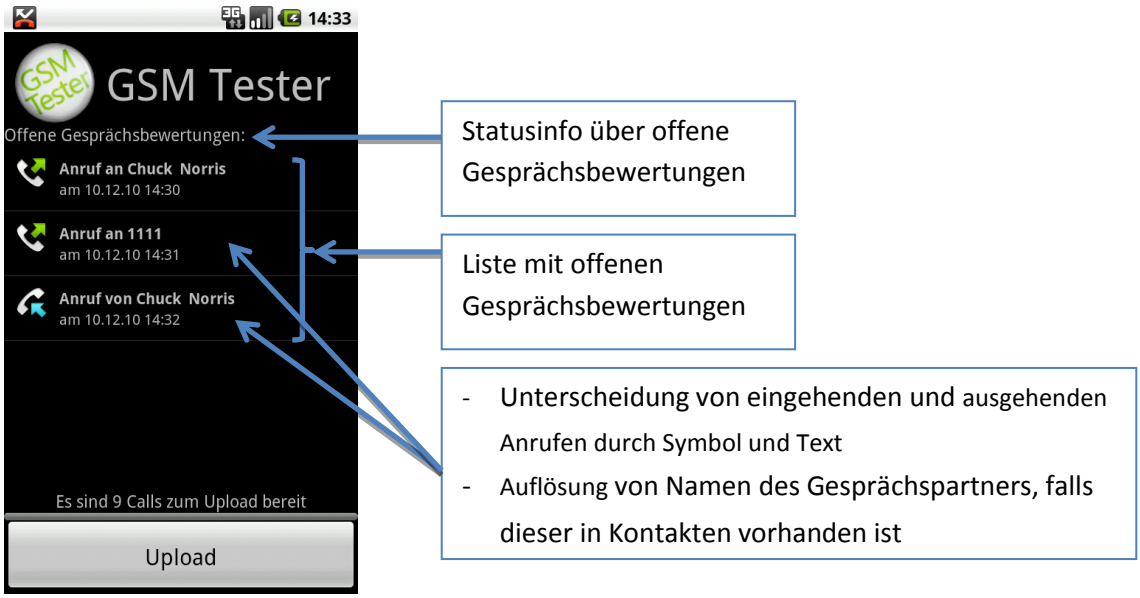
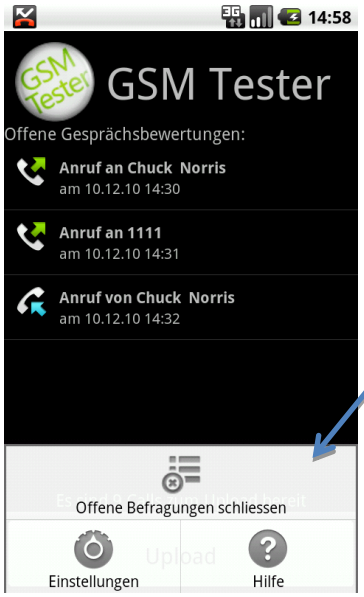


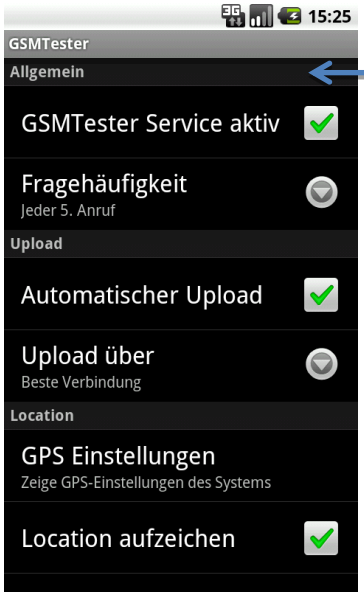
Abbildung 26: StartActivity2



Menü:
- Offene Befragungen schliessen: markiert die Befragungen in Liste als abgeschlossen. Somit sind sie zum Upload bereit
- Einstellungen: öffnet die Einstellungsseite
- Hilfe: öffnet die Hilfeseite

Abbildung 27: StartActivity3

8.3.2. ConfigActivity |preferences.xml



Enthält die möglichen Einstellungen. Genauere Spezifikation siehe Kapitel 9.2.2. Einstellungen im Anhang

Abbildung 28: ConfigActivity

8.3.3. HelpActivity | help.html

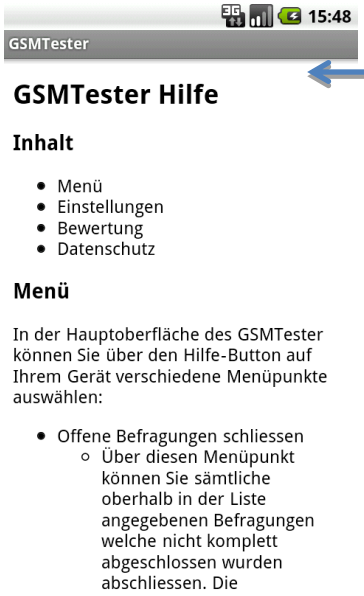


Abbildung 29: HelpActivity

8.3.4. SurveyActivity | survey.xml

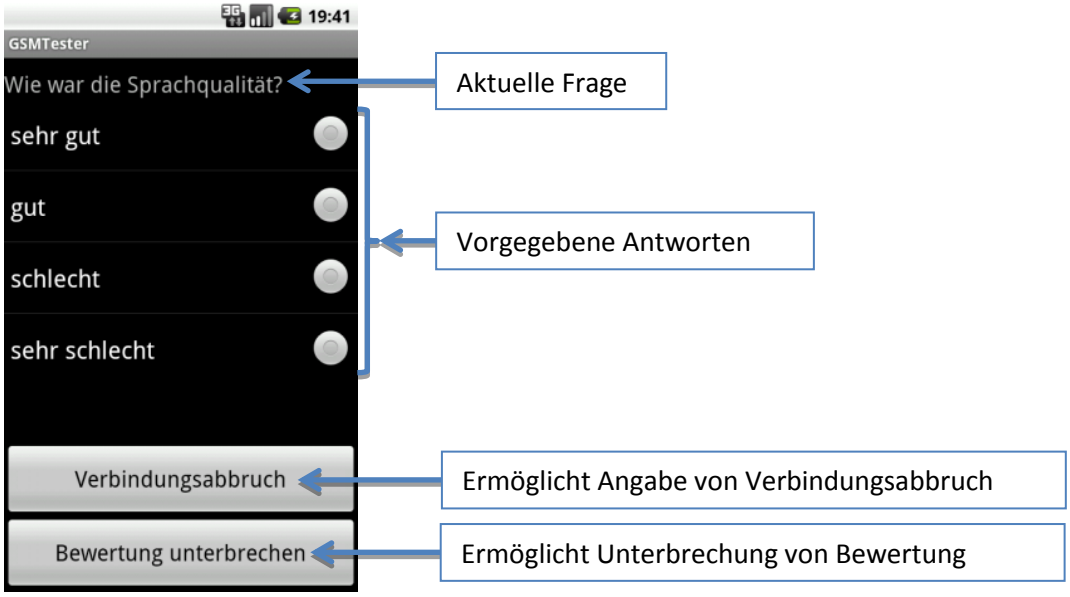


Abbildung 30: SurveyActivity

8.4. XML Struktur

Der GSM Tester legt die Messdaten eines Anrufs im XML-Format auf einem FTP-Server ab. In diesem Abschnitt soll das verwendete XML-Schema erläutert werden.

8.4.1. Struktur

```
<GSMTesterReport>
  <!-- CallInfo Beinhaltet allgemeine Informationen zum Anruf -->
  <callInfo>
    <type type="Outgoing"/>
    <durration start="09.12.10 15:55:49" stop="09.12.10 15:55:51" seconds="2"/>
    <phoneNumbers opposite="+4178732XXXX"/>
    <operator simoperator="Swisscom" netoperator="Swisscom"/>
    <diverses networkCountryIso="ch"/>
  </callInfo>
  <!-- survey beinhaltet die Resultate der Befragung -->
  <!-- ist nicht vorhanden wenn keine Befragung durchgeführt wurde-->
  <survey>
    <canceled value="0"/>
    <quality value="2"/>
    <location value="0"/>
    <irritation value="2"/>
  </survey>
  <!-- measurements beinhaltet sämtliche Messungen zum Anruf-->
  <measurements>
    <i ty="0" ti="09.12.10 15:55:48" ber="-1" asu="22"/>
    <i ty="1" ti="09.12.10 15:55:49" long="8.818198" accuracy="2965.0"
      provider="network" lat="47.223869"/>
    <i ty="2" ti="09.12.10 15:55:51" cid="1233" lac="3423"/>
  </measurements>
</GSMTesterReport>
```

Die Tags der Messungen wurden bewusst sehr minimalistisch gewählt, um die Grösse der XML-Datei zu reduzieren. Entsprechende Beschreibungen zu **ty** und **ti** können dem folgenden Kapitel entnommen werden.

8.4.2. Wertetabellen

Diverse Tags können verschiedene Werte beinhalten. In diesem Abschnitt sollen zu den Tags die möglichen Werte mit einer entsprechenden Beschreibung festgehalten werden:

8.4.2.1. callInfo → type → type

Beschreibung Parameter: Gibt den Typ des Anrufs an
Mögliche Werte: Incoming/Outgoing/NoAccess

8.4.2.2. survey → canceled → value

Beschreibung Parameter: Antwort des Benutzers, ob der Anruf abgebrochen wurde.

Mögliche Werte:

Wert	Beschreibung
0	Normal (Gespräch normal beendet)
1	Call Drop (Gespräch unterbrochen)

8.4.2.3. survey → quality → value

Beschreibung Parameter: Antwort des Benutzers über die Qualität des Gesprächs

Mögliche Werte:

Wert	Beschreibung
0	Sehr gut
1	Gut
2	Schlecht
3	Sehr schlecht

8.4.2.4. survey → location → value

Beschreibung Parameter: Antwort des Benutzers, wo er sich während des Anrufs befunden hat.

Mögliche Werte:

Wert	Beschreibung
0	Im Gebäude
1	Im Freien
2	Im Auto/Bus/Tram
3	Im Zug
4	Anderer Ort

8.4.2.5. survey → irritation → value

Beschreibung Parameter: Antwort des Benutzers, wie verärgert er über die Qualität des Anrufs war.

Mögliche Werte:

Wert	Beschreibung
0	Gar nicht
1	Ein wenig
2	Stark
3	Sehr stark

8.4.2.6. measurements → i → ty

Beschreibung Parameter: Für jede Messung wird ein i-Element erstellt. Es gibt verschiedene Arten von Messungen. Die Art (Typ) der Messung wird im ty-Attribut festgehalten.

Mögliche Werte:

Wert	Beschreibung
0	Signalstärke
1	Location
2	Cell-Location

8.4.2.7. measurement → i → ti

Beschreibung Parameter: Beinhaltet den Zeitpunkt, an dem die Messung durchgeführt wurde.

8.4.3. Dateigrösse pro Gespräch

Wir nehmen an, dass ein durchschnittliches Gespräch drei Minuten dauert. Der GSM Tester macht ca. jede Sekunde eine Messung (sämtliche Messungen miteinberechnet) Entsprechend würden wir mit unsere XML-Struktur die folgende Dateigrösse erhalten:

Dauer Gespräch:	3*60s	180s
Durchschnittliche Grösse pro Messung:		70 Byte
Dateigrösse Messung:	180s * 70 Byte	5600 Byte
Dateigrösse Anrufinformationen:		500 Byte
Total:		**6100 Byte**

Bei diesen Werten handelt es sich um Annahmen welche durch Messungen verifiziert wurden.

8.5. Permissions

Der GSM Tester benötigt eine ganze Reihe von Android-Permissions. In diesem Abschnitt wird festgehalten, welche Permissions warum benötigt werden:

Permission	Verwendung
ACCESS_COARSE_LOCATION	Zugriff auf Location-Daten des Network-Providers
ACCESS_COARSE_UPDATES	Zugriff auf Location-Updates des Network-Providers, welche per Location Listener publiziert werden.
ACCESS_FINE_LOCATION	Zugriff auf Locatoin-Daten des GPS-Providers
ACCESS_NETWORK_STATE	Zugriff auf Location-Updates des GPS-Providers, welche per Location Listener publiziert werden.
ACCESS_WIFI_STATE	Zugriff auf WLAN(WiFi)-Status
CHANGE_NETWORK_STATE	Wird benötigt um das aktive Netzwerk auszulesen
CHANGE_WIFI_STATE	Wird benötigt um das aktive Netzwerk auszulesen
INTERNET	Zugriff auf Internet (wird für FTP Upload benötigt)
PROCESS_OUTGOING_CALLS	Abfangen von ausgehenden Anrufen. Wird benötigt um die Telefonnummer von ausgehenden Anrufen zu erfassen.
READ_CONTACTS	Der GSM Tester benötigt Zugriff auf das Adressbuch, um Telefonnummern nach Namen aufzulösen.
READ_PHONE_STATE	Zugriff auf den Phone-State (Siehe dazu Kapitel 6.2.1. PhoneState)
RECEIVE_BOOT_COMPLETED	Um den CallReporterService beim booten des Geräts zu starten, wird die Permission benötigt, um den Boot-Complete Broadcast zu erhalten.
WRITE_EXTERNAL_STORAGE	Daten werden vor dem Upload temporär gespeichert. Wenn möglich auf das externe Storage.

8.6. Konfigurierbarkeit

Die Architektur bietet Ansatzpunkte, an der Konfigurationen durchgeführt werden können.

8.6.1. Mehrsprachig

Android bietet von sich aus eine mehrsprachige Umgebung. D.h. es wird beim Laden der Ressourcen einer Applikation versucht, lokalisierte Ressourcen zu laden. Falls diese nicht vorhanden sind, werden die Standard-Ressourcen verwendet.

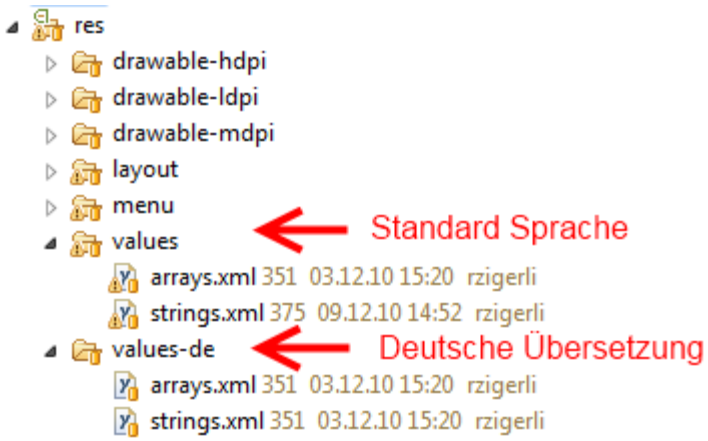


Abbildung 31: Struktur Mehrsprachigkeit

Für jede weitere Sprache muss nur ein neuer values-Ordner mit den entsprechenden Dateien erstellt werden. Einfachste Möglichkeit ist das Kopieren der bestehenden und dann die Anpassung der entsprechenden Einträge im XML auf die neue Sprache.

8.6.2. Survey

Eine Anpassung der Befragungsscreens muss an folgenden Stellen stattfinden:

```
<TextView
    android:id="@+id/tv_survey"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:paddingTop="10sp"/>
<ListView
    android:id="@+id/lv_survey"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
```

Abbildung 32: Codeausschnitt Konfiguration Survey XML

Dieser Ausschnitt zeigt einen Teil der Datei survey.xml. Diese beinhaltet eine TextView mit der ID tv_survey, die die gestellte Frage repräsentiert. In der ListView lv_survey werden die Antwortmöglichkeiten dargestellt. Die Definition der Antworten ist im arrays.xml definiert.

Abbildung 33: Konfiguration Fragen Eclipse/XML

Die eigentlichen Fragen sind im dazugehörigen strings.xml definiert.

```
<!--  survey.xml -->
<string name="txt_bu_callDrop">Verbindungsabbruch</string>
<string name="txt_irritation">Wie sehr sind Sie verärgert?</string>
<string name="txt_locality">Wo waren Sie während dem Gespräch?</string>
<string name="txt_quality">Wie war die Sprachqualität?</string>
<string name="txt_bu_survey_pause">Bewertung unterbrechen</string>
```

Abbildung 34: Konfiguration Survey Antworten

8.6.3. Konfiguration

Die Konfiguration ist nur Begrenzt anpassbar. Dies liegt daran, dass die meisten Parameter gegeben sind (z.B. automatischer Upload → Ja/Nein)

Einzig der Konfigurationspunkt der „**Fragehäufigkeit**“ lässt sich anpassen. Dazu muss die **arrays.xml** im aus dem Ordner **res/values** geöffnet werden. Dort haben wir die Werte zur Fragehäufigkeit entsprechend angegeben:

```
<string-array name="survey_frequency">
    <item>2</item>
    <item>3</item>
    <item>4</item>
    <item>5</item>
    <item>6</item>
    <item>7</item>
    <item>8</item>
    <item>9</item>
    <item>10</item>
</string-array>
<string-array name="survey_frequency_values">
    <item>2</item>
    <item>3</item>
    <item>4</item>
    <item>5</item>
    <item>6</item>
    <item>7</item>
    <item>8</item>
    <item>9</item>
    <item>10</item>
</string-array>
```

Abbildung 35: Codeausschnitt Konfiguration Fragehäufigkeit

Man sieht hier einerseits die angezeigten Anzeigedaten (name = survey_frequency) und andererseits die entsprechenden Werte dazu (name=survey_frequency_value).

Es kann nun einerseits der Anzeigename geändert werden (z.B. Anstatt 2 könnte man „zwei“ schreiben“) oder es können Einträge entfernt/gelöscht werden. Dabei ist einzig zu beachten, dass die entsprechenden Werte bei den Values zu den Werten der Anzeigedaten passen (Android geht hier nach der Reihenfolge). Weiter müssen die Werte in values Ganzzahlen sein, welche die Befragungsfrequenz ergeben.

8.6.4. Decision Maker

Der Decision-Maker entscheidet anhand eines Algorithmus, ob nach einem Anruf die Befragung durchgeführt wird.

Folgender Codeausschnitt zeigt, wie im CallReporterService der Decisionmaker mit einer definierten Frequenz befragt wird, ob die Benutzerbefragung durchgeführt werden soll. Falls ja, wird in der Datenbank eine neue Survey erstellt und die SurveyActivity angezeigt.

```
// Prüfen ob der Benutzer nach dem Anruf befragt werden soll
if (DecisionMaker.shouldRequestUser(currentCall, getSurveyFrequence())) {
    // Da ein Survey durchgeführt wird dieses erst in der Datenbank
    // erstellen
    GSMTesterDatabase.query(new CreateSurveyHandler(currentCall.id));

    // Benutzer befragen
    showSurveyActivity();

}
```

Abbildung 36: Codeausschnitt CallReporterService

Falls nun ein eigener Algorithmus für die Entscheidung implementiert werden soll, muss nur die shouldRequestUser()-Methode des DecisionMakers angepasst werden. Der GSM Tester implementiert bereits einen zweiten Algorithmus welcher die „Frequenz“ zufällig nach jedem Anruf wählt. Diese Implementation soll hier nicht weiter dokumentiert werden und dient im Code als Beispiel für einen alternativen Algorithmus.

8.6.5. FTP Host

Der FTP-Host, auf welchen die CallReports transferiert werden, kann in der Klasse **FTPHost** konfiguriert werdenn.

```
/**
 * FTP-Hostname auf welchem die Daten abgelegt werden
 */
public static final String hostname = "exampleHostName";

/**
 * Username mit dem sich das Programm am FTP-Server anmeldet
 */
public static final String ftpUserName = "exampleUsername";

/**
 * Passwort für den in {@link #ftpUserName} angegebenen Benutzer
 */
public static final String ftpPassword = "examplePassword";
```

Abbildung 37: Codeausschnitt FTPHost

8.6.6. Weitere Einstellungen

In der Klasse **Call**:

```
/**
 * Maximale Anzahl Messungen die ein Anruf haben kann
 * Sobald mehr Messungen rein kommen werden diese verworfen
 */
private static final Long maxMeasurementCount = 100001;
```

Abbildung 38: Codeausschnitt Setting Call

In der Klasse **CallReporterService**:

```
/**
 * Anzahl Anrufe die bereit zum Upload sein dürfen bis der Benutzer eine
 * Notification erhält er solle doch die vorhandenen Anrufe uploaden
 */
private static final int MaxReadyCallCount = 300;

/**
 * Minimaler Zeitabstand in Sekunden zwischen zwei Locationmessungen
 */
private static final int minTimeBetweenLocationMeas = 1;

/**
 * Minimaler Distanzabstand zwischen zwei Locationmessungen
 */
private static final int minMeterBetweenLocationMeas = 10;
```

Abbildung 39: Codeausschnitt Setting CallReporterService

8.7. Weitere Arbeiten

8.7.1. Gerätespezifische Anpassungen

Auf den von uns getesteten Geräten müssen keine gerätespezifischen Anpassungen vorgenommen werden, solange die Anpassungen an die entsprechende Android-Version durchgeführt wurden.

8.7.2. HSR Testlauf

Um einen repräsentativen Testlauf sowohl der Applikation an sich, als auch der ermittelten Daten durchzuführen, haben wir uns ein Szenario an der HSR vorgestellt. Für diese Idee spricht die Tatsache, dass die Anzahl an Android Smartphones an der HSR bereits sehr hoch ist. Ebenfalls sind die Studenten naturgemäss sehr experimentierfreudig. Um jedoch genug Interessenten für ein solches Vorhaben gewinnen zu können, müsste man den Probanden eine gewisse Gegenleistung bieten. Unser Vorschlag dazu ist, dass man für 50 bewertete und übertragene Anrufe einen Schweizer Franken erhält. Denkbar wäre auch ein anderes Bonussystem, z.B. in Zusammenarbeit mit einem Provider der Gesprächsminuten zur Verfügung stellen würde. Um einen solchen Testlauf effizient durchführen zu können, müsste ein Bugtracking System aufgesetzt werden, indem die User direkt gefundene Bugs eintragen könnten. Ebenfalls müsste man die Auswertung der gewonnenen Daten festlegen.

8.8. Gerätetests

Wir haben zu Testzwecken die Applikation auf Ihre Funktionalität mit mehreren Android-Geräten getestet:

Gerät	Android Version	Status
HTC Desire	2.2	OK
HTC Desire HD	2.2	OK
HTC Nexus One	2.2	OK
Emulator	2.1	OK
Emulator	2.2	OK
Emulator	2.3	OK

8.9. Android Version

8.9.1. Auswahl minimaler SDK

GSM Tester wurde unter der Android Version 2.2 entwickelt. Dies haben wir so gewählt, um Neuerungen die laufend in die Android API einfliessen benutzen zu können und die meisten neuen Geräte mit Android 2.2 erhältlich sind.

8.9.2. Anpassungen für tiefere SDK

Um GSM Tester auf tieferen Android-API's auszuführen, sind einige Anpassungen nötigt. Hier soll nicht direkt die Implementation einer solchen Anpassung niedergeschrieben, sondern lediglich auf die Änderungen der API hingewiesen werden, welche eine Änderung am GSM Tester benötigen.

Um den GSM Tester für tiefere Android-Versionen vorzubereiten, müssen die folgenden Unterkapitel in Absteigender Form durchgearbeitet werden.

8.9.2.1. Android 2.3

Änderungen an der API durch Android 2.3:
http://developer.android.com/sdk/android-2.3.html

Die angegebenen Änderungen haben keinen Einfluss auf den GSM Tester.

8.9.2.2. Android 2.2

Änderungen an der API durch Android 2.2:
http://developer.android.com/sdk/android-2.2.html

Der GSM Tester wurde unter dieser Version entwickelt und braucht keine Änderungen.

8.9.2.3. Android 2.1

Änderungen an der API durch Android 2.1:
http://developer.android.com/sdk/android-2.1.html

Folgende Änderung hat Einfluss auf den GSM Tester:

Telephony
New SignalStrength class provides information about the device's current network signal. This can be acquired from the new onSignalStrengthsChanged(SignalStrength) callback.

Bis Version 2.1 hatte der PhoneStateListener die folgende Methode:

```
public void onSignalStrengthChanged (int asu)
```

Neu ab 2.1 wurde die alte Methode als deprecated markiert und die folgende eingeführt:

```
public void onSignalStrengthsChanged (SignalStrength signalStrength)
```

Der GSM Tester verwendet die neue Variante um über die SignalStrength-Klasse auf die Signalstärke (asu) sowie die Bit Error Rate (ber) zuzugreifen.

8.9.2.4. Android 2.0.1

Änderungen an der API durch Android 2.0.1:
http://developer.android.com/sdk/android-2.0.1.html

Die angegebenen Änderungen haben keinen Einfluss auf den GSM Tester.

8.9.2.5. Android 2.0

Änderungen an der API durch Android 2.0:
http://developer.android.com/sdk/android-2.0.html

Folgende Änderung hat Einfluss auf den GSM Tester:

Contacts
New contacts APIs that allow for data from multiple accounts

Die gesamte Contact-API wurde durch Android überarbeitet. Der GSM Tester verwendet die neue Methode um über die Telefonnummer an den Namen einer Person zu gelangen.

Folgende Seite beschreibt die beiden Methoden die in der neuen resp. der älteren Android-Versionen eingesetzt werden:

Android bis Version 2.0
http://www.higherpass.com/Android/Tutorials/Working-With-Android-Contacts/2/

Android ab Version 2.0
http://www.higherpass.com/Android/Tutorials/Working-With-Android-Contacts/

8.10. Test

Da der Gesamtrahmen des Projektes überschaubar ist, haben wir uns dazu entschlossen, auf UnitTests zu verzichten. Dafür haben wir uns auf die Ausarbeitung und die Durchführung eines ausführlichen Systemtests konzentriert. Dieser ist zur Einsicht dem Anhang angefügt.

9. Anhang

9.1. Abbildungsverzeichnis

9.2. Bedienungsanleitung

9.2.1. Allgemeine Infos

9.2.1.1. Was ist GSM Tester

GSM Tester ist eine Android Applikation, die es ermöglicht, Angaben zur GSM Netzwerkqualität zu machen. Dabei wird sowohl die gefühlte als auch die technische Qualität betrachtet. Dazu werden während einem Gespräch verschiedene Werte vom System ausgelesen und anschliessend auf dem Smartphone abgelegt. Zusätzlich wird der Benutzer nach dem Gespräch über die gefühlte Netzwerkqualität befragt. Diese Werte werden ebenfalls lokal gespeichert und anschliessend zusammen mit den technischen Daten auf einem Server abgelegt. Diese Daten helfen dabei, eine systematische Untersuchung von Verbindungsproblemen wie Verbindungsunterbruch, nicht erreichbare Gesprächspartner oder schlechter Sprachqualität durchzuführen.

9.2.1.2. Welche Daten werden aufgezeichnet

Je nach Einstellungen werden folgende Daten aufgezeichnet:

- Lokationsdaten (metergenau via GPS, falls aktiviert, sonst auf ca. 2km genau via GSM Netz)
- Start- und Endzeit von Gespräch
- Signalstärke während dem Gespräch
- Welcher Provider verwendet wurde
- Telefonnummer der Gesprächspartner (Anonymisierung der letzten vier Stellen)

9.2.2. Einstellungen

Um GSM Tester Ihren persönlichen Wünschen anzupassen, stehen Ihnen folgende Einstellungen zur Verfügung.

Einstellung	Erklärung
GSM Tester Service aktiv	Aktivieren oder deaktivieren Sie den Services, der die Aufzeichnungen im Hintergrund tätigt. **Standard: aktiviert**
Fragehäufigkeit	Wählen Sie aus, nach wie vielen Anrufen Sie befragt werden möchten. 5 bedeutet, dass nach jedem 5. Anruf der Befragungsdialog erscheint. **Standard: 5**
Automatischer Upload	Die gespeicherten Daten werden im Hintergrund automatisch an den Server gesendet. (empfohlen, da Ergebnisse nur ca. 6 kByte gross sind – dies entspricht einem Bruchteil einer MMS) Wenn die Option deaktiviert ist, erhalten Sie die Möglichkeit, die Daten manuell an den Server zu senden. **Standard: aktiviert**
Upload über	Wählen Sie aus, ob der Upload über die beste zur Verfügung stehende Verbindung getätigt werden soll. (empfohlen) Falls ein WLAN zur Verfügung steht, wird dieses genommen. Ansonsten wird die beste Mobilfunkverbindung genommen. Dabei können Kosten für die Datenübertagung über die Mobilfunkverbindung entstehen. Wenn Sie die Option „WLAN“ auswählen, wird der Upload nur dann durchgeführt, wenn ein WLAN zur Verfügung steht. Dabei entstehen keine weiteren Kosten für die Datenübertragung. Diese Einstellungen wirken, wenn der automatische Upload aktiviert ist. Falls dieser deaktiviert ist, wird immer die beste Verbindung verwendet. **Standard: Beste Verbindung**
GPS Einstellungen	Hier gelangen Sie auf die GPS Systemeinstellungen. Dort erhalten Sie die Möglichkeit, das GPS Modul zu aktivieren oder zu deaktivieren. Dies hilft uns, schlecht versorgte Orte genau zu identifizieren.
Location aufzeichnen	Stellen Sie ein, ob GSM Tester die Lokationsdaten aufzeichnen soll oder nicht. Falls die Option deaktiviert ist, werden keine Lokationsdaten aufgezeichnet. **Standard: aktiviert**

9.2.3. Navigation

9.2.3.1. Offene Bewertungen schliessen

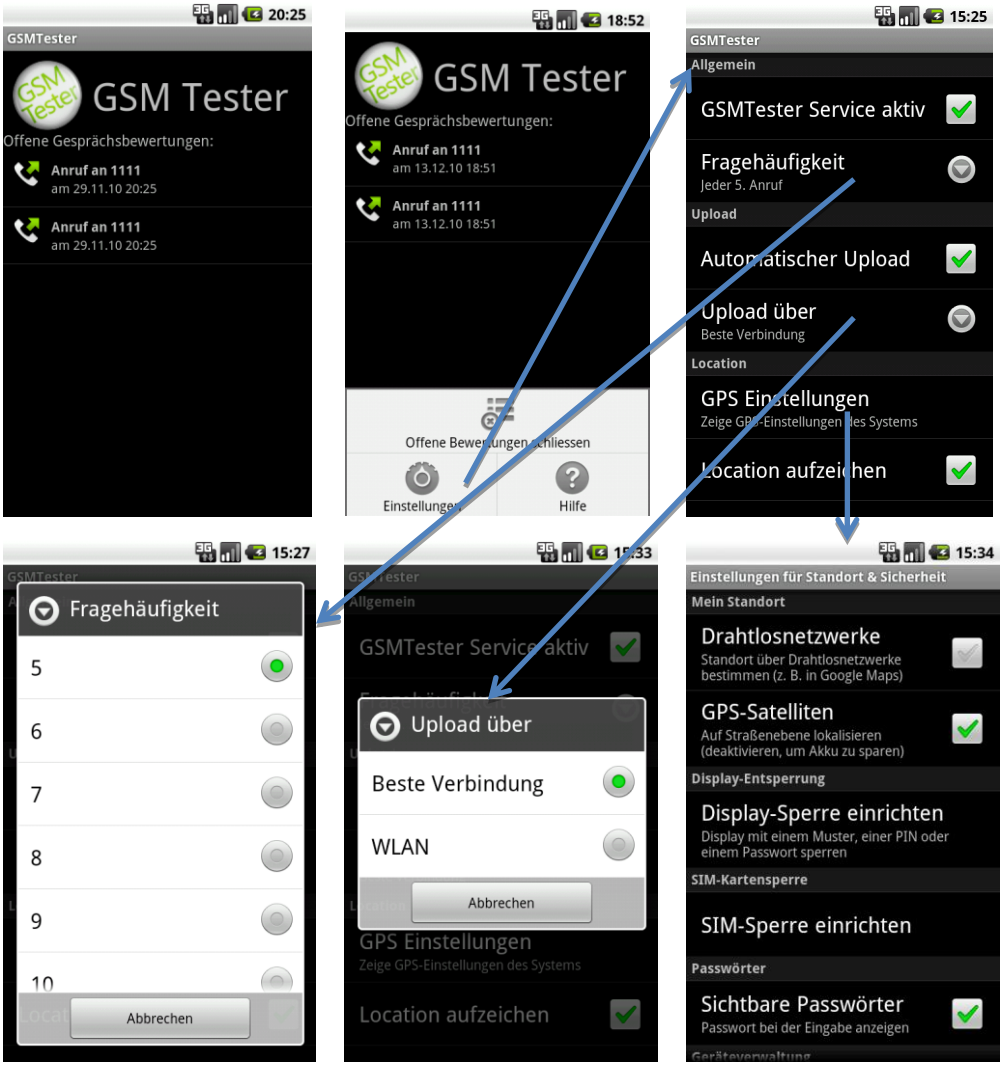
Über den Menü Button besteht die Möglichkeit, offene Bewertungen als abgeschlossen zu markieren. Sobald die Option gewählt wurde, werden alle offenen Bewertungen dementsprechend markiert und verschwinden aus der Liste mit den offenen Gesprächsbewertungen.

9.2.3.2. Hilfe

Ebenfalls über den Menü Button ist die Hilfe erreichbar. Hier werden die wichtigsten Elemente von GSM Tester beschrieben sowie zusätzliche Informationen gegeben.

9.2.3.3. Einstellungen

Nach einem Klick auf den Menü Button erscheint das Menü mit den Einträgen „Offene Bewertungen schliessen“, „Einstelllungen“ und „Hilfe“. Um die Einstellungen für die GSM Tester Applikation festzulegen, klickt man auf „Einstellungen“. Hier können Sie die im Kapitel 4 erwähnten Einstellungen vornehmen.

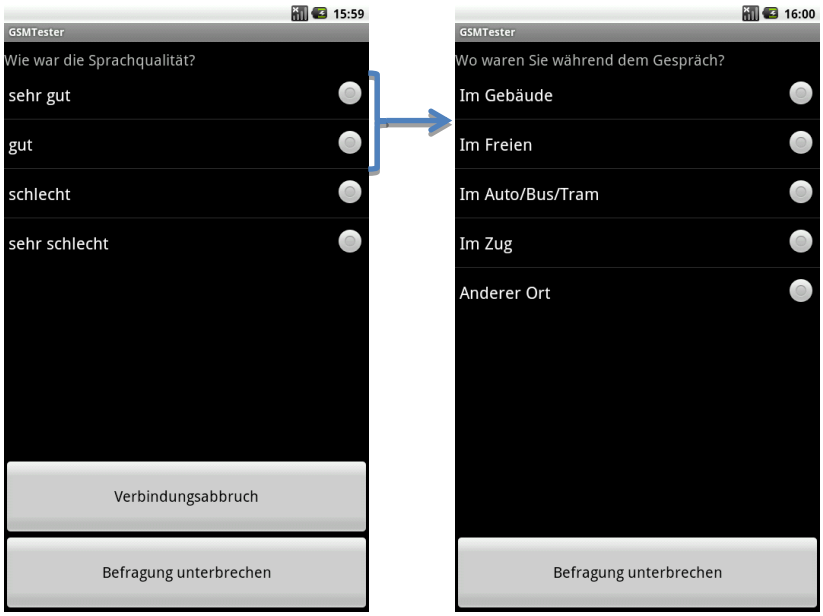


9.2.3.4. Befragung

Die Häufigkeit des Erscheinens des Fragedialogs ist von den Einstellungen, die unter Kapitel 4.3 beschrieben wurden, abhängig. Die Anzahl der gestellten Fragen ist von den getätigten Antworten abhängig, jedoch können alle Befragungen mit maximal drei Klicks beantwortet werden. Es besteht ebenfalls zu jedem Zeitpunkt die Möglichkeit, die Befragung zu unterbrechen. Dies geschieht durch einen Klick auf den Button „Befragung unterbrechen“. Die Befragung startet nach dem Beenden eines Gespräches automatisch. Ebenfalls steht ein Button „Verbindungsabbruch“, mit dem ein Verbindungsabbruch des Telefongespräches notiert werden kann, zur Verfügung.

<u>Ablauf 1: Gespräch mit guter/sehr guter Sprachqualität</u>

Hier wird ein Ablauf dargestellt, bei dem das Gespräch mit guter oder sehr guter Sprachqualität im Zug stattgefunden hat. Zuerst wird nach der Sprachqualität gefragt. Diese kann durch vier mögliche Auswahlelemente beantwortet werden. Nach einem Klick auf „sehr gut“ oder „gut“ erscheint die nächste Frage. Diese betrifft den Ort, an dem sich der Benutzer zum Zeitpunkt des Gespräches aufgehalten hat. Hier stehen fünf Auswahlmöglichkeiten zur Verfügung. Nach einem Klick auf ein Element, wird eine kurze Information angezeigt, dass die Befragung erfolgreich abgeschlossen wurde und es erscheint wieder die Ansicht, die der Benutzer vor der Befragung aktiv hatte.



Ablauf 2: Gespräch mit schlechter/sehr schlechter Sprachqualität

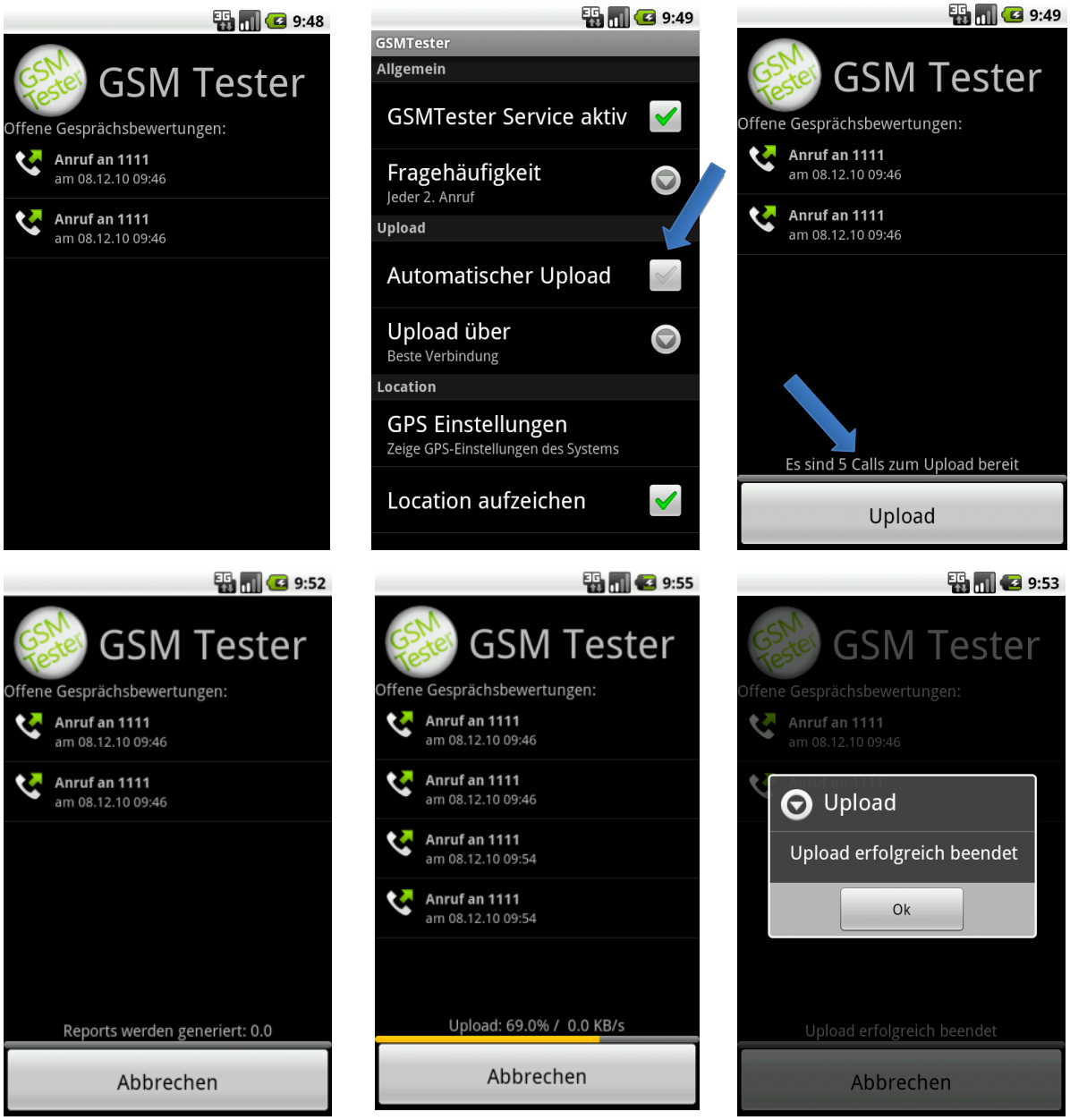
Nun wird ein Ablauf dargestellt, bei dem die Sprachqualität schlecht bzw. sehr schlecht war. Es erscheint wieder zuerst die Frage nach der Sprachqualität. Nach einem Klick auf „schlecht“ oder „sehr schlecht“ erscheint die Frage nach der Verärgerung. Diese Frage kann ebenfalls mit vier Möglichkeiten beantwortet werden. Nach der Beantwortung dieser Frage wird als Letztes wieder nach dem Ort gefragt, an dem sich der Benutzer befunden hat. Nach dem Klick auf den Ort wird eine kurze Information angezeigt, dass die Befragung erfolgreich abgeschlossen wurde und es erscheint wieder die Ansicht, die der Benutzer vor der Befragung aktiv hatte.

Ablauf 3: Gespräch mit Verbindungsabbruch

Hier wird ein Ablauf dargestellt, bei dem die das Gespräch durch einen Verbindungsabbruch unterbrochen wurde. Es erscheint wieder zuerst die Frage nach der Sprachqualität. Nach einem Klick auf den Button „Verbindungsabbruch“ erscheint die Frage nach der Verärgerung, gefolgt von der Frage nach dem Ort, an dem sich der Benutzer befunden hat. Nach dem Klick auf den Ort wird eine kurze Information angezeigt, dass die Befragung erfolgreich abgeschlossen wurde und es erscheint wieder die Ansicht, die der Benutzer vor der Befragung aktiv hatte.

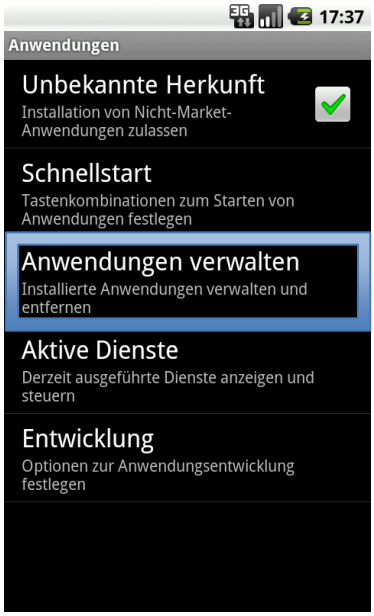
Ablauf 4: Manueller Upload

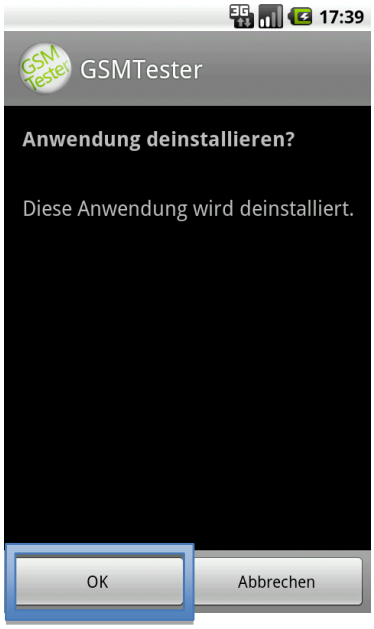
In diesem Ablauf wird beschrieben, wie die Navigation betreffend manuellem Upload funktioniert. Dafür wird in den Einstellungen die Option Automatischer Upload deaktiviert (siehe Kapitel 3 und 4.3). Anschliessend kann mit dem Backbutton auf den Startbildschirm von GSM Tester gewechselt werden. Nun erscheint am unteren Bildschirmrand ein Uploadbutton. Sobald mindestens ein Anruf zum uploaden bereit steht, kann mit einem Klick auf den Button der Upload Vorgang gestartet werden.



9.2.3.5. Deinstallation

Um GSM Tester zu deinstallieren, gehen Sie bitte ins Menü Ihres Android Mobiltelefon. Dort wählen Sie Einstellungen / Anwendungen / Anwendungen verwalten / GSM Tester. Anschliessend klicken Sie auf den Button „deinstallieren“. Damit wurde GSM Tester komplett deinstalliert.





9.3. Kritik GUI Gestaltung Verbindungsabbruch

2.1. Variante 1: Zusätzliche Frage

Wie war die Sprachqualitaet:

- Sehr gut
- Gut
- Schlecht
- Unverstaendlich

Diese Auflistung suggeriert eine immer schlechtere Rangfolge, allerdings ist es nicht klar welches der letzten beiden schlechter ist (schlecht oder unverstaendlich). Eindeutiger waere als letzter Punkt “sehr schlecht” (statt unverstaendlich).

Wenn es um die Verstaendlichkeit geht koennte man auch direkt nach der Verstaendlichkeit fragen.

2.2. Variante 2: Listelement in Sprachqualität

Wie war die Sprachqualitaet:

- Sehr gut
- Gut
- Schlecht
- Unverstaendlich
- Verbindungsabbruch

Dassselbe Problem wie oben. Zusaetzlich: Was soll ein User anklicken wenn die Qualitaet schlecht war und es einen Verbindungsabbruch gab? Fuer die User werden gibt es hier kein klares Antwortschema. Zusaetzlich sind wahrscheinlich eher schlechte Verbindungen von einem Abbruch betroffen.

2.3. Variante 3: Zusätzlicher Button in Sprachqualität

Hier wird das Problem des Verbindungsabbruchs aus Variante 2 besser geloest. Verbindungsabbruch wird hier als eigenstaendiges Element gefuehrt.

Ist den beteiligten hier klar was mit Verbindungsabbruch gemeint ist (nicht die Umfrage wird abgebrochen sondern das vorherige Gespraech wurde unterbrochen)?

9.4. Systemtest

ID	Aktion	erwartetes Ergebnis	Resultat
T1	Start der Applikation	leerer Startscreen erscheint keine Progressbar kein UploadButton	**ok**
T2	Click auf Menü Click auf Einstellungen Ändern der Fragehäufigkeit auf 2 Automatischer Upload deaktivieren Click auf Backbutton	Einstellungen werden korrekt übernommen Startscreen erscheint Progressbar erscheint UploadButton erscheint (deaktiviert) Meldung: Es sind keine Calls zum Upload bereit	**ok**
T3	Click auf Home Button	Homescreen wird angezeigt	**ok**
T4	Tätige einen Anruf	es erscheint keine Befragung	**ok**
T5	Tätige einen zweiten Anruf	Nach Beenden des Gesprächs erscheint die Frage nach der Sprachqualität	**ok**
T6	Beantworte die Fragen	Befragung wird durchgeführt Zum Schluss erscheint eine Meldung, dass die Befragung erfolgreich durchgeführt wurde Screen wechselt auf Homescreen	**ok**
T7	Starten von GSM Tester	Es sind zwei Calls zum Upload bereit Uploadbutton aktiv	**ok**
T8	Tätige 10 Anrufe und unterbreche die jeweiligen Befragungen	Es erscheinen fünf Calls in den offenen Bewertungen Der Uploadbutton ist aktiv Es erscheint die Meldung, dass 5 Calls zum Upload bereit stehen	**ok**
T9	Click auf Menü Button Click auf alle offenen Befragungen abschliessen	Es erscheint die Meldung, dass keine offenen Bewertungen vorhanden sind Die Liste mit den offenen Bewertungen ist leer Es erscheint die Meldung, dass 10 Calls zum Upload bereit stehen	**ok**
T10	Click auf Upload	Meldung: Reports werden generiert mit Prozentanzeige Progressbar füllt sich Meldung: Upload wird gestartet Meldung: Upload mit Prozentanzeige und Geschwindigkeit Progressbar füllt sich Meldung, dass Upload erfolgreich beendet inkl. Klick auf OK nötig	**ok**
T11	Click auf ok	Meldung: Es sind keine Calls zum Upload bereit Uploadbutton: deaktiviert	**ok**

T12	Click auf Menü Autoupload aktivieren Click auf Backbutton	Uploadelemente sind nicht mehr sichtbar	**ok**
T14	Tätige zwei Calls Beantworte eine Befragung warte eine Minute Starte GSM Tester deaktiviere AutoUpload in den Einstellungen	Meldung: es sind keine Calls zum Upload bereit	**ok**
T15	generiere 50 Calls Klick auf Upload Klick auf Abbrechen währen Generierung	Generierung wird gestartet Text gibt an, dass Generierung gestartet wurde und zeigt aktuellen Fortschritt in % an. Progressbar füllt sich gemäss Fortschritt. Nach Klick auf abbrechen, wird wieder angezeigt, dass 50 Calls zum Upload bereit sind	**ok**
T16	Start wieder mit 50 generierten Calls Klick auf Upload Klick auf Abbrechen während Uplaod (min. 5 s warten, damit einige Calls übertragen wurden)	wie bei T15, jedoch wird Generierung abgeschlossen und Uplaod startet. Text zeigt aktuellen Fortschritt in % und Übertragungsgeschwindigkeit in KBit/s an. Progressbar füllt sich gemäss Fortschritt. Nach Klick auf abbrechen, wird angezeigt, dass nun weniger als 50 Calls zum Upload bereit sind	**ok**
T17	Start mit übrigen Calls aus T16 Klick auf Upload Nach Upload, Klick auf Ok	wie bei T16, jedoch wird Upload nun komplett durchgeführt. Es erscheint Statusmeldung über Progressbar, dass Uplaod erfolgreich beendet wurde. Ebenfalls erscheint ein Popup, dass Uplaod erfolgreich beendet wurde. Nach klick auf Ok, erscheint wieder Meldung, dass keine Calls zum Upload bereit sind und Uploadbutton ist deaktiviert	**ok**
T18	Tätige Anrufe, bis Befragung startet. Klick auf Bewertung unterbrechen Starte GSM Tester	folgende Elemente werden angezeigt: Frage: Wie war die Sprachqualität? Antworten: sehr gut / gut / schlecht / sehr schlecht Button: Verbindungsabbruch / Bewertung unterbrechen nach Klick auf Bewertung unterbrechen wird Ausgangsscreen angezeigt Nach Start von GSM Tester erscheint Call in offenen Gesprächsbewertungen mit Nummer, Datum und Zeit	**ok**
T19	Klick auf Call unter Offenen Gesprächsbewertungen Click auf schlecht / stark / Im Zug	Bewertung wird wieder an Verlassungspunkt gestartet. Nach Klick auf schlecht erscheint Frage: "Wie sehr sind Sie verärgert?". Nach Klick auf stark erscheint Frage: "Wo waren Sie während dem Gespräch?" Nach Klick auf "Im Zug" wechselt Screen auf StartActivity und zeigt kurzzeitig einen Toast an, dass Bewertung erfolgreich abgeschlossen wurde	**ok**
T20	Android Einstellung Wi-Fi aktivieren	Applikation kann gestartet werden	**ok**
T21	Android Einstellung Wi-Fi deaktivieren	Applikation kann gestartet werden	**ok**
T22	Android Einstellung GPS aktivieren	Applikation kann gestartet werden	**ok**
T23	Android Einstellung GPs deaktivieren	Applikation kann gestartet werden	**ok**

T24	Android Ort Einstellungen "Wireless nutzen" und "GPS-Satelliten verwenden" deaktivieren	Applikation kann gestartet werden	**ok**
T25	Android Ort Einstellungen "Wireless nutzen" aktivieren und "GPS-Satelliten verwenden" deaktivieren	Applikation kann gestartet werden	**ok**
T26	Android Ort Einstellungen "Wireless nutzen" aktivieren und "GPS-Satelliten verwenden" aktivieren	Applikation kann gestartet werden	**ok**
T27	Android Ort Einstellungen "Wireless nutzen" deaktivieren und "GPS-Satelliten verwenden" aktivieren	Applikation kann gestartet werden	**ok**

9.6. Sitzungsprotokolle

9.6.1. Sitzungsprotokoll 23.09.10

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektübersicht verschaffen
- Fragekatalog durcharbeiten

Diskussion / Beschlüsse:

- 1. Challenge: GSM-Stack analysieren und erreichen, dass die drei Unterbruchszenarien erkennbar sind (zu Beginn kein Netz, Unterbruch während Gespräch, Gespräch wurde gewollt beendet)
- Upload WLAN: Definition von gewünschten WLAN's oder jedes WLAN ermöglichen
- Abfrage vor Upload oder nicht? -> muss einstellbar sein, default auf nein.
- Schauen, wie Android WLAN's erkennt (offenes WLAN ja, aber ist auch Internet vorhanden)
- Anforderungsanalyse an Herrn Glatz, anschliessend ok und Mail an Herrn Fiedler
- Quellcode inkl. Doku zum Schluss auch an Herrn Fiedler
- Allfälliges Zwischenziel: möglichst viele Infos bei Anrufende aus GSM-Stack lesen
- Evtl. mit Debuginfos vom Emulator arbeiten
- Counter einbauen, der Anzahl erfolgreiche und Anzahl unterbrochene Anrufe führt
- Fokussierung bei Tests auf die drei Hauptprobleme
- Analyse inkl. Dokumentation des GSM-Stacks scheint zentraler Aspekt der Arbeit zu sein

Offene Punkte Verantwortung (erledigt vor nächster Sitzung):

- Möglichkeiten untersuchen, GSM Infos auf Android zu erlangen

Nächster Termin:

Datum: Donnerstag, den 30.09.10
Zeit: 17.00
Dauer: ca. 1h
Ort: Zimmer 6.112

9.6.2. Sitzungsprotokoll 30.09.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

Fragen von unserer Seite:

- Ist die weitere Analyse betreffend Zugriff auf Daten des RIL gewünscht?
- Können wir nun Fokus auf die App mit bestehenden Funktionen legen?
- Dokument mit definierten Fragen von Herr Fiedler an uns schicken
- Virtueller Server?

Diskussion:

- Virtueller Server noch offen
- Sprachqualität hoch, wenn Biterrorate tief
- GSM Zustände (Zustandsmaschine) in RFC nachschauen
- In welchem Zustand ist Protokollstack
- Analyseseiten (Homepages) die aufschlussreich waren, lokal speichern
- Benutzer nicht unnötig mit Abfragen belästigen
- Dokumentieren, was erreichbar ist und was nicht.
- Was ist opensource, was closed source
- Hack auch dokumentieren
- Auch von Hand sollten Kommentare zu Gespräch gemacht werden können
- Fragen an Herr Fiedler:
    - Hack weiterverfolgen?
    - 3 Indikatoren die wir haben, aufzählen und fragen, ob wir aus diesen die nötigen Infos ziehen können
    - Betreffend Testgerät (HTC Desire) fragen
    - Evtl. Feldversuche mit statistischer Auswertung erstellen
    - Mit welchen Bedingungen funktioniert Verbindung noch?
    - Mindestens 15 Werte (Mittelwert, Medianwert, Streuung)

Nächster Termin:

Datum: Donnerstag, den 07.10.2010
Zeit: 17.00
Dauer: 30 min
Ort: Cafeteria

9.6.3. Sitzungsprotokoll 07.10.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

- Virtual Server ohne W2k8
- Momentan Testapp für Feldtests
- Gesprächsdauer auch aufzeichnen

Bis nächste Woche:

- Analyse Parameter
- Allenfalls Paperprototype
- Zeitplan definitv

Nächster Termin:

Datum: Donnerstag, den 14.10.2010
Zeit: 17.00 Uhr
Dauer: 30min
Ort: Cafeteria

9.6.4. Sitzungsprotokoll 14.10.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen (Erkenntnis, dass Funktionsumfang der Applikation stark eingeschränkt ist, da wir zu wenig technische Daten erhalten)
- Weiteres Vorgehen besprechen
- Paperprototypes
- Projektplan
- Signalstärke mit Logs zeigen

Diskussion / Beschlüsse:

- Paperprototypes dokumentieren und an Herr Fiedler schicken
- Mibilität des Benutzers aufzeichnen
- Zusammentragen, was geht, was nicht:
  - Problem aufzeigen, dass zu wenig für Arbeit
  - Ideen für Herrn Fielder für neue Aufgabe
  - Plan-B erarbeiten
- Verbindungsproblem erkennen

Auf nächste Woche:

- Zuverlässigkeit der Erkennung des Verbindungsunterbruches untersuchen
- Dokumentieren → Machbarkeitsstudie
- Inkl. Vorschläge, was mit diesen Werten gemacht werden könnte
- Themen für Plan B

Nächster Termin:

Datum: 21.10.10
Zeit: 17.00 Uhr
Dauer: 30 min
Ort: Cafeteria
Info: Allenfalls Zusatztermin mit Herr Fiedler für Besprechung weiteres Vorgehen für Arbeit

9.6.5. Sitzungsprotokoll 29.10.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

Mail an Herr Fiedler:

- Fragen, ob Telefonnummer überhaupt nötig ist (Datenschutzbedenken)
- Falls ja, unter welchen Bedingungen?

Nächster Termin:

Datum: 04.11.10
Zeit: 17.00 Uhr
Dauer: 30 min
Ort: Cafeteria

9.6.6. Sitzungsprotokoll 11.11.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

Besprechung mit Herr Fiedler:

- Festlegen von XML-Struktur
- Test XML in Excel öffnen
- Format der Bedienungsanleitung besprechen

Nächster Termin:

Datum: 18.11.10
Zeit: 17.00
Dauer: 30min
Ort: Cafeteria

9.6.7. Sitzungsprotokoll 18.11.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

- Infos von Besprechung mit Herr Fiedler per Mail an Herr Glatz senden

Nächster Termin:

Datum: 02.12.10
Zeit: 13.30
Dauer: 30 min
Ort: Cafeteria

9.6.8. Sitzungsprotokoll 19.11.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Präsentation der Applikation
  - Hauptscreen
  - Einstellungen
  - Eine Befragung durchspielen
  - Eine Befragung abbrechen
  - Durchgang mit Autoupload / ohne Autoupload
- XML-Datei auf Server betrachten
  - Struktur der XML-Datei besprechen und gewünschte Änderungen notieren
- Allgemeine Fragen
  - Befragungskonzept in Ordnung?
  - Bedienungsanleitung:
  - Wie soll diese aussehen? Auf Papier? Im Hilfemenu?
  - Wo bzw. für was einen Button?
  - Wo soll Dokumentation der Deinstallation gemacht werden?
  - Nachbarzellen können während Call nicht aufgezeichnet werden

Diskussion / Beschlüsse:

- Mehrsprachigkeit gewünscht, falls zeitlich möglich
- Fragehäufigkeit: Methode soweit vorbereiten, dass mittels Option auch Random-Verhalten konfiguriert werden kann
- Schätzung von Datenmenge im Bericht:
  - Wieviele Daten gehen über Netzwerk?
- Verbindungsabbruch:
  - Mail an Herr Fiedler mit verschiedenen Vorschlägen mit Screens zur Darstellung von Elementen, die den Verbindungsabbruch abfragen (zusätzlicher Button, in bestehende Listview integriert usw.) -> wird für Besprechung mit Usability-Psychologen verwendet
- Momentan ist kein Retour in Befragung möglich, dies soll dokumentiert werden. Ebenfalls erwähnen, dass es erweitert werden kann.
- Bei Abbruch, nur nach Verärgerung und Ort fragen
- Bei No Access, Veränderung und Ort fragen -> Aufzeichnen von letzter Location mit Zeitdelta(wie lange her)
- Dokumentation, welche Möglichkeiten mit gerootetem Phone bestehen (auf einer Seite zusammenfassen)
- Anpassung XML-File:
  - Zeit besser ausgeben
  - Für Befragung → nur Zahlen speichern, Mapping zu Werten in Dokumentation
- In Bericht:
  - Statediagram von Screens der Befragung
  - Falls Zeit vorhanden: Wenn man Testlauf der Applikation an der HSR durchführen möchte, überlegen, auf was geachtet werden muss. z.B. Anreize für Installation der Anwendung usw.
- In Bedienungsanleitung:
  - Befragung umfasst maximal vier Fragen
  - Es sind maximal vier Clicks nötig, um wieder bei der ursprünglichen Position zu sein
  - Unterbrechungsfunktion von Befragung beschreiben
  - Wesentliche Menüführung dokumentieren

- Upload ca. xx-Bytes, daher merkt man nichts (entspricht 2 SMS)
- Deinstallation (wie diese durchgeführt wird)

Nächster Termin:

Datum: Donnerstag, den 02.12.10
Zeit: 17.00 Uhr
Dauer: 30 min
Ort: Cafeteria

9.6.9. Sitzungsprotokoll 02.12.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

- Button: Verbindungsabbruch orange anzeigen
- Hilfetext an Herr Fiedler
- Bericht vorher an Herr Fiedler und Herr Glatz schicken, für Feedback falls noch Themen offen sind, diese in Bericht als offen deklarieren
- Nummerierte Liste für Betrieb auf anderen Geräten (Test mit anderen Phones)
  →in Bericht einfliessen lassen

- Bericht: Mehrsprachigkeit dokumentieren, was man noch machen muss um zusätzliche Sprachen hinzuzufügen
- Weitere Arbeiten:
  - Betrieb auf anderen Geräten
  - Weitere Ideen
- Bericht am 23.12.10 in Postfach bei Sekretariat (Gebäude 4 links) für Herr Heinzmann und Herr Glatz (je 1x gedruckter Bericht inkl. CD)
- Für Herr Fiedler CD mit Bericht und Code, fragen ob gedruckte Version erwünscht

Nächster Termin:

Datum: 09.12.10
Zeit: 13.30
Dauer: 30 min
Ort: Cafeteria

9.6.10. Sitzungsprotokoll 09.12.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen
- Gestaltung Bericht:
  - Gehört Bedienungsanleitung in Bericht?
  - Muss im Bericht stehen, wieso wir das so gemacht haben? → soll Entscheidungsfluss ersichtlich sein oder nur Endresultat?
  - Separates SAD nötig?
  - Anpassung von „Verbindungsabbruch“-Button auf orange verfälscht Usability
  - UnitTest?

Diskussion / Beschlüsse:

- Bedienungsanleitung in Anhang
- Entscheidungsfluss muss ersichtlich sein
- Kein separates SAD
- Evtl. farbigen Rahmen um Button
- Da Applikation überschaubar, keine UnitTests nötig. Jedoch diesen Sachverhalt in Bericht festhalten. → Konzentration auf Integrationstest / Funktionstest / Systemtest
- Anforderungsspezifikationen in Analyse
- Was muss gemacht werden in Analyse
- Hauptteile von Bericht: Analyse / Design / Implementierung
- Projektmanagement egal ob Hauptteil oder Anhang
- Sitzungsprotokolle in Anhang
- Codebeschreibung im Detail als Javadoc
- Wichtig für Bericht: Vergleich Resultat mit Aufgabenstellung:
  - Schlussfolgerung, was wurde erreicht
  - Was nicht
  - Was könnte noch gemacht werden
  - Was möchte Herr Fiedler mit Produkt machen

Nächster Termin:

Datum: Donnerstag, den 16.12.10
Zeit: 13.30
Dauer: 30 min
Ort: Cafeteria
Wichtig: Bis zu diesem Termin Bericht so weit als möglich als PDF an Herr Glatz

9.6.11. Sitzungsprotokoll 16.12.2010

Teammitglieder / Kürzel: Reto Zigerlig / rez
Stefan Schöb / scs

Traktanden:

- Projektstatus besprechen
- Weiteres Vorgehen besprechen

Diskussion / Beschlüsse:

- GUI Navigation für Befragung auch nochmals in Bericht oder reicht Benutzeranleitung?
  - höchstens Verweis / Überlegungen und Feedback von Kunde in Bericht
- Kann Poster (Header) anders gestaltet werden? → Frau Furrer fragen
- Wie soll Projektplan in Bericht gebunden werden?
  - Zeitauswertungsdiagramme wurden erstellt, muss nun komplexer Projektplan noch in Anhang? → A3-Querformat
- Wann Feedback zu Bericht? → Anfangs nächste Woche
- Abstract zuoberst in Bericht
- Projektreview auf Grobaufteilung anwenden
- Für Präsentation für Herr Fiedler:
  - Motivation
  - Umsetzung
  - Zusammenfassung (Nutzen von Applikation, Aussichten, Verbesserungen)

Detailplan GSM Tester

Phasen				Inception				Elaboration																Construction I												Construction II								Transition							
Meilensteine	MS									MS1								MS2						MS3												MS4						MS5						MS6			
Woche (von - bis)		Total		1 20.09.10 RZ 26.09.10 SCS				2 27.09.10 RZ 3.10.10 SCS				3 4.10.10 RZ 10.10.10 SCS				4 11.10.10 RZ 17.10.10 SCS				5 18.10.10 RZ 24.10.10 SCS				6 25.10.10 RZ 31.10.10 SCS				7 1.11.10 RZ 7.11.10 SCS				8 8.11.10 RZ 14.11.10 SCS				9 15.11.10 RZ 21.11.10 SCS				10 22.11.10 RZ 28.11.10 SCS				11 29.11.10 REZ 5.12.10 SCS				12 6.12.10 REZ 12.12.10 SCS				13 13.12.10 REZ 19.12.10 SCS				14 20.12.10 REZ 26.12.10 SCS			
Task	Nr.	SOLL	IST	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I	S	I
Projekt Managment	**0x**	**34.0**	**16.0**	8.0	5.0	7.0	3.0	7.5	6.8	6.5	0.3	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.5	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0
Projektplan (inkl. Zeitplan)	1	20.0	8.5	5.0	4.0	5.0	2.0	5.0	2.5	5.0																																														
Risiko Management	2	4.0	4.0	2.0				2.0	4.0																																												
Dokumentvorlagen	3	4.0	2.0	1.0	1.0	2.0	1.0			1.0																																											
Zeiterfassung	4	6.0	1.5					0.5	0.3	0.5	0.3					0.5		0.5						0.5		0.5										0.5		0.5						0.5	1.0	0.5						0.5		0.5	
Requirements	**2x**	**14.0**	**6.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	3.5	7.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Use Case	20	4.0	2.5									2.0		2.0	2.5																																							
Anforderungsanalyse	21	10.0	3.5									5.0	3.5	5.0																																							
Analyse	**4x**	**90.0**	**75.5**	0.0	2.0	0.0	2.0	0.0	8.0	0.0	24.5	6.0	0.0	6.0	12.5	10.0	4.0	20.0	11.5	20.0	0.0	18.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Domain Model	40	4.0	2.0									2.0		2.0																																							
System sequenzdiagramme	41	4.0	2.0									2.0		2.0																																							
Activity Diagramm	42	4.0	4.0									2.0		2.0																																							
Analyse: Datenhaltung	44	8.0	8.5								5.5							8.0																																			
Analyse: Serverkommunikation / Datenupload	45	30.0	5.5								5.5							12.0				18.0																																
Analyse: Connectivity (GSM/WLAN)	46	40.0	53.5		2.0		2.0		8.0		13.5				12.5	10.0	4.0		11.5	20.0				10.0																															
Design	**6x**	**55.0**	**43.5**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	15.5	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	20.0	0.0	5.0	0.0	10.0	17.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Activity Architektur	60	5.0	2.0																					5.0																	2.0															
Datenbank Architektur	61	15.0	21.0																							15.0																											
Paperprototype	43	10.0	15.5													10.0	15.5																																					
Gesamt Architektur	62	25.0	5.0																					5.0		5.0		5.0		10.0								4.0		5.0													
Implementation	**8x**	**140.0**	**150.8**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	12.5	0.0	16.8	0.0	10.5	10.0	17.5	5.0	17.0	15.0	12.0	15.0	14.5	15.0	10.0	15.0	8.5	17.0	4.0	18.0	3.5	15.0	7.0	15.0	5.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0
Implementation Datenaufzeichnung	80	50.0	52.5																				12.5				10.5			5.0	9.0		15.0	14.5			15.0	6.0		15.0													
Implementation UI	81	50.0	80.8																		9.0				16.8			10.0	17.5							15.0	10.0			10.0	4.0		3.5	7.0				3.0							
Implementation Datenupload	82	40.0	17.5																												8.0								2.5	7.0		3.0		15.0		15.0	5.0								
Tests / Refactoring	**10x**	**70.0**	**45.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	6.0	0.0	10.0	0.0	10.5	20.0	5.0	20.0	8.5	15.0	3.0	15.0	0.0
Refactoring	100	25.0	35.0																																				2.0		6.0		6.0		10.5		2.0	20.0	8.5		5.0			
Systemtest	101	20.0	10.0																																											4.0		20.0	3.0						
Unit Tests	102	25.0	0.0																																														15.0		10.0					
Bericht	**12x**	**34.0**	**91.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	4.5	0.0	0.0	0.0	14.0	0.0	7.0	0.0	24.0	0.0	17.5	17.0	10.0	17.0	8.0
Bericht	120	20.0	91.0																				3.0																		3.0						4.5				14.0		7.0		24.0		17.5	10.0	10.0	10.0	8.0
Poster	121	10.0	0.0																																																				5.0		5.0	
Persönliche Berichte	122	4.0	0.0																																																			2.0		2.0	
Weiteres	**14x**	**43.0**	**30.0**	4.0	4.0	4.0	2.5	4.0	4.0	4.0	1.0	1.5	12.0	1.5	0.5	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Meetings	140	24.0	11.0		3.0		0.5				1.0				0.5				0.5				0.5												2.0								2.0													
Projekt Einarbeitung	141	19.0	19.0	4.0	1.0	4.0	2.0	4.0	4.0	4.0		1.5	12.0	1.5											0.5																				0.5													
Wochen Subtotal				12.0	11.0	11.0	7.5	11.5	18.8	10.5	25.8	14.5	15.5	14.5	15.5	20.5	19.5	20.5	12.0	20.0	9.0	18.0	16.0	20.5	17.3	20.5	10.5	15.0	17.5	15.0	34.0	15.0	15.0	15.0	14.5	15.0	12.0	15.0	17.5	17.5	17.0	18.5	9.5	15.0	21.5	15.0	17.5	20.5	23.5	20.5	17.5	15.0	27.0	15.0	17.5	17.5	10.0	17.5	8.0
Wochentotal		**480.0**	**457.8**			23.0	18.5			22.0	44.5			29.0	31.0			41.0	31.5			38.0	25.0			41.0	27.8			30.0	51.5			30.0	29.5			30.0	29.5			36.0	26.5			30.0	39.0			41.0	41.0			30.0	44.5			35.0	18.0

9.7. Quellverzeichnis

9.7.1. Abbildungen

Abbildung 1: Android Komponenten des GSM-Prozesses
Link: http://www.netmite.com/android/mydroid/development/pdk/docs/telephony.html
Letzter Zugriff: 07.10.10

Abbildung 3: PhoneState Diagramm
http://2.bp.blogspot.com/_FgDIN0F67B0/TGAIgFWv4GI/AAAAAAAACc8/wOADF8MDkPc/s1600/BasicInformation.png
Letzter Zugriff: 07.10.10

9.7.2. Bücher

Arno Becker, Marcus Pant: Android 2 - Grundlagen und Programmierung. dpunkt.verlag, 2010
ISBN: 978-3-89864-677-2

9.7.3. Weblinks

Android Development Guide
http://developer.android.com/guide/index.html
Letzter Zugriff: 21.12.10

Stackoverflow
http://stackoverflow.com/questions/tagged/android
Letzter Zugriff: 21.12.10

anddev.org - Android Development Community
http://www.anddev.org/
Letzter Zugriff: 21.12.10

9.7.4. Library

ftp4j Release 1.5.1 http://www.sauronsoftware.it/projects/ftp4j/

9.8. Glossar

Begriff	Beschreibung
Activity	Klasse aus dem Android-Framework. Controler für die Views.
Android Market	Auf dieser Webseite können Entwickler Ihre Applikationen für den Endbenutzer zur Verfügung stellen. Es können Applikationen gekauft oder kostenlos geladen werden. Der volle Zugriff auf den Market erfolgt nur über ein Androidfähiges Endgerät. Siehe http://www.android.com/market.
Android SDK	Software Development Kit, stellt eine Sammlung von Werkzeugen und Anwendungen zur Entwicklung von Android-Apps zur Verfügung. Es existiert für jede Android-Version ein eingenständiger SDK.
Cell Id	Identifikation der GSM Zelle
FTP	File Transfer Protocol, Protokoll zur Datenübertragung basierend auf TCP
GPS	Global Positioning System, globales Navigationssatetellitensystem welches von dem moisten Smartphones durch einen integrierten Chip unterstützt wird.
GSM	Global System for Mobile Communications; Ein Standard welcher in der Telefonie genutzt wird
Intent	Klasse aus dem Android-Framework. Dient zur Kommunikation und zur Datenübertragung zwischen Activities.
Location Area Code LAC	Code der aktuellen Location Area (Aufenthaltsbereich)
Logcat	Android Logsystem
NO ACCESS	Benutzer möchte ein Telefongespröch starten befindet sich aber in einem Gebiet in welchem er keinen Empfang hat.
RIL	Radio Interface Layer, stellt eine Schnittstelle zur Radio-Hardware des Smartphones bereit.
Service	Klasse aus dem Android-Framework. Stellt Grundfunktionalitäten für einen Hintergrund-Service zur Verfügung.
UMTS	Universal Mobile Telecommunications System