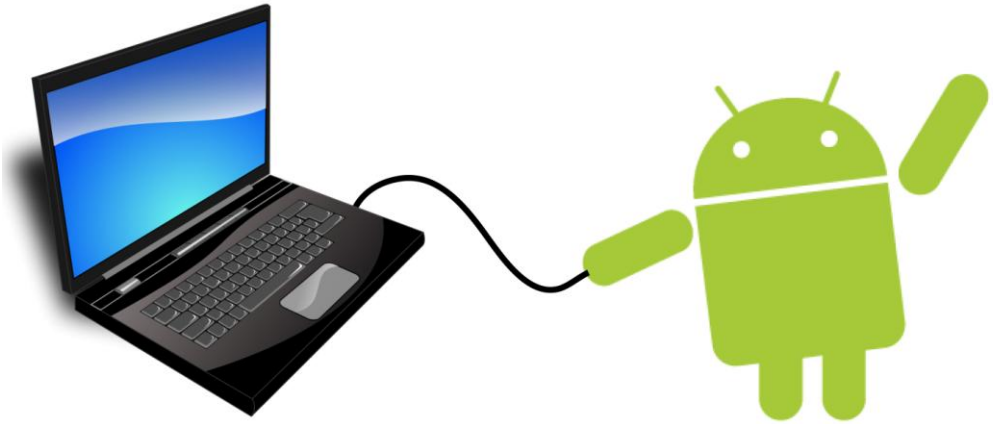
HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Android Control Framework

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil



Herbstsemester 2010

Autor(en): Daniela Meier, Ramona Rudnicki
Betreuer: Thomas Letsch
Gegenleser: Andreas Steffen

Inhaltsverzeichnis

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Kurzfassung der Studienarbeit

Abteilung	Informatik
Namen der Studierenden	Ramona Rudnicki Daniela Meier
Studienjahr	HS 2010
Titel der Studienarbeit	Android Control Framework
Examinatorin / Examinator	Thomas Letsch
Themengebiet	Software
Projektpartner	--
Institut	--

Um eine PC-Suite mit einer Verbindung zu einem Android-Gerät zu erstellen, muss ohne Zuhilfenahme des Android Control Framework (AnCoF) jeder Zugriff auf das Android-Betriebssystem neu programmiert werden. Zudem sind nicht alle Bereiche und Funktionen direkt über das öffentliche API verfügbar.

AnCoF bietet eine einfache Schnittstelle für die Nutzung ausgewählter Android-Funktionalitäten. Die Funktionen, welche AnCoF zur Verfügung stellt, decken folgende Bereiche ab:

- Kalender: Synchronisation
- Telefonie: Entgegennehmen und Auslösen von Anrufen via PC
- Telefoneinstellungen: Verändern der wichtigen Gesprächseinstellungen (z.B. Lautsprecher)
- Textnachrichten: Empfangen und Senden von SMS via PC
- Dateien und Ordner: Synchronisation
- Kontakte: Synchronisation

Zudem ist das „Task Management Framework on Smart-Phone“, welches das Erstellen und Austauschen von Aufgabenlisten ermöglicht, in AnCoF eingebunden. Es wird auch für die Verbindung zwischen PC und dem Mobiltelefon genutzt.

Projektplan
«Android Control Framework»

Version 1.3

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 6.

1.2 Gültigkeitsbereich

Dieses Dokument dient als Grundlage für das Projekt Android Control Framework (AnCoF) und behält daher seine Gültigkeit während der gesamten Projektdauer.

2 Projektübersicht

2.1 Ziel und Zweck

Es wird ein Framework für Android-Geräte entwickelt, welches Remote-Zugriff auf bestimmte Funktionen des Android-Systems bietet und damit die gängigen Aufgaben von PC-Suiten, wie z.B. Telefonieren, SMS versenden und Kalender oder Daten synchronisieren, ermöglicht.

Zudem wird das bestehende Task-Management-Framework on Smart-Phone (TaMaF) in das Framework eingebunden.

2.2 Vorgehen

Die Vorgehensweise entspricht dem Unified Process (UP), wie er in [2] beschrieben wird.

2.3 Annahmen und Einschränkungen

Die Soll-Arbeitszeit pro Projektmitarbeiter und Woche entspricht in etwa 18.5 Stunden. Längere Absenzen sind keine zu erwarten. Bei Ausfällen wegen Krankheit oder unerwarteten Problemen wird der Zeitplan angepasst bzw. die Arbeitszeiten erweitert.

3 Projektorganisation

3.1 Organisationsstruktur

Tabelle 3.1 zeigt die am Projekt beteiligten Personen und ihre Zuständigkeitsbereiche.

Bereich	**Verantwortlich**	**Artefakten**
Planung und Analyse	Daniela Meier (dm)	• Projektplan (Pp) • Anforderungsspezifikation (As) • Domainanalyse (Da)
Machbarkeit	Ramona Rudnicki (rr)	• Prototyp
Umsetzung	rr	• Software Architecture Document (SAD) • Code • Releases
Testen	dm	• Unit Tests • Systemtests

Tabelle 3.1: Organisationsstruktur

3.2 Externe Schnittstellen

Thomas Letsch (tl)	Projektbetreuer

4 Managementabläufe

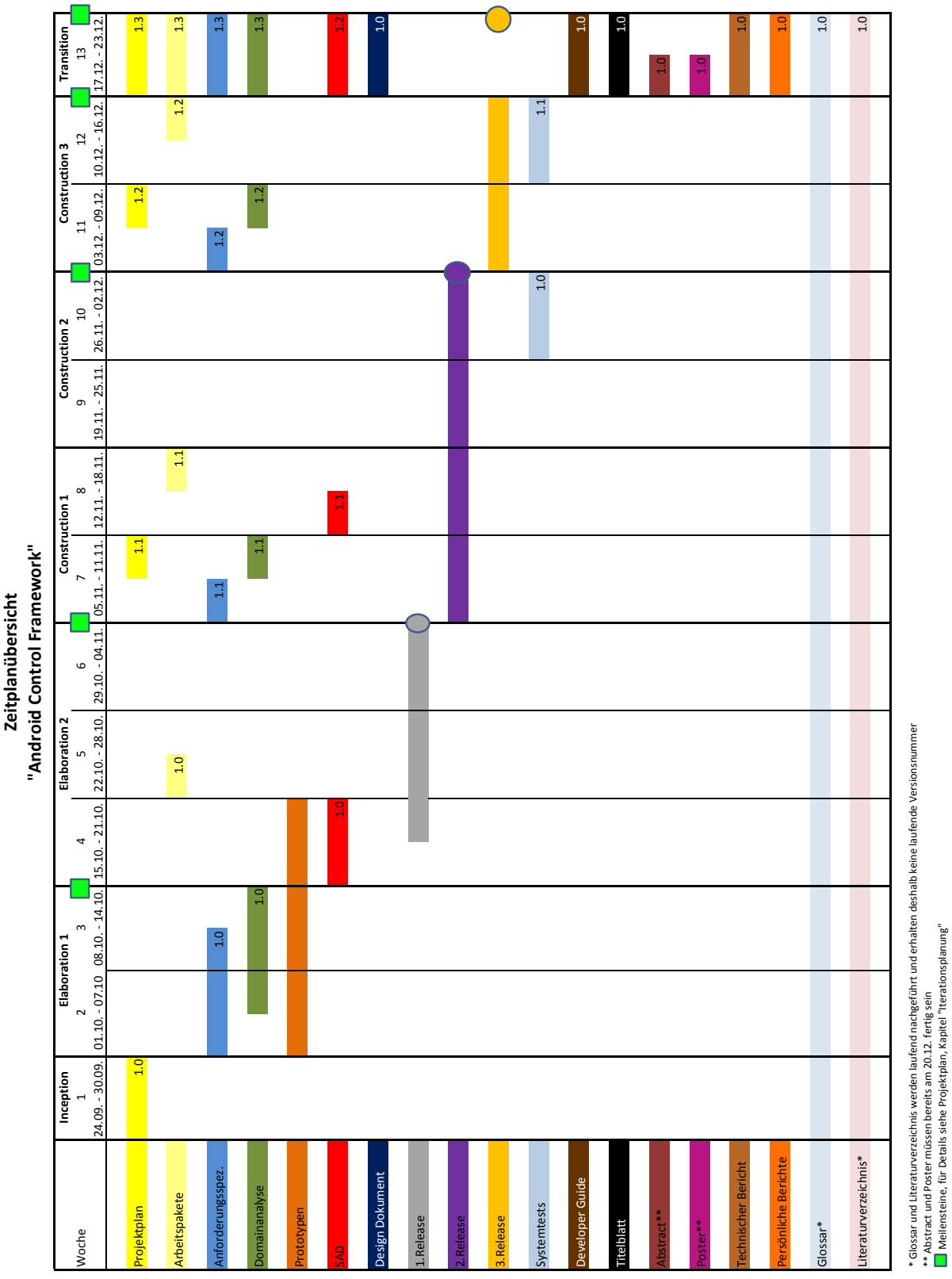
4.1 Projektdauer

Der Zeitplan (siehe Kapitel 4.2.1, Seite 4) basiert auf einer Arbeitszeit von 240 Stunden pro Teammitglied, was ein Total von 480 Stunden sowie eine durchschnittliche Wochenarbeitszeit von 18 bis 19 Stunden ergibt. Das Projekt beginnt am 23.09.2010 und endet mit der Abgabe am 23.12.2010.

4.2 Projektplan

4.2.1 Zeitplan

Die Abbildung 4.1 auf Seite 5 bietet eine Übersicht über den gesamten Projektverlauf, das heisst wann, welche Artefakte erstellt werden.


Zeitplanübersicht
"Android Control Framework"
Woche
Inception 1 24.09. - 30.09.
Elaboration 1 2 01.10. - 07.10 3 08.10. - 14.10.
4 15.10. - 21.10.
Elaboration 2 5 22.10. - 28.10. 6 29.10. - 04.11.
Construction 1 7 05.11. - 11.11. 8 12.11. - 18.11.
Construction 2 9 19.11. - 25.11. 10 26.11. - 02.12.
Construction 3 11 03.12. - 09.12. 12 10.12. - 16.12.
Transition 13 17.12. - 23.12.
Projektplan
Arbeitspakete
Anforderungsspez.
Domainanalyse
Prototypen
SAD
Design Dokument
1.Release
2.Release
3.Release
Systemtests
Developer Guide
Titelblatt
Abstract**
Poster**
Technischer Bericht
Persönliche Berichte
Glossar*
Literaturverzeichnis*
* Glossar und Literaturverzeichnis werden laufend nachgeführt und erhalten deshalb keine laufende Versionsnummer
** Abstract und Poster müssen bereits am 20.12. fertig sein
Meilensteine, für Details siehe Projektplan, Kapitel "Iterationsplanung"

4.2.2 Beschreibung der Artefakte

Projektplan

Der Pp definiert die Planung und Organisation des Projekts. Ausserdem dient er als Grundlage für weitere Projektdokumente. Geplante Versionen: Siehe Tabelle 4.1.

Version	Inhalt
1.0	Planung bis und mit Meilenstein 2.
1.1	Planung bis und mit Meilenstein 3.
1.2	Planung bis und mit Meilenstein 4.
1.3	Gesamtes Projekt.

Tabelle 4.1: Projektplan

Anforderungsspezifikation

Im Dokument As sind die funktionalen und nicht funktionalen Anforderungen an das Produkt AnCoF ersichtlich. Geplante Versionen: Siehe Tabelle 4.2.

Version	Inhalt
1.0	Anforderungen bezüglich Handling von Telefonanrufen und Synchronisation Kalendereinträge.
1.1	Anforderungen bezüglich Handling von SMS, Daten- bzw. Kontaktsynchronisation und Telefoneinstellungen.
1.2	Anforderungen an definitiv umgesetzte Funktionalitäten.
1.3	Final Review, keine inhaltlichen Änderungen, nur Überprüfung Rechtschreibung, Verweise, Abkürzungen.

Tabelle 4.2: Anforderungsspezifikation

Domainanalyse

Die Da ermöglicht es, sich eine Übersicht über die Business Logik des Projektes zu verschaffen. Für einen Gesamtüberblick kann das Domain Modell «Domainmodel.eap» zu rate gezogen werden. Geplante Versionen: Siehe Tabelle 4.3.

Version	Inhalt
1.0	Analyse Handling Telefonanrufe respektive Synchronisation Kalendereinträge.
1.1	Analyse Handling SMS, Daten- bzw. Kontaktsynchronisation und Telefoneinstellungen.
1.2	Analyse definitiv umgesetzter Funktionalitäten.
1.3	Final Review, keine inhaltlichen Änderungen, nur Überprüfung Rechtschreibung, Verweise, Abkürzungen.

Tabelle 4.3: Domainanalyse

Prototypen

Es werden mehrere kleine Prototypen erstellt, die jeweils einzelne Funktionalitäten implementieren.

Software Architecture Document

Das SAD liefert eine Übersicht der Architektur respektive über den logischen und physischen Aufbau des AnCoF. In den einzelnen Kapiteln wird eingehend auf die «Logical view», die «Process view», die «Deployment view» und die «Data view» eingegangen, wie sie in [2] beschrieben sind. Geplante Versionen: Siehe Tabelle 4.4.

Version	Inhalt
1.0	Architekturübersicht über die einzelnen Subsysteme und Schichten.
1.1	Ergänzung der neuen Aufgabenbereiche wie Handling SMS, Daten- bzw. Kontaktsynchronisation und Telefoneinstellungen.
1.2	Final Review, keine inhaltlichen Änderungen, nur Überprüfung Rechtschreibung, Verweise, Abkürzungen.

Tabelle 4.4: Software Architecture Document

Design Dokument

Das Design Dokument (DD) enthält Informationen über das Design respektive Design-Entscheide. Es ist eine Version geplant, Version 1.0, die erst zum Schluss erstellt wird.

Releases

Siehe 4.2.4, Seite 9.

Developer Guide

Der Developer Guide (DG) ist eine Anleitung, mit der ein Java-Programmierer erlernen kann, wie das Framework eingesetzt wird.

Es ist nur eine Version geplant, Version 1.0, welche alle von aussen zugänglichen Funktionen erläutert.

Systemtests

Die Systemtests werden in zwei Teilen durchgeführt. Erste Tests finden in der Woche 10 statt, weitere Tests folgen in der Woche 12. Die Resultate werden in der Testdokumentation festgehalten. Diese enthält die Dokumentation aller durchgeführter Systemtests und eine Übersicht aller Bereiche, die durch Unit Tests abgedeckt sind.

Allgemeine Arbeiten

Titelblatt, Abstract, Poster, technischer und persönlicher Bericht werden gemäss den Vorgaben der Abteilung Informatik erstellt.

Glossar

Das Glossar (Gl) listet wichtige Abkürzungen und Begriffe im Rahmen des Projektes auf und wird laufend ergänzt.

Literaturverzeichnis

Das Literaturverzeichnis enthält Angaben zur verwendeten Literatur und wird laufend ergänzt. Jeweils am Ende jedes Dokuments befindet sich eine Übersicht der referenzierten Werke. Somit wird zum Verständnis der einzelnen Dokumente nicht zusätzlich ein Literaturverzeichnis benötigt wird.

4.2.3 Iterationsplanung

Iteration Assessments

Die Inhalte der Iterationen werden an vorangehenden Iteration Assessments festgelegt. Für Details siehe Abschnitt 8.4.3, Seite 16.

Iterationen

Im UP sind vier Phasen definiert, welche in mehreren Iterationen durchlaufen werden. Die Tabelle 4.5 listet die geplanten Iterationen sowie deren Inhalte auf. Die genauen Daten, die Dauer sowie der Inhalt der Iterationen können der Zeitplanung entnommen werden (siehe Abschnitt 4.2.1, Seite 4).

Name	Beschreibung
Inception 1	Start des Projekts, Detailplanung des Projekts.
Elaboration 1	Erfassung Requirements und Analyse für Telefonanruf-Handling sowie Kalendersynchronisation. Mehrere kleine Prototypen implementiert.
Elaboration 2	Fokus auf 1. Release und SAD.
Construction 1	Erweitern der Requirements und Überarbeitung Analyse und SAD für SMS-Handling, Daten- bzw. Kontaktsynchronisation und Telefoneinstellungen.
Construction 2	Fokus auf 2. Release, Durchführung erster Systemtests.
Construction 3	Fokus auf 3. Release, erweitern der Requirements, Überarbeitung Analyse und SAD und Erstellen des Design-Dokumentes. Durchführung abschliessender Systemtests.
Transition	Verfassen vorgegebener Dokumente gemäss [1].

Tabelle 4.5: Liste der Iterationen

Meilensteine

Tabelle 4.6, Seite 9 listet die geplanten Meilensteine sowie die vorgesehenen Arbeitsresultate auf. Die genauen Daten der Meilensteine können der Zeitplanung entnommen werden (siehe Kapitel 4.2.1, Seite 4).

Nr.	Name	Arbeitsresultat
MS1	Pp, As und Da	Anforderungen an das Projekt und dessen Analyse durchgeführt.
MS2	1. Release	Handling von Telefonanrufen und Synchronisation Kalender ist möglich.
MS3	2. Release	Handling von SMS, Synchronisation der Daten bzw. Kontakte und Telefoneinstellungen.
MS4		Daten- und Kontaktsynchronisation fertiggestellt, Kalendersynchronisation bereinigt, Code-Freeze.
MS5	3. Release	Präsentation und Abgabe des Projekts.

Tabelle 4.6: Liste der Meilensteine

4.2.4 Releases

Am Ende von Elaboration 2, Construction 2 und Transition wird ein Release der Software erstellt. Die genauen Daten können der Zeitplanung entnommen werden (siehe Kapitel 4.2.1, Seite 4), die Releases und ihre Inhalte der Tabelle 4.7, Seite 9. Zu jedem Release wird ein Release-Dokument erstellt, welches Informationen zu speziellen Konfigurationen enthält.

Die Inhalte der Releases werden in der jeweiligen Projektplan-Version festgelegt.

Name	Iteration	Inhalt
0.1	Elaboration 2	Handling von Telefonanrufen sowie Kalendersynchronisation implementiert.
0.2	Construction 2	Handling von SMS, Daten- bzw. Kontaktsynchronisation und Telefoneinstellungen implementiert.
0.3	Transition	Abgabe der Studienarbeit.

Tabelle 4.7: Liste der Releases

4.2.5 Code Reviews

Während den Construction-Iterationen werden wöchentlich Code Reviews durchgeführt (für Details siehe Kapitel 8.4.2, Seite 15). Die Daten und Inhalte der geplanten Code Reviews können Tabelle 4.8 entnommen werden.

Datum	Review-Gegenstand	Revision
27.10.2010	Verbindungsaufbau Kalender, Telefon	159
03.11.2010	Kalendersynchronisation	201
10.11.2010	Kalendersynchronisation	215
17.11.2010	Kalendersynchronisation	254
24.11.2010	Datensynchronisation	281
01.12.2010	Kalendersynchronisation	308
15.12.2010	Gesamter Code	391

Tabelle 4.8: Liste der Code Reviews

4.3 Besprechungen

Wöchentliche Teamsitzungen sind freitags zwischen 10:10 und 11:50 angesetzt, jene mit dem Betreuer finden donnerstags zwischen 15.30 und 17.30 statt. Die Dauer kann je nach Bedarf variieren. Zusätzlich finden kürzere Besprechungen nach Absprache statt.

Teilnehmer:	dm, rr
Ort:	Hochschule für Technik Rapperswil (HSR), 1.262
Zeit:	Freitag, 10:10 bis 11:50 (wöchentlich)

Teilnehmer:	dm, rr, tl
Ort:	HSR, 5.207
Zeit:	Donnerstag, 15:30 bis 17:30 (wöchentlich)

5 Risiko-Management

Die Risiken, ihren Einfluss auf den Projekterfolg und die Gegenmassnahmen sind im separaten Dokument «Risikomanagement» beziehungsweise `Risikomanagement.xlsx` beschrieben.

6 Arbeitspakete

Die einzelnen Arbeitspakete (Ap) sind im Dokument Ap ersichtlich, welches als Ergänzung zum Zeitplan (siehe Kapitel 4.2.1, Seite 4) dient. Es definiert die Inhalte beziehungsweise Resultate der Ap, ihre Priorität, die Abhängigkeiten von anderen Ap sowie die Personen, welche die Umsetzung der Ap überwachen.

7 Infrastruktur

7.1 Räumlichkeiten

Teamsitzungen, Reviews und Präsentationen finden in den Räumlichkeiten der HSR statt.

7.2 Hardware

Jedes Projektmitglied arbeitet mit dem privaten Personal Computer (PC). Bei Ausfall eines PC stehen Ersatzrechner oder die HSR-Arbeitsplatzrechner im Raum 1.262 zur Verfügung.

7.3 Software

- Programmiersprache:
  - Java 6
- Entwicklungsumgebung:
  - Eclipse
  - Enterprise Architect (EA)
- Versionsverwaltungssoftware: Subversion[1], Zuständigkeit: rr
- Code-Analyse: Findbugs[2]
- Dokumentation: LaTeX

7.4 Kommunikation

Folgende Kommunikationsmittel werden eingesetzt:

- E-Mail
- Windows Messenger

[1] `https://svns.hsr.ch/AndroidControlFramework/`
[2] `http://findbugs.sourceforge.net/`

8 Qualitätsmassnahmen

8.1 Dokumentation

8.1.1 Allgemein

Die Personen, welche für die einzelnen Ap verantwortlich sind, identifizieren welche Dokumente im Zuge der Umsetzung eines Ap erstellt, aktualisiert beziehungsweise erweitert werden müssen. Sie kümmern sich auch um deren Umsetzung. Anschliessend werden die Dokumente einem Review unterzogen (siehe Kapitel 8.4.1, Seite 15).

8.1.2 Sitzungsprotokolle

Sämtliche Sitzungen werden protokolliert und innerhalb 24 Stunden dem anderen Projektmitglied respektive dem Betreuer per Mail als Attachement weitergeleitet. So wird sichergestellt, dass alle Entscheide und Diskussionspunkte festgehalten sind, ausserdem können Missverständnisse vermieden werden.

8.1.3 Zeitplan

Jedes Teammitglied erfasst seine Arbeitszeiten selbst und ist dafür verantwortlich, den Bedarf für die einzelnen Ap in die Gesamtübersicht des Dokuments «Zeitplan» respektive `Zeitplan.xlsx` einzutragen. Die Erfassung der täglichen Arbeiten erfolgt in einer separaten Excel-Tabelle «Zeiterfassung» beziehungsweise `Zeiterfassung.xlsx`, welche jedes Teammitglied für sich selbst zur eigenen Übersicht führt.

8.2 Aufgabenverwaltung

Todos, Tasks und Bugs werden in der bereitgestellten «Todo List» erfasst, welche es ermöglicht, die Aufgaben den einzelnen Teammitgliedern zuzuweisen.

8.3 Style Guides

8.3.1 Code

Der Programmcode richtet sich grundsätzlich nach den Code Conventions von Sun[1]. Diese werden mithilfe des Profils AnCoF von Eclipse durchgesetzt. Folgende Anpassungen gegenüber den Standardeinstellungen wurden vorgenommen:

- Indentation
  - «Align fields in columns» gesetzt.
  - «Statements within 'switch' body» gesetzt.
- White Space
  - Expression: Bei «Type cast» kein Abstand zwischen schliessender Klammer und Objekt.

[1] `http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html`

  - Arrays: Bei «Array initializer» keinen Abstand nach öffnender und vor schliessender geschweifter Klammer.

- Blank Lines
  - Zusätzliche Leerzeile vor erster Deklaration in einer Klasse.
- New Lines
  - Keine neuen Zeilen in jeglichen leeren Klassen, Methoden und Blöcken respektive zwischen leeren geschweiften Klammern.
- Control Statements
  - Bei einzeiligem «if-Statement» wird alles auf eine Zeile geschrieben.
- Line Wrapping
  - 'enum' declaration: Bei Konstanten erfolgt Zeilenumbruch dort, wo es nötig ist.
- Comments
  - «Enable header comment formatting» gesetzt.

Zudem beginnen Interfacenamen immer mit einem «I».

8.3.2 Dokumente

Für Dokumente stehen LaTeX-Dokumentvorlagen zur Verfügung, die eine einheitliche Formatierung begünstigen.

8.4 Reviews

8.4.1 Dokumentations-Reviews

Dokumente werden von den zuständigen Personen erarbeitet und vom jeweils anderen Teammitglied gegengelesen. Danach werden die Entwürfe im Team besprochen und inhaltlich überarbeitet. Eine Person übernimmt dann die Abschlussredaktion, welche die Überarbeitung durch zwei Korrekturlesungen beinhaltet. Jeder Revisor trägt sich in die «Revisionshistorie» ein, damit jederzeit Änderungen und Ergänzungen verfolgt werden können.

8.4.2 Code Reviews

Es werden regelmässig Code Reviews wichtiger Komponenten durchgeführt. Dabei liegt der Schwerpunkt bei folgenden Kriterien:

- Code-Richtlinien (nicht alles lässt sich mit dem erstellten Profil AnCoF formatieren)
- «Code Smell»
- Verständlichkeit

Unterstützt werden die Code Reviews durch eine Code-Analyse mit Findbugs, welche das Projekt unter anderem auf duplizierten oder nicht erreichbaren Code untersucht. Die Code Reviews werden im Check-in-Kommentar erfasst.

8.4.3 Iteration Assessments

An der letzten Sitzung vor dem Ende einer jeden Iteration wird ein Iteration Assessment durchgeführt. Es wird der aktuelle Status der Arbeiten beurteilt und die zu bearbeitenden Arbeitspakete beziehungsweise Ziele für die nächste Iteration festgelegt. In diesem Rahmen wird der Pp aktualisiert beziehungsweise revidiert.

8.5 Versionsverwaltungssystem

Im Subversion (SVN)-Repository werden sämtliche UP-Artefakte, die Dokumentation sowie der gesamte Quellcode der Software verwaltet. Alle Projektteilnehmer können auf diese Weise gemeinsam am Projekt arbeiten. Es dürfen nur funktionsfähige, funktional zusammenhängende Patches eingecheckt werden.

8.6 Tests

8.6.1 Unit Tests

Es wird auf Test Driven Development (TDD) gesetzt, das heisst, die Projektmitglieder sind verpflichtet, fortlaufend Unit Tests zu schreiben und gegen diese Tests zu entwickeln.

8.6.2 Systemtests

Anhand der Use Case (UC) werden Systemtests durchgeführt und protokolliert, um sicherzustellen, dass das System den funktionalen Anforderungen genügt.

8.7 Logging

Es wird ein Logging mit zweidimensionaler Laufzeit-Konfiguration eingesetzt. Dabei wird nach Kategorie (z.B. Package) und Log-Level gelogged.

Bei dichtem Logging, z.B. in Loops, wird zuerst der momentane Loglevel abgefragt, um unnötige Aufbereitung von Logausgaben zu verhindern.

So weit als möglich wird der Logging-Mechanismus des TaMaF übernommen.

8.7.1 Loglevels

Die Loglevels entsprechen denjenigen des TaMaF.

Loglevel	Wann benutzt
Severe	Dieser Level wird bei schweren Fehlern, welche das System zum Absturz bringen können, verwendet.
Warning	Aktionen, die zu schweren Fehlern und Abstürzen führen können, werden mit diesem Level gelogged.
Info	Normale Aktionen und Events, z.B. Aktion gestartet/beendet, werden mit Info gelogged.
Fine, Finer	Dieser Level wird für Debug-Nachrichten verwendet.

Tabelle 8.1: Loglevel

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	22.09.2010	dm	Dokument erstellt.
1.0rc02	22.09.2010	rr	Überarbeiten der Kapitel 1,2,3,5,7.
1.0rc03	26.09.2010	dm	Beschreibung Artefakte in Kapitel 4 ergänzt.
1.0rc04	28.09.2010	rr	Unterkapitel Logging in Kapitel 8 hinzugefügt.
1.0rc04	29.09.2010	dm	Allfällige Abkürzungen ergänzt.
1.0rc05	29.09.2010	rr	Review: Diverse Korrekturen.
1.0rc06	29.09.2010	dm	Allfällige Abkürzungen ergänzt.
1.0rc07	29.09.2010	dm	Artefaktbeschreibung «Design Dokument ergänzt».
1.0rc08	03.10.2010	dm	Kapitel gemäss Besprechung angepasst und ergänzt.
1.0rc09	08.10.2010	dm	Änderungen gemäss Besprechung angepasst.
1.0rc10	13.10.2010	dm	Review: Diverse Korrekturen.
1.0	15.10.2010	dm	Akzeptierte Version 1.0
1.1rc01	15.10.2010	dm	Änderungen gemäss Besprechung angepasst.
1.1rc02	10.11.2010	dm, rr	Kapitel Management Process angepasst.
1.1	20.11.2010	dm	Akzeptierte Version 1.1
1.2rc01	27.11.2010	dm	Testdokumentation-Beschreibung ergänzt.
1.2rc02	04.12.2010	dm	Dokument gemäss Stand Release 2 überarbeitet.
1.2	11.12.2010	dm	Akzeptierte Version 1.2
1.3rc01	19.12.2010	rr	Rechtschreibung und Layout überarbeitet.
1.3	20.12.2010	dm	Akzeptierte Version 1.3

Tabelle 8.2: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
Ap Arbeitspakete
As Anforderungsspezifikation
Da Domainanalyse
DD Design Dokument
DG Developer Guide
dm Daniela Meier
EA Enterprise Architect
Gl Glossar
HSR Hochschule für Technik Rapperswil
PC Personal Computer
Pp Projektplan
rr Ramona Rudnicki
SA Studienarbeit
SAD Software Architecture Document
SMS Short Message Service
SVN Subversion
TaMaF Task-Management-Framework on Smart-Phone
TDD Test Driven Development
tl Thomas Letsch
UC Use Case
UP Unified Process

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] HSR: *Anleitung: Dokumentation Semester-, Bachelor- und Diplomarbeiten*

[2] LARMAN, Craig: *Applying UML and Patterns*. Prentice Hall, 2004. – ISBN 978–0–13–148906–2

Risikoanalyse
«Android-Control-Framework»

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

21. Dezember 2010

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	22.09.2010	rr	Dokument erstellt
1.0rc02	13.10.2010	dm	Risikos aktualisiert
1.0	20.12.2010	dm	Akzeptierte Version

Nr	Risikotitel	Risikobeschreibung	max. Schaden [h]	Eintrittswahr-scheinlichkeit [%]	gewichteter Schaden [h]	Massnahmen zur Vermeidung/Verminderung	Vorgehen bei Eintreffen
R1	Ausfall Arbeitsstation	HW eines Projektmitgliedes fällt aus	8.00	5.00%	0.40	Alternative HW organisieren und mind. alle 4 Stunden erledigte Arbeiten im Repository einchecken	an HSR-Rechner bzw. eigenem Laptop arbeiten
R2	Ausfall SVN-Server	SVN-Server der HSR fällt aus	20.00	2.00%	0.40	lokale Kopien aktuell halten	Ausweichen auf anderen SVN-Server (z.B. Google)
R4	(Teil-) Ausfall eines Projektmitgliedes	Ausfall eines Projektmitgliedes aufgrund Krankheit, Unfall, Studiumabbruch	240.00	2.00%	4.80	Verantwortlichkeiten in Team klar definieren, Stellvertreter definieren	Mehrarbeit in Folgewochen, zusätzliche Arbeit auf anderes Teammitglied verteilen, Funktionsumfang kürzen
R5	Erfahrung der Projektmanager	Die Projektmanager haben zu wenig Erfahrung mit ähnlichen Projekten und können daher den Aufwand der Arbeitspakete nicht korrekt schätzen	10.00	30.00%	3.00	kritische Arbeitspakete frühzeitig bearbeiten und bei Unklarheiten mit Betreuer Rücksprache halten, iterativ planen	nicht alle geplanten Funktionen und Features umsetzen, Mehrarbeit, Sitzung mit Betreuer, um weitere Planung zu besprechen
R6	Probleme mit Technologien	Verwendete Technologien führen zu Problemen da das Basiswissen fehlt	50.00	30.00%	15.00	Versuchen zu vereinfachen, iterativ planen	Dokumentationen und Beispiele im Internet suchen, systematisch in Themen einarbeiten
R7	Ausfall des Testgerätes	Das verwendete Android-Telefon ist defekt und muss repariert/ersetzt werden	20.00	5.00%	1.00	Alternativgerät organisieren	soweit möglich Emulator benutzen; neues Gerät organisieren; später angesetzte Funktionalitäten vorziehen falls Emulator für Tests nicht ausreicht
Summe			**328**	**69.00 %**	**23.60**		

Arbeitspakete
«Android Control Framework»

Version 1.3

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Name	Resultat	Vorgang	Priorität	Wer
Projekt Management				
Projektplan	Projektplan erstellen	Aufgabenstellung	hoch	Team
Codierungsrichtlinien	Erstellen von Profil in Eclipse	–	tief	Team
Zeitplanung	Zeitplanung erstellen und in Projektplan einfliessen lassen	Releases	hoch	Team
Risikomanagement	Risiken analysieren und Gegenmassnahmen festlegen, in Projektplan einfliessen lassen	–	hoch	rr
Qualitätsmassnahmen	Qualitätsmassnahmen festlegen und in Projektplan einfliessen lassen	–	hoch	dm
Requirements				
Use Case (UC) brief	UC brief erstellen	–	hoch	dm
UC Diagram	UC Diagram erstellen	UC brief	tief	rr
Anforderungsspezifikation	Anforderungsspezifikationen erstellen	–	hoch	Team
Operation Contracts	Operation Contracts erstellen	System Sequence Diagramme	mittel	Team
Analyse				
Analyse Task-Management-Framework on Smart-Phone (TaMaF)	–	–	mittel	Team
Domainanalyse	Dokument Domainanalyse erstellen	–	mittel	dm
Domain Model	Domain Model erstellen	UC brief	hoch	rr
System Sequence Diagramme	System Sequence Diagramme erstellen	UC brief	mittel	Team
Design				
Design Model	Design Model erstellen	Domain Model	hoch	Team
Desing Dokument	Design Dokument erstellt	Design Model	hoch	Team
Logische Architektur	Logical Architecture Diagram erstellen	Anforderungsspezifikation	hoch	rr
Software Architecture Document (SAD)	SAD erstellen	Anforderungsspezifikation	mittel	rr
Implementation				
Architekturprototyp	Kalender- und Telefonprototypen implementieren	Anforderungspezifikation	hoch	Team
Verbindung	Verbindungsaufbau von Personal Computer (PC) zu Mobile-Device implementieren	Domain Model	hoch	rr
Kalender	Kalenderfunktionen implementieren	Domain Model	mittel	dm
Telefon	Telefonfunktionen implementieren	Domain Model	mittel	rr
Short Message Service (SMS)	SMS-Funktionen implementieren	Domain Model	mittel	rr

	Resultat	Vorgang	Priorität	Wer
Daten	Datenfunktionen implementieren	Domain Model	mittel	dm
Kontakte	Kontaktfunktionen implementieren	Domain Model	mittel	rr
Telefoneinstellungen	Telefoneinstellungs-Funktionen implementieren	Domain Model	mittel	rr
Code Review	Kontinuierliche Überprüfung der Code-Qualität wichtiger Komponenten	–	mittel	Team
Allgemeine Arbeiten	«Aufräumarbeiten»	–	tief	Team
Test				
Unit-Tests	Unit-Tests erstellen für TDD	Artefakte Design	mittel	Team
Systemtest	Funktionale Systemtests definieren und durchführen	Artefakte Implementierung	mittel	Team
Bugfixing	Verbliebene Fehler beheben	–	hoch	Team
Dokumentation				
Titelblatt	Titelblatt für Gesamtabgabe erstellen	alle Dokumente	tief	rr
Abstract	Übersicht über Projekt	–	tief	dm
Literaturverzeichnis	Übersicht über alle benutzten Werke und Internetreferenzen	–	tief	Team
Glossar	Definition projektspezifischer Begriffe	–	tief	Team
Technischer Bericht	Übersicht über Aufgabenstellung, Arbeit und Resultate	-	tief	Team
Persönlicher Bericht	Erfahrungsbericht	–	tief	Team
Poster	Übersicht über Projekt	–	tief	Team
Developer Guide	Anleitung für Entwickler die Android Control Framework (AnCoF) nutzen	Artefakte Implementierung	hoch	Team
Präsentation	Präsentation Studienarbeit (SA)	–	mittel	Team
Allgemeine Bereinigung Dokumente	Korrekturlesung	alle Dokumente	mittel	Team
Dokumentvorlagen	Vorlagen erstellt	–	mittel	dm
Javadoc	Alle Funktionen sind beschrieben	Anforderungsspezifikation	mittel	Team
Testdokumentation	Testdokumentation erstellen	Systemtests, Unit Tests	mittel	dm
Sitzungen				
Sitzungen mit Betreuer	Gemeinsame Projektplanung	–	hoch	Team
Teamsitzungen	Detailplanung	Sitzungen mit Betreuer	hoch	Team
Infrastruktur				
Arbeitsplatz einrichten	Subversion (SVN), Eclipse einrichten	–	hoch	Team

Tabelle 0.1: Liste der Arbeitspakete

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	26.09.2010	dm	Dokument erstellt.
1.0rc02	28.09.2010	dm	Dokument an Projekt angepasst.
1.0rc03	29.09.2010	dm	Arbeitspakete überarbeitet.
1.0rc04	27.10.2010	Team	Arbeitspakete überarbeitet
1.0	29.10.2010	dm	Akzeptierte Version 1.0
1.1rc01	17.11.2010	rr	Neue Arbeitspakete hizugefügt.
1.1	20.11.2010	dm	Akzeptierte Version 1.1
1.2rc01	15.12.2010	dm	Beschreibung fehlender Arbeitspakete.
1.2	19.12.2010	dm	Akzeptierte Version 1.2
1.3rc01	19.12.2010	rr	Rechtschreibkorrekturen und Layoutänderungen
1.3	20.12.2010	dm	Akzeptierte Version 1.3

Tabelle 0.2: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
PC Personal Computer
SA Studienarbeit
SAD Software Architecture Document
SMS Short Message Service
SVN Subversion
TaMaF Task-Management-Framework on Smart-Phone
TDD Test Driven Development
UC Use Case

Tabellenverzeichnis

Anforderungsspezifikation «Android Control Framework»

Version 1.3

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 6.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 Funktionale Anforderungen

2.1 Aktoren

Als Aktor wird die auf dem Framework aufbauende Applikation bezeichnet, mit anderen Worten die Applikation auf dem Personal Computer (PC) und jene auf dem Mobile.

2.2 UC-Diagramm

Das UC-Diagramm wird zwecks besserer Lesbarkeit in die einzelnen Teilbereiche aufgesplittet, welche sich jedoch alle auf dasselbe Framework beziehen.

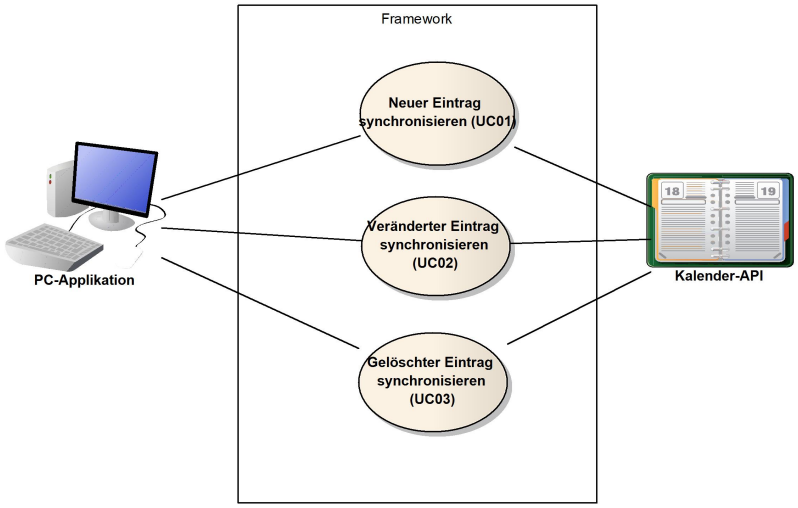


Abbildung 2.1: UC-Diagramm Kalender

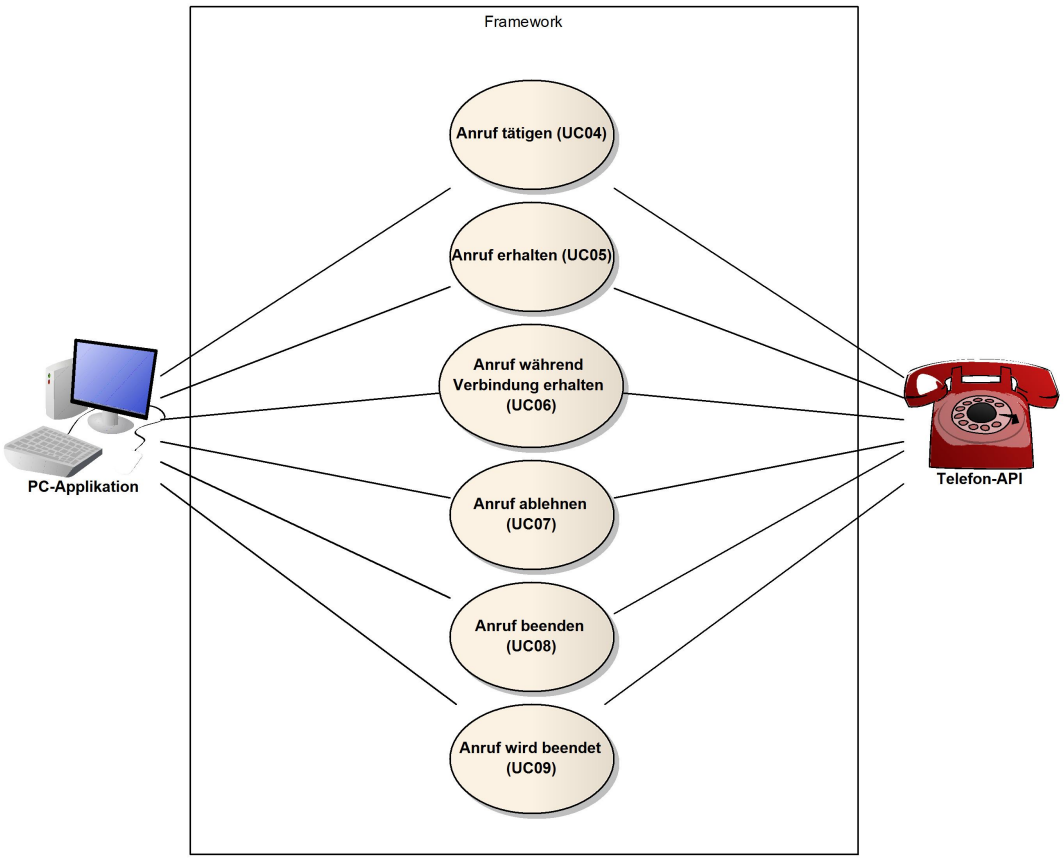


Abbildung 2.2: UC-Diagramm Telefon

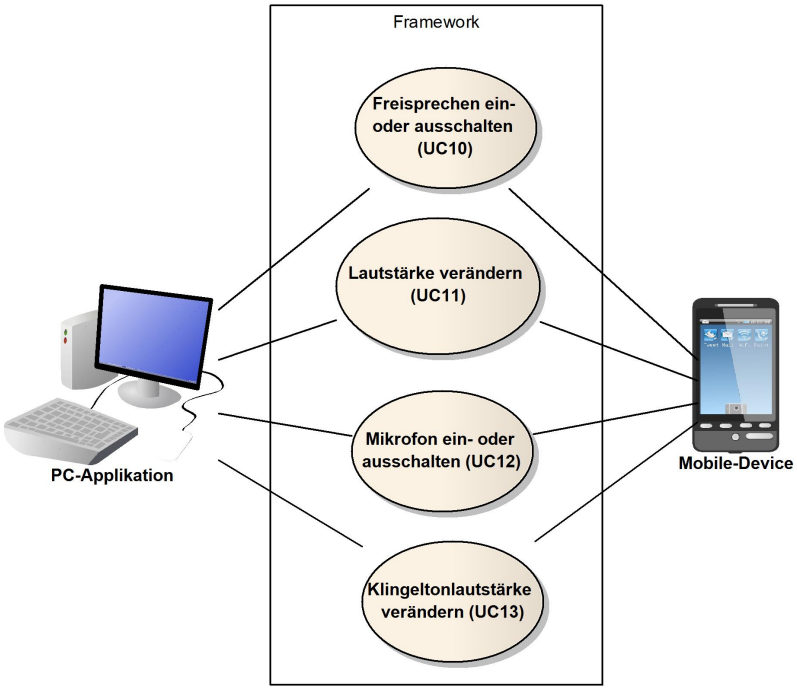


Abbildung 2.3: UC-Diagramm Telefoneinstellungen

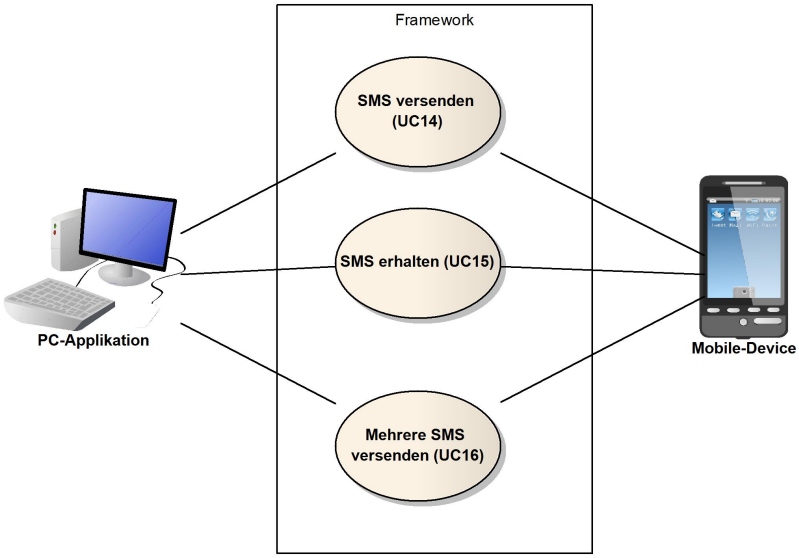


Abbildung 2.4: UC-Diagramm SMS

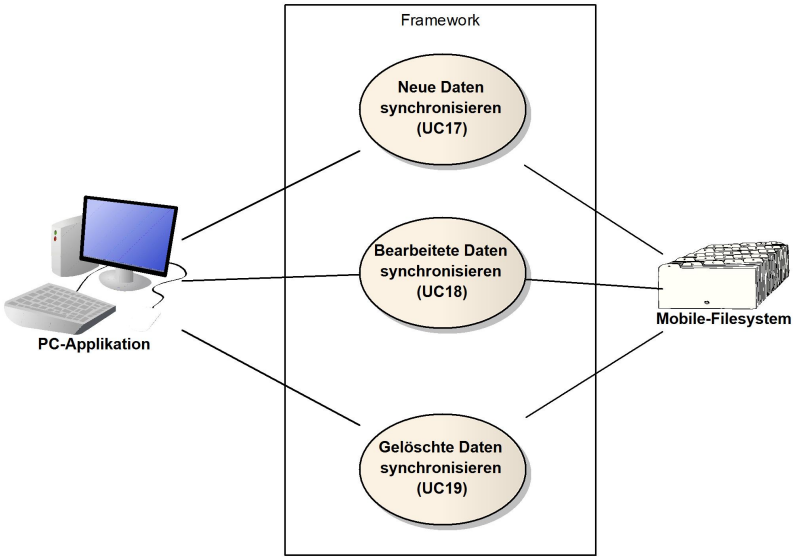


Abbildung 2.5: UC-Diagramm Datensynchronisation

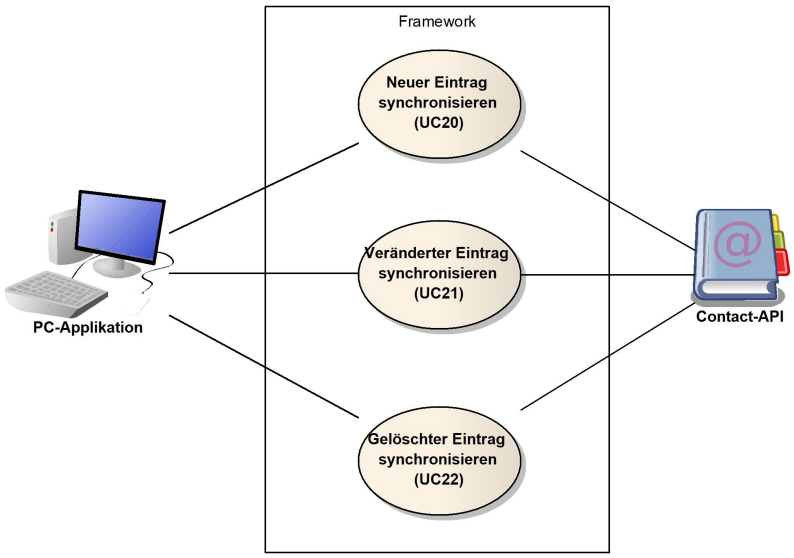


Abbildung 2.6: UC-Diagramm Kontakte

2.3 UC brief

Es wird angenommen, dass bereits eine Verbindung zwischen dem Mobile-Device und dem PC unter Zuhilfenahme des Task-Management-Framework on Smart-Phone (TaMaF) aufgebaut wurde. Die UC gehen davon aus, dass zu einem bestimmten Zeitpunkt nur ein Mobile-Device mit dem PC verbunden ist.

2.3.1 Kalendersynchronisation

Annahme: Es macht keinen Unterschied, ob Kalenderdaten auf dem Mobile-Device oder dem PC bearbeitet werden. Da das Android Control Framework (AnCoF) davon ausgeht, dass die Daten in einer kompatiblen Form vorliegen.

Neuer Eintrag synchronisieren (UC01)

Ein neuer Eintrag auf dem Mobile-Device/PC wurde erstellt und wird auf dem anderen Gerät hinzugefügt.

Veränderter Eintrag synchronisieren (UC02)

Ein Eintrag auf dem Mobile-Device/PC wurde bearbeitet und wird auf dem anderen Gerät aktualisiert.

Gelöschter Eintrag synchronisieren (UC03)

Ein Eintrag auf dem Mobile-Device/PC wurde gelöscht und wird auf dem anderen Gerät ebenfalls entfernt.

2.3.2 Telefonieren

Anruf tätigen (UC04)

Ein Anruf wird vom PC her aufgebaut.

Anruf erhalten (UC05)

Ein Anruf wird vom PC her entgegengenommen.

Anruf während Verbindung erhalten (UC06)

Während dem Verbindungsaufbau respektive einer aktiven Verbindung wird ein Anruf erhalten. Annahme: Eingehende Anrufe während dem Verbindungsaufbau und solche während einer bereits aufgebauten Verbindung werden gleich behandelt.

Anruf ablehnen (UC07)

Ein Anruf wird vom PC her abgelehnt.

Anruf beenden (UC08)

Ein aktiver Anruf wird vom PC her getrennt.

Anruf wird beendet (UC09)

Ein aktiver Anruf wird vom anderen Gesprächsteilnehmer getrennt.

2.3.3 Telefoneinstellungen verändern

Freisprechen ein- oder ausschalten (UC10)

Die Freisprechanlage wird vom PC her ein- oder ausgeschaltet.

Lautstärke verändern (UC11)

Die Gesprächslautstärke wird vom PC her verändert.

Mikrofon ein- oder ausschalten (UC12)

Das Mikrofon wird vom PC her ein- oder ausgeschaltet.

Klingeltonlautstärke verändern (UC13)

Die Klingeltonlautstärke wird vom PC her verändert.

2.3.4 SMS

SMS versenden (UC14)

Eine SMS wird vom PC her versendet.

SMS erhalten (UC15)

Eine SMS wird vom PC her empfangen.

SMS an mehrere Empfänger versenden (UC16)

Es wird eine SMS zum gleichen Zeitpunkt an unterschiedliche Empfänger versendet.

2.3.5 Datensynchronisation

Annahme: Es macht keinen Unterschied, ob Daten auf dem Mobile-Device oder dem PC bearbeitet werden. Das AnCoF geht davon aus, dass die Daten in der Form, wie sie auf dem Mobile-Device gespeichert werden, vorliegen. Folgende Punkte können konfiguriert werden:

- Rekursion: Wird rekursiv synchronisiert oder nicht?
- Synchronisationsmodus: Wird rückgefragt, bevor eine Änderung an den bestehenden Dateien vorgenommen wird oder der Zeitstempel verglichen und automatisch geändert?
- Flag Winner: Welche Dateien werden bei einer Kollision übernommen?

Neue Daten synchronisieren (UC17)

Neue Daten wurden auf dem Mobile-Device/PC erstellt und werden auf dem anderen Gerät hinzugefügt.

Bearbeitete Daten synchronisieren(UC18)

Daten wurden auf dem Mobile-Device/PC verändert und werden auf dem anderen Gerät aktualisiert.

Gelöschte Daten synchronisieren (UC19)

Daten wurden auf dem Mobile-Device/PC gelöscht und werden auf dem anderen Gerät ebenfalls entfernt.

2.3.6 Kontakte

Neuer Kontakt synchronisieren (UC20)

Ein neuer Kontakt wurde auf dem Mobile-Device/PC erstellt und wird auf dem anderen Gerät hinzugefügt.

Veränderter Kontakt synchronisieren (UC21)

Ein Kontakt wurde auf dem Mobile-Device/PC bearbeitet und wird auf dem anderen Gerät aktualisiert.

Gelöschter Kontakt synchronisieren (UC22)

Ein Kontakt wurde auf dem Mobile-Device/PC gelöscht und wird auf dem anderen Gerät ebenfalls entfernt.

2.4 Interfaces

Das Diagramm 2.7 zeigt die Schnittstellen, über welche eine PC-Applikation, welche auf dem Framework AnCoF aufsetzt, mit dem Mobile-Device kommunizieren kann.

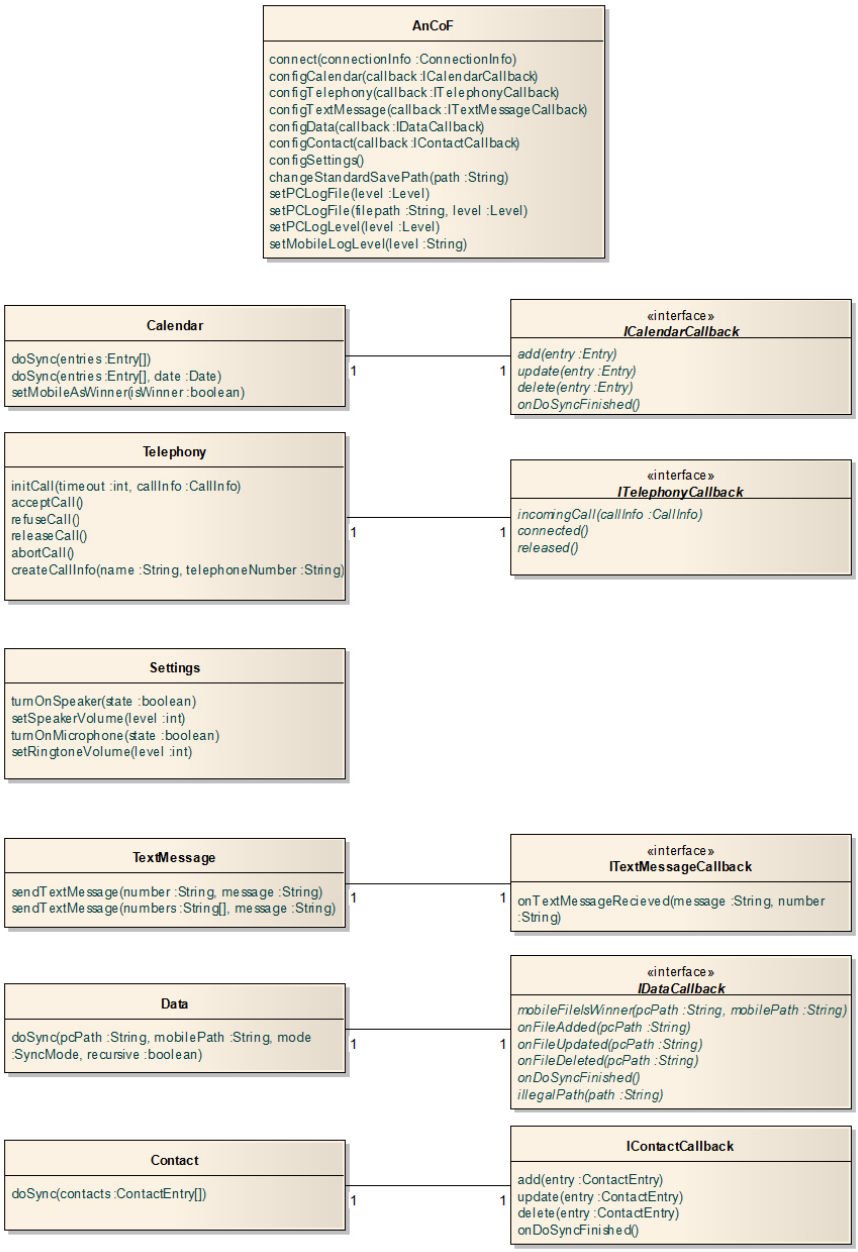


Abbildung 2.7: Interface Diagramm

2.4.1 AnCoF

connect(connectionInfo)

Stellt eine Verbindung mit einem Mobile-Device her.

config...(callback)

Erzeugt eine neue Instanz des jeweiligen Systems mit der im Callback angegebenen Einstellungen und gibt die Instanz zurück.

changeStandardSavePath(path)

Ersetzt den Standardpfad für die gespeicherten Applikationsdaten durch den angegebenen Pfad. Wird der Pfad nicht verändert, wird im ApplicationData-Ordner von Windows gespeichert.

setPCLogFile(filepath, level)

Bestimmt, welche Levels das Logfile enthält und wo es gespeichert wird. Der Parameter filepath ist optional, ist er nicht gesetzt, wird ein Standardpfad verwendet.

setPCLogLevel(level)

Bestimmt den Loglevel auf der PC-Seite.

setMobileLogLevel(level)

Bestimmt den Loglevel auf der Mobile-Device-Seite.

2.4.2 Calendar

doSync(entries, date)

Ist Teil der UC01-03. Diese Funktion vergleicht die Einträge auf dem Mobile-Device mit jenen der PC-Applikation. Der Parameter date ist optional und bestimmt, von welchem Datum an synchronisiert wird. Ist er nicht gesetzt, wird ein Standardwert verwendet.

setMobileAsWinner(isWinner)

Bestimmmt, welche Kalendereinträge bei Synchronisationskonflikten bevorzugt werden.

2.4.3 ICalendarCallback

Wird von der PC-Applikation implementiert, um Kalenderfunktionalitäten zu ermöglichen.

add(entry)

Ist Teil des UC01. Bei der Synchronisation wurde ein Eintrag gefunden, welcher auf der PC-Applikation noch nicht existiert und dessen Erstelldatum neuer als jenes der letzten Synchronisation ist. Der Eintrag muss daher hinzugefügt werden.

update(entry)

Ist Teil des UC02. Bei der Synchronisation wurde ein Eintrag gefunden, welcher auf der PC-Applikation existiert, jedoch auf dem Mobile-Device neuer ist. Die fehlenden respektive geänderten Informationen müssen daher auf den Stand des Mobile-Devices aktualisiert werden.

delete(entry)

Ist Teil des UC03. Bei der Synchronisation wurde ein Eintrag gefunden, welcher auf dem Mobile-Device nicht existiert und dessen Erstelldatum älter als jenes der letzten Synchronisation ist. Der Eintrag muss daher gelöscht werden.

onDoSyncFinished()

Wird von AnCoF aufgerufen sobald der Synchronisationsvorgang abgeschlossen ist.

2.4.4 Telephony

initCall(callInfo, timeout)

Ist Teil des UC04. Eine Verbindung soll aufgebaut werden. Details zur Verbindung (z.B. Telefonnummer) werden mit CallInfo mitgegeben. Der timeout definiert, wann ein erfolgloser Verbindungsversuch abgebrochen werden soll.

acceptCall()

Ist Teil von UC05 und UC06. Der eingehende Anruf wird von der PC-Applikation her entgegengenommen.

refuseCall()

Ist Teil des UC07. Der eingehende Anruf wird von der PC-Applikation her abgelehnt.

releaseCall()

Ist Teil des UC08. Der aktive Anruf wird von der PC-Applikation her beendet.

abortCall()

abort() wird verwendet, um eine Verbindungsanfrage, die mit initCall() angestossen wurde, jedoch von der Gegenseite noch nicht akzeptiert wurde, zu verwerfen.

createCallInfo(name, telephoneNumber)

Wird verwendet, um eine neue CallInfo-Instanz zu erzeugen.

2.4.5 ITelephonyCallback

Wird von der PC-Applikation implementiert. Es spezifiziert, wie mit Anruf-Informationen umgegangen wird.

incomingCall(callInfo)

Ist Teil von UC05 bis UC07. Wird von AnCoF aufgerufen, wenn ein eingehender Anruf vorliegt.

connected()

Ist Teil von UC04 bis UC06. Wird von AnCoF aufgerufen, sobald eine Verbindung zu einem anderen Gesprächsteilnehmer besteht.

released()

Ist Teil von UC08 und UC09. Wird von AnCoF aufgerufen, sobald eine Verbindung vollständig abgebaut ist.

2.4.6 Settings

turnOnSpeaker(state)

Ist Teil des UC10. Diese Funktion ermöglicht es, die Freisprechanlage ein- oder auszuschalten.

setSpeakerVolume(level)

Ist Teil des UC11. Diese Funktion ermöglicht es, die Gesprächslautstärke zu regulieren.

turnOnMicrophone(state)

Ist Teil des UC12. Diese Funktion ermöglicht es, das Mikrofon ein- oder auszuschalten.

setRingtoneVolume(level)

Ist Teil des UC13. Diese Funktion ermöglicht es, die Klingeltonlautstärke zu regulieren.

2.4.7 TextMessage

sendTextMessage(message, number)

Ist Teil von UC14 und UC16. Die Funktion wird zum versenden von neuen Textnachrichten an eine oder mehrere Telefonnummern gebraucht.

2.4.8 ITextMessageCallback

Wird von der PC-Applikation implementiert.

onTextMessageRecieved(message, number)

Ist Teil des UC15. Die Funktion wird von AnCoF aufgerufen, wenn eine neue Textnachricht vorliegt.

2.4.9 Data

doSync(pcPath, mobilePath, syncMode, recursive)

Ist Teil der UC17 bis UC19. Die Funktion vergleicht die Einträge des Verzeichnisses, welches mit pcPath angegeben wurde, mit jenen vom mobilePath. Der syncMode gibt dabei an, ob nach Zeitstempel oder mit Rückfrage verglichen werden soll. Mit recursive wird zusätzlich bestimmt, ob nur Dateien des angegebenen Ordners verglichen werden oder ob alle Dateien in den Unterordnern einbezogen werden.

2.4.10 IDataCallback

Wird von der PC-Applikation implementiert.

mobileFileIsWinner(pcPath, mobilePath)

Wird von AnCoF aufgerufen, um zu ermitteln, wie mit dem unter mobilePath angegebenen, veränderten Datei verfahren werden soll.

onFileAdded(pcPath)

Wird von AnCoF aufgerufen, wenn eine Datei auf dem PC hinzugefügt wurde.

onFileUpdated(pcPath)

Wird von AnCoF aufgerufen, wenn eine Datei auf dem PC aktualisiert wurde.

onFileDeleted(pcPath)

Wird von AnCoF aufgerufen, wenn eine Datei auf dem PC gelöscht wurde.

onDoSyncFinished()

Wird von AnCoF aufgerufen, sobald der Synchronisationsvorgang abgeschlossen ist.

illegalPath(path)

Wird von AnCoF aufgerufen, wenn ein angegebener Pfad nicht vorhanden ist.

2.4.11 Contact

doSync(contacts)

Ist Teil der UC20 bis UC22. Die Funktion vergleicht die Kontakteinträge auf dem Mobile-Device mit jenen in der contacts-Liste.

2.4.12 IContactCallback

Wird von der PC-Applikation implementiert.

add(entry)

Ist Teil des UC20. Bei der Synchronisation wurde ein Kontakt gefunden, welcher auf der PC-Applikation noch nicht existiert und dessen Erstelldatum neuer als jenes der letzten Synchronisation ist. Der Kontakt muss daher hinzugefügt werden.

update(entry)

Ist Teil des UC21. Bei der Synchronisation wurde ein Kontakt gefunden, welcher auf der PC-Applikation existiert, jedoch auf dem Mobile-Device neuer ist. Die fehlenden respektive geänderten Informationen müssen daher auf den Stand des Mobile-Devices aktualisiert werden.

delete(entry)

Ist Teil des UC22. Bei der Synchronisation wurde ein Kontakt gefunden, welcher auf dem Mobile-Device nicht existiert und dessen Erstelldatum älter als jenes der letzten Synchronisation ist. Der Kontakt muss daher gelöscht werden.

onDoSyncFinished()

Wird von AnCoF aufgerufen sobald der Synchronisationsvorgang abgeschlossen ist.

3 Nichtfunktionale Anforderungen

3.1 Funktionalität

Das AnCoF dient als Vermittler zwischen zwei Applikationen (Mobile-Device und PC). Es hilft Daten zu synchronisieren bzw. unterstützt gängige Funktionen von PC-Suiten.

Das Verbinden von mehreren Mobile-Devices ist möglich, mit der Einschränkung, dass zu einem Zeitpunkt immer nur ein Mobile-Device mit dem PC verbunden ist. Auch der umgekehrte Fall, ein Mobile-Device wird mit mehreren PCs synchronisiert, wird unterstützt. Hier gilt die gleiche Einschränkung, immer nur ein PC ist zu einem Zeitpunkt verbunden. Um Verwirrungen auf Seiten des Benutzers zu vermeiden, kann ein Mobile-Device bzw. PC zu jeder Zeit mit seinem Namen identifiziert werden.

3.2 Zuverlässigkeit

Nach einer Unterbrechung der Synchronisation, z.B. bei Verbindungsverlust durch Trennen der USB-Verbindung, darf kein inkonsistenter Zustand entstehen. Bei erneuter Verbindung müssen die Daten weiter respektive neu synchronisiert werden können.

Auch bei Telefongesprächen muss bei einem plötzlichen Unterbruch die Verbindung korrekt beendet werden, sodass bei einem erneuten Anruf eine neue Telefonverbindung aufgebaut werden kann.

3.3 Erlernbarkeit

Das Framework wird so aufgebaut, dass ein Java-Programmierer dieses mit Hilfe des Dokuments «Developer Guide» innerhalb eines Tages erlernen und danach verwenden kann.

3.4 Wartbarkeit

Das AnCoF muss zu jedem Zeitpunkt erweiterbar bzw. veränderbar sein, damit ein Entwickler jederzeit Anpassungen an eine neue Version des Android-Systems vornehmen oder Funktionen, welche hier weggelassen wurden, ergänzen kann.

3.5 Effizienz

Das System muss Daten in O(n) synchronisieren können.

3.6 Unterstützte Systeme

Das AnCoF unterstützt Android-Betriebssysteme ab Version 1.6. AnCoF ist getestet bis Version 2.1.

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	29.09.2010	dm	Dokument erstellt.
1.0rc02	29.09.2010	dm	Kapitel Funktionale Anforderungen bearbeitet.
1.0rc03	29.09.2010	rr	Kapitel Nichtfunktionale Anforderungen bearbeitet.
1.0rc04	29.09.2010	dm	Review Kapitel NF Anforderungen.
1.0rc05	03.10.2010	dm	Änderungen und Ergänzungen gemäss Besprechung.
1.0rc06	05.10.2010	rr	UseCase Diagramm hinzugefügt.
1.0rc07	08.10.2010	dm	Interface Diagramm hinzugefügt und Kalenderoperationen beschrieben.
1.0rc08	10.10.2010	rr	Telefonoperationene beschrieben.
1.0rc09	13.10.2010	dm	Review, diverse Korrekturen.
1.0	15.10.2010	dm	Akzeptierte Version 1.0
1.1rc01	15.10.2010	dm	Interface-Diagramm angepasst
1.1rc02	10.11.2010	rr	Fehlende Usecases ergänzt, Interface erweitert.
1.1rc03	10.11.2010	dm	Review, diverse Korrekturen.
1.1rc04	17.11.2010	dm	Erweiterung Interface-Diagramm.
1.1	20.11.2010	dm	Akzeptierte Version 1.1
1.2rc01	20.11.2010	dm	UC und Interface Datensynchronisation angepasst.
1.2rc02	08.12.2010	dm, rr	Interface angepasst.
1.2	11.12.2010	dm	Akzeptierte Version 1.2
1.3rc01	19.12.2010	rr	Einarbeitung Änderungen an Rechtschreibung und Layout.
1.3	20.12.2010	dm	Akzeptierte Version 1.3

Tabelle 3.1: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
PC Personal Computer
SA Studienarbeit
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone
UC Use Case

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Domainanalyse
«Android Control Framework»

Version 1.3

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Siehe «Projektplan», Kapitel 4.2.2, Seite 6.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 Domainmodel

2.1 Strukturdiagramm

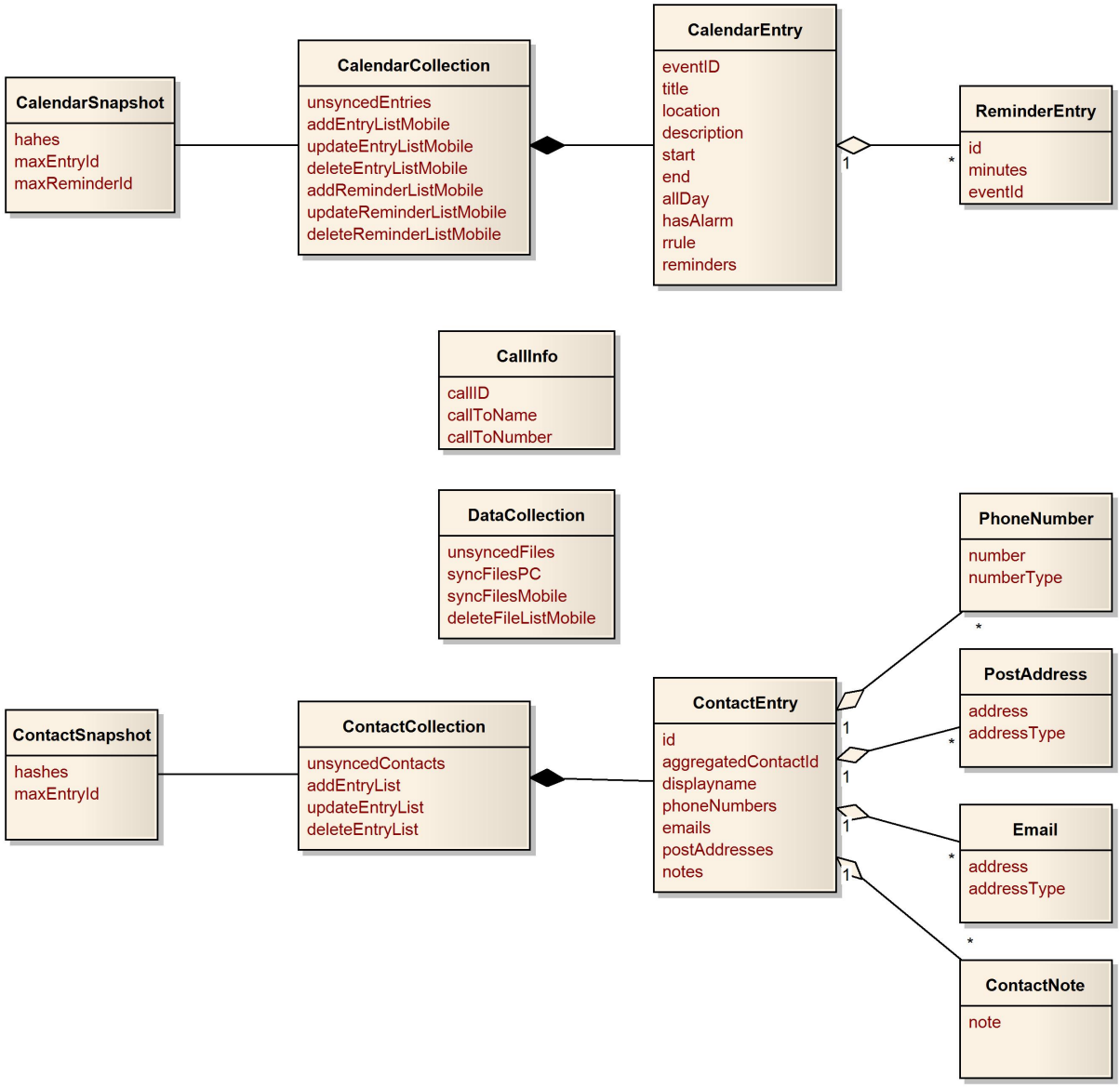


Abbildung 2.1: Domain Model

2.2 Konzeptbeschreibung

CalendarCollection

Das Objekt CalendarCollection enthält die einzelnen Kalender und ihre Einträge.

Attribute	unsyncedEntries	Liste noch nicht synchronisierter Einträge.
	addEntryListMobile	Einträge, die auf dem Mobile-Device hinzugefügt werden müssen.
	updateEntryListMobile	Einträge, die auf dem Mobile-Device aktualisiert werden müssen.
	deleteEntryListMobile	Einträge, die auf dem Mobile-Device gelöscht werden müssen.
	addReminderListMobile	Erinnerungen, die auf dem Mobile-Device hinzugefügt werden müssen.
	updateReminderListMobile	Erinnerungen, die auf dem Mobile-Device aktualisiert werden müssen.
	deletReminderListMobile	Erinnerungen, die auf dem Mobile-Device gelöscht werden müssen.
Beziehungen	Ist eine Art «Behälter» für Kalender.	

Tabelle 2.1: Attribute CalendarCollection

CalendarEntry

Das Objekt CalendarEntry enthält die spezifischen Informationen eines Kalendereintrags.

Attribute	eventID	Eindeutige Id zur Identifizierung des Eintrags.
	title	Überschrift des Eintrags.
	location	Ortsangabe des Eintrags.
	description	Beschreibung des Eintrags.
	start	Startzeit des Eintrags.
	end	Endzeit des Eintrags.
	allDay	Dauert der Eintrags den ganzen Tag?
	hasAlarm	Sind Erinnerungseinträge vorhanden?
	rrule	Informationen über regelmässige Wiederholungen des Eintrags.
	reminders	Liste von Erinnerungseinträgen.
Beziehungen	Ist Teil einer CalendarCollection.	

Tabelle 2.2: Attribute CalendarEntry

ReminderEntry

Das Objekt ReminderEntry enthält Informationen der Erinnerungseinstellungen eines Kalendereintrags.

Attribute	id	Eindeutige Id zur Identifizierung der Erinnerung.
	minutes	Wie lange vor dem Termin die Erinnerung aktiv werden soll.
	eventId	Identifiziert den Termin, zu dem die Erinnerung gehört.
Beziehungen	Ist Teil einer CalendarEntry.	

Tabelle 2.3: Attribute ReminderEntry

CalendarSnapshot

Das Objekt CalendarSnapshot enthält die gespeicherten Hash-Werte der einzelnen Einträge.

Attribute	hashes	Hash-Werte der einzelnen Einträge, sortiert nach Kalender.
	maxEntryId	Maximale Id der Kalendereinträge des zuletzt gespeicherten Zustandes.
	maxReminderId	Maximale Id der Erinnerungseinträge des zuletzt gespeicherten Zustandes.
Beziehungen	Wird von der CalendarCollection benutzt.	

Tabelle 2.4: Attribute CalendarSnapshot

CallInfo

Das Objekt CallInfo enthält alle Details eines Anrufs und dient dazu, die einzelnen Anrufe zu identifizieren.

Attribute	callID	Eindeutige Id zur Identifizierung des Anrufs. Wird automatisch zugewiesen.
	callToName	Name des anderen Gesprächsteilnehmers.
	callToNumber	Telefonnummer des anderen Gesprächsteilnehmers.

Tabelle 2.5: Attribute CallInfo

DataCollection

Das Objekt DataCollection enthält die einzelnen Pfade der zu synchronisierenden Dateien.

Attribute	unsyncedFiles	Liste noch nicht synchronisierter Dateien.
	syncFilesPC	Dateien, die auf dem PC hinzugefügt, respektive aktualisiert werden müssen.
	syncFilesMobile	Dateien, die auf dem Mobile-Device hinzugefügt, respektive aktualisiert werden müssen.
	deleteFileListMobile	Dateien, die auf dem Mobile-Device gelöscht werden müssen.
Beziehungen	Ist eine Art «Behälter» für Dateipfade.	

.

Tabelle 2.6: Attribute DataCollection

ContactCollection

Das Objekt ContactCollection enthält die einzelnen Kontakte und ihre Einträge.

Attribute	unsyncedContacts	Liste noch nicht synchronisierter Kontakte.
	addEntryList	Kontakte, die auf dem Mobile-Device hinzugefügt werden müssen.
	updateEntryList	Kontakte, die auf dem Mobile-Device aktualisiert werden müssen.
	deleteEntryList	Kontakte, die auf dem Mobile-Device gelöscht werden müssen.
Beziehungen	Ist eine Art «Behälter» für Kontakte.	

Tabelle 2.7: Attribute ContactCollection

ContactEntry

Das Objekt ContactEntry enthält die spezifischen Informationen eines Kontakteintrags.

Attribute	id	Eindeutige Id zur Identifizierung des Kontakts.
	aggregatedContactId	Id des zugehörigen Kontakts.
	displayname	Name.
	phoneNumbers	Telefonnummern.
	emails	Mail-Adressen.
	postAdresses	Post-Anschriften.
	notes	Notizen.
Beziehungen	Ist Teil einer ContactCollection.	

Tabelle 2.8: Attribute ContactEntry

PhoneNumber

Das Objekt PhoneNumber enthält die spezifischen Informationen einer Telefonnummer.

Attribute	number	Telefonnummer.
	numberType	Kennzeichnung, z.B. «private».
Beziehungen	Ist Teil eines ContactEntry-Objekts.	

Tabelle 2.9: Attribute PhoneNumber

PostAddress

Das Objekt PostAddress enthält die spezifischen Informationen einer Post-Anschrift.

Attribute	address	Anschrift.
	addressType	Kennzeichnung, z.B. «private».
Beziehungen	Ist Teil eines ContactEntry-Objekts.	

Tabelle 2.10: Attribute PostAddress

Email

Das Objekt Email enthält die spezifischen Informationen einer Mail-Adresse.

Attribute	address	Mail-Adresse.
	addressType	Kennzeichnung, z.B. «private».
Beziehungen	Ist Teil eines ContactEntry-Objekts.	

Tabelle 2.11: Attribute Email

ContactNote

Das Objekt ContactNote enthält die spezifischen Informationen einer Notiz.

Attribute	note	Notiztext.
Beziehungen	Ist Teil eines ContactEntry-Objekts.	

Tabelle 2.12: Attribute ContactNote

ContactSnapshot

Das Objekt ContactSnapshot enthält die gespeicherten Hash-Werte der einzelnen Einträge.

Attribute	hashes	Hash-Werte der einzelnen Einträge.
	maxEntryId	Maximale Id der Kontakte des zuletzt gespeicherten Zustandes.
Beziehungen	Wird von der ContactCollection benutzt.	

Tabelle 2.13: Attribute ContactSnapshot

3 SSD

Zum besseren Verständnis werden Interface (z.B. IMobileCallback) und Framework-Zugriff (z.B. Mobile) getrennt aufgeführt. Die Blackbox-Darstellung nach [1] wird dabei nicht verletzt, da nur die Benutzerschnittstellen aufgezeigt werden.

3.1 Kalendersynchronisation

3.1.1 UC01 Neuer Eintrag synchronisieren

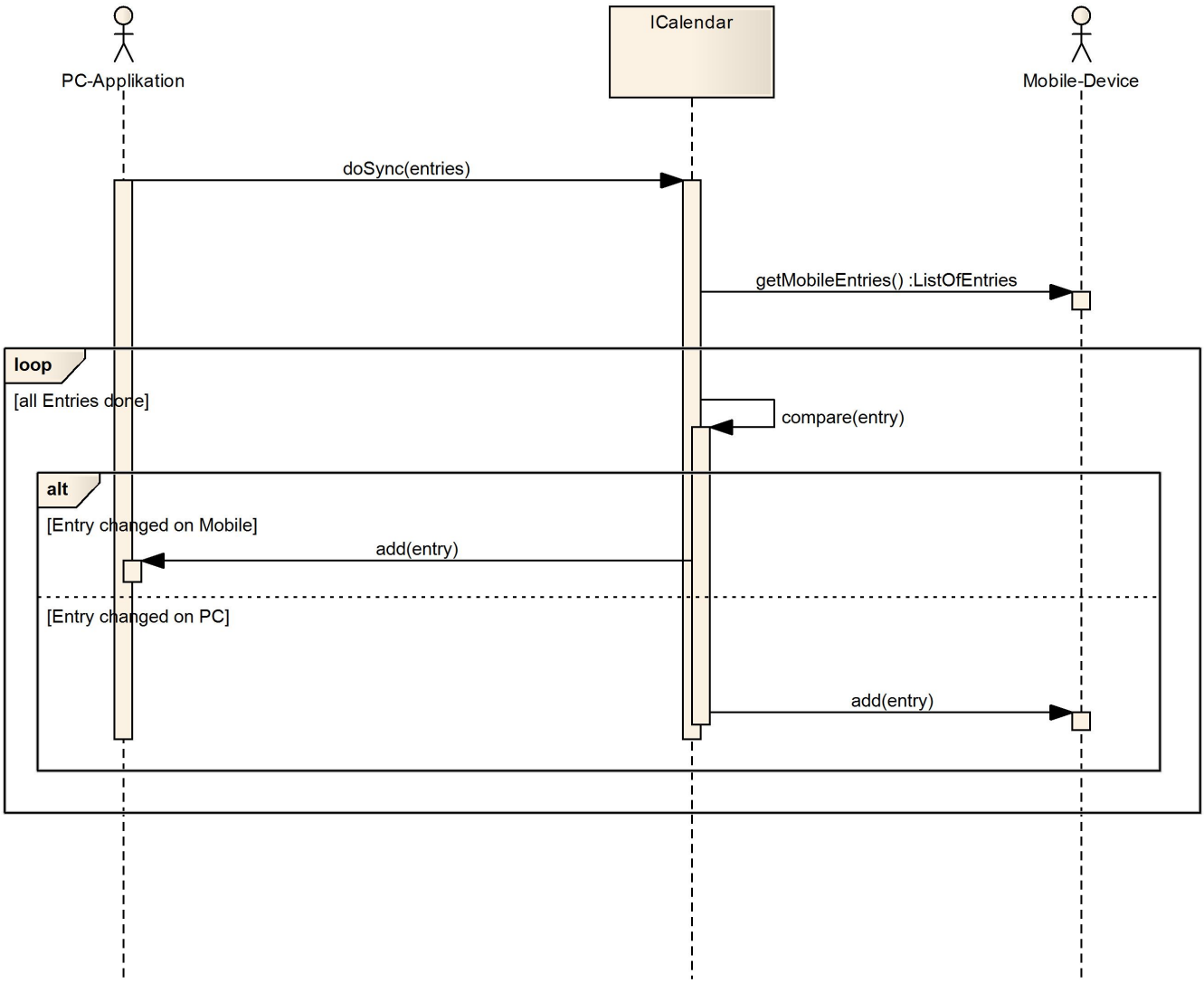


Abbildung 3.1: Neuer Eintrag synchronisieren

3.1.2 UC02 Veränderter Eintrag synchronisieren

Der Ablauf ist der selbe wie in UC01. Für Details siehe Abbildung 3.1, Seite 8.

3.1.3 UC03 Gelöschter Eintrag synchronisieren

Der Ablauf ist der selbe wie in UC01. Für Details siehe Abbildung 3.1, Seite 8.

3.2 Telefonieren

Der Befehl `configTelephony(mobileCallback)` wird in den UC04-07 zur Verständlichkeit jedes Mal aufgeführt. Normalerweise wird die Funktion nur zu Beginn einmal aufgerufen, um einen «Callback» zu ermöglichen.

3.2.1 UC04 Anruf tätigen

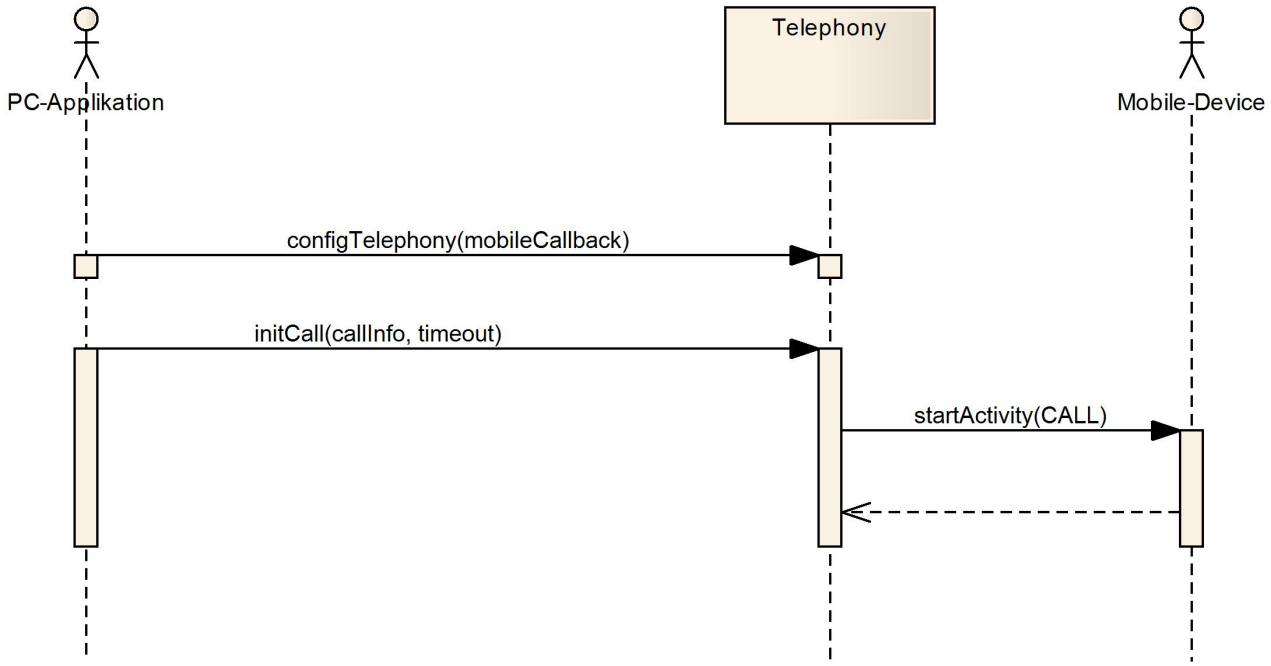


Abbildung 3.2: Anruf tätigen

3.2.2 UC05 Anruf erhalten

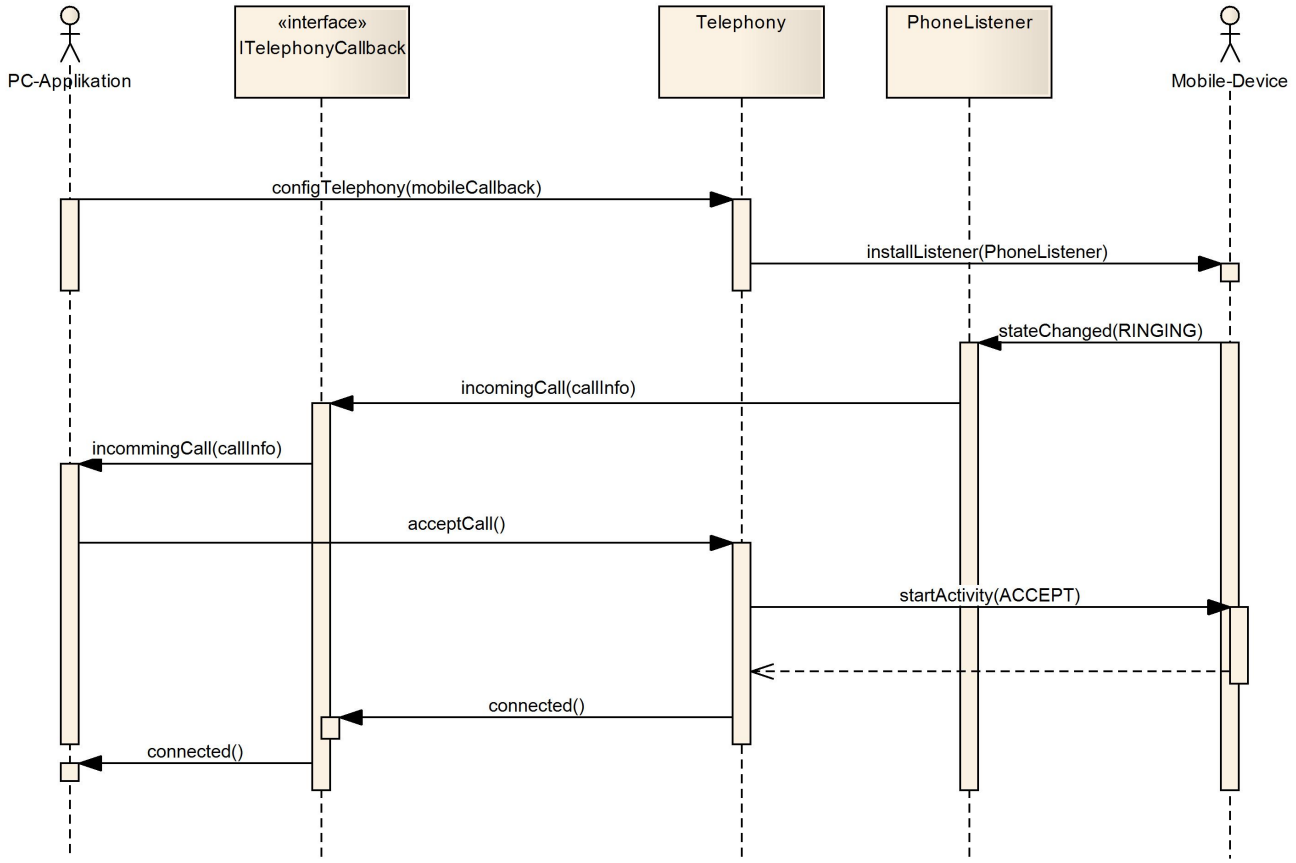


Abbildung 3.3: Anruf erhalten

3.2.3 UC06 Anruf während Verbindung erhalten

Der Ablauf ist der selbe wie in UC05. Für Details siehe Abbildung 3.3.

3.2.4 UC07 Anruf ablehnen

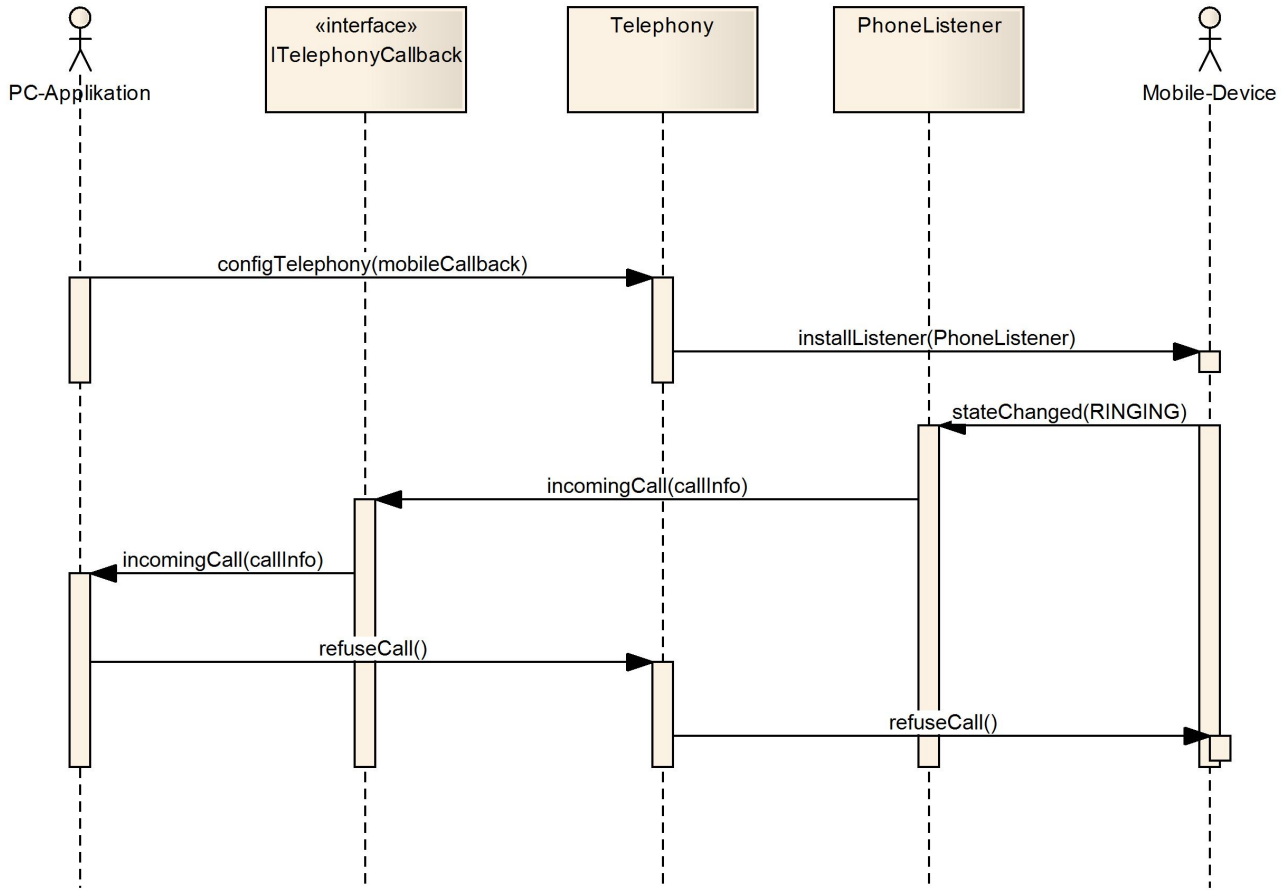


Abbildung 3.4: Anruf ablehnen

3.2.5 UC08 Anruf beenden

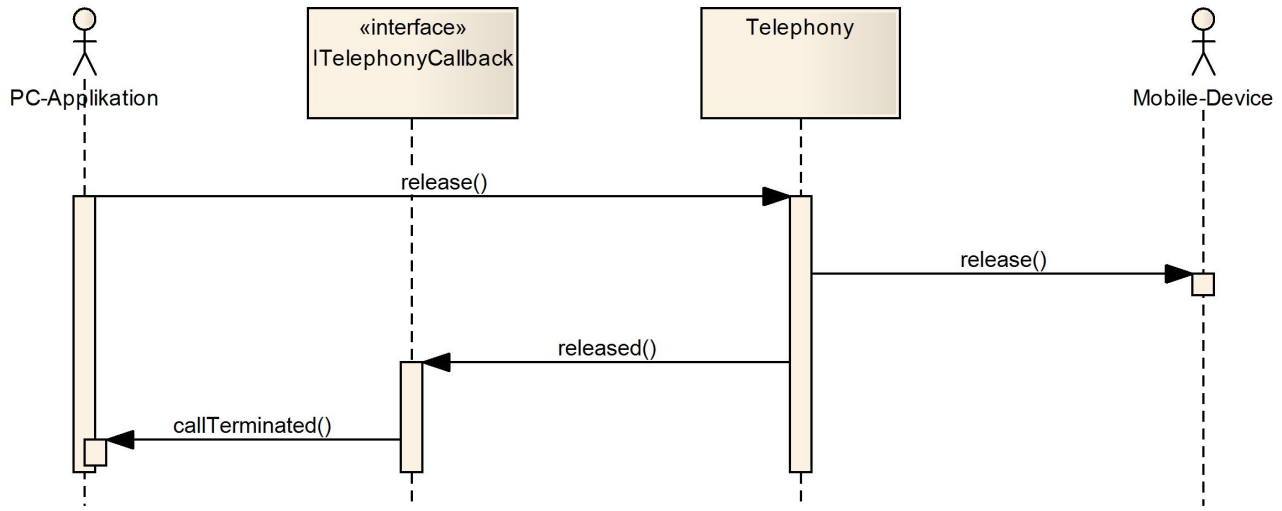


Abbildung 3.5: Anruf beenden

3.2.6 UC09 Anruf wird beendet

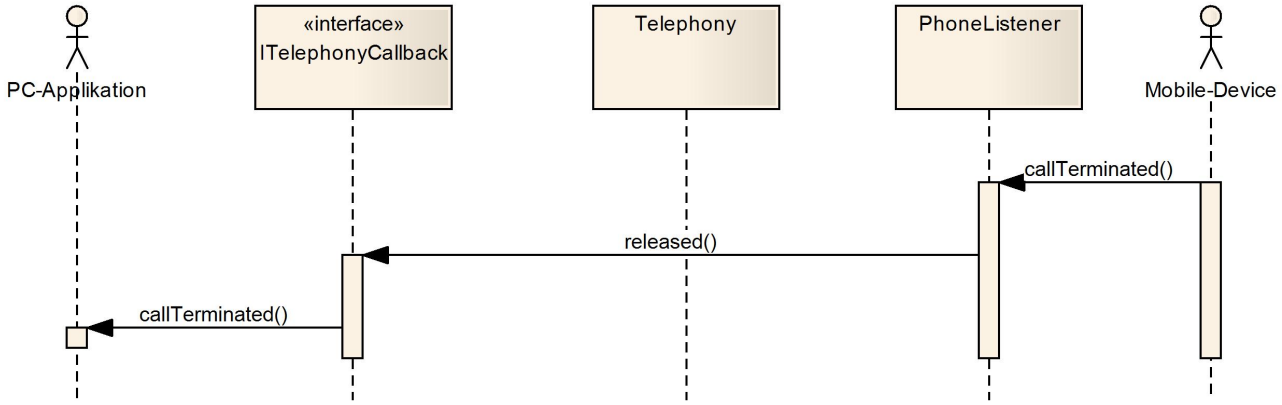


Abbildung 3.6: Anruf wird beendet

3.3 Telefoneinstellungen verändern

3.3.1 UC10 Freisprechen ein- oder ausschalten

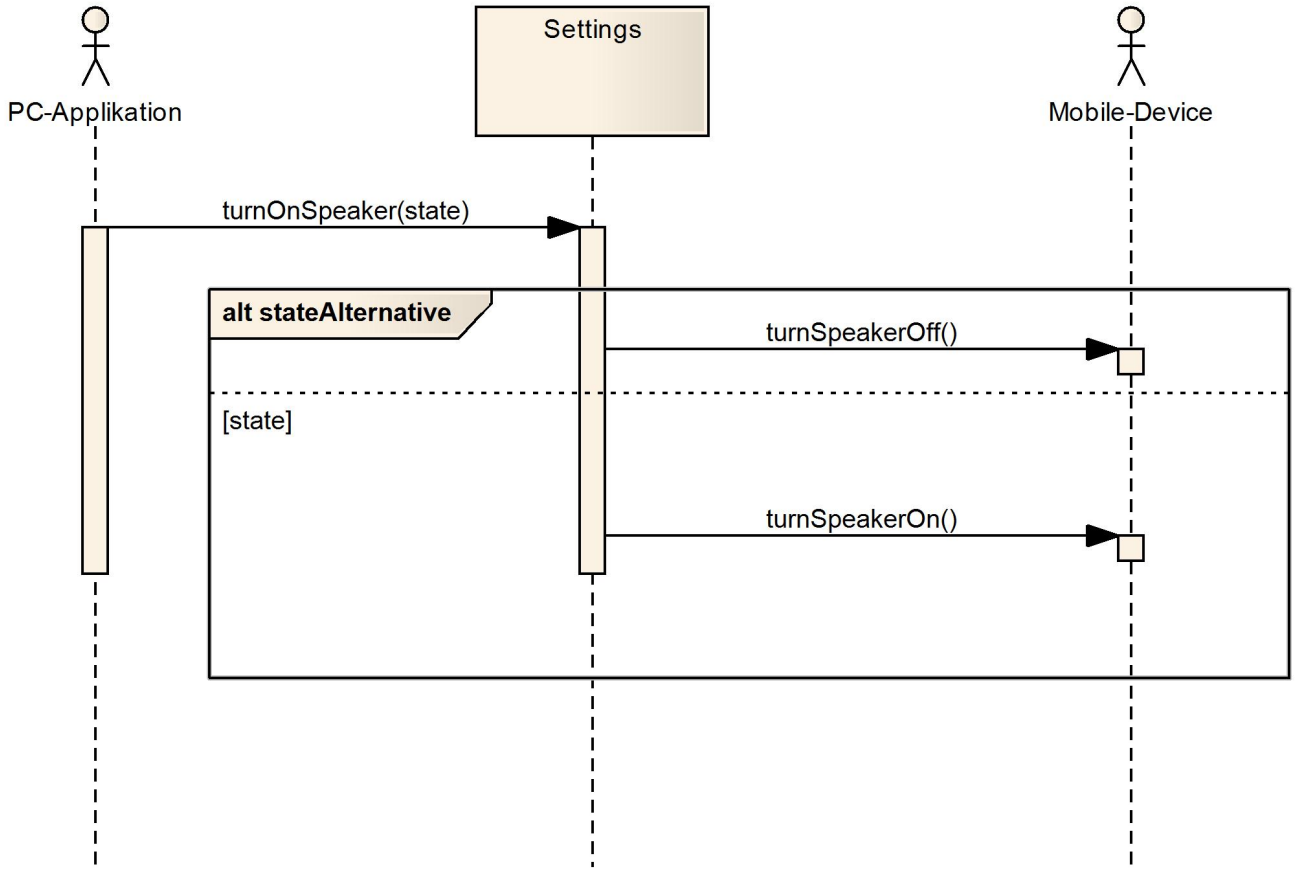


Abbildung 3.7: Lautsprecher wird ein- oder ausgeschaltet

3.3.2 UC11 Lautstärke verändern

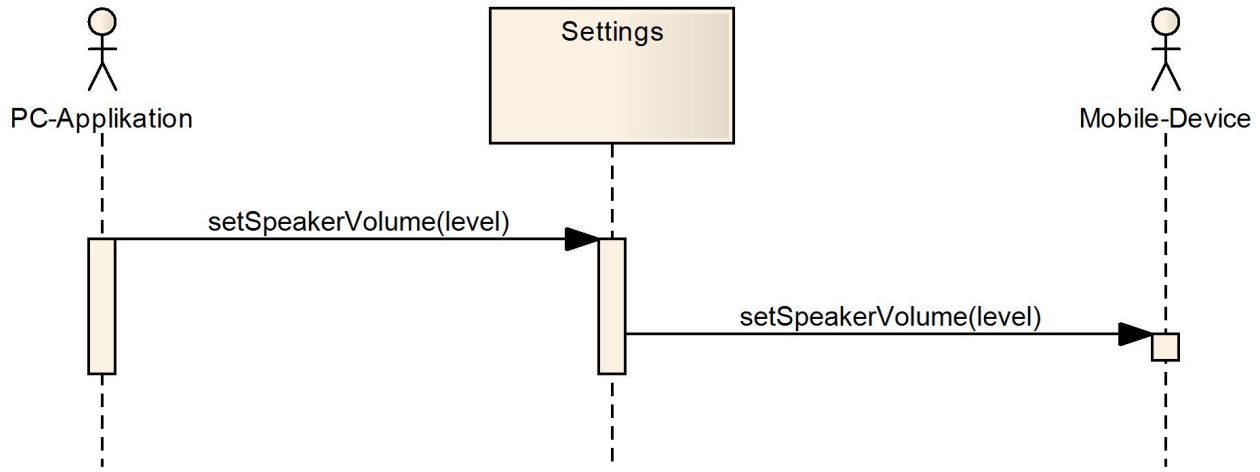


Abbildung 3.8: Lautstärke wird verändert

3.3.3 UC12 Mikrofon ein- oder ausschalten

Der Ablauf ist der selbe wie in UC10. Für Details siehe Abbildung 3.7, Seite 13.

3.3.4 UC13 Klingeltonlautstärke verändern

Der Ablauf ist der selbe wie in UC11. Für Details siehe Abbildung 3.8.

3.4 SMS

3.4.1 UC14 SMS versenden

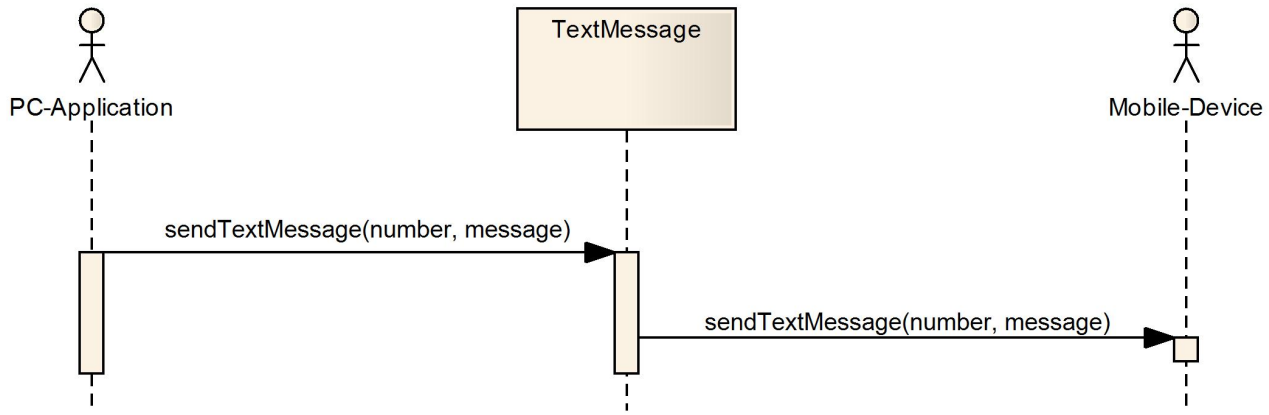


Abbildung 3.9: SMS wird versendet

3.4.2 UC15 SMS erhalten

Es wird angenommen, dass der Benutzer bereits das Callback-Objekt gesetzt hat.

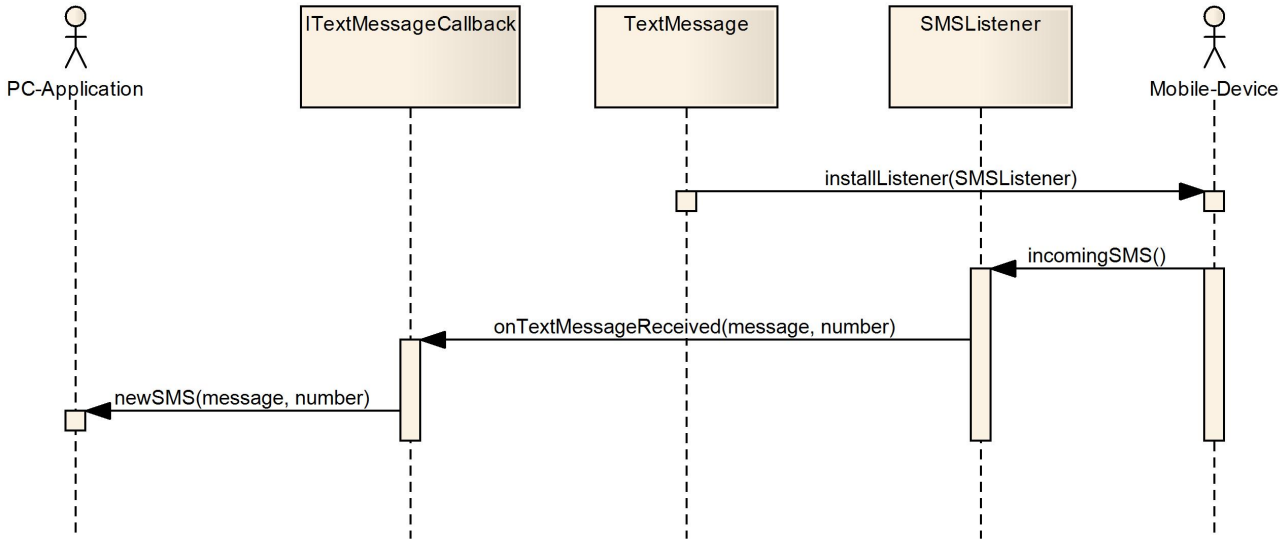


Abbildung 3.10: SMS wird empfangen

3.4.3 UC16 SMS an mehrere Empfänger versenden

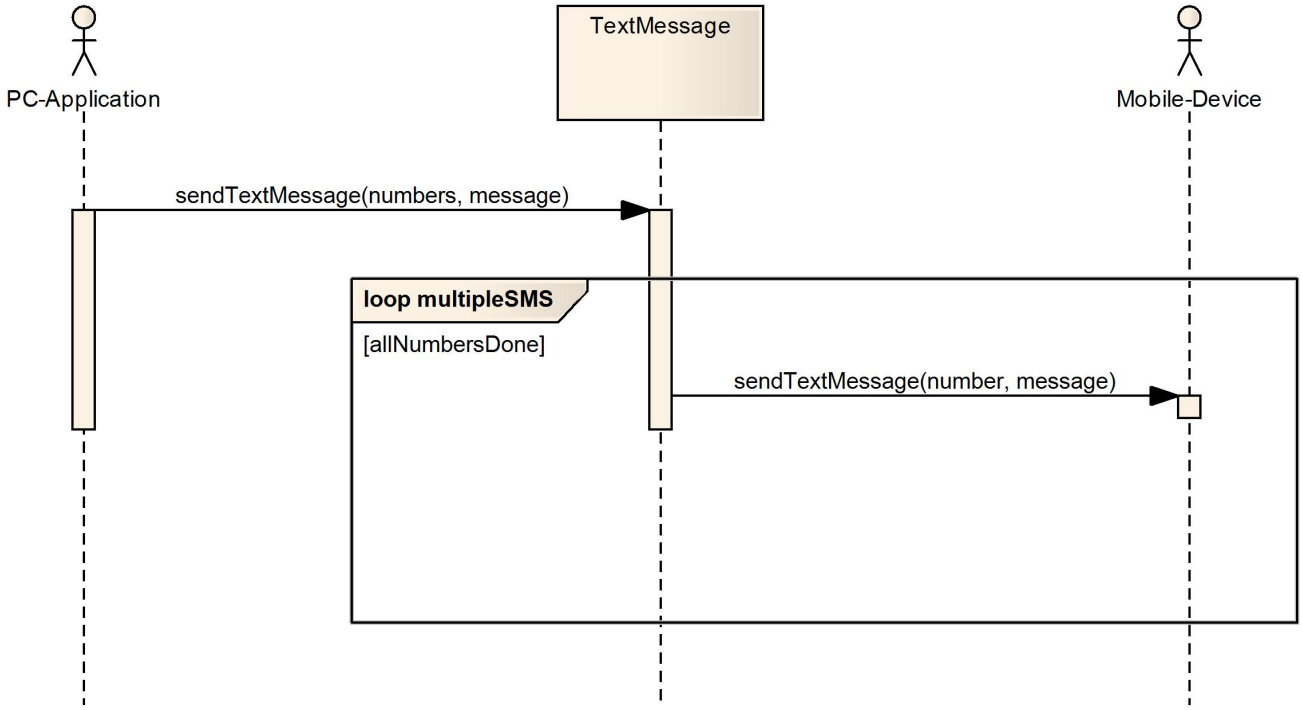


Abbildung 3.11: SMS wird an mehrere Empfänger versendet

3.5 Datensynchronisation

3.5.1 UC17 Neue Daten synchronisieren

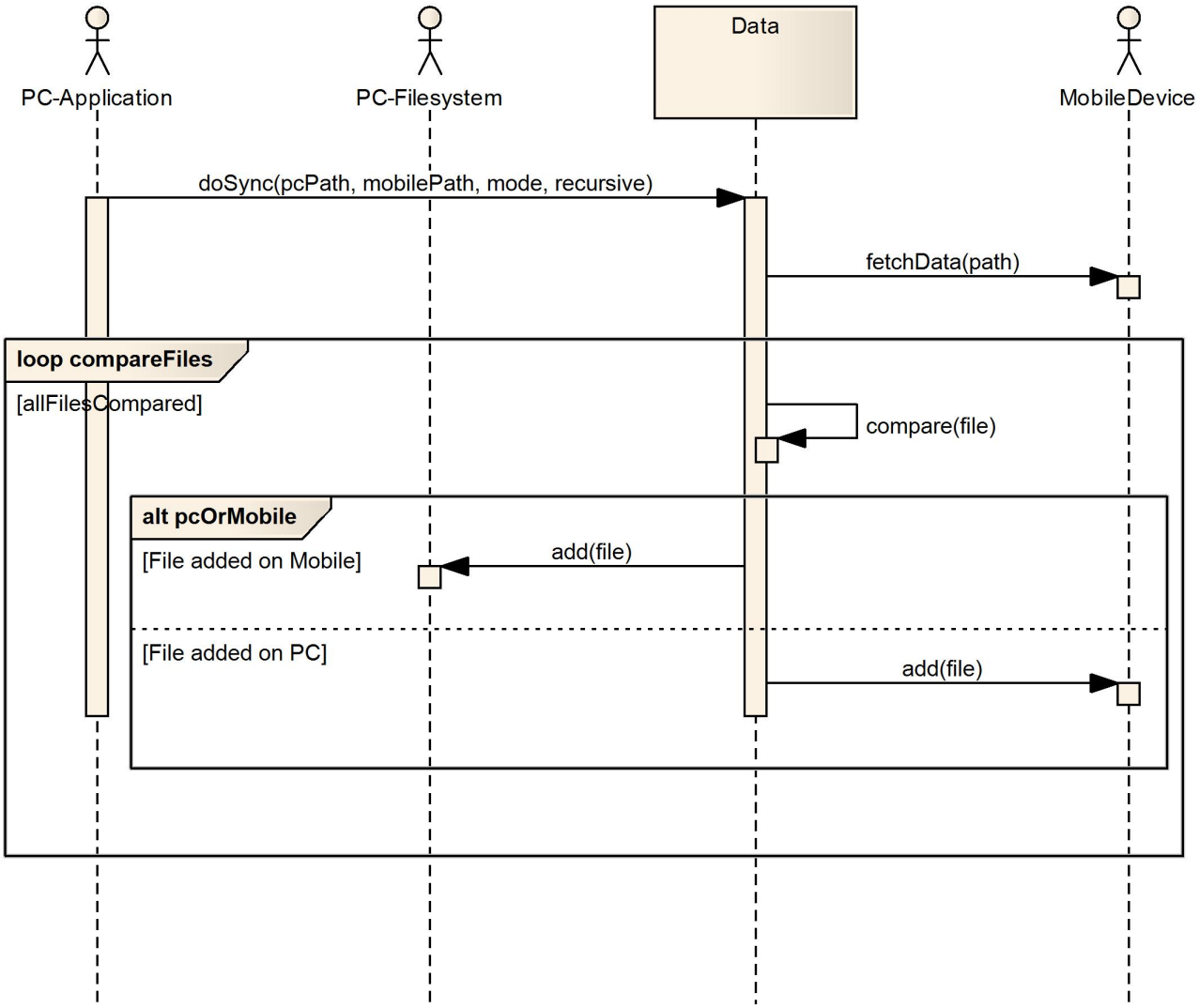


Abbildung 3.12: Neue Daten werden synchronisiert

3.5.2 UC18 Bearbeitete Daten synchronisieren

In der Abbildung 3.13 wird nur die Version im Rückfragemodus gezeigt. Die Version im Zeitstempelmodus ist analog zu UC17, Details können der Abbildung 3.12 entnommen werden.

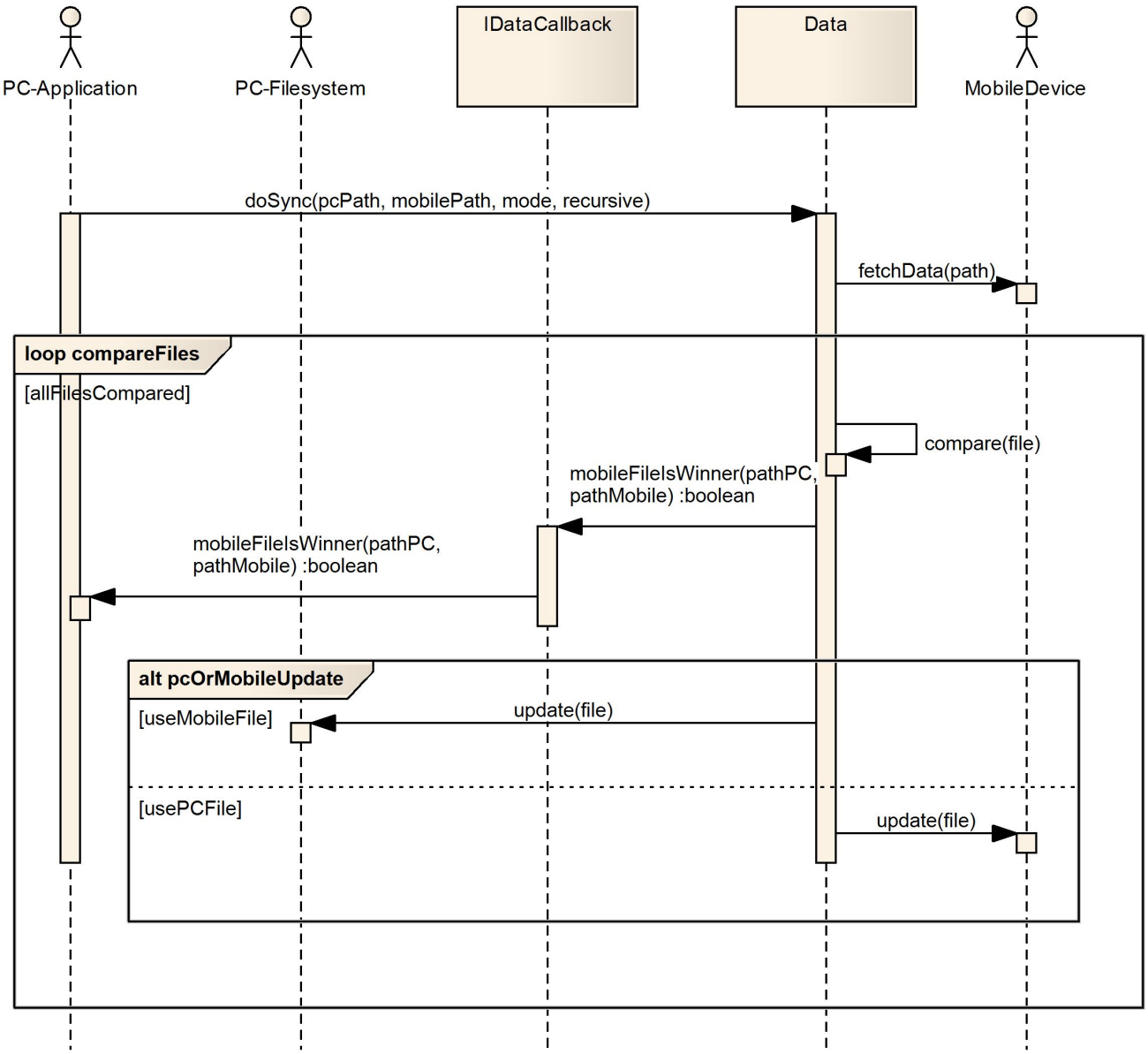


Abbildung 3.13: Bearbeitete Daten werden synchronisiert

3.5.3 UC19 Gelöschte Daten synchronisieren

Der Ablauf im Zeitstempelmodus ist der selbe wie in UC17. Für Details siehe Abbildung 3.12, Seite 17. Der Ablauf im Rückfragemodus ist der selbe wie in UC18. Für Details siehe Abbildung 3.13.

3.6 Kontaktsynchronisation

Die Synchronisation der Kontakte läuft genau gleich ab wie die Kalendersynchronisation. Für Details siehe Kapitel 3.1, Seite 8.

4 Operation Contracts

4.1 OC01: doSync

Operation	doSync(entries)
Cross References	UC01-03
Preconditions	• Auf dem Mobile-Device gibt es eine Kalenderapplikation. • Die Liste «entries» ist mit den Kalendereinträgen gefüllt. • Zwischen PC und Mobile-Device wurde eine Verbindung hergestellt.
Postconditions	Alle Einträge wurden mit einander verglichen und entsprechend angepasst. Für Details siehe Tabellen 4.2, 4.3 und 4.4.

Tabelle 4.1: Operation Contract 01

4.2 OC02: add

Operation	add(entry)
Cross References	UC01
Preconditions	• Die Synchronisation war erfolgreich. • Auf dem Mobile-Device wurde ein Eintrag gefunden, der in der Entry-Liste noch nicht vorhanden ist.
Postconditions	Der neue Eintrag wurde auf dem Gerät hinzugefügt.

Tabelle 4.2: Operation Contract 02

4.3 OC03: update

Operation	update(entry)
Cross References	UC02
Preconditions	• Die Synchronisation war erfolgreich. • In der Entry-Liste oder auf dem Mobile-Device wurde ein Eintrag gefunden, der nicht dieselben Informationen wie auf dem anderen Gerät enthält.
Postconditions	Im älteren Eintrag wurden die Informationen entsprechend angepasst.

Tabelle 4.3: Operation Contract 03

4.4 OC04: delete

Operation	delete(entry)
Cross References	UC03
Preconditions	• Die Synchronisation war erfolgreich. • In der Entry-Liste wurde ein Eintrag gefunden, der auf dem Mobile-Device nicht mehr vorhanden ist.
Postconditions	Der Eintrag wurde auf dem Gerät gelöscht.

Tabelle 4.4: Operation Contract 04

4.5 OC05: configTelephony

Operation	configTelephony(mobileCallback)
Cross References	UC04-07
Preconditions	• Objekt ITelephonyCallback existiert. • Verbindung zum Mobile-Device existiert.
Postconditions	Objekt ITelephonyCallback wurde gespeichert.

Tabelle 4.5: Operation Contract 05

4.6 OC06: initCall

Operation	initCall(callInfo, timeout)
Cross References	UC04
Preconditions	ITelephonyCallback ist gesetzt.
Postconditions	Verbindungsaufbau wurde ausgelöst.

Tabelle 4.6: Operation Contract 06

4.7 OC07: connected

Operation	connected()
Cross References	UC04-06
Preconditions	Verbindung zu einem anderen Gesprächsteilnehmer besteht.
Postconditions	-

Tabelle 4.7: Operation Contract 07

4.8 OC08: incomingCall

Operation	incomingCall(callInfo)
Cross References	UC05-07
Preconditions	Eingehender Anruf ist vorhanden.
Postconditions	-

Tabelle 4.8: Operation Contract 08

4.9 OC09: acceptCall

Operation	acceptCall()
Cross References	UC05
Preconditions	• ITelephonyCallback ist gesetzt. • Eingehender Anruf ist vorhanden.
Postconditions	Anruf wurde entgegengenommen.

Tabelle 4.9: Operation Contract 09

4.10 OC10: refuseCall

Operation	refuseCall()
Cross References	UC07
Preconditions	• ITelephonyCallback ist gesetzt. • Eingehender Anruf ist vorhanden.
Postconditions	Anruf wurde abgelehnt.

Tabelle 4.10: Operation Contract 10

4.11 OC11: release

Operation	release()
Cross References	UC08
Preconditions	Aktive Verbindung zu einem anderen Gesprächsteilnehmer besteht.
Postconditions	Anruf wurde beendet.

Tabelle 4.11: Operation Contract 11

4.12 OC12: released

Operation	released()
Cross References	UC08-09
Preconditions	Aktive Verbindung zu einem anderen Gesprächsteilnehmer wurde beendet.
Postconditions	-

Tabelle 4.12: Operation Contract 12

4.13 OC13: turnSpeakerOff

Operation	turnSpeakerOff()
Cross References	UC10
Preconditions	-
Postconditions	Der Lautsprecher wurde ausgeschaltet.

Tabelle 4.13: Operation Contract 13

4.14 OC14: turnSpeakerOn

Operation	turnSpeakerOn()
Cross References	UC10
Preconditions	-
Postconditions	Der Lautsprecher wurde eingeschaltet.

Tabelle 4.14: Operation Contract 14

4.15 OC15: setSpeakerVolume

Operation	setSpeakerVolume(level)
Cross References	UC11
Preconditions	-
Postconditions	Die Lautstärke des Lautsprechers wurde verändert.

Tabelle 4.15: Operation Contract 15

4.16 OC16: sendTextMessage

Operation	sendTextMessage(number, message)
Cross References	UC14, UC16
Preconditions	Das Objekt TextMessage wurde bereits mit einem ITextMessageCallback konfiguriert.
Postconditions	Die Kurzmitteilung wurde mit dem Inhalt message an number versendet.

Tabelle 4.16: Operation Contract 16

4.17 OC17: onTextMessageRecieved

Operation	onTextMessageRecieved(message, number)
Cross References	UC15
Preconditions	Eingehende SMS ist vorhanden.
Postconditions	-

Tabelle 4.17: Operation Contract 17

4.18 OC18: doSync

Operation	doSync(pcPath, mobilePath, mode, recursive)
Cross References	UC17-19
Preconditions	• Das Objekt Data wurde bereits mit einem IDataCallback konfiguriert. • Die mit pcPath und MobilePath angegebenen Ordner existieren.
Postconditions	-

Tabelle 4.18: Operation Contract 18

4.19 OC19: add

Operation	add(file)
Cross References	UC17
Preconditions	• Die Synchronisation war erfolgreich. • Auf dem Mobile-Device oder dem PC wurde eine Datei gefunden, die auf einem der beiden Geräte noch nicht vorhanden ist.
Postconditions	Die neue Datei wurde auf dem anderen Gerät hinzugefügt.

Tabelle 4.19: Operation Contract 19

4.20 OC20: update

Operation	update(file))
Cross References	UC18
Preconditions	• Die Synchronisation war erfolgreich. • Die Datei existiert auf beiden Geräten und wurde verändert. • Mit dem Zeitstempel bzw. mit Benutzereingabe wurde ermittelt, welche der beiden Dateien zu behalten ist.
Postconditions	Die gewählte Datei überschreibt die andere.

Tabelle 4.20: Operation Contract 20

4.21 OC21: delete

Operation	delete(file))
Cross References	UC19
Preconditions	• Die Synchronisation war erfolgreich. • Eine Datei ist auf einem der beiden Geräte gelöscht worden.
Postconditions	Die Datei wurde auf dem anderen Gerät gelöscht.

Tabelle 4.21: Operation Contract 21

4.22 OC22: mobileFileIsWinner

Operation	mobileFileIsWinner(pathPC, pathMobile))
Cross References	UC18, UC19
Preconditions	• Es existiert eine Datei, die auf beiden Geräten verändert wurde. • Der Vergleichsmodus ist auf Rückfrage eingestellt.
Postconditions	Die Funktion gibt true zurück, falls die Datei auf dem Mobile-Device behalten werden soll, sonst false.

Tabelle 4.22: Operation Contract 22

5 Voraussetzungen

5.1 Kalender

5.1.1 Datenbankschema Google-Kalender

In einem Google-Kalender können Termine eingetragen werden, in denen keine, eine oder mehrere Erinnerungen, sogenannte «Reminders», gesetzt sind. Diese sind in einer separaten Datenbank-Tabelle gespeichert und referenzieren jeweils den zugehörigen Termin über einen Fremdschlüssel. Jeder Termin wiederum referenziert einen Kalender mittels Fremdschlüssel.

Jeder Kalender, jeder Eintrag und jede Erinnerung besitzt eine eindeutige Id, mittels dieser sie identifiziert werden können. Die Id's werden für jede Tabelle in aufsteigender Reihenfolge vergeben und entsprechen der jeweils höchsten, noch nicht vergebenen Id. Id's gelöschter Kalender, Einträge und Erinnerungen stehen ebenfalls erneut zur Verfügung, sofern sie nicht tiefer als eine bereits besetzte Id sind.

Wird ein Eintrag sowohl auf der Server- und Mobile-Device-Seite geändert, gewinnt bei der Synchronisation der Server und die Änderungen auf der Mobile-Device-Seite werden mit jenen des Servers überschrieben.

Kommen auf der Server- und Mobile-Device-Seite neue Kalendereinträge hinzu, werden die Id's der Einträge auf der Server-Seite so angepasst, dass keine doppelten Id's vorhanden sind.

Für weitere Informationen zum Datenbankschema siehe Anhang, Google Datenbankschema.

5.2 Telefon

Android macht bei den Telefonfunktionen folgende Einschränkung: Es kann maximal ein Anruf in den Halten-Modus versetzt werden. Kommt ein weiterer hinzu, wird der erste Anruf automatisch beendet.

5.3 Kontakte

Die Kontakt-API wurde zwischen den Android-Versionen 1.6 und 2.0 stark verändert. Dies vor allem bei der Struktur der Tabellen, sowie im Zugriff darauf.

5.4 Daten

5.4.1 Zeitmessung

Das Feld «last modified» einer Datei auf dem Android-Betriebssystem enthält die letzte Bearbeitungszeit in Bezug auf den 01.01.1970 auf die Sekunden genau gerundet. Der Wert wird jedoch in Millisekunden abgespeichert.

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	22.09.2010	dm	Dokument erstellt.
1.0rc02	10.10.2010	rr	Domainmodel, SSD und Contracts für Telefon UseCases hinzugefügt.
1.0rc03	13.10.2010	dm	Review, diverse Korrekturen.
1.0	15.10.2010	dm	Akzeptierte Version 1.0
1.1rc01	10.11.2010	dm, rr	Kapitel Voraussetzungen hinzugefügt.
1.1rc02	10.11.2010	rr	Überarbeitung des Dokuments, hinzufügen der Erweiterungen.
1.1rc03	10.11.2010	dm	Review, diverse Korrekturen.
1.1rc04	17.11.2010	dm, rr	Korrekturen in Kapitel Domainmodel gemäss Besprechung.
1.1	20.11.2010	dm	Akzeptierte Version 1.1
1.2rc01	20.11.2010	dm	Kapitel Voraussetzungen angepasst.
1.2rc02	08.12.2010	dm	Domainmodel angepasst.
1.2	11.12.2010	dm	Akzeptierte Version 1.2
1.3rc01	12.12.2010	dm	Kapitel Voraussetzungen ergänzt
1.3rc02	19.12.2010	rr	Rechtschreibung und Layout verbessert.
1.3	20.12.2010	dm	Akzeptierte Version 1.3

Tabelle 5.1: Revisionshistorie

Anhang

```
CREATE TABLE Attendees (_id INTEGER PRIMARY KEY,event_id INTEGER,attendeeName
TEXT,attendeeEmail TEXT,attendeeStatus INTEGER,attendeeRelationship INTEGER,attendeeType
INTEGER);

CREATE TABLE CalendarAlerts (_id INTEGER PRIMARY KEY,event_id INTEGER,begin INTEGER NOT
NULL,end INTEGER NOT NULL,alarmTime INTEGER NOT NULL,creationTime INTEGER NOT
NULL,receivedTime INTEGER NOT NULL,notifyTime INTEGER NOT NULL,state INTEGER NOT
NULL,minutes INTEGER,UNIQUE (alarmTime, begin, event_id));

CREATE TABLE CalendarCache (_id INTEGER PRIMARY KEY,key TEXT NOT NULL,value TEXT);

CREATE TABLE CalendarMetaData (_id INTEGER PRIMARY KEY,localTimezone TEXT,minInstance
INTEGER,maxInstance INTEGER);

CREATE TABLE Calendars (_id INTEGER PRIMARY KEY,_sync_account TEXT,_sync_account_type
TEXT,_sync_id TEXT,_sync_version TEXT,_sync_time TEXT,_sync_local_id INTEGER,_sync_dirty
INTEGER,_sync_mark INTEGER,url TEXT,name TEXT,displayName TEXT,hidden INTEGER NOT NULL
DEFAULT 0,color INTEGER,access_level INTEGER,selected INTEGER NOT NULL DEFAULT 1,sync_events
INTEGER NOT NULL DEFAULT 0,location TEXT,timezone TEXT,ownerAccount TEXT,
organizerCanRespond INTEGER NOT NULL DEFAULT 1);

CREATE TABLE Events (_id INTEGER PRIMARY KEY,_sync_account TEXT,_sync_account_type
TEXT,_sync_id TEXT,_sync_version TEXT,_sync_time TEXT,_sync_local_id INTEGER,_sync_dirty
INTEGER,_sync_mark INTEGER,calendar_id INTEGER NOT NULL,htmlUri TEXT,title TEXT,eventLocation
TEXT,description TEXT,eventStatus INTEGER,selfAttendeeStatus INTEGER NOT NULL DEFAULT
0,commentsUri TEXT,dtstart INTEGER,dtend INTEGER,eventTimezone TEXT,duration TEXT,allDay
INTEGER NOT NULL DEFAULT 0,visibility INTEGER NOT NULL DEFAULT 0,transparency INTEGER NOT
NULL DEFAULT 0,hasAlarm INTEGER NOT NULL DEFAULT 0,hasExtendedProperties INTEGER NOT
NULL DEFAULT 0,rrule TEXT,rdate TEXT,exrule TEXT,exdate TEXT,originalEvent
TEXT,originalInstanceTime INTEGER,originalAllDay INTEGER,lastDate INTEGER,hasAttendeeData
INTEGER NOT NULL DEFAULT 0,guestsCanModify INTEGER NOT NULL DEFAULT
0,guestsCanInviteOthers INTEGER NOT NULL DEFAULT 1,guestsCanSeeGuests INTEGER NOT NULL
DEFAULT 1,organizer STRING, deleted INTEGER NOT NULL DEFAULT 0, dtstart2 INTEGER, dtend2
INTEGER, eventTimezone2 TEXT, syncAdapterData TEXT);

CREATE TABLE EventsRawTimes (_id INTEGER PRIMARY KEY,event_id INTEGER NOT NULL,dtstart2445
TEXT,dtend2445 TEXT,originalInstanceTime2445 TEXT,lastDate2445 TEXT,UNIQUE (event_id));

CREATE TABLE ExtendedProperties (_id INTEGER PRIMARY KEY,event_id INTEGER,name TEXT,value
TEXT);

CREATE TABLE Instances (_id INTEGER PRIMARY KEY,event_id INTEGER,begin INTEGER,end
INTEGER,startDay INTEGER,endDay INTEGER,startMinute INTEGER,endMinute INTEGER,UNIQUE
(event_id, begin, end));
```

Abbildung 5.1: Google-Datenbankschema

```
CREATE TABLE Reminders (_id INTEGER PRIMARY KEY,event_id INTEGER,minutes INTEGER,method
INTEGER NOT NULL DEFAULT 0);

CREATE TABLE _sync_state (_id INTEGER PRIMARY KEY,account_name TEXT NOT NULL,account_type
TEXT NOT NULL,data TEXT,UNIQUE(account_name, account_type));

CREATE TABLE _sync_state_metadata (version INTEGER);

CREATE TABLE android_metadata (locale TEXT);

CREATE VIEW view_events AS SELECT Events._id AS
_id,htmlUri,title,description,eventLocation,eventStatus,selfAttendeeStatus,commentsUri,dtstart,dte
nd,duration,eventTimezone,allDay,visibility,timezone,selected,access_level,transparency,color,hasAl
arm,hasExtendedProperties,rrule,rdate,exrule,exdate,originalEvent,originalInstanceTime,originalAllD
ay,lastDate,hasAttendeeData,calendar_id,guestsCanInviteOthers,guestsCanModify,guestsCanSeeGue
sts,organizer,deleted,Events._sync_id AS _sync_id,Events._sync_version AS
_sync_version,Events._sync_dirty AS _sync_dirty,Events._sync_account AS
_sync_account,Events._sync_account_type AS _sync_account_type,Events._sync_time AS
_sync_time,Events._sync_local_id AS _sync_local_id,Events._sync_mark AS
_sync_mark,url,ownerAccount,sync_events FROM Events JOIN Calendars ON
(Events.calendar_id=Calendars._id);

CREATE INDEX attendeesEventIdIndex ON Attendees (event_id);

CREATE INDEX calendarAlertsEventIdIndex ON CalendarAlerts (event_id);

CREATE INDEX eventSyncAccountAndIdIndex ON Events (_sync_account_type, _sync_account,
_sync_id);

CREATE INDEX eventsCalendarIdIndex ON Events (calendar_id);

CREATE INDEX extendedPropertiesEventIdIndex ON ExtendedProperties (event_id);

CREATE INDEX instancesStartDayIndex ON Instances (startDay);

CREATE INDEX remindersEventIdIndex ON Reminders (event_id);

CREATE TRIGGER calendar_cleanup DELETE ON Calendars BEGIN DELETE FROM Events WHERE
calendar_id = old._id;END;

CREATE TRIGGER events_cleanup_delete DELETE ON Events BEGIN DELETE FROM Instances WHERE
event_id = old._id;DELETE FROM EventsRawTimes WHERE event_id = old._id;DELETE FROM
Attendees WHERE event_id = old._id;DELETE FROM Reminders WHERE event_id = old._id;DELETE
FROM CalendarAlerts WHERE event_id = old._id;DELETE FROM ExtendedProperties WHERE event_id
= old._id;END;
```

Abbildung 5.2: Google-Datenbankschema

Abkürzungsverzeichnis

API Application Programming Interface
PC Personal Computer
OC Operation Contract
SA Studienarbeit
SMS Short Message Service
SSD Systemsequenz-Diagramm
UC Use Case

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] LARMAN, Craig: *Applying UML and Patterns*. Prentice Hall, 2004. – ISBN 978–0–13–148906–2

Software Architecture Document
«Android Control Framework»

Version 1.2

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 7.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 Logical View

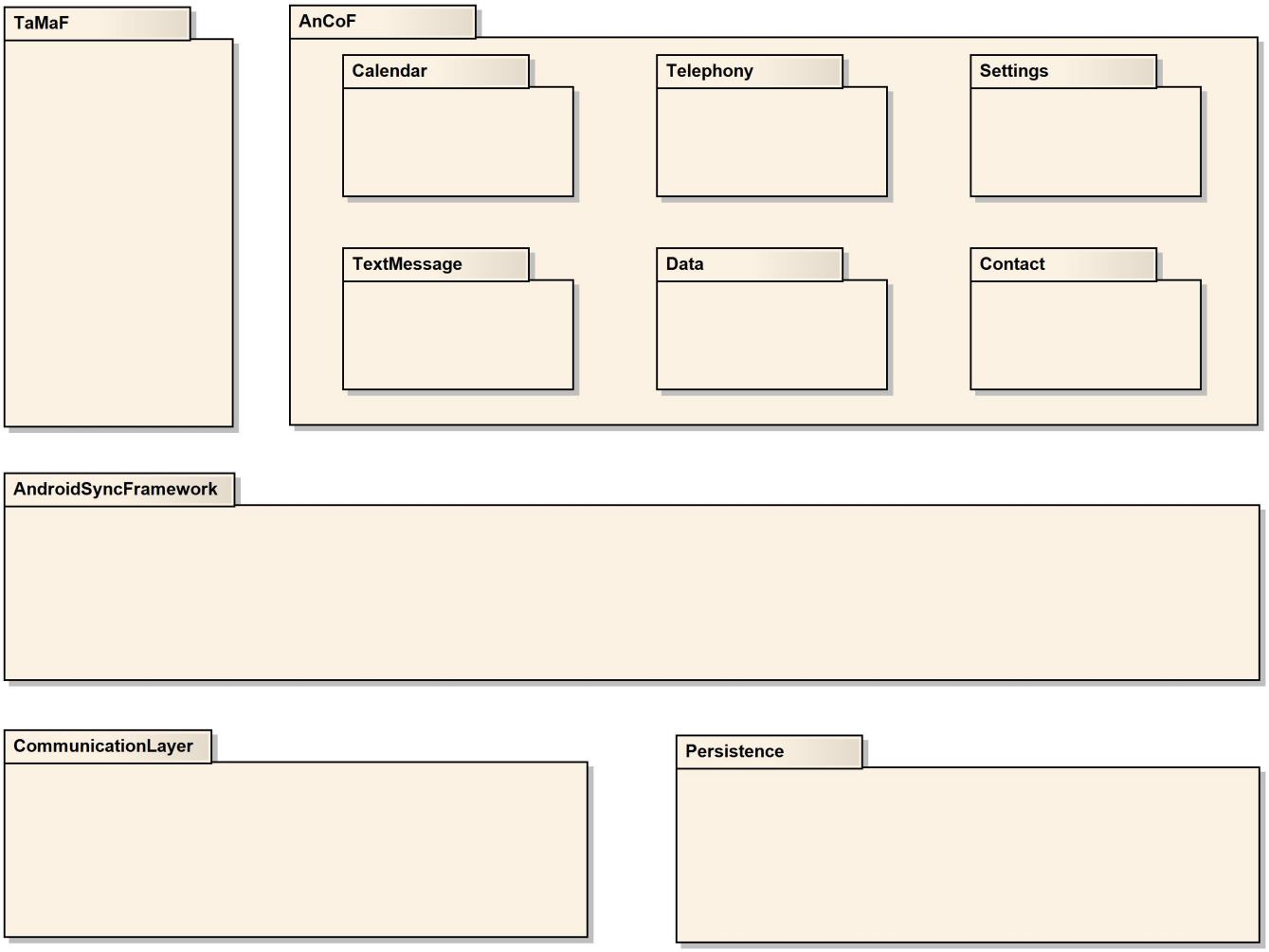


Abbildung 2.1: Packages (Alle Subpackages in AnCoF sind gleichrangig und wurden nur aus Darstellungsgründen untereinander angeordnet.)

2.1 AnCoF

Das Package AnCoF bietet für jeden Teilbereich von AnCoF eine Sammlung von Subpackages an.

2.1.1 Calendar

Im Package «Calendar» werden alle Klassen zusammengefasst, die mit der Kalendersynchronisation zu tun haben.

2.1.2 Telephony

Im Package «Telephony» werden alle Klassen zusammengefasst, die mit der Telefonie (z.B. Anruf tätigen) zu tun haben.

2.1.3 Settings

Im Package «Settings» werden alle Klassen zusammengefasst, die mit dem Verändern der Telefoneinstellungen zu tun haben.

2.1.4 TextMessage

Im Package «TextMessage» werden alle Klassen zusammengefasst, die mit dem SMS-Handling zu tun haben.

2.1.5 Data

Im Package «Data» werden alle Klassen zusammengefasst, die mit der Datensynchronisation zu tun haben.

2.1.6 Contact

Im Package «Contact» werden alle Klassen zusammengefasst, die mit der Kontaktsynchronisation zu tun haben.

2.2 TaMaF

Dieses Package enthällt alle Funktionen des Task-Management-Framework on Smart-Phone (TaMaF). Für Details siehe Dokumentation TaMaF.

2.3 AndroidSyncFramework

AnCoF baut mit diesem Package die Verbindung zu den Mobile-Devices auf. Dieses Package wird von TaMaF übernommen und erweitert, damit es auch die AnCoF spezifischen Funktionen unterstützt. Details können dem Design Dokument (DD) bzw. der Dokumentation von TaMaF entnommen werden.

2.4 CommunicationLayer

Dieses Package wird von TaMaF übernommen. Für Details siehe Dokumentation TaMaF.

2.5 Persistence

Dieses Package wird von TaMaF übernommen. Für Details siehe Dokumentation TaMaF.

3 Deployment View

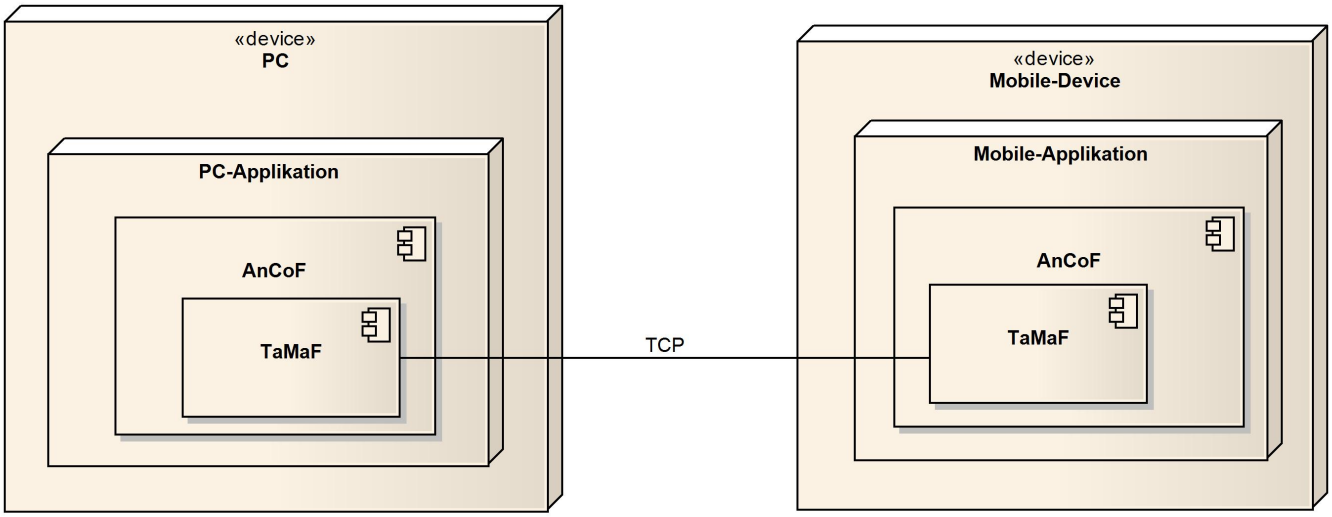


Abbildung 3.1: Deployment

Auf der Personal Computer (PC)-Seite bildet AnCoF einen Bestandteil der darauf aufbauenden PC-Applikation. Für die Mobile-Applikation bietet AnCoF verschiedene Services an, welche bei Bedarf benutzt werden können.

4 Process View

PC und Mobile-Device stellen zwei separate Prozesse dar, die von TaMaF verwaltet werden. Details können der Dokumentation von TaMaF entnommen werden.

Zudem wird auf dem Mobile-Device jeder Bereich von AnCoF als eigener Thread geführt. Bei weiteren Parallelitäten, z.B. zwei eingehende Telefonanrufe zur gleichen Zeit, werden Android-Mechanismen ausgenutzt.
Auf der PC-Applikation wird davon ausgegangen, dass keine parallelen Aktionen erfolgen.

5 Data View

5.1 Kalender

Für die Synchronisation der Kalendereinträge wird der letzte logische Zeitstempel der synchronisierten Kalendereinträge als Hash-Wert dauerhaft auf der Festplatte des PCs gespeichert.

5.2 Kontakte

Für die Synchronisation der Kontakteinträge wird der letzte logische Zeitstempel der synchronisierten Kontakteinträge als Hash-Wert dauerhaft auf der Festplatte des PCs gespeichert.

5.3 Daten

Für die Synchronisation der Daten wird sowohl der letzte Synchronisationszeitpunkt als auch der letzte Stand der synchronisierten Dateien als Pfadangaben dauerhaft auf der Festplatte des PCs gespeichert.

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	13.10.2010	dm	Dokument erstellt.
1.0rc02	19.10.2010	rr	Kapitelentwürfe erstellt.
1.0	23.10.2010	dm	Akzeptierte Version 1.0
1.1rc01	17.11.2010	dm, rr	Erweiterung um neue Bereiche
1.1	20.11.2010	dm	Akzeptierte Version 1.1
1.2rc01	19.12.2010	rr	Rechtschreibung und Layout korrigiert.
1.2	20.11.2010	dm	Akzeptierte Version 1.2

Tabelle 5.1: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
DD Design Dokument
PC Personal Computer
SA Studienarbeit
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Design Dokument
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 7.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 AnCoF

2.1 Klassenstruktur

2.1.1 Subpackages von ch.hsr.ancof

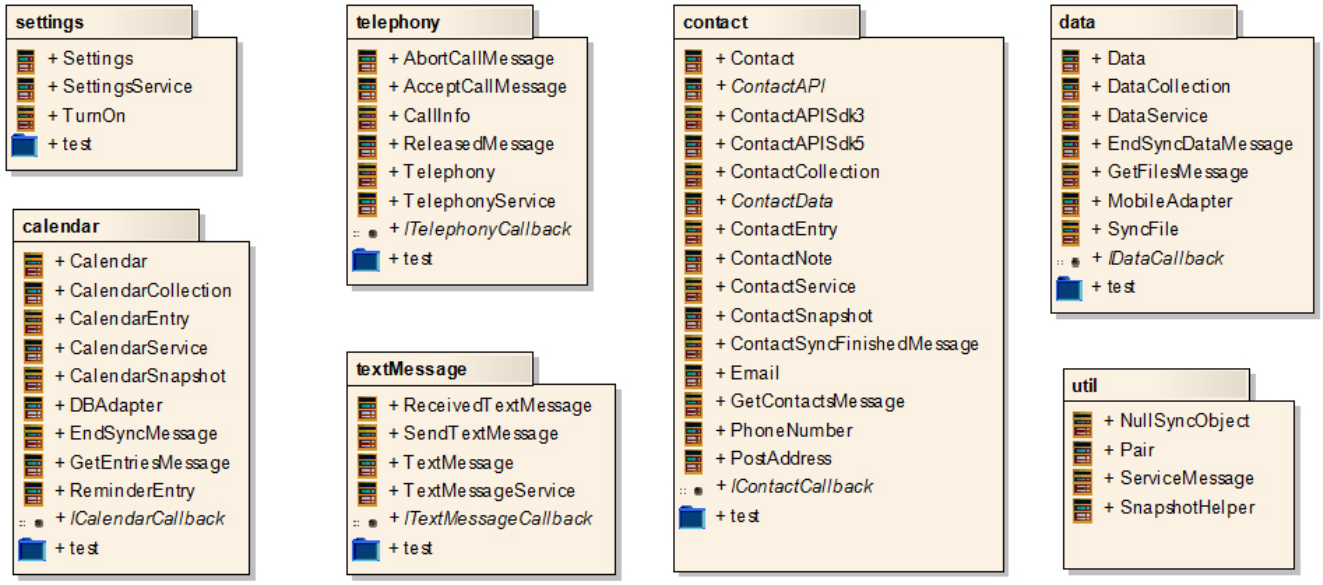


Abbildung 2.1: Subpackages AnCoF

Die Subpackages von «ancof» sind, mit Ausnahme von «util», unabhängig voneinander. Auf die Inhalte der Packages wird in den entsprechenden Kapiteln näher eingegangen.

2.1.2 ch.hsr.ancof

Neben den Subpackages enthält das Package auch noch die Hauptklassen für die Kommunikation zwischen Mobile-Device und PC. Zudem ist die Elter-Klasse der Features und das Elter-Interface der Callback-Interfaces enthalten.

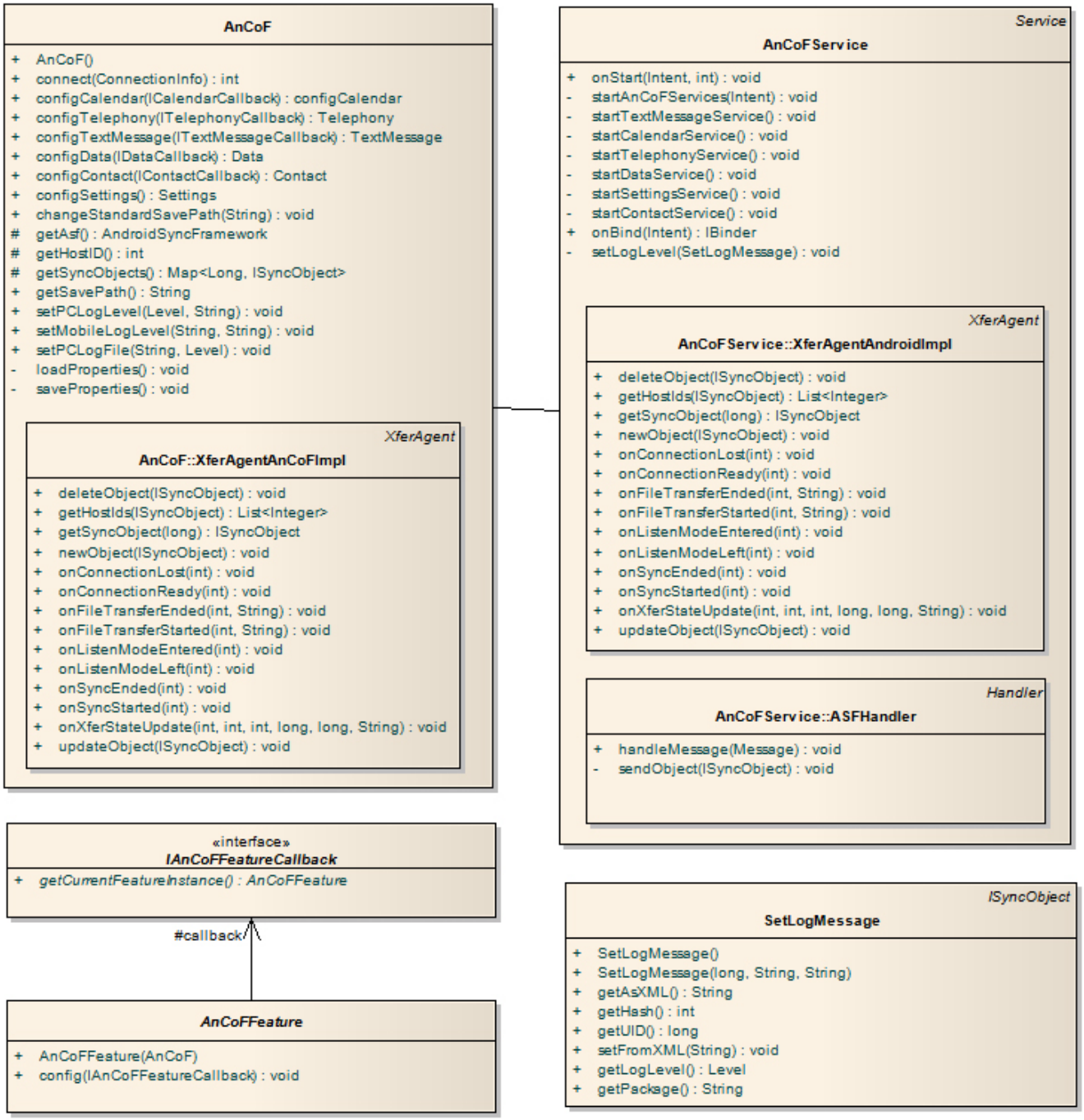


Abbildung 2.2: Package AnCoF

ch.hsr.ancof.AnCoF

Die Klasse «AnCoF» dient als Einstiegsklasse für PC-Applikationen. Mit dieser Klasse kann die Kommunikation mit dem Mobile-Device aufgebaut und es können die Features von AnCoF konfiguriert werden.

Die Kommunikation zwischen Mobile-Device und PC wird an das «AndroidSyncFrameword (ASF)» weitergeleitet. Details dazu können der Dokumentation von Task-Management-Framework on Smart-Phone (TaMaF) entnommen werden.

Um eine Instanz eines Features zu bekommen sollte unbedingt auf die «config»-Funktionen zurückgegriffen werden. Diese geben Funktionen eine Instanz des Features zurück, in der alle wichtigen Parameter

um mit dem Mobile-Device kommunizieren zu können, bereits gesetzt sind.

Auf die Rolle der Implementation des «XferAgent» wird im Kapitel 2.3, Seite 5 näher eingegangen.

ch.hsr.ancof.AnCoFService

Die Klasse «AnCoFService» ist die zweite Hauptklasse. Diese wird nur auf dem Mobile-Device ausgeführt. Sie wird, wie im «Developer Guide» beschrieben, mit einem Intent gestartet und startet danach alle «Unterservices».

«AnCoFService» ist ein Android-Service, der darauf ausgelegt ist, dass er solange läuft, bis er mit `stopService()` angehalten wird. Zudem läuft die komplette Kommunikation mit dem PC über diese Klasse. Die Unterservices senden android.os.Messages mit dem zu versendenden Objekt an den Thread des «AnCoFService». Zusätzlich beinhalten die Messages eine Beschreibung des Inhalts. Im «AnCoFService» wird die Message entpackt und je nach Beschreibung an den PC weitergeleitet. Dies geschieht mit dem «ASF».

ch.hsr.ancof.IAnCoFFeature und ch.hsr.ancof.IAnCoFFeatureCallback

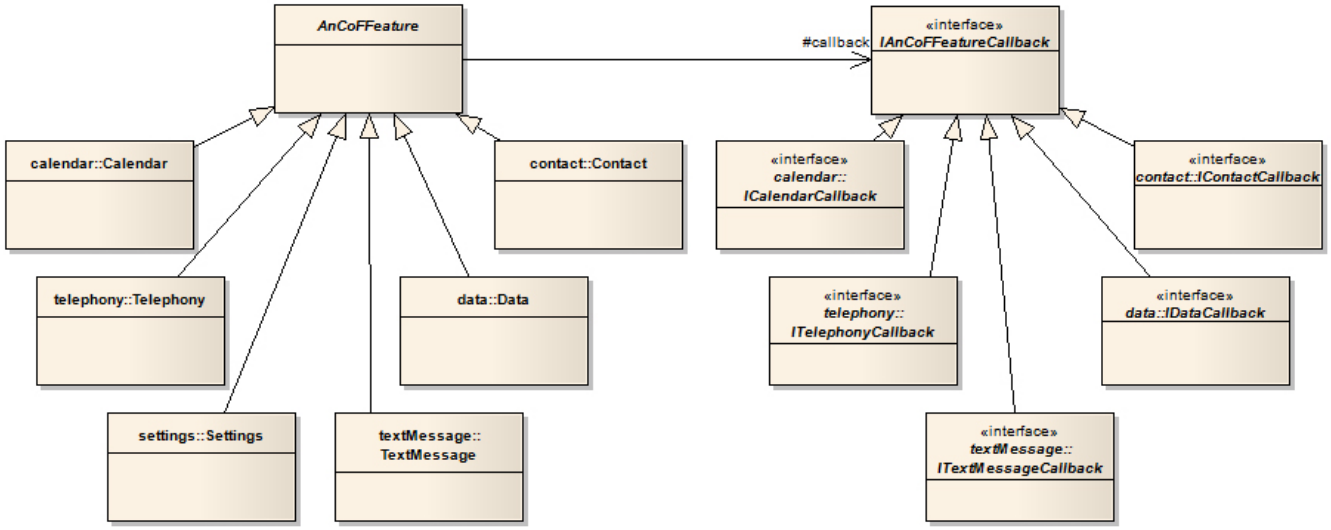


Abbildung 2.3: Features mit Vererbung

Diese beiden Klassen wurden eingeführt, damit das Aufsetzen und die Verbindung mit der Klasse «AnCoF» nicht jedes mal neu implementiert werden muss.

ch.hsr.ancof.SetLogMessage

Die Klasse «SetLogMessage» implementiert das Interface «ISyncObject» des «ASF». Sie wird gebraucht, um eine Meldung, zwecks Loglevel-Veränderung, an das Mobile-Device zu senden. Details zur Kommunikation zwischen Mobile-Device und PC können im Kapitel 2.3, Seite 5 nachgelesen werden.

2.2 Zusammenspiel der Features von AnCoF

Jedes Feature von AnCoF wurde so entwickelt, dass es unabhängig von den anderen existieren kann. Eine Verbindung besteht nur zu den beiden Hauptklassen «AnCoF» und «AnCoFService».

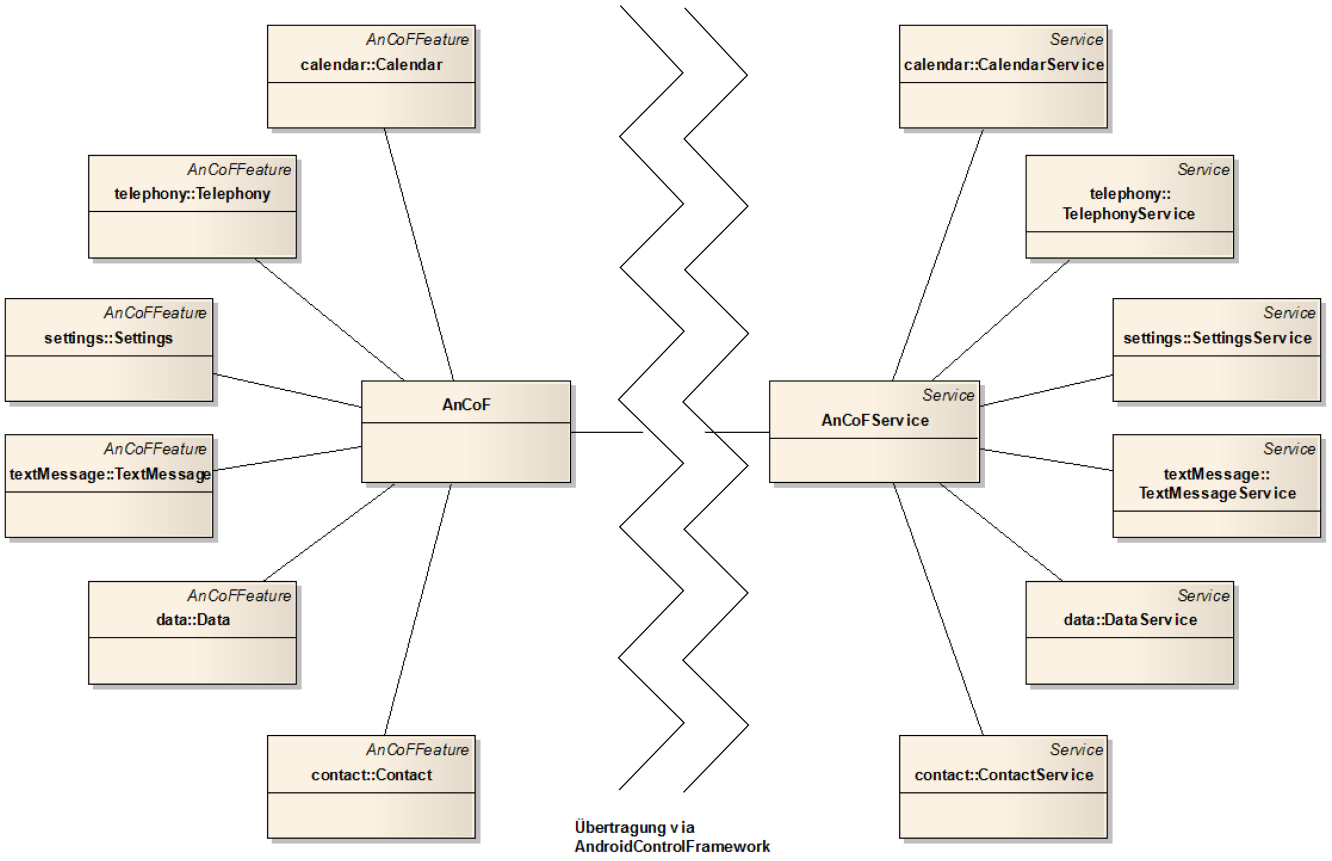


Abbildung 2.4: Zusammenspiel der AnCoF Features

Dies wurde eingeführt, damit das Framework keinen Overhead generiert, wenn nur ein Bereich genutzt wird. Aus dem selben Grund wurden auf der Mobile-Device-Seite Services gewählt und nicht Activities. Somit kann ein Benutzer frei entscheiden, was für eine Applikation bzw. welches Design er entwickeln möchte.

Die «Unterservices» (zum Beispiel CalendarService, TelephonyService) werden vom «AnCoFService» mit `BIND_AUTO_CREATE` gestartet. Der «AnCoFService» wird dabei an den neuen Service gebunden. Die Inneren XXXBinder-Klassen werden gebraucht, um eine Instanz des gestarteten Services im «AnCoFService» zu halten.

2.3 Kommunikation zwischen Mobile-Device und PC

Die Kommunikation zwischen Mobile-Device und PC konnte nicht für beide Geräte gleich implementiert werden. In diesem Kapitel wird nur auf die AnCoF spezifischen Details eingegangen. Wie die Kommunikation aufgebaut ist, kann in der Dokumentation von TaMaF nachgelesen werden.

2.3.1 Versenden von Objekten auf dem PC

Da davon ausgegangen wird, dass auf dem PC keine Parallelität existiert, kann jede Feature-Klasse das Versenden von Nachrichten selbst übernehmen. Das Message-Objekt wird in der Map «syncObjects» mit der UID als Key gespeichert, damit das «ASF» das Objekt zum Versenden wiederfinden kann. Danach wird das Objekt über das «ASF» versendet.

2.3.2 Versenden von Objekten auf dem Mobile-Device

Das Versenden von Objekten auf dem Mobile-Device gestaltet sich etwas komplizierter. Hier kann nicht davon ausgegangen werden, dass keine Parallelität existiert. Aus diesem Grund wurde das Versenden in zwei Schritten durchgeführt.

Im ersten Schritt verpackt der «Unterservice» das Objekt in eine Message und fügt es der Message-Queue des AnCoF-Services hinzu. Im zweiten Schritt überprüft der «AnCoFService», um was für eine Nachricht es sich handelt, fügt dem Objekt die eindeutige UID hinzu und speichert bzw. versendet es analog, wie dies auch auf dem PC geschieht.

Die einzige Ausnahme hierbei bildet der «TelephonyService». Er fügt die Message an erster Stelle der Queue ein. Somit sind Telefonanrufe höher priorisiert als z.B. eine Kalendersynchronisation.

2.3.3 Empfangen der Nachrichten

Die Implementation des XferAgent empfängt die vom anderen Gerät kommenden Nachrichten und reicht diese, falls nötig, an die entsprechenden Verarbeitungsorte weiter. Auf die Verarbeitung der Nachrichten wird in den jeweiligen Kapiteln eingegangen.

2.4 Verwendete Libraries

Um das Framework auch für die neue Kalender-API nutzen zu können, muss die Android-Library der Version 2.0 eingebunden werden. Dies hat zur Folge, dass AnCoF nicht mehr für Geräte mit einem Android-System kleiner 2.0 verwendet werden kann. In diesem Fall muss die Android-Library der Version 1.6 eingebunden werden. Die entstandenen Fehler können ignoriert werden, da sie in einer Klasse auftreten, welche auf Android 1.6 nicht genutzt wird.

3 Kalender

Für die Implementierung der Kalenderfunktionalitäten im AnCoF wurde eine Lösung gesucht, die möglichst einfach und effizient zu aktualisieren ist, falls der Google-Standardkalender neue Funktionalitäten erhält oder Änderungen erfährt. Folgende Lösungsvorschläge wurden erarbeitet (Vorschlag, Begründung):

- Nutzung «Google Data API»: Falls der Kalender in einer späteren Version die API ändert, muss man sich nicht um die Anpassungen zu kümmern.
- Erstellen einer eigenen Kalender-Klasse: Die Datenstruktur kann frei gewählt werden.
- Erstellen einer eigenen Kalender-Klasse, die mit der «Google Data API» interagiert: Methoden der «Google Data API» können genutzt werden.

3.1 Lösungen

3.1.1 «Google Data API»

Die «Google Data API» kann nur für PC-Applikationen verwendet werden. Eine neue Version, Version 1.1.0-alpha released [1], bietet laut Beschreibung Support für die «Java client library», welche Bestandteil der «Google Data API» ist. Die erforderlichen Funktionen sind jedoch nicht wie angegeben verfügbar. Hinzu kommt, dass man, um die API nutzen zu können, eine Verbindung zum User Account respektive dem Google Kalender im Internet herstellen muss.

3.1.2 Eigene Klasse

Mittels Auslesen aus der Datenbank können die lokal gespeicherten Daten des Kalenders erfasst und bearbeitet werden. Geänderte Daten können auch zurückgeschrieben respektive Einträge erfasst oder gelöscht werden[2]. Der Nachteil einer eigenen Klasse ist, dass bei Änderungen an der «Google Data API» unter Umständen das AnCoF nicht mehr genutzt werden kann respektive angepasst werden muss.

3.1.3 Eigene Klasse, die mit «Google Data API» interagiert

Um den Zugriff über das Internet zu umgehen, kann eine eigene Klasse erstellt werden, welche die Daten aus der Datenbank ausliest respektive zurückschreibt und sie für die Synchronisation in der Datenstruktur, welche das «Google Data API» vorgibt, zwischenspeichert. So könnten die Vorteile einer eigenen Klasse genutzt werden und zugleich auch auf die Synchronisationsmechanismen der «Google Data API» zurück gegriffen werden. Gemäss Abschnitt 3.1.1 ist dies jedoch nicht möglich, da die Klassen gar nicht verfügbar sind.

3.2 Entscheid

Aufgrund der obigen Recherchen kann die «Google Data API» für unser Framework nicht zunutze gemacht werden. AnCoF realisiert die Umsetzung der Kalender-Funktionalitäten mit einer eigenen Kalender-Klasse,

[1] `http://code.google.com/p/google-api-java-client/`

[2] `http://www.developer.com/ws/article.php/3850276/Working-with-the-Android-Calendar.htm`

welche eigenständig und unabhängig von der «Google Data API» ist. Es muss das Risiko eingegangen werden, dass allfällige Änderungen am API von Google auch Wartungsarbeiten am AnCoF nach sich ziehen.

3.3 Klassenstruktur

3.3.1 ch.hsr.ancof.calendar

Abbildung 3.1: Package Calendar

Die Funktionalitäten der Kalender- und Erinnerungseinträge wurden je in einer eigenen Klasse umgesetzt. Grund dafür ist die Struktur der Datenbank des Mobile-Devices, welche jeweils für die Kalender, deren Einträge und die dazugehörigen Erinnerungen eine eigene Tabelle besitzt. Für Details zum Datenbankschema siehe «Domainanalyse», Kapitel 5.1.1, Seite 27. Eine übergreifende Klasse bewältigt sodann die Kalendersynchronisation und holt sich die entsprechenden Daten aus den anderen beiden Klassen. Zusätzlich wird eine Sicherung des synchronisierten Zustandes benötigt, ein Snapshot.

ch.hsr.ancof.calendar.Calendar

Die Klasse «Calendar» startet den Synchronisationsprozess. Zusätzlich kann hier konfiguriert werden, welche Seite bei Konflikten als Gewinner hervorgeht.

ch.hsr.ancof.calendar.CalendarCollection

Die Klasse «CalendarCollection» beinhaltet alle Methoden, die für die eigentliche Synchronisation verlangt werden. Für Details zur Synchronisation siehe 3.4, Seite 10.

ch.hsr.ancof.calendar.CalendarEntry

Die Klasse «CalendarEntry» beinhaltet alle Methoden, welche benötigt werden, um einen Kalendereintrag zu erstellen.

ch.hsr.ancof.calendar.ReminderEntry

Die Klasse «ReminderEntry» beinhaltet alle Methoden, die für eine Erinnerungsfunktion benötigt werden.

ch.hsr.ancof.calendar.CalendarSnapshot

Die Klasse «CalendarSnasphot» dient der Speicherung des synchronisierten Zustandes, damit man für die nächste Synchronisation einen Vergleichswert hat. Dieser erlaubt es zu entscheiden, welche Einträge gelöscht, welche verändert und welche Einträge neu hinzugefügt wurden. Dazu werden jeweils pro Kalender die Id-Werte jedes Eintrags mit einem Hash-Wert über die gesamten dazugehörigen Felder gespeichert. Der Hash-Wert übernimmt keine Sicherheitsfunktion, er dient nur dem Vergleich. Zusätzlich werden für die Kalendereinträge und Erinnerungen die Maximal-Werte der Id's festgehalten.

ch.hsr.ancof.calendar.DBAdapter

Die Klasse «DBAdapter» dient als Bindeglied zwischen PC und Mobile-Device. Sie kümmert sich darum, die Daten aus der Datenbank des Mobile-Devices zu laden und diese entsprechend auch wieder zurück zu speichern.

3.4 Synchronisation

Um die Synchronisation zu initiieren, wird mit der Klasse «GetEntriesMessage» eine Nachricht versendet, die nur dazu dient, den Synchronisationsprozess zu starten. Danach werden die aktuellen Kalendereinträge auf dem Mobile-Device ausgelesen und auf den PC übermittelt, da die Synchronisation aus Performance-Gründen dort stattfindet. Hier werden diese mit jenen des PC's verglichen und danach wiederum zum Mobile übertragen, wo sie entsprechend hinzugefügt, aktualisiert oder gelöscht werden. Sind die Einträge auf dem Mobile fertig bearbeitet, wird mit der Klasse «EndSyncMessage» eine Nachricht zurück zum PC geschickt. Sie signalisiert das Ende der Synchronisation und übermittelt die maximalen Id-Werte, um diese im Snapshot (siehe Kapitel 3.3.1, Seite 9) zu aktualisieren.

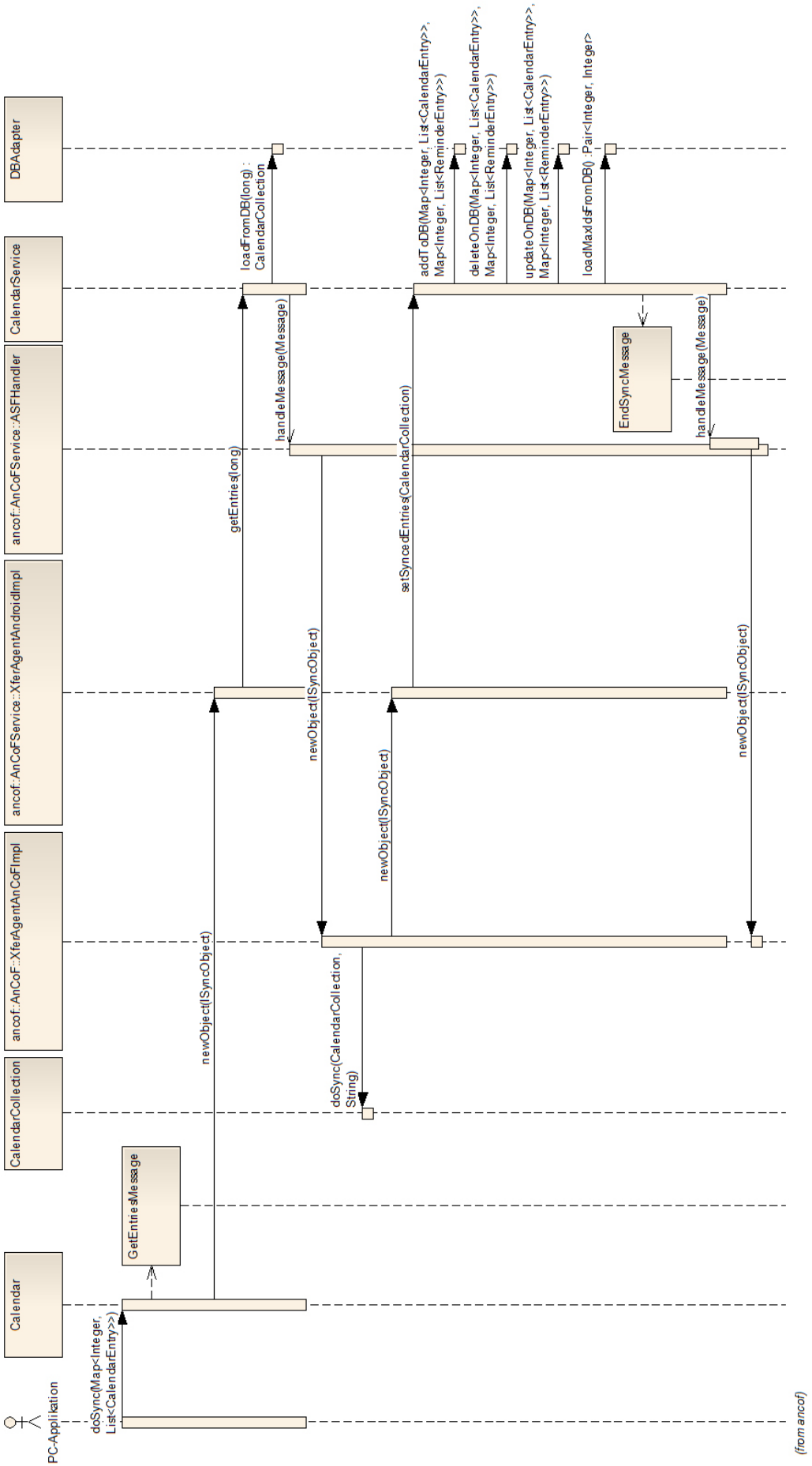


Abbildung 3.2: Ablauf Synchronisation

3.4.1 Datenbankschema Google-Kalender

Im Gegensatz zum Google-Kalender, kann bei AnCoF konfiguriert werden, wer bei Synchronisationskonflikten gewinnt. Für Details zum Datenbankschema des Google-Kalenders siehe «Domainanalyse», Kapitel 5.1.1, Seite 27.

3.4.2 Verbindungsaufbau

Die Kalender, ihre Einträge und die Erinnerungen werden vom Mobile-Device auf den PC übertragen (siehe Abbildung 3.2, Seite 11). Während dem Auslesen der Einträge aus der Datenbank auf dem Mobile-Device wird die höchste Id der Einträge und jene der Erinnerungen bestimmt. Die Kalender und Einträge werden in einer Map gespeichert, welche mitsamt den maximalen Id-Werten an den PC übermittelt wird. Jeder Eintrag enthält nebst Feldern wie Titel, Ort, Zeit etc. eine Liste mit den dazugehörigen Erinnerungen.

3.4.3 Synchronisationsalgorithmus

Entgegen den Voraussetzungen des Google Kalenders (für Details siehe Domainanalyse, Kapitel 5.1.1, Seite 27) gewinnt bei beidseitigen Änderungen standardmässig nicht die Version des PCs, sondern diejenige des Mobile-Devices. Würde die Strategie von Google in diesem Punkt übernommen werden, gäbe es einen «Lost Update». Änderungen auf dem Google Server, welche durch eine vorgängige Synchronisation auf dem Mobile-Device übernommen wurden, würden mit den Änderungen der PC-Seite überschrieben werden. Wie in Kapitel 3.4.1 beschrieben, kann diese Einstellung konfiguriert werden.

- Vorbereitung
  1. Listen: Es werden für die Einträge und Erinnerungen, welche zurück zum Mobile-Device geschickt werden müssen, eine add-, update- und delete-Liste erstellt.
  2. Offset: Um Kollisionen zu verhindern, müssen die Id's auf der PC-Seite um einen Offset-Wert verschoben werden. Dieser wird anhand der maximalen Id-Werten auf dem Mobile-Device und jenen zum Zeitpunkt der letzten Synchronisation berechnet.
- PC
  1. Neue Einträge und Erinnerungen: Alle Einträge und Erinnerungen auf der PC-Seite, welche eine Id höher als jene der letzten Synchronisation besitzen, erhalten eine neue Id. Diese wird um den Offset verschoben.
     Einträge und Erinnerungen ergänzen: Die aktualisierten Einträge und Erinnerungen der PC-Seite können nun in die entsprechende Liste kopiert werden. Diese werden nach Abschluss der Synchronisation auf das Mobile-Device übermittelt und der Datenbank hinzugefügt. Zudem wird der Eintrag im «Snapshot» gespeichert.
  2. Gelöschte Einträge und Erinnerungen: Alle Einträge und Erinnerungen, deren Id kleiner oder gleich jener der letzten Synchronisation ist und die nur noch auf der PC-Seite existieren, werden auf der PC-Seite gelöscht, sowie aus dem «Snapshot» entfernt.
  3. Geänderte Einträge und Erinnerungen: Für alle Einträge und Erinnerungen, deren Id kleiner oder gleich jener der letzten Synchronisation ist und die sowohl auf der PC-Seite als auch auf dem Mobile-Device existieren, muss der Hash-Wert des Eintrags berechnet werden. Dieser wird mit dem gespeicherten «Snapshot» der letzten Synchronisation verglichen wird. Für Details zum Snapshot, siehe Kapitel 3.3.1, Seite 9.
     a) Hash Mobile-Device und Hash PC-Seite entsprechen demjenigen des «Snapshots»: Nichts wurde verändert. Der Eintag muss im «Snapshot» markiert werden, damit er nicht gelöscht wird.
     b) Hash Mobile-Device entspricht Hash «Snapshot» und Hash PC-Seite hat geändert: PC-Seite wurde verändert.

i. PC-Eintrag der entsprechenden update-Liste hinzufügen und im «Snapshot» überschreiben.

ii. Falls Erinnerungen vorhanden sind, überprüfen, ob es gelöschte, neue oder geänderte Erinnerungen gibt und diese der entsprechenden Liste hinzufügen.

c) Hash Mobile-Device hat geändert und Hash PC-Seite entspricht demjenigen des «Snapshots»: Mobile-Device-Seite wurde verändert.

i. Eintrag auf PC-Seite aktualisieren und im «Snapshot» überschreiben.

d) Hash Mobile-Device und Hash PC-Seite haben geändert: Beide Seiten wurden verändert. Wurde das Mobile-Device als Gewinner definiert(siehe Kapitel 3.3.1, Seite 9), wird dieser Fall gleich behandelt, wie 3c, ansonsten wie 3b

- Mobile-Device
  1. Neue Einträge und Erinnerungen: Alle Einträge und Erinnerungen auf dem Mobile-Device, welche eine Id höher als jene der letzten Synchronisation haben, werden auf der PC-Seite hinzugefügt.
  2. Gelöschte Einträge und Erinnerungen: Alle Einträge und Erinnerungen, deren Id kleiner oder gleich jener der letzten Synchronisation ist und die nur noch auf dem Mobile-Device existieren, werden der entsprechenden delete-Liste hinzugefügt.

- Neue Kalender:
  1. Mobile-Kalender: Kalender, die auf der PC-Seite noch nicht vorhanden sind, werden hinzugefügt.
  2. PC-Kalender: Kalender, die auf dem Mobile-Device noch nicht vorhanden sind, können nicht hinzugefügt werden, da auf dem Mobile-Device keine neuen Kalender erstellt werden können. In diesem Fall muss ein entsprechender Kalender auf dem Google-Server erstellt werden, welcher, sobald er auf dem Mobile-Device synchronisiert wurde, mit den entsprechenden Werten gefüllt werden kann.

- Abschluss:
  1. Die Listen, welche nun die Einträge und Erinnerungen enthalten, welche auf dem Mobile-Device hinzugefügt, aktualisiert oder gelöscht werden müssen, werden an das Mobile-Device übermittelt, wo die Einträge und Erinnerungen entsprechend in der Datenbank bearbeitet werden (siehe Abbildung 3.2, Seite 11).
  2. Das Snapshot-XML wird gespeichert.

4 Telefon

4.1 Klassenstruktur

4.1.1 ch.hsr.ancof.telephony

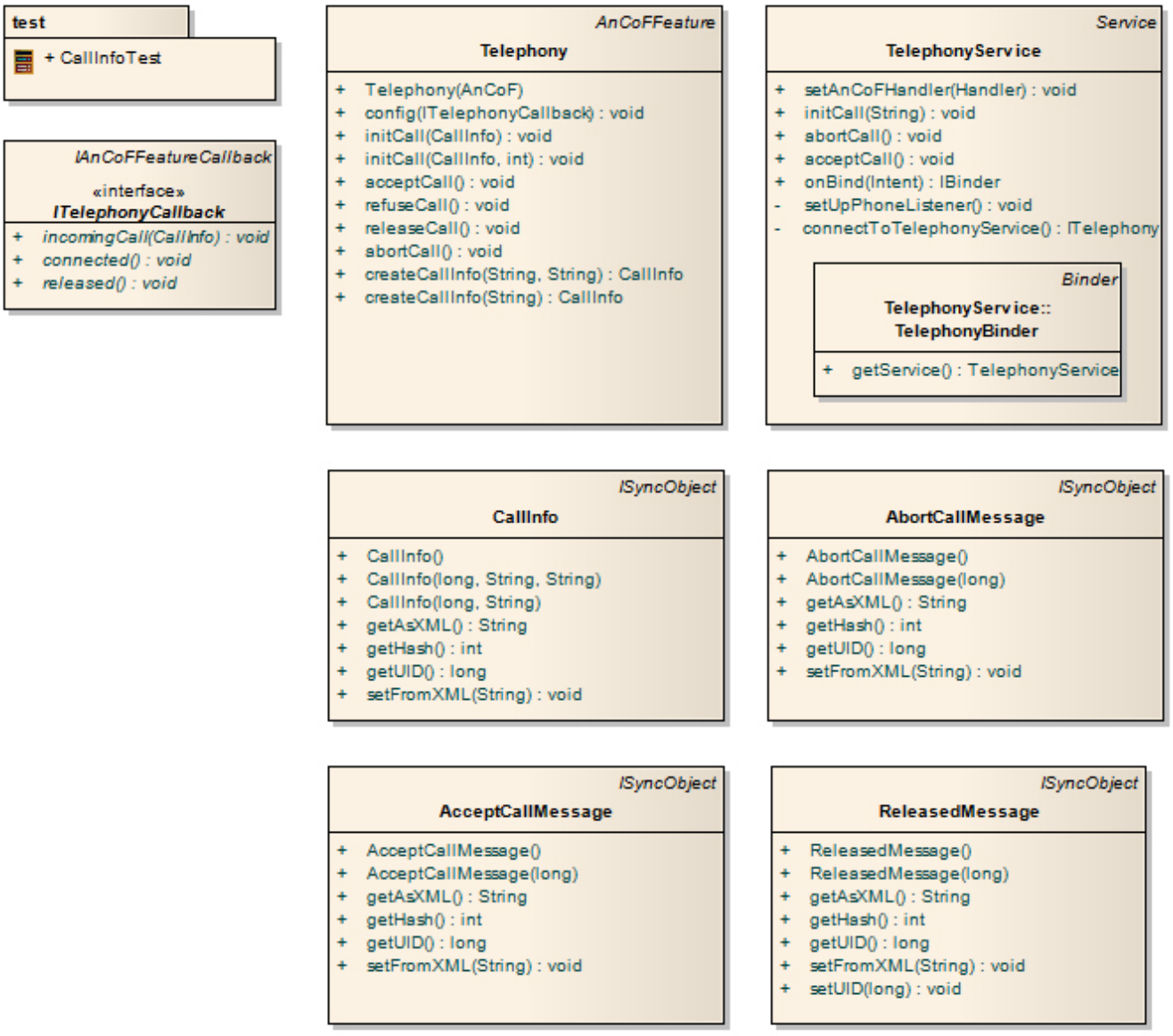


Abbildung 4.1: Package Telephony

ch.hsr.ancof.telephony.Telephony

Die Klasse «Telephony» enthält alle Funktionen um einen Telefonanruf zu tätigen bzw. anzunehmen.

Die Methoden `release-`, `refuse-` und `abortCall` werden exakt gleich implementiert. Deshalb wurde nur die Methode `releaseCall` implementiert. Die anderen Methoden wurden nur aus Gründen der Benutzerfreundlichkeit im Userinterface belassen.

ch.hsr.ancof.telephony.TelephonyService

Die Klasse «TelephonyService» führt die ankommenden Befehle auf dem Mobile-Device aus.

Während der Service startet, wird ein «PhoneStateListener» auf dem Mobile-Device registriert. Eingehende Anrufe können somit ermittelt und an den PC weitergeleitet werden.

Da der offizielle Android-Software Development Kit (SDK) das Entgegennehmen eines Anrufs bzw. das Auflegen des Höhrers nicht unterstützt, musste ein «Workaround» mit Java-Reflection und einem Android Interface Definition Language (AIDL)-Interface[1] gefunden werden.

ch.hsr.ancof.telephon.*Message

Die drei Message-Klassen sind ausschliesslich Nachrichten, die vom Mobile-Device an den PC bzw. in der anderen Richtung versendet werden.

[1] `http://developer.android.com/guide/developing/tools/aidl.html`

5 Einstellungen

Die Einstellungen beziehen sich nur auf Einstellungen, die auf dem Mobile-Device gemacht werden können. Einstellungen an AnCoF selbst können über die Hauptklasse «AnCoF» getätigt werden.

Zudem mussten bei diesem Bereich von AnCoF aus Zeitgründen Einschränkungen gemacht werden. Details können dem Kapitel 10.4, Seite 26 entnommen werden.

5.1 Klassenstruktur

5.1.1 ch.hsr.ancof.settings

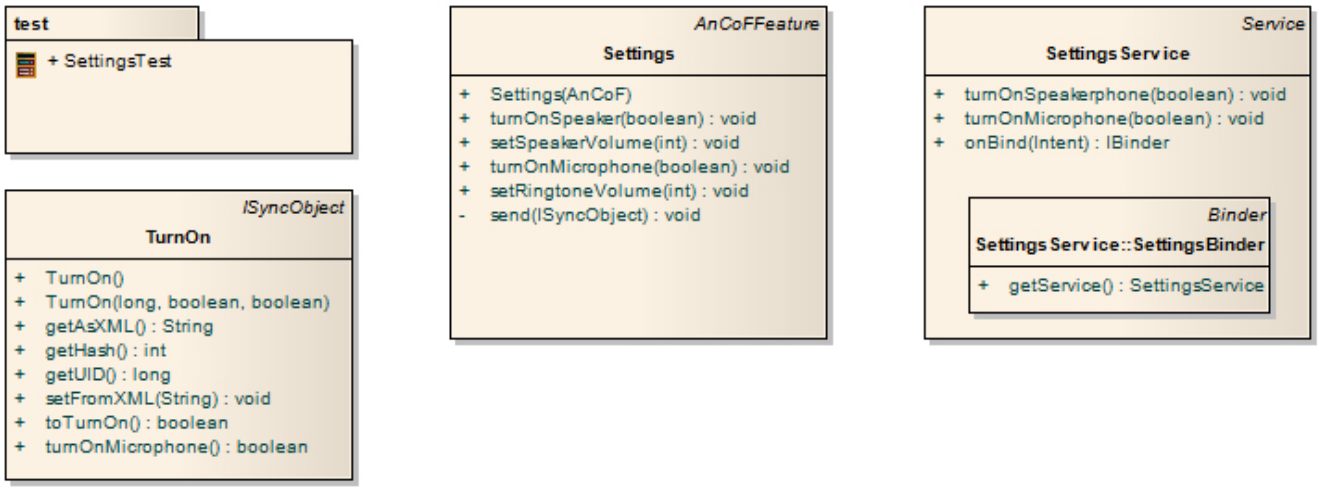


Abbildung 5.1: Package Settings

In diesem Package existiert kein Callback-Interface. Dies aus dem Grund, da Einstellungen am Mobile-Device nur vorgenommen werden und kein Rückwert liefern.

ch.hsr.ancof.settings.Settings

Die Klasse «Settings» bietet dem Benutzer Funktionen, um Telefoneinstellungen zu verändern. Die Klasse wirkt dabei als reine «Weiterleiterin» und schickt die gewünschte Funktion als Message an das Mobile-Device.

ch.hsr.ancof.settings.SettingsService

Diese Klasse verarbeitet die per Message eingegangenen Befehle und führt sie auf dem Mobile-Device aus.

ch.hsr.ancof.settings.TurnOn

«TurnOn» ist die Nachricht, welche an das Mobile-Device geschickt wird. Sie enthält Informationen darüber, welches von beiden Geräten (Lautsprecher oder Mikrofon) angesprochen und ob dieses ein- oder ausgeschaltet werden soll.

6 SMS

6.1 Klassenstruktur

6.1.1 ch.hsr.ancof.textMessage

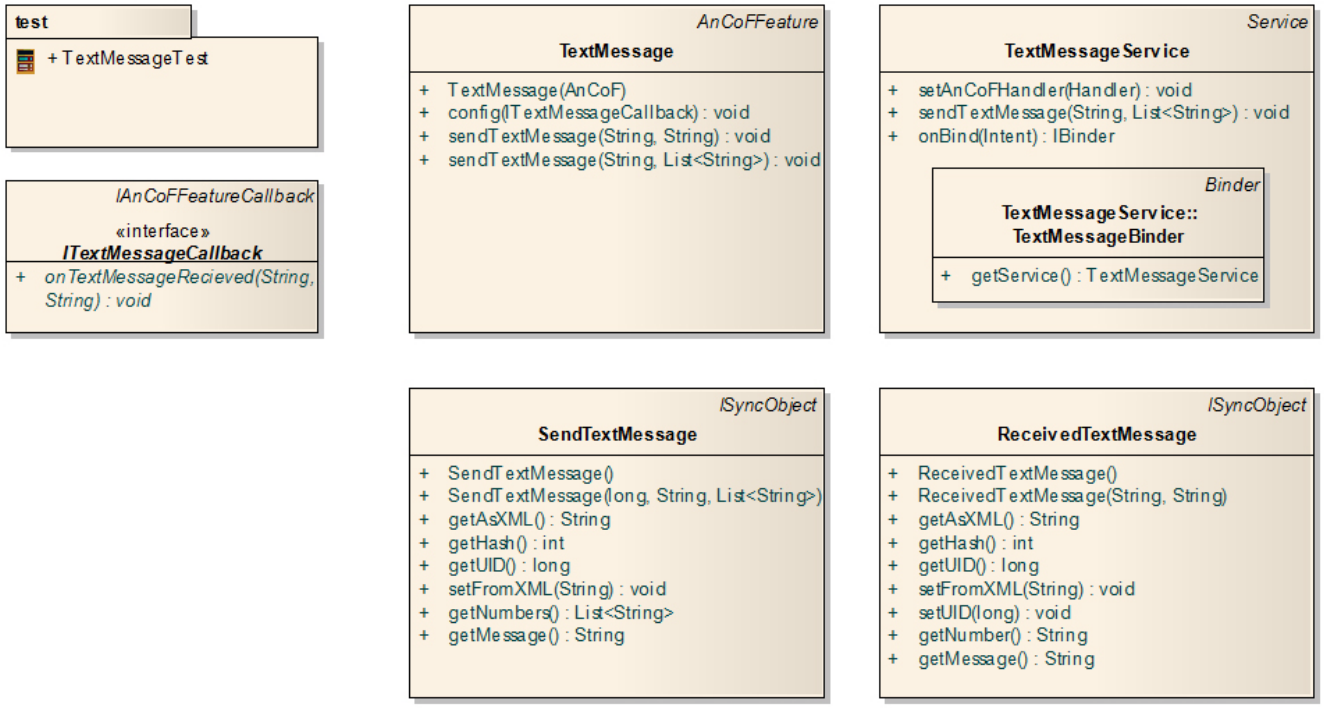


Abbildung 6.1: Package textMessage

ch.hsr.ancof.textMessage.TextMessage

Die Klasse «TextMessage» wird für das Versenden von SMS genutzt.

ch.hsr.ancof.textMessage.TextMessageService

Die Klasse «TextMessageService» versendet die Textnachrichten, indem sie die Strings, falls nötig, in einzelne Nachrichten trennt. Danach wird jede Nachricht an die entsprechenden Empfänger übermittelt.

Um Textnachrichten empfangen zu können, wird ein BroadcastReceiver implementiert und beim Betriebssystem registriert.

7 Daten

7.1 Klassenstruktur

7.1.1 ch.hsr.ancof.data

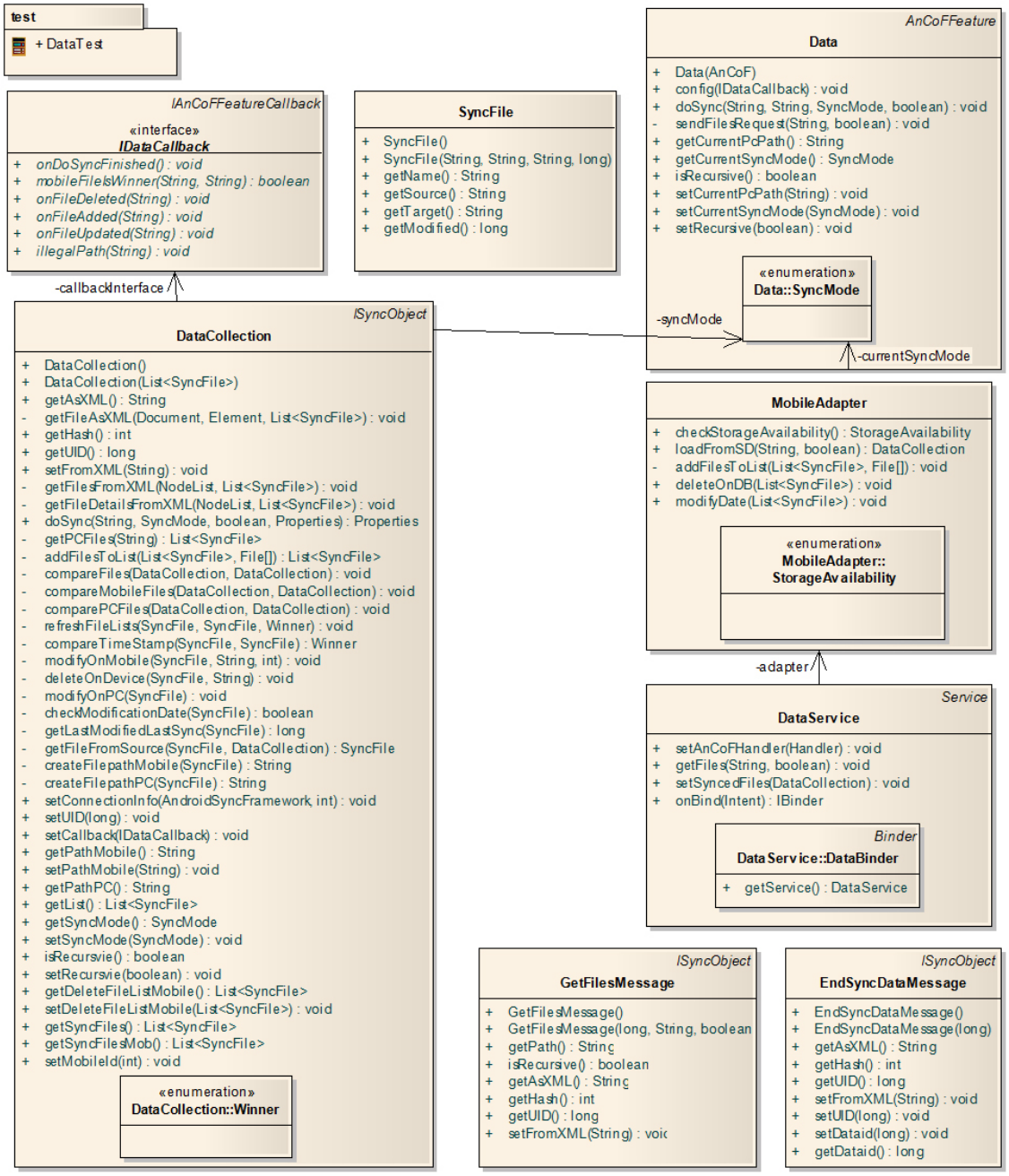


Abbildung 7.1: Package Data

Der Klassenaufbau der Datensynchronisation wurde analog derjenigen für die Kalendersynchronisation strukturiert(siehe Kapitel 3.3, Seite 8). Die Synchronisation findet ebenfalls in der Klasse «DataCollection» statt. Für Details zur Synchronisation siehe Kapitel 7.2. Gestartet wird der Synchronisationsprozess entsprechend in der Klasse «Data». Das Bindeglied zum Mobile-Device bildet hier die Klasse «MobileAdapter». Um eine Datei abzubilden, wurde entschieden eine eigene Klasse zu gestalten, die Klasse «SyncFile». Es hätte zwar die File-Klasse von Java genutzt werden können, da aber die dort gebotenen Funktionalitäten nicht ausgenutzt worden wären, wurde eine für AnCoF zugeschnittene Klasse gestaltet. Zur Sicherung des Synchronisationszustandes wird auf ein Property-File zurückgegriffen.

Property-File

Das Property-File enthält die Pfade und das Bearbeitungsdatum aller Dateien, die bis zu diesem Zeitpunkt jemals synchronisiert wurden. Es werden sowohl die Pfade der Mobile-Dateien als auch diejenigen der PC-Dateien gespeichert. Entgegen der Situation beim CalendarSnasphot (siehe Kapitel 3.3.1, Seite 9), wurde keine eigene Klasse für die Sicherung geschaffen. Das Laden und Speichern des Propertyfiles wurde in die Klasse «AnCoF» ausgelagert. Wäre das File direkt in der Klasse «DataCollection» geladen und gespeichert worden, wo es effektiv gebraucht wird, hätte dies zu Konflikten mit Android geführt, da der Pfad für die Speicherung respektive das Laden dort nicht existiert (Gespeichert wird das Property-File auf dem PC).

7.2 Synchronisation

7.2.1 File-System Android

Das File-System auf der Secure Digital Memory (SD)-Karte entspricht grundsätzlich demjenigen auf dem Betriebssystem Windows. Einziger Unterschied ist die Genauigkeit, mit welcher die Bearbeitungszeiten der Dateien bestimmt werden. Bei Android werden diese auf die Sekunde genau gerundet, bei Windows auf die Millisekunde genau. Für Details siehe «Domainanalyse», Kapitel 5.4.1, Seite 27. Diesem Umstand wird AnCoF folgendermassen gerecht: Die Werte der Dateien der PC-Seite werden für den Synchronisations-Vergleich entsprechend der Genauigkeit derjenigen des Mobile-Devices aufbereitet respektive mittels Modulo die «Millisekunden-Stellen» abgeschnitten.

7.2.2 Verbindungsaufbau

Die Pfade, Namen und Bearbeitungszeit der zu synchronisierenden Dateien werden übermittelt.

7.2.3 Synchronisationsalgorithmus

Der Synchronisationsmodus kann folgendermassen eingestellt werden:

- TIMESTAMP: Das Bearbeitungsdatum bestimmt die Synchronisation.
- ASKING: Bevor eine Datei bearbeitet wird, wird gefragt, wie verfahren werden soll.

Der hier beschriebene Fall verfährt nach dem Timestamp-Modus.

- Vorbereitung
  1. Folgende Listen werden erstellt:
     - syncPC-List: Um Dateien mittels TaMaF vom Mobile-Device auf den PC zu übermitteln.
     - delete-List: Um gelöschte Dateien auf dem Mobile-Device zu löschen.

  - syncMob-List: Um für Dateien, welche mittels TaMaF vom PC her übertragen wurden, das Bearbeitungsdatum entsprechend zu setzen, da TaMaF dies unterlässt.

2. Das aktuelle Property-File, welches Angaben zum Stand der letzten Synchronisation enthält, wird geladen. Für Details zum Property-File siehe Kapitel 7.1.1, Seite 20.

- PC
  1. Neue Dateien: Alle Dateien auf der PC-Seite, welche ein Bearbeitungsdatum höher als jenes der letzten Synchronisation haben und auf dem Mobile-Device nicht vorhanden sind, werden mittels TaMaF auf dem Mobile-Device hinzugefügt und in der syncMob-Liste für die Aktualisierung des Bearbeitungsdatums eingetragen. Sie werden ebenfalls im Property-File ergänzt.
  2. Gelöschte Dateien: Alle Dateien, deren Bearbeitungsdatum kleiner oder gleich jenem der letzten Synchronisation ist und die nur noch auf der PC-Seite existieren, werden auf der PC-Seite gelöscht und aus dem Property-File entfernt.
  3. Geänderte Dateien: Für alle Dateien, deren Bearbeitungsdatum kleiner oder gleich jenem der letzten Synchronisation ist und die sowohl auf der PC-Seite als auch auf dem Mobile-Device existieren, muss der Zeitstempel verglichen werden.
     a) Der Zeitstempel der Datei der PC-Seite ist grösser als jener der Datei des Mobile-Devices: Die PC-Datei gewinnt. Die Datei wird mittels TaMaF auf dem Mobile-Device aktualisiert und für die Aktualisierung des Bearbeitungsdatums in der syncMob-Liste eingetragen. Das Bearbeitungsdatum wird im Property-File ebenfalls aktualisiert.
     b) Der Zeitstempel der Datei der PC-Seite ist kleiner als jener der Datei des Mobile-Devices: Die Mobile-Device-Datei gewinnt. Die Datei wird in der syncPC-Liste für die Übertragung an den PC eingetragen und das Bearbeitungsdatum im Property-File aktualisiert.
     c) Beide Zeitstempel sind genau gleich:
        i. Beide Zeitstempel entsprechen demjenigen der letzten Synchronisation: Keine der Dateien wurde verändert.
        ii. Die Zeitstempel entsprechen nicht demjenigen der letzten Synchronisation: Der vorher definierte Gewinner gewinnt. Ist der Gewinner die PC-Seite, wird wie bei 3a vorgegangen, ansonsten wie bei 3b.

- Mobile-Device
  1. Neue Dateien: Alle Dateien auf dem Mobile-Device, welche ein Bearbeitungsdatum höher als jenes der letzten Synchronisation haben und auf der PC-Seite nicht vorhanden sind, werden in der syncPC-Liste für die Übertragung an den PC eingetragen und im Property-File ergänzt.
  2. Gelöschte Dateien: Alle Dateien, deren Bearbeitungsdatum kleiner oder gleich jenem der letzten Synchronisation ist und die nur noch auf Mobile-Device existieren, werden der delete-Liste hinzugefügt und im Property-File gelöscht.

- Abschluss:
  1. Das aktuelle Property-File wird auf der PC-Seite gespeichert.
  2. Folgende Listen werden an das Mobile-Device übermittelt: Eine Listen von Dateien, welche auf dem Mobile-Device gelöscht werden müssen oder deren Bearbeitungsdatum aktualisiert werden muss und die Liste, welche die Dateien enthält, die mittels TaMaF auf der PC-Seite hinzugefügt werden müssen.Dort werden die Dateien entsprechend bearbeitet oder verschickt. Sobald die Dateien auf dem PC hinzugefügt wurden, können auch deren Bearbeitungsdaten aktualisiert werden.

8 Kontakte

8.1 Klassenstruktur

8.1.1 ch.hsr.ancof.contact

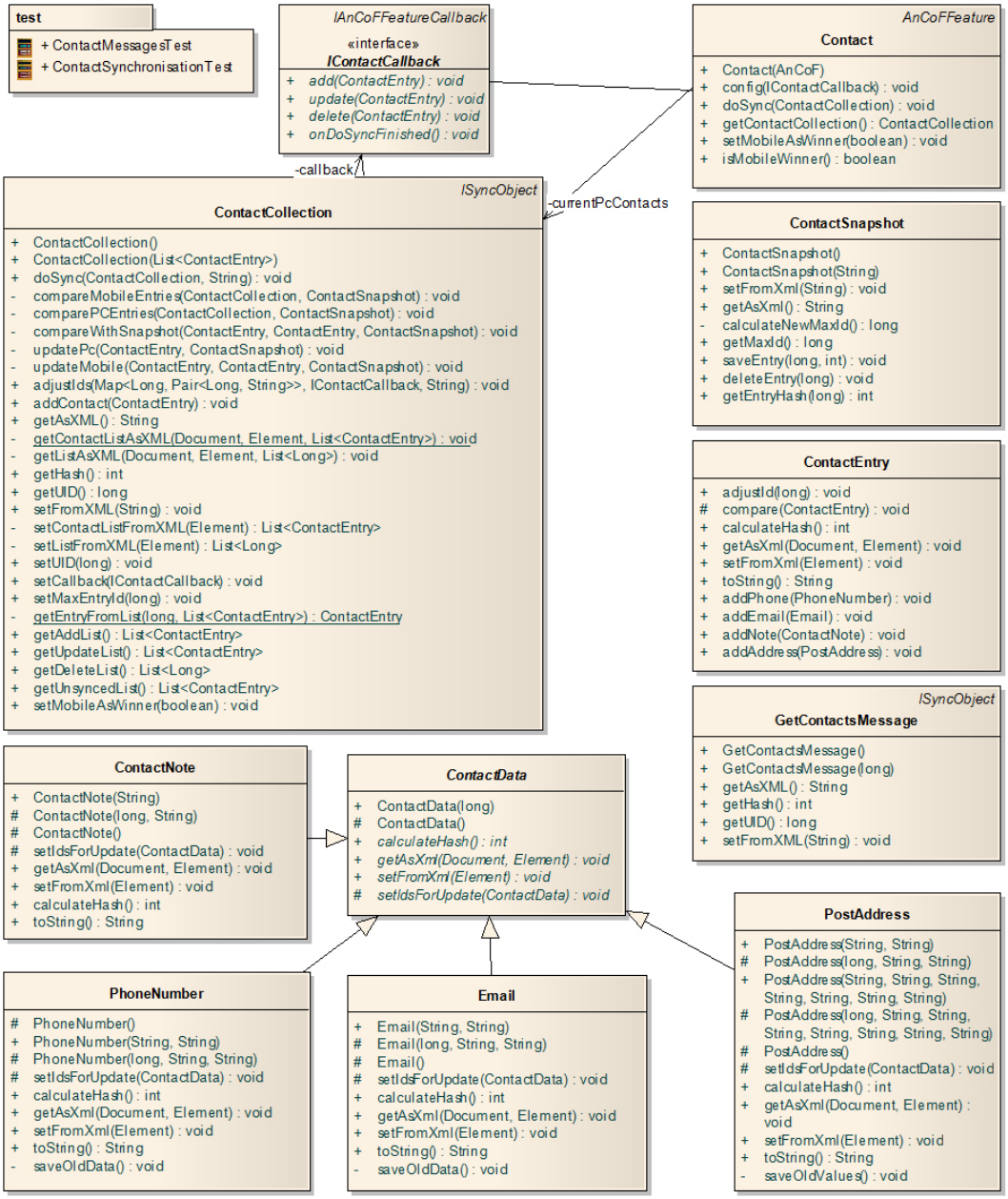


Abbildung 8.1: Package Contact

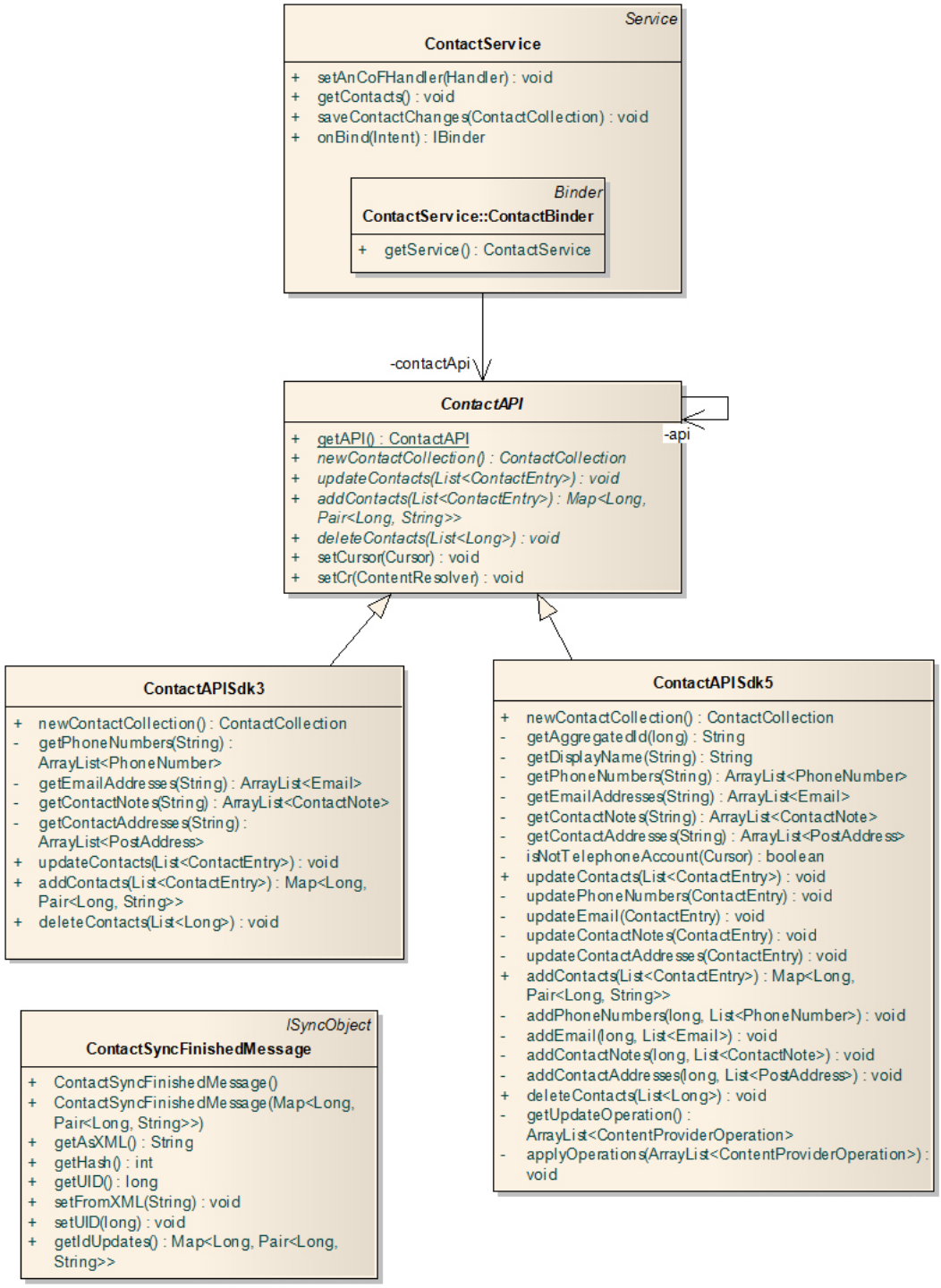


Abbildung 8.2: Package Contact

Der Aufbau der Klassen wurde analog derjenigen der Kalendersynchronisation (siehe Kapitel 3.3, Seite 8) strukturiert. Auch hier findet die Synchronisation in der Klasse «ContactCollection» statt.

8.2 Datenstruktur

Auch die Datenstruktur gleicht derjenigen der Kalendersynchronastion. Genauere Angaben zu den unterstützen Feldern können im «Developer Guide» im Kapitel 8.1.2, Seite 11 nachgelesen werden.

Da die Kalender-Application Programming Interface (API) von Android drei-stufig aufgebaut ist, musste entschieden werden, welche Art von Kontakten synchronisiert wird. Der Entscheid fiel auf die sogenannten «RawContacts» mit einem Vermerk, zu welchem «Contact» sie gehören. Dies deshalb, weil verschiedene «RawContacts» zu einem «Contact» verbunden werden können, wenn bestimmte Bedingungen[1] erfüllt sind. Unter diesen Bedingungen müsste nach der Synchronisation nochmals der gesamte Datenbankinhalt gelesen und analysiert werden.

8.3 Zugriff auf die Datenbank

Wie bereits im Kapitel 5.3, Seite 27 im Dokument «Domainanalyse» erwähnt, hat sich der Aufbau der Kontakt API sehr verändert. Um sowohl die Versionen vor als auch nach Android 2.0 zu unterstützen, muss der Zugriff auf die Datenbank doppelt implementiert werden.

Aus diesem Grund wird zur Laufzeit in der Klasse «ContactAPI» die Version des SDK überprüft und je nach Ergebnis, die Klasse «ContactAPISdk3» oder «ContactAPISdk5» als API-Instanz zurückgegeben.

8.4 Synchronisation

Der Synchronisationsalgorithmus ist im Grundaufbau derselbe, wie derjenige der Kalendersynchronisation. Deshalb wird an dieser Stelle auf das Kapitel 3.4.3, Seite 12 verwiesen. Der Abschnitt über «Neue Kalender» muss nicht beachtet werden.

8.5 Android Version 1.6 und 2.0

Für Entwicklungszwecke wird empfohlen mit der Android-Library der Version 2.0 zu arbeiten. Sie enthält sowohl die alten Funktionen, als auch die neuen. Für Testzwecke muss die Library unter Umständen wie im Kapitel 2.4, Seite 6 erwähnt, ausgetauscht werden.

[1] `http://developer.android.com/resources/articles/contacts.html` im Abschnitt «Automatic aggregation»

9 Util

9.1 Klassenstruktur

9.1.1 ch.hsr.ancof.util

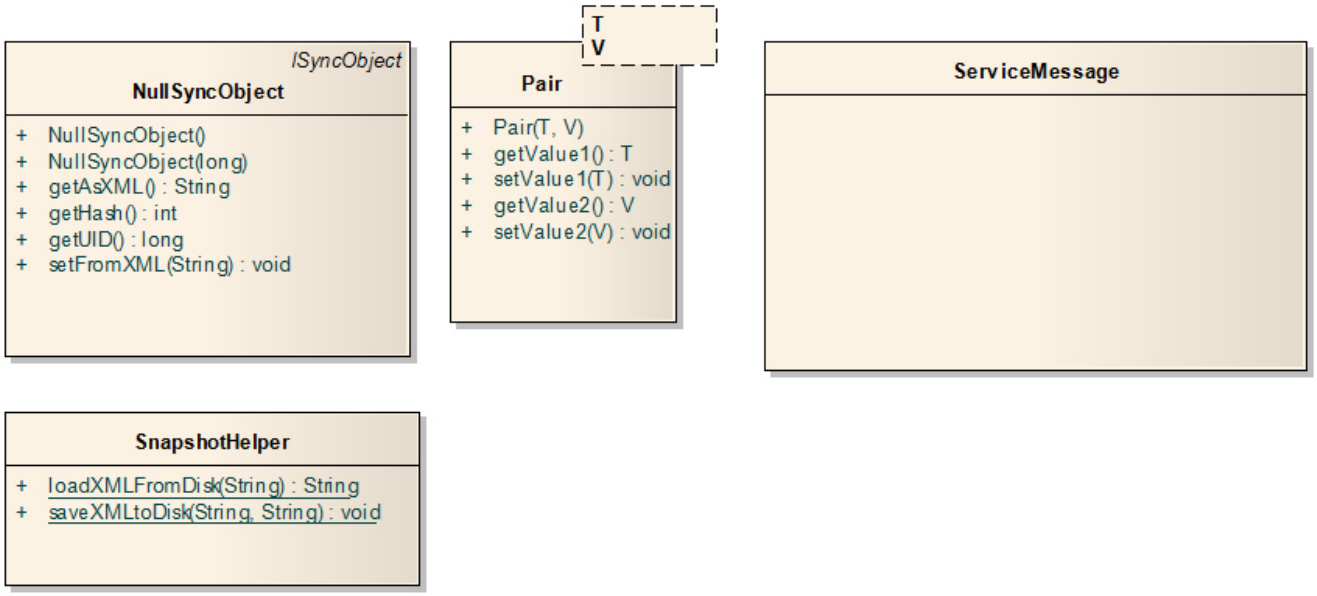


Abbildung 9.1: Package Util

ch.hsr.ancof.util.NullSyncObject

Das «NullSyncObject» wurde eingeführt, weil in AnCoF für alle Bereiche die ausgetauschten Nachrichten nicht persistiert werden und diese daher nach einem erneuten Start von AnCoF nicht mehr vorhanden sind. Da das «ASF» sich die ausgetauschten Objekte merkt, um sie unter Umständen auch mit weiteren Geräten synchronisieren zu können, musste ein leeres Objekt eingeführt werden.

ch.hsr.ancof.util.SnapshotHelper

Die Klasse «SnapshotHelper» dient dazu, die als Extensible Markup Language (XML) abgespeicherten Snapshots in einen String zu laden bzw. den String als Datei auf der Festplatte zu speichern.

ch.hsr.ancof.util.ServiceMessage

Diese Klasse enthält nur statische Meldungsnummern, die für den Betreff, der unter den Services ausgetauschten Meldungen, genutzt werden. Sie wurde geschaffen, um die Nummern der Betreffzeilen sortiert zu halten und um keine Zahl versehentlich doppelt zu belegen.

10 Erweiterungen und Einschränkungen

Zu den Erweiterungen und Einschränkungen sollten auch die «TODOs» im Code beachtet werden.

10.1 AnCoF

Bei der Verbindung zwischen Mobile-Device und PC wurde ersteres als Server bestimmt. Eine mögliche Erweiterung wäre, dies aufzubrechen und für den Benutzer konfigurierbar zu machen.

Im Moment laufen alle Services in eigenen Threads, welche sich aber im gleichen Prozess befinden müssen. Um dem Benutzer mehr Freiheiten zu ermöglichen könnte dieses Problem mit AIDL-Interfaces gelöst werden, so dass in Zukunft die Services auch in eigenen Prozessen laufen können.

Eine momentane Einschränkung ist zudem, dass ein Gerät nicht mit einem Namen identifiziert werden.

10.2 Kalender

In der produktiven Version von AnCoF werden im Moment noch keine Umlaute und sonstige nicht-ASCII-Zeichen unterstützt. Dieses Problem besteht, weil vom übertragenen XML die Informationen nicht mehr korrekt zurückgewonnen werden können.

10.3 Telefon

Die ursprünglich vorgesehene Funktion, um eine zustande gekommene Verbindung auf dem PC anzuzeigen, wurde aus Zeitgründen weggelassen und stellt nun eine mögliche Erweiterung des Telefon Bereichs von AnCoF dar.

Zudem kann geprüft werden, ob das wechseln zwischen zwei Telefonanrufen in Zukunft möglich ist.

10.4 Einstellungen

Bei den Einstellungen sind im Moment, nur die Methoden, um den Lautsprecher ein- bzw. auszuschalten, implementiert. Zudem besteht ein Anfang für das Ein-/Ausschalten des Mikrofons. Die anderen Funktionen sind nur als Grundgerüst vorhanden. Dies geschah, wie im Kapitel erwähnt, aus Zeitgründen.

Zudem können hier beliebig viele weitere Telefoneinstellungen hinzugefügt werden, wie z.B. das Verändern oder Aktivieren des Vibrationsalarms.

10.5 SMS

Auch bei den SMS besteht das Problem der Umlaute, welche nicht richtig übertragen werden. Deshalb werden im Moment nur ASCII-Zeichen in den Nachrichtentexten unterstützt. Eine mögliche Erweiterung des Frameworks auf UTF-8 Unterstützung wäre wünschenswert.

10.6 Daten

Für eine Weiterentwicklung von AnCoF ist geplant, dass zusätzlich entschieden werden kann, ob rekursiv synchronisiert wird oder nicht.

10.7 Kontakt

Im Moment können Daten (z.B. Telefonnummern) wenn ein Kontakt bereits synchronisiert wurde, nur verändert, nicht aber hinzugefügt oder gelöscht werden. Die Grundstruktur für diese Erweiterung wurde bereits erzeugt, indem die Daten, wenn sie erzeugt oder gelöscht werden, eine bestimmte negative Id erhalten. Diese kann beim Zurückspeichern in die Datenbank bzw. beim Vergleichen der Kontakte überprüft und entsprechend behandelt werden.

Zudem sollte der Display Name eines Kontaktes geändert werden können. Dies kann z.B. dadurch ermöglicht werden, dass der «RawContact» nach dem Speichern nochmals die Id des «Contacts», zu dem er gehört, überprüft.

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	29.09.2010	dm	Dokument erstellt.
1.0rc02	20.11.2010	dm	Kapitel Kalender erstellt.
1.0rc03	12.12.2010	dm	Kapitel Kalender angepasst und Kapitel Daten erstellt.
1.0rc04	16.12.2010	dm	Kapitel Kalender, Daten überarbeitet.
1.0rc05	17.12.2010	dm	Bilder hinzugefügt
1.0rc06	18.12.2010	rr	Kapitel AnCoF geschrieben.
1.0rc07	18.12.2010	rr	Kapitel Telefon geschrieben.
1.0rc08	18.12.2010	rr	Kapitel Einstellungen geschrieben.
1.0rc09	18.12.2010	rr	Kapitel Kontakte, SMS und util geschrieben.
1.0rc10	18.12.2010	rr	Kapitel Erweiterungen geschrieben.
1.0rc11	20.12.2010	rr	Rechtschreibung und Layout angepasst.
1.0	20.12.2010	dm	Akzeptierte Version 1.0

Tabelle 10.1: Revisionshistorie

Abkürzungsverzeichnis

AIDL Android Interface Definition Language
AnCoF Android Control Framework
API Application Programming Interface
ASCII American Standard Code for Information Interchange
ASF AndroidSyncFrameword
PC Personal Computer
SA Studienarbeit
SD Secure Digital Memory
SDK Software Development Kit
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone
UTF UCS Transformation Format
XML Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Testdokumentation
«Android Control Framework»

Version 1.2

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 7.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 Systemtest

2.1 Voraussetzung

Der Benutzer verfügt über ein Mobile-Device mit dem Betriebssystem Android der minimalen Version 1.6 bzw. 2.0 für die Kontaktsynchronisation und einen Personal Computer (PC) mit einer Java Installation der minimalen Version 1.6. Um eine Verbindung zwischen dem Mobile-Device und dem PC aufzubauen, wird ein USB Kabel benötigt. Zudem wird davon ausgegangen, dass die Tests mit den beiden zur Verfügung gestellten Testprojekten durchgeführt werden.

2.2 Vorbereitung

Bevor getestet werden kann, muss die Datei «AnCoF.apk» auf dem Mobile-Device installiert werden. Um eine Verbindung zwischen dem Mobile-Device und dem PC herzustellen, das USB-Kabel beim PC und beim Mobile-Device einstecken.

Die Installation wird folgendermassen vorgenommen: Auf der Konsole in das Verzeichnis, in dem «AnCoF.apk» gespeichert ist, wechseln und den Befehl `adb install AnCoF.apk` eingeben. Ist die Installation erfolgreich abgeschlossen, die Applikation auf dem Mobile-Device vom Menu aus starten.

Danach in das Verzeichnis wechseln, indem sich «AnCoFPC.jar» befindet und die PC-Applikation mit dem Konsolen-Befehl `java -jar AnCoFPC.jar` starten.

2.3 Speicherort Snapshotfiles

Die Snapshotfiles, welche für die Synchronisation benötigt werden, werden in folgendem Ordner gespeichert:

- Windows XP: C:\Dokumente und Einstellungen\<Benutzername>\Anwendungsdaten\AnCoF
- Windows Vista, Windows 7: C:\Users\<Benutzername>\AppData\Roaming\AnCoF

Die Ordner «Anwendungsdaten» und «AppData» sind versteckte Ordner, diese müssen erst sichtbar gemacht werden.

2.4 Ablauf

Alle im Dokument Anforderungsspezifikation (As) beschriebenen Use Cases (UCs) werden getestet und die Resultate bzw. Abläufe festgehalten.

2.5 Systemtest

2.5.1 Kalendersynchronisation

Name	**UC01: Neuer Eintrag synchronisieren**
Beschreibung	Es werden neue Einträge auf dem Mobile-Device und dem PC erstellt.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die neuen Einträge jeweils ergänzt.
Voraussetzung	Auf dem Mobile-Device muss ein neuer Eintrag erstellt werden. Das Snapshotfile darf nicht vorhanden sein.
Testdaten	Eintrag des Mobile-Devices und drei Standardeinträge des PC-Testprojekts.
Befehl	`sync calendar`
Resultat	Auf dem Mobile wurden drei Einträge, auf dem PC einer hinzugefügt.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.1: Systemtest UC01

Name	**UC02: Veränderter Eintrag synchronisieren**
Beschreibung	Es wird je ein vorhandener Eintrag auf dem Mobile-Device und dem PC verändert.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die veränderten Einträge jeweils aktualisiert.
Voraussetzung	Es existiert ein synchronisierter Zustand. Auf dem Mobile-Device wurde ein Eintrag verändert.
Testdaten	Veränderte Einträge.
Befehl	`change calendarEntry <description>`, danach `sync calendar`.
Resultat	Die veränderten Einträge wurden erkannt und aktualisiert.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.2: Systemtest UC02

Name	**UC03: Gelöschter Eintrag synchronisieren**
Beschreibung	Es wird je ein vorhandener Eintrag auf dem Mobile-Device und dem PC gelöscht.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die gelöschten Einträge jeweils entfernt.
Voraussetzung	Es existiert ein synchronisierter Zustand. Auf dem Mobile-Device wurde ein Eintrag gelöscht.
Testdaten	Gelöschte Einträge.
Befehl	`delete calendarEntry`, danach `sync calendar`.
Resultat	Die Einträge wurden als gelöscht erkannt und entfernt.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.3: Systemtest UC03

2.5.2 Telefonieren

Name	**UC04: Anruf tätigen**
Beschreibung	Es wird ein Anruf vom PC her aufgebaut.
Erwartetes Resultat	Auf dem angerufenen Telefon klingelt es.
Voraussetzung	Das angerufene Telefon ist eingeschaltet.
Testdaten	–
Befehl	`call <phonenumber>`
Resultat	Ein Anruf geht auf dem angerufenen Telefon ein.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.4: Systemtest UC04

Name	**UC05: Anruf erhalten**
Beschreibung	Es wird ein Anruf vom PC her entgegengenommen.
Erwartetes Resultat	Auf dem Mobile-Device wird ein «aktiver» Anruf angezeigt.
Voraussetzung	Ein nicht entgegen genommener Anruf ist vorhanden.
Testdaten	–
Befehl	`accept call`
Resultat	Verbindung wurde aufgebaut.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.5: Systemtest UC05

Name	**UC06: Anruf während Verbindung erhalten**
Beschreibung	Während einer aktiven Verbindung wird ein Anruf erhalten.
Erwartetes Resultat	Ein eingehender Anruf wir auf dem PC signalisiert.
Voraussetzung	Ein nicht entgegen genommener Anruf ist vorhanden.
Testdaten	–
Befehl	–
Resultat	«Incoming call with number: xxx» wird auf dem PC angezeigt.
Kommentar	Wird der Anruf akzeptiert, wird der erste Anruf auf «halten» gesetzt. Es kann nicht zwischen den beiden Anrufen gewechselt werden. Sobald der zweite Anruf aufgelegt wird, ist der ursprüngliche Anruf wieder aktiv.
Testdatum	30.11.2010

Tabelle 2.6: Systemtest UC06

Name	**UC07: Anruf ablehnen**
Beschreibung	Ein Anruf wird vom PC her abgelehnt.
Erwartetes Resultat	Auf dem Mobile-Device wird der eingehende Anruf abgelehnt.
Voraussetzung	Ein nicht entgegen genommener Anruf ist vorhanden.
Testdaten	–
Befehl	`refuse call`
Resultat	Der Anruf wird abgelehnt.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.7: Systemtest UC07

Name	**UC08: Anruf beenden**
Beschreibung	Ein aktiver Anruf wird vom PC her getrennt.
Erwartetes Resultat	Auf dem Mobile-Device wird die aktive Verbindung getrennt.
Voraussetzung	Eine aktive Verbindung ist vorhanden.
Testdaten	–
Befehl	`release call`
Resultat	Auf dem PC wird «Current call was released» angezeigt.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.8: Systemtest UC08

Name	**UC09: Anruf wird beendet**
Beschreibung	Ein Anruf wird vom anderen Gesprächsteilnehmer beendet.
Erwartetes Resultat	Auf dem PC wird eine Benachrichtigung angezeigt.
Voraussetzung	Eine aktive Verbindung ist vorhanden.
Testdaten	–
Befehl	–
Resultat	Auf dem PC wird «Current call was released» angezeigt.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.9: Systemtest UC09

2.5.3 Telefoneinstellungen verändern

Name	**UC10: Freisprechen ein- oder ausschalten**
Beschreibung	Die Freisprechanlage wird vom PC her ein- respektive ausgeschaltet.
Erwartetes Resultat	Auf dem Mobile-Device wird die Freisprechanlage entsprechend ein- oder ausgeschaltet.
Voraussetzung	Eine aktive Verbindung ist vorhanden.
Testdaten	–
Befehl	`turn on speaker` respektive `turn off speaker`.
Resultat	Die Freisprechanlage wurde ein- respektive ausgeschaltet.
Kommentar	–
Testdatum	01.12.2010

Tabelle 2.10: Systemtest UC10

Name	**UC11: Lautstärke verändern**
Beschreibung	Die Gesprächslautstärke wird vom PC her verändert.
Erwartetes Resultat	Die Gesprächslautstärke wird auf dem Mobile-Device entsprechend angepasst.
Voraussetzung	Eine aktive Verbindung ist vorhanden.
Testdaten	–
Befehl	–
Resultat	–
Kommentar	Noch nicht implementiert.
Testdatum	–

Tabelle 2.11: Systemtest UC11

Name	**UC12: Mikrofon ein- oder ausschalten**
Beschreibung	Das Mikrofon wird vom PC her ein- respektive ausgeschaltet.
Erwartetes Resultat	Auf dem Mobile-Device wird das Mikrofon entsprechend ein- oder ausgeschaltet.
Voraussetzung	Eine aktive Verbindung ist vorhanden.
Testdaten	–
Befehl	–
Resultat	–
Kommentar	Noch nicht implementiert.
Testdatum	–

Tabelle 2.12: Systemtest UC12

Name	**UC13: Klingeltonlautstärke verändern**
Beschreibung	Die Klingeltonlautstärke wird vom PC her verändert.
Erwartetes Resultat	Auf dem Mobile-Device wird die Klingeltonlautstärke entsprechend angepasst.
Voraussetzung	Ein eingehender Anruf ist vorhanden.
Testdaten	–
Befehl	–
Resultat	–
Kommentar	Noch nicht implementiert.
Testdatum	–

Tabelle 2.13: Systemtest UC13

2.5.4 SMS

Name	**UC14: SMS versenden**
Beschreibung	Es wird eine SMS vom PC her versendet.
Erwartetes Resultat	Der Empfänger erhält eine SMS.
Voraussetzung	–
Testdaten	–
Befehl	`send textMessage <phonenumber> <message>`
Resultat	Die erstellte Nachricht ist bei <phonenumber> angekommen.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.14: Systemtest UC14

Name	**UC15: SMS erhalten**
Beschreibung	Eine SMS wird auf dem PC empfangen.
Erwartetes Resultat	Die SMS ist auf dem PC angekommen.
Voraussetzung	Eine SMS wurde von extern verschickt.
Testdaten	Empfangene SMS
Befehl	–
Resultat	Auf dem PC wird «New SMS from: xxx» gefolgt von Mitteilungsinhalt angezeigt.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.15: Systemtest UC15

Name	**UC16: SMS an mehrere Empfänger versenden**
Beschreibung	Es wird auf dem PC eine SMS erstellt, die zum gleichen Zeitpunkt an unterschiedliche Empfänger versendet wird.
Erwartetes Resultat	Die Empfänger erhalten eine SMS.
Voraussetzung	–
Testdaten	–
Befehl	`add number <phonenumber>` für jeden Empfänger, danach `send multiple textMessages <message>`.
Resultat	Die Nachrichten sind angekommen.
Kommentar	–
Testdatum	30.11.2010

Tabelle 2.16: Systemtest UC16

2.5.5 Datensynchronisation

Name	**UC17: Neue Daten synchronisieren**
Beschreibung	Es werden neue Dateien auf dem Mobile-Device und dem PC erstellt.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die neuen Dateien jeweils ergänzt.
Voraussetzung	Auf dem Mobile-Device muss eine neue Datei im Ordner «\AnCoFTestData» erstellt werden. Das Snapshotfile darf nicht vorhanden sein.
Testdaten	Drei Standard-Dateien im PC-Testprojekt und eine auf dem Mobile-Device.
Befehl	`sync data`
Resultat	Die Dateien wurden auf dem anderen Gerät hinzugefügt.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.17: Systemtest UC17

Name	**UC18: Bearbeitete Daten synchronisieren**
Beschreibung	Es wird je eine Datei auf dem Mobile-Device und dem PC bearbeitet.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die Dateien jeweils aktualisiert.
Voraussetzung	Auf dem Mobile-Device muss eine Datei bearbeitet werden.
Testdaten	Veränderte Dateien.
Befehl	`set mode <mode>` um den Synchronisationsmodus zu setzen, `change file`, danach `sync data`.
Resultat	Asking mode: Das Mobile-Device gewinnt standardmässig, die Änderungen auf dem PC wurden mit dem Stand auf dem Mobile-Device überschrieben. Timestamp mode: Die neuere Datei wurde jeweils übernommen.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.18: Systemtest UC18

Name	**UC19: Gelöschte Daten synchronisieren**
Beschreibung	Es wird je eine Datei auf dem Mobile-Device und dem PC gelöscht.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die gelöschten Dateien jeweils entfernt.
Voraussetzung	Auf dem Mobile-Device muss eine Datei gelöscht werden.
Testdaten	Gelöschte Files.
Befehl	`delete file`, danach `sync data`.
Resultat	Die Dateien wurden entsprechend entfernt.
Kommentar	–
Testdatum	15.12.2010

Tabelle 2.19: Systemtest UC19

2.5.6 Kontakte

Name	**UC20: Neuer Eintrag synchronisieren**
Beschreibung	Es werden neue Kontakte auf dem Mobile-Device und dem PC erstellt.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die neuen Kontakte jeweils ergänzt.
Voraussetzung	Auf dem Mobile-Device muss ein neuer Kontakt erstellt werden.
Testdaten	Erstellte Kontakte.
Befehl	`sync contacts`
Resultat	Die Kontakte wurden auf dem anderen Gerät hinzugefügt.
Kommentar	Läuft nur im Debug-Modus.
Testdatum	15.12.2010

Tabelle 2.20: Systemtest UC20

Name	**UC21: Veränderter Eintrag synchronisieren**
Beschreibung	Es wird je ein Kontakt auf dem Mobile-Device und dem PC verändert.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die Kontakte jeweils aktualisiert.
Voraussetzung	Auf dem Mobile-Device muss ein Kontakt verändert werden.
Testdaten	Veränderte Einträge.
Befehl	`change contact <number>`, danach `sync contacts`.
Resultat	Die Einträge wurden aktualisiert.
Kommentar	Läuft nur im Debug-Modus.
Testdatum	15.12.2010

Tabelle 2.21: Systemtest UC21

Name	**UC22: Gelöschter Eintrag synchronisieren**
Beschreibung	Es wird je ein Kontakt auf dem Mobile-Device und dem PC gelöscht.
Erwartetes Resultat	Auf dem Mobile-Device und dem PC wurden die gelöschten Kontakte jeweils entfernt.
Voraussetzung	Auf dem Mobile-Device muss ein Kontakt gelöscht werden.
Testdaten	Gelöschte Kontakte.
Befehl	`delete contact`, danach `sync contacts`.
Resultat	Die Kontakte wurden entsprechend gelöscht.
Kommentar	Läuft nur im Debug-Modus.
Testdatum	15.12.2010

Tabelle 2.22: Systemtest UC22

3 Unit Test

Dieses Kapitel dient der Übersicht über die getesteten Fälle. Details können der Javadoc der entsprechenden Klassen entnommen werden.

3.1 ch.hsr.ancof.calendar.test

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Neue, veränderte und gelöschte Kalendereinträge auf PC- sowie Mobile-Device-Seite. Erstellung des XML's für die Datenübermittlung sowie das Laden und Speichern des Snapshotfiles.

Tabelle 3.1: Unit Test Kalender

3.2 ch.hsr.ancof.telephony.test

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Erstellung des XML's für die Datenübermittlung.

Tabelle 3.2: Unit Test Telefonieren

3.3 ch.hsr.ancof.settings.test

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Erstellung des XML's für die Datenübermittlung.

Tabelle 3.3: Unit Test Einstellungen

3.4 ch.hsr.ancof.textMessage.test

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Erstellung des XML's für die Datenübermittlung.

Tabelle 3.4: Unit Test SMS

3.5 ch.hsr.ancof.data.test

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Erstellung des XML's für die Datenübermittlung.

Tabelle 3.5: Unit Test Daten

3.6 ch.hsr.ancof.contact.test.ContactSynchronisationTest

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Neue, veränderte und gelöschte Kontakte auf PC- sowie Mobile-Device-Seite. Erstellung des XML's für die Datenübermittlung sowie das Laden und Speichern des Snapshotfiles.

Tabelle 3.6: Unit Test Kontakte

3.7 ch.hsr.ancof.contact.test.ContactMessagesTest

Testdatum	Testbeschreibung
15.12.2010	Folgende Fälle werden mittels Unit Tests überprüft: Erstellung des XML's für `ContactSyncFinishedMessage` und `GetContactsMessage`.

Tabelle 3.7: Unit Test Kontakt-Messages

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	27.11.2010	dm	Dokument erstellt.
1.0rc02	01.12.2010	dm, rr	Systemtests durchgeführt.
1.0	15.12.2010	rr	Akzeptierte Version 1.0
1.1rc01	15.12.2010	dm, rr	Systemtests und Unit Tests durchgeführt.
1.1	19.12.2010	dm	Akzeptierte Version 1.1
1.2rc01	19.12.2010	rr	Rechtschreibung und Layoutkorrekturen.
1.2	20.12.2010	dm	Akzeptierte Version 1.2

Tabelle 3.8: Revisionshistorie

Abkürzungsverzeichnis

As Anforderungsspezifikation
PC Personal Computer
SA Studienarbeit
SMS Short Message Service
UC Use Case
USB Universal Serial Bus

Tabellenverzeichnis

Developer Guide
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck des Dokuments

Siehe «Projektplan», Kapitel 4.2.2, Seite 7.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 AnCoF

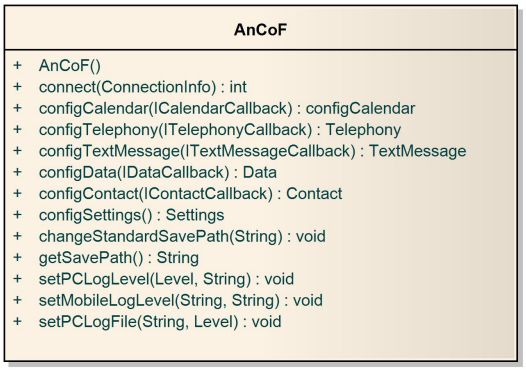


Abbildung 2.1: Benutzerinterface Kalendersynchronisation

2.1 PC

Auf dem PC kann mit `connect(...)` eine Verbindung aufgebaut werden. Der PC übernimmt dabei die Rolles des Clients und das Mobile-Device die Rolle des Servers. Die ConnectionInfo-Klasse wird in der Dokumentation des Task-Management-Framework on Smart-Phone (TaMaF) beschrieben.

2.1.1 Nutzen eines Bereichs von AnCoF

Um einen Bereich von AnCoF nutzen zu können, muss dieser zuerst konfiguriert werden. Erst danach kann die Feature-Klasse aktiv genutzt werden. Details zur Konfiguration können den jeweiligen Kapiteln entnommen werden.

2.1.2 Setzen des Snapshot-Speicherorts

Da in den drei Bereichen Kalender, Daten und Kontakte Snaphots gespeichert werden müssen, muss ein Speicherort dafür gesetzt sein.
Wird nichts geändert, ist dies «APPDATA + \AnCoF». Mit `setStandardSavePath(path)` kann dieser den eigenen Bedürfnissen angepasst werden. Zu Beachten ist, dass in diesem Fall der Ordner bereits existieren muss.

2.2 Mobile-Device

AnCoF arbeitet auf dem Mobile-Device mit Services. Diese Services müssen im Prozess der Applikation selbst laufen und müssen im Manifest bekannt gemacht werden. Details können den jeweiligen Kapiteln entnommen werden.

2.2.1 Starten von AnCoF

In einer Activity wird die Klasse AnCoFService durch einen Intent gestartet. Um AnCoF zu konfigurieren, muss die Portnummer für die Verbindung als «Extra» mit dem Namen «port» mitgegeben werden.

Zudem muss dem Intent bekannt gemacht werden, welche Features von AnCoF genutzt werden sollen. Dies geschieht, indem man mit `setData(...)` die gewünschten Features mitgibt. Die vordefinierten Strings können in der Klasse `AnCoFService` gefunden werden.

Kalendersynchronisation	AnCoFService.CALENDAR
Telefonie	AnCoFService.TELEPHONY
Telefoneinstellungen	AnCoFService.SETTINGS
SMS	AnCoFService.TEXT_MESSAGE
Datensynchronisation	AnCoFService.DATA
Kontaktsynchronisation	AnCoFService.CONTACT

Tabelle 2.1: Namen der AnCoF-Bereiche auf dem Mobile-Device

Beispiel

Dies ist ein Beispiel, welches AnCoF mit einer Verbindung auf Port 8080 startet. Zudem wurden die zwei Bereiche Telefonie und Kalendersynchronisation ausgewählt.

```
Intent ancofService = new Intent(this, AnCoFService.class);
ancofService.setData(Uri.parse(AnCoFService.TELEPHONY + AnCoFService.CALENDAR);
ancofService.putExtra("port", 8080);
startService(ancofService);
```

Zudem können die beiden Testprojekte von AnCoF zu Hilfe genommen werden.

2.2.2 Manifest Einträge

Auch AnCoF selbst ist ein Service welcher im Manifest eingetragen werden muss:

```
<service android:name="ch.hsr.ancof.AnCoFService"/>
```

Zudem müssen folgende Permissions gesetzt werden:

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

3 Kalender

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

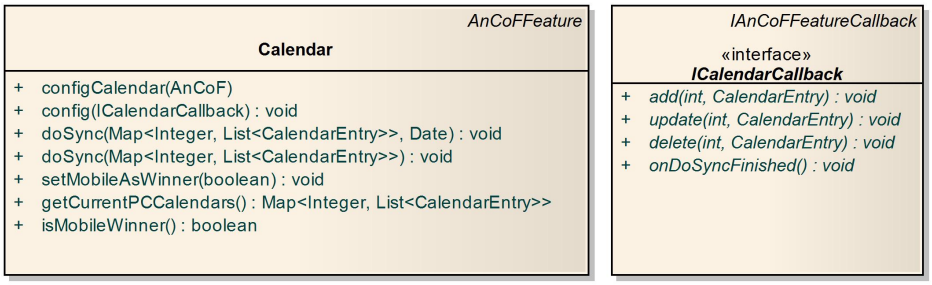


Abbildung 3.1: Benutzerinterface Kalendersynchronisation

3.1 PC

3.1.1 Konfigurieren der Kalenderklasse

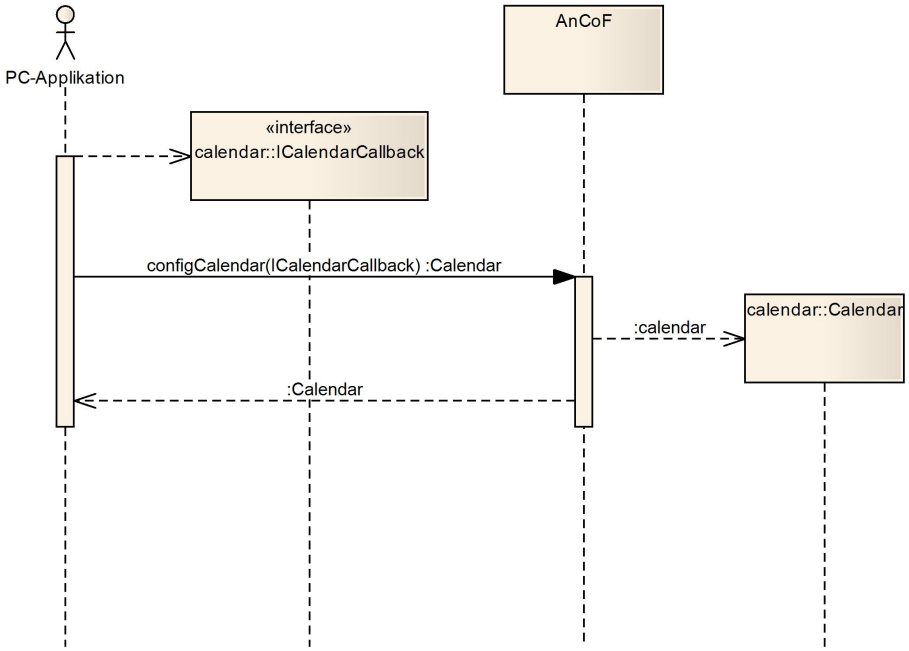


Abbildung 3.2: Konfiguration Kalenderklasse

Eine korrekt konfigurierte Instanz von Calendar bekommt man, wenn der Befehl `configCalendar(callback)` der Klasse AnCoF aufgerufen wird. callback ist dabei das implementierte ICalendarCallback-Interface. Für Details zum Konfigurationsablauf kann die Abbildung 3.2, Seite 4 zu Rate gezogen werden.

3.1.2 Datenstruktur

Gemäss «Design Dokument», Kapitel 3.3.1, Seite 8 benutzt AnCoF eigene Klassen, um die Kalender- und Erinnerungseinträge darzustellen. Daher müssen die zu vergleichenden Daten ebenfalls in dieser Form vorliegen.

- Kalender: Jeder Kalender wird über eine Kalendernummer, dargestellt als ein Integer-Wert, identifiziert.
- Kalendereintrag: Die Klasse «CalendarEntry» unterstützt folgende Felder:
  - `int eventID`
  - `String title`
  - `String description`
  - `String location`
  - `long start`
  - `long end`
  - `int allDay`
  - `int hasAlarm`
  - `String rrule`
  - `List<ReminderEntry> reminders`
- Erinnerung: Die Klasse «ReminderEntry» unterstützt folgende Felder:
  - `int minutes`
  - `int id`
  - `int eventId`

Details zu den einzelnen Feldern können in der Javadoc nachgelesen werden. Erlaubte Werte für das Feld «rrule» können aus dem Beispiel im Anhang entnommen werden.

3.1.3 Synchronisation

Um den Synchronisationsprozess zu starten, wird `doSync(pcCalendars, syncFrom)` der Klasse Calendar aufgerufen. pcCalendars ist dabei eine Map der Klasse «java.util.Map», welche als Key die Kalendernummer und als dazugehöriger Value jeweils eine Liste mit Kalendereinträgen enthält. Für Details zur Datenstruktur siehe 3.1.2. syncFrom bezeichnet das Datum, ab dem die Kalendereinträge verglichen werden und ist als Date der Klasse «java.util.Date» anzugeben. Details zum Synchronisationsalgorithmus können dem «Design Dokument», Kapitel 3.4.3, Seite 12 entnommen werden.

3.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.calendar.CalendarService"/>
```

3.2.1 Permissions

Die Klasse CalendarService benötigt folgende Permissions:

- android.permission.READ_CALENDAR
- android.permission.WRITE_CALENDAR

4 Telefon

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

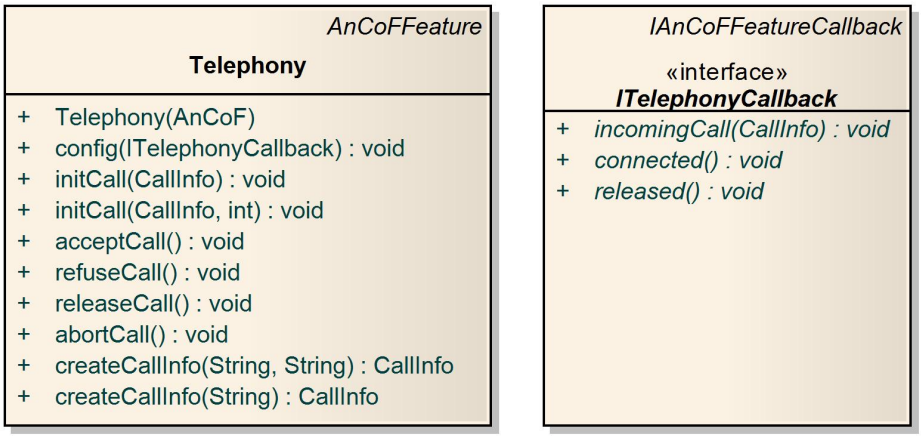


Abbildung 4.1: Benutzerinterface Telefon

4.1 PC

4.1.1 Konfigurieren der Telefonklasse

Die Konfiguration erfolgt gleich wie bei der Kalendersynchronisation und kann dem Kapitel 3.1.1, Seite 4 entnommen werden. Zu beachten ist, dass «Calendar» durch «Telephony» zu ersetzen ist.

Um einen Anruf tätigen zu können wird ein «CallInfo»-Objekt benötigt, welches am besten mit `createCallInfo` erstellt wird, da es dann bereits die für das Versenden wichtigen Informationen enthält.

4.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.telephony.TelephonyService"/>
```

4.2.1 Permissions

Die Klasse TelephonyService benötigt folgende Permissions:

- android.permission.READ_PHONE_STATE
- android.permission.MODIFY_PHONE_STATE
- android.permission.CALL_PHONE

5 Einstellungen

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

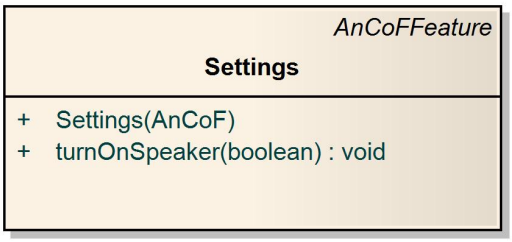


Abbildung 5.1: Benutzerinterface Einstellungen

5.1 PC

5.1.1 Konfigurieren der Settingsklasse

Eine korrekt konfigurierte Instanz von Settings bekommt man, wenn der Befehl `configSettings()` der Klasse «AnCoF» aufgerufen wird.

5.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.settings.SettingsService"/>
```

5.2.1 Permissions

Die Klasse SettingsService benötigt folgende Permissions:

- android.permission.MODIFY_AUDIO_SETTINGS
- android.permission.RECORD_AUDIO

6 SMS

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

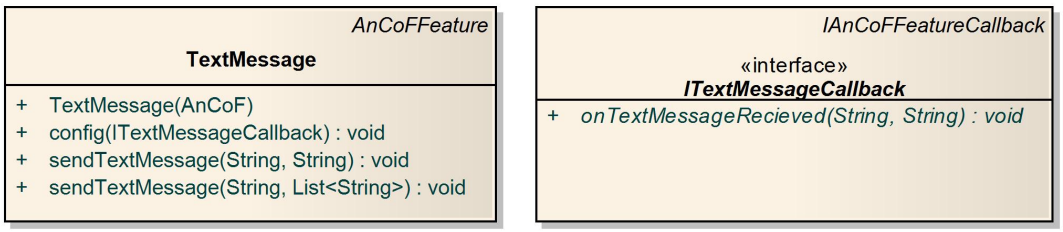


Abbildung 6.1: Benutzerinterface SMS

6.1 PC

6.1.1 Konfigurieren der SMS-Klasse

Die Konfiguration erfolgt gleich wie bei der Kalendersynchronisation und kann dem Kapitel 3.1.1, Seite 4 entnommen werden. Zu beachten ist, dass «Calendar» durch «TextMessage» zu ersetzen ist.

6.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.textMessage.TextMessageService"/>
```

6.2.1 Permissions

Die Klasse TextMessageService benötigt folgende Permissions:

- android.permission.SEND_SMS
- android.permission.RECEIVE_SMS

7 Data

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

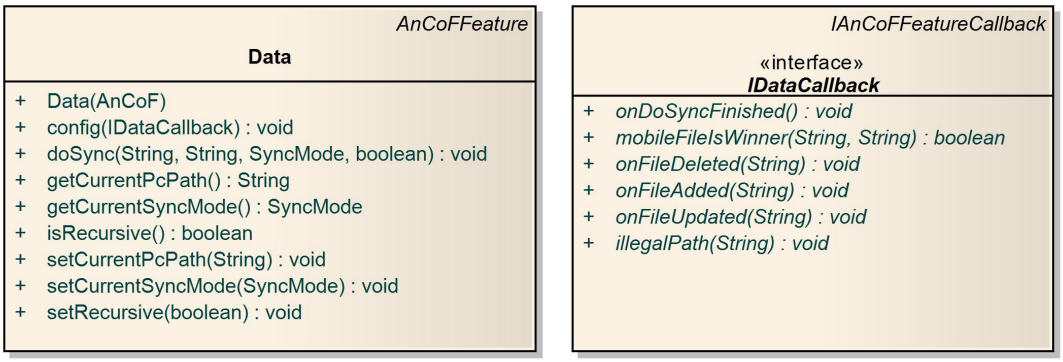


Abbildung 7.1: Benutzerinterface Daten

7.1 PC

7.1.1 Konfigurieren der Datenklasse

Die Konfiguration erfolgt gleich wie bei der Kalendersynchronisation und kann dem Kapitel 3.1.1, Seite 4 entnommen werden. Zu beachten ist, dass «Calendar» durch «Data» zu ersetzen ist.

7.1.2 Synchronisation

Um den Synchronisationsprozess zu starten, wird `doSync(pcPath, mobilePath, mode, recursive)` der Klasse Data aufgerufen. pcPath und mobilePath sind dabei die Pfade zu den zu synchronisierenden Ordnern als String gespeichert, mode ist der SyncMode (TIMESTAMP oder ASKING) und recursive ein boolean, der angibt, ob rekursiv oder nicht synchronisiert wird. Details zum Synchronisationsalgorithmus können dem «Design Dokument», Kapitel 7.2.3, Seite 20 entnommen werden.

7.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.data.DataService"/>
```

7.2.1 Permissions

Die Klasse DataService benötigt folgende Permission:

- android.permission.WRITE_EXTERNAL_STORAGE

8 Kontakte

Details zu den einzelnen Funktionen kann der Javadoc entnommen werden.

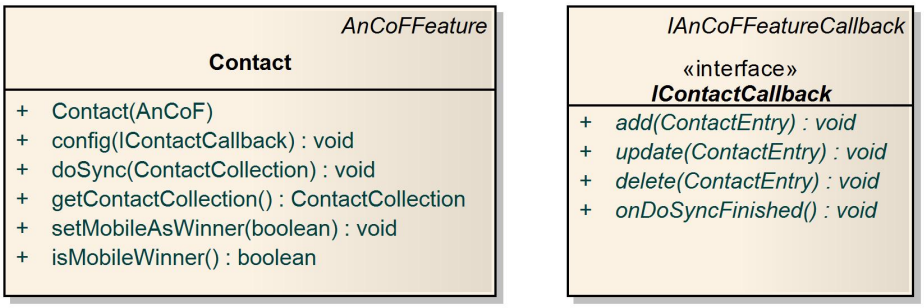


Abbildung 8.1: Benutzerinterface Kontakte

8.1 PC

8.1.1 Konfigurieren der Kontaktklasse

Die Konfiguration erfolgt gleich wie bei der Kalendersynchronisation und kann dem Kapitel 3.1.1, Seite 4 entnommen werden. Zu beachten ist, dass «Calendar» durch «Contact» zu ersetzen ist.

8.1.2 Datenstruktur

Die Abbildung 8.2 dient als Übersicht über die, für die Synchronisation benötigte, Datenstruktur. Zudem zeigt die Abbildung, welche Felder eines Kontakts unterstützt werden.

Zu beachten ist, dass der «display name» nach der initialisierung nicht mehr geändert werden darf. Ist dies trotzdem erforderlich, muss der alte Kontakt gelöscht und ein neuer mit dem neuen Namen erstellt werden.

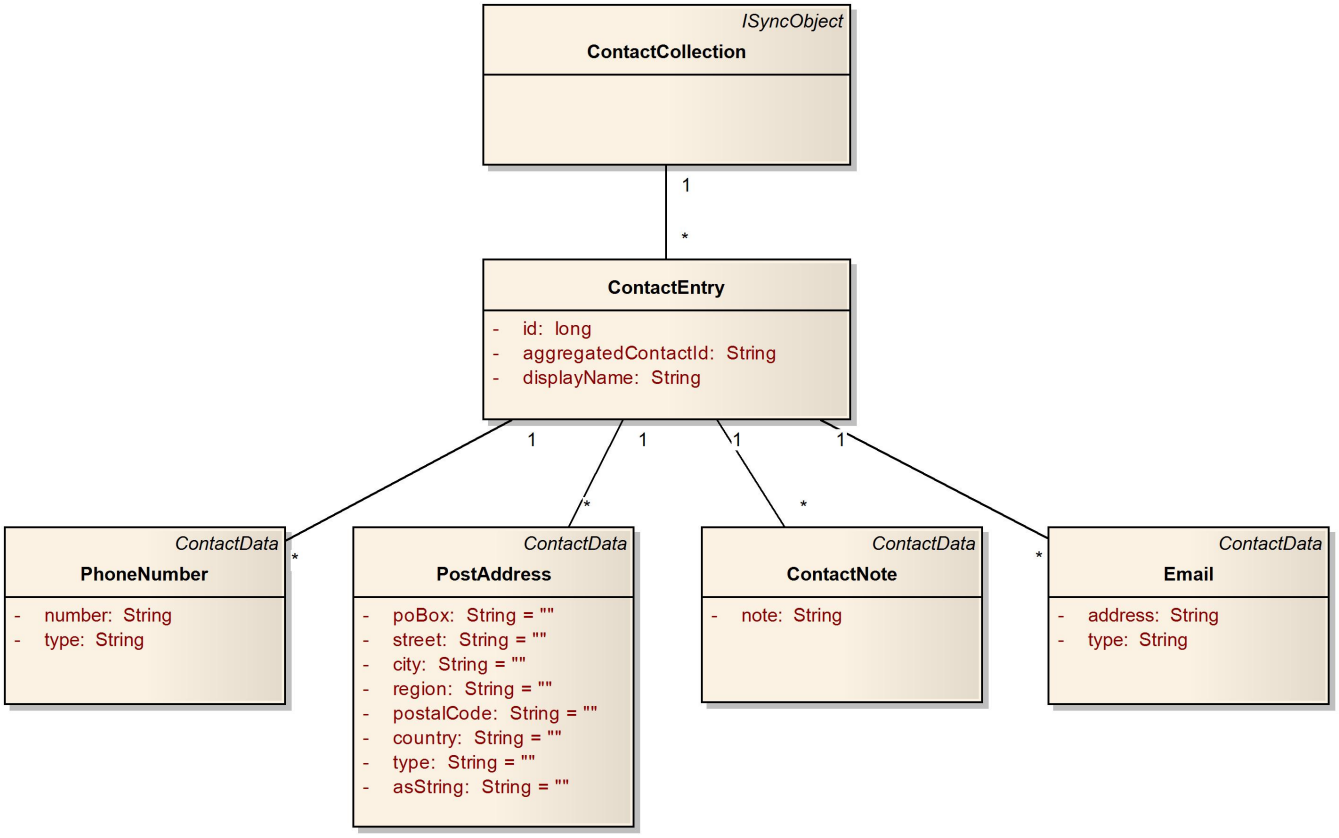


Abbildung 8.2: Datenstruktur Kontakte

8.1.3 Synchronisation

Um den Synchronisationsprozess zu starten, wird `public void doSync(contacts)` der Klasse Contact aufgerufen. contacts muss als ContactCollection mitgegeben werden. Details zum Synchronisationsalgorithmus können dem «Design Dokument», Kapitel 8.4, Seite 24 entnommen werden.

Bei diesem Bereich von AnCoF bestehen noch Einschränkungen bezüglich der Synchronisation. Details dazu können ebenfalls im «Design Dokument», Kapitel 10.7, Seite 27 nachgeschlagen werden.

8.2 Mobile-Device

Der Service wird automatisch gestartet, sofern er beim Start von AnCoF angegeben und im Manifest bekannt gemacht wurde:

```
<service android:name="ch.hsr.ancof.contact.ContactService"/>
```

8.2.1 Permissions

Die Klasse ContactService benötigt folgende Permissions:

- android.permission.READ_CONTACTS
- android.permission.WRITE_CONTACTS

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	29.09.2010	dm	Dokument erstellt.
1.0rc02	17.12.2010	dm	Kapitel ergänzt.
1.0rc03	17.12.2010	rr	Benutzerinterface ergänzt.
1.0rc04	17.12.2010	rr	Anpassungen in Kapiteln, besonders bei Kontakte.
1.0rc05	20.12.2010	rr	Rechtschreibung und Layout angepasst.
1.0	20.12.2010	dm	Akzeptierte Version 1.0

Tabelle 8.1: Revisionshistorie

Anhang

Original event is on Thursday, 21 October 2010

- One-time event: 10-21 04:35:04.433: INFO/CalendarTest(579): Event ID: 3, Title: Test1, Repetition: null
- Daily: 10-21 04:35:04.443: INFO/CalendarTest(579): Event ID: 4, Title: Test2, Repetition: FREQ=DAILY;WKST=MO
- Every weekday(Mon-Fri): 10-21 04:35:04.463: INFO/CalendarTest(579): Event ID: 79, Title: Test3, Repetition: FREQ=WEEKLY;WKST=MO;BYDAY=MO,TU,WE,TH,FR
- Weekly(every Thursday): 10-21 04:35:04.473: INFO/CalendarTest(579): Event ID: 80, Title: Test4, Repetition: FREQ=WEEKLY;WKST=MO;BYDAY=TH
- Monthly(every third Thu): 10-21 04:35:04.483: INFO/CalendarTest(579): Event ID: 81, Title: Test5, Repetition: FREQ=MONTHLY;WKST=MO;BYDAY=3TH
- Monthly(on day 21): 10-21 04:35:04.443: INFO/CalendarTest(579): Event ID: 72, Title: Test6, Repetition: FREQ=MONTHLY;WKST=MO;BYMONTHDAY=21
- Yearly(on 21 October): 10-21 04:35:04.453: INFO/CalendarTest(579): Event ID: 73, Title: Test7, Repetition: FREQ=YEARLY;WKST=MO

Abkürzungsverzeichnis

AnCoF Android Control Framework
PC Personal Computer
SA Studienarbeit
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Glossar
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

21. Dezember 2010

Inhaltsverzeichnis

1 Einführung

1.1 Zweck

Siehe «Projektplan», Kapitel 4.2.2, Seite 8.

1.2 Gültigkeitsbereich

Das Dokument behält seine Gültigkeit während der gesamten Projektdauer.

2 Begriffe

Tabelle 2.1 (Seite 4) listet wichtige Begriffe und ihre Bedeutung im Rahmen des Projekts auf. In der Spalte Kontext ist aufgeführt, in welchem Zusammenhang welche Definition des Begriffs verwendet wird.

Begriff	Kontext	Definition und Information
AnCoF	–	Das Android Control Framework (AnCoF) stellt Methoden zur Verfügung, um Funktionalitäten des Mobile-Devices über den Personal Computer (PC) abzuwickeln.
AnCoF Feature	–	AnCoF Feature bezeichnet einen Teilbereich von AnCoF, z.B. Telefon, Kalender, etc. Als Synonym wird in der Dokumentation auch der Begriff «Bereich» verwendet.
Android	–	Betriebssystem sowie Software-Plattform für mobile Geräte wie Smartphones, Mobiltelefone und Netbooks, die von der Open Handset Alliance entwickelt werden.
Ap	Planung	Dokument: Arbeitspakete (Ap)
APPDATA	Implementation, Dessign	Der in der «Pfad-Variablen» angegebene Speicherort des Ordners «Anwendungsdaten» (Windows XP) bzw. «AppData» (Windows Vista und höher).
As	Anforderungen	Dokument: Anforderungsspezifikation (As)
Callback	–	Der Callback ist ein Objekt, welches ermöglicht, Informationen an die PC-Applikation weiterzugeben.
CRUD	–	Create, read, update, delete (CRUD) Operationen.
Da	Analyse	Dokument: Domainanalyse (Da)
DD	Design	Dokument: Design Dokument (DD)
Device	–	Ein Gerät, welches AnCoF benutzen kann (z.B. PC, Mobile-Device). In der Dokumentation wird der Begriff «Gerät» als Synonym verwendet.
DG	Entwicklung	Dokument: Developer Guide (DG)
DRY	Dokumentation und Implementation	Don't repeat yourself (DRY), das heisst: Keinen doppelten Text oder Code verfassen.
EA	Design	Der Enterprise Architect (EA) ist ein Unified Modeling Language (UML)- und Business-Modellierungs-Tool.
Eclipse	Entwicklung	Entwicklungsumgebung
Extra	Intent	Extra bezeichnet weitere Daten, die mit einem Intent mitgegeben werden können.
Gl	Dokumentation	Dokument: Glossar (Gl)
Haltemodus	Telefon	Ein aktiver Anruf wurde unterbrochen und zugunsten eines anderen Anrufs zurückgestellt.

Begriff	Kontext	Definition und Information
Intent	Implementation, Dessign	Startet eine Aktion auf dem Mobile-Device.
Java	Implementation	Objektorientierte Programmiersprache
Mobile-Device	–	Mobiles Gerät wie zum Beispiel ein Smartphone oder Mobiltelefon
PC-Applikation	–	PC-Applikation, welche auf AnCoF aufbaut.
Pp	Planung	Dokument: Projektplan (Pp)
SAD	Design	Dokument: Software Architecture Document (SAD)
SMS	–	Mittels Short Message Service (SMS) können Nachrichten zwischen Mobile-Devices ausgetauscht werden.
Td	Construction	Dokument: Testdokumentation (Td)
USB	–	Der Universal Serial Bus (USB) ist ein serielles Bussystem zur Verbindung eines PC's mit externen Geräten.
TaMaF	–	Mittels Task-Management-Framework on Smart-Phone (TaMaF) kann eine Verbindung über USB oder Wireless Local Area Network (WLAN) zwischen einem Mobile-Device und dem PC aufgebaut werden.

Tabelle 2.1: Liste der projektrelevanten Begriffe

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	22.09.2010	dm	Dokument erstellt.
1.0rc02	13.10.2010	dm	Begriffe ergänzt.
1.0rc03	27.11.2010	dm	Begriffe ergänzt.
1.0rc04	20.12.2010	rr	Begriffe ergänzt, Rechtschreibung und Layout angepasst.
1.0	21.12.2010	dm	Akzeptierte Version 1.0

Tabelle 2.2: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
Ap Arbeitspakete
As Anforderungsspezifikation
CRUD Create, read, update, delete
Da Domainanalyse
DD Design Dokument
DG Developer Guide
DRY Don't repeat yourself
EA Enterprise Architect
Gl Glossar
PC Personal Computer
Pp Projektplan
SAD Software Architecture Document
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone
Td Testdokumentation
UML Unified Modeling Language
USB Universal Serial Bus
WLAN Wireless Local Area Network

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

21. Dezember 2010

Verwendete Werke

Larman, Craig (2004): Applying UML and Patterns. 3. Auflage, Upper Saddle River.

Internetreferenzen

Allgemein

Anleitung für Dokumentation Semester-, Bachelor- und Diplomarbeiten (2010):
https://www.hsr.ch/index.php?eID=tx_nawsecuredl&u=5695&file=fileadmin/user_upload/customers/hsr/HSR-INTERN/Bachelor-Studiengaenge/Informatik/Vorlagen/DokuAnleitungBA_DA_SA_100304.pdf&t=1292945638&hash=ae2dc3219ba98501c99a905f9f05ae37, 20. Dezember 2010.

Android

Android: http://developer.android.com/, 20. Dezember 2010.

Kalender

Baccega, Andrea (2010):
http://www.andreabaccega.com/blog/2010/08/09/add-events-on-google-calendar-on-android-froyo/, 20. Dezember 2010.
Blackler, Jim (2009): http://jimblackler.net/blog/?p=151, 20. Dezember 2010.
Conder, Shane und Darcey, Lauren (2009):
http://www.developer.com/ws/article.php/3850276/Working-with-the-Android-Calendar.htm, 20. Dezember 2010.
Google (2010): http://code.google.com/p/google-api-java-client/, 20. Dezember 2010.

Telefon

AIDL-Interface:
http://code.google.com/p/teddsdroidtools/source/browse/branches/beta2-devel/teddsdroidtools/src/com/android/internal/telephony/ITelephony.aidl?r=33, 20. Dezember 2010.
Prasanta, Paul (2010): http://prasanta-paul.blogspot.com/2010/09/call-control-in-android.html, 20. Dezember 2010.

SMS

Lee, Wei-Meng (2008): http://mobiforge.com/developing/story/sms-messaging-android, 20. Dezember 2010.

Kontakte

Kontakte: http://www.higherpass.com/Android/Tutorials/Working-With-Android-Contacts/1/, 20. Dezember 2010.

Bilder

Open Clipart Library: http://www.openclipart.org/, 20.Dezember 2010.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Android Control Framework


Daniela Meier


Ramona Rudnicki

Betreuer: Thomas Letsch

Studienarbeit Herbstsemester 2010

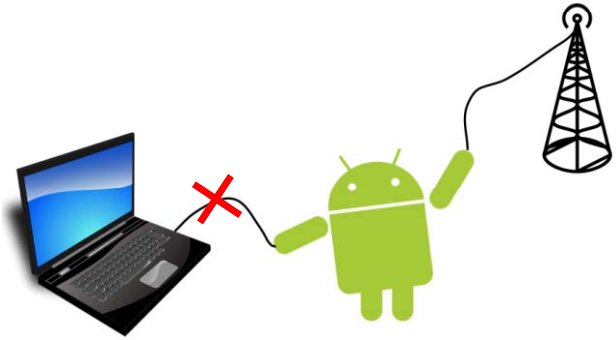
Themengebiet: Software

Situtation …

… ohne Android Control Framework:

- ✕ Synchronisation von Kalender und Kontakten benötigt Internetverbindung zu Google
- ✕ Anrufe tätigen und Versenden von SMS via PC ist nicht möglich
- ✕ Telefoneinstellungen können nur direkt am Gerät verändert werden
- ✕ Synchronisation von Dateien nur „von Hand“ möglich

oder



oder

… mit Android Control Framework:

- ✓ Direkte Verbindung von Mobiltelefon und PC zur Synchronisierung
- ✓ Steuern der Anrufe und senden/empfangen von SMS via PC möglich
- ✓ Telefoneinstellungen direkt am PC konfigurierbar
- ✓ Automatische Ordnersynchronisation

Technischer Bericht
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Inhaltsverzeichnis

1 Einleitung

Für den remote-Zugriff auf ein Android-Mobiltelefon wird aktuell eine Verbindung zum Google-Server respektive ein eigener Web-Server auf dem Mobiltelefon benötigt. Dieser Umstand sollte mit dem «Android Control Framework (AnCoF)» beseitigt werden. Folgende Bereiche sollten mit Funktionen ausgestattet werden, die es zum Beispiel ermöglichen, eine PC-Suite zu implementieren.

- Kalender: Synchronisation der Kalendereinträge ohne Datenverbindung ins Internet.
- Telefon: Anrufe empfangen und tätigen.
- Einstellungen: Lautsprecher, Mikrophon und Lautstärke konfigurieren.
- SMS: Nachrichten versenden und empfangen.
- Daten: Synchronisation der Dateien.
- Kontakte: Synchronisation der Kontakte.

Um den Verbindungsaufbau zwischen Mobiltelefon und Personal Computer (PC) nicht nochmals von Grund auf zu implementieren, sollte auf Funktionen von Task-Management-Framework on Smart-Phone (TaMaF) zurückgegriffen werden. TaMaF kann auch im Bereich Daten eingesetzt werden, da es bereits Funktionen für die Datei-Übermittlung zwischen zwei Geräten zur Verfügung stellt.

Um diese Anforderungen umzusetzen, wurde gemäss Unified Process (UP), wie er in [1] beschrieben wird, geplant. Die Dokumente wurden so kurz wie nötig gehalten, es wurde darauf geachtet, dass das DRY-Prinzip nicht verletzt wurde. Auf das Design Dokumente wurde besonderen Wert gelegt, da es die gesamte Architektur beschreibt und Entscheidungen und mögliche Erweiterungen dokumentiert.

Nebst den Dokumenten und der eigentlichen Implementierung wurden Prototypen erstellt, welche zu beginn einen hohen Stellenwert hatten. Sie halfen zu entscheiden, ob die vorgenommenen Bereiche überhaupt umsetzbar sind. Für das Test Driven Development (TDD) waren auch die Unit Tests von Bedeutung, sie ermöglichten es, implementierte Codeblöcke fortlaufend auf ihre Korrektheit zu überprüfen.

2 Ergebnisse

Während der Umsetzung der Funktionalitäten von AnCoF sind immer wieder Probleme aufgetaucht, deren Lösungen nicht immer ganz offensichtlich waren.

2.1 Kalender

Da für die Kalendersynchronisation der Umweg über den Google Server «ausgeschaltet» werden sollte, tauchten bereits zu beginn die ersten Schwierigkeiten auf. Das Application Programming Interface (API) von Android gewährt keinen offiziellen Zugriff auf die Kalenderdaten in der Datenbank. Um die Kalendersynchronisation trotzdem zu ermöglichen, musste auf die in Internetforen präsentierte Lösung, die interne API zu nutzen, zurück gegriffen werden. Mit dieser Möglichkeit konnte eine eigene Kalenderklasse mit Synchronisationsalgorithmus implementiert werden. Dafür musste in Kauf genommen werden, das allfällige Änderungen an der Kalenderstruktur von Google unter Umständen ebenfalls Anpassungen an AnCoF nach sich ziehen, die nicht in jedem Fall offensichtlich sein müssen.

2.2 Telefon

Die Telefonimplementation hingegen gestaltete sich einfacher. Es konnten sogar die vorgesehenen Funktionen «Auflegen» und «Ablehnen» in einer Methoden zusammengefasst werden, was die Arbeit erleichterte. Trotzdem lief auch hier nicht alles reibungslos. Das Beenden eines Anrufs ist mit der öffentlichen API nicht möglich. Weiter merkt sich Android maximal zwei Anrufe auf einem Gerät, ein Anruf ist aktiv, ein weiterer im Haltemodus. Eine Konferenzschaltung, wie man sie von Festnetztelefonen her kennt, konnte daher nicht implementiert werden. Das Beenden des Anrufs hingegen konnte trotz anfänglicher Schwierigkeiten umgesetzt werden. Mittels Java Reflection konnten die nötigen Befehle in Erfahrung gebracht werden.

2.3 Einstellungen

Bei den Einstellungen konnten nicht alle Funktionalitäten umgesetzt werden. Aus Zeitgründen wurde entschieden, nur die Lautsprecherkonfiguration zu implementieren. Die Konfiguration der Lautstärke respektive des Mikrophons musste zurückgestellt werden.

2.4 SMS

Ein etwas anderes Problem boten die SMS. Schuld daran, dass der implementierte Code nicht sofort funktionierte, waren nicht etwa Überlegungsfehler, sondern «Schreibfehler» im Manifest. Dieser Umstand kostete einiges an Zeit, machte aber auch Bewusst, dass nebst Code immer auch ein Manifest Bestandteil eines Android-Projektes ist.

2.5 Daten

Einfacher als Gedacht stellte sich die Datensynchronisation heraus. Zwei kleinere Hindernisse fanden sich in der Übertragung der zu aktualisierenden Dateien. TaMaF hat nur das Versenden von Dateien implementiert, nicht aber der Empfang von solchen. Weiter wird nach erfolgreicher Übermittlung das Bearbeitungsdatum nicht gesetzt. Dieses ist jedoch von hoher Wichtigkeit für die Synchronisation, da ansonsten nicht entschieden werden kann, welche Datei neuer ist. Um die Dateien vom Mobiltelefon an den PC zu übermitteln, wurden diese in einem ersten Schritt mittels XML an das Mobiltelefon gesendet und von dort aus dann mit der von TaMaF vorgesehenen Funktion an den PC übertragen. Das fehlende Bearbeitungsdatum wird in AnCoF im Nachhinein gesetzt.

Was wiederum aus Zeitgründen nicht implementiert wurde, jedoch wünschenswert wäre, ist die «rekursive» Datensynchronisation.

2.6 Kontakte

Das zweite «Sorgenkind» nebst der Kalendersynchronisation waren die Kontakte. Hier mussten mehrere Einschränkungen gemacht werden. Es mussten Prioritäten gesetzt werden, welche Kontakte überhaupt synchronisiert werden. Die Entscheidung fiel auf die Telefonkontakte, jene auf der SIM-Karte und die Google-Kontakte wurden zurückgestellt. Weiter kann die Id der einzelnen Kontaktdaten nicht explizit gesetzt werden. Gelöst wurde dieses Problem, indem die Id erst bei der nächsten Synchronisation übernommen wird. Es ist vorgesehen, diese Id in Zukunft für die Erkennung von neuen oder gelöschten Kontaktdaten zu verwenden. Offen ist zur Zeit, wie mit Kontakten mit demselben Namen vorgegangen wird. Diese werden auf dem Mobiltelefon automatisch zusammengefasst. Es besteht daher die Einschränkung, dass die Namen nicht verändert werden dürfen.

3 Fazit

Den Zeitaufwand für die Teilbereiche von AnCoF einzuschätzen, stellte sich als die grösste Herausforderung dar. Der Schwierigkeitsgrad der einzelnen Bereiche wurde völlig falsch eingeschätzt und war daher keine grosse Hilfe in der Zeitplanung. Einzelne Bereiche (Telefon, SMS oder Daten) waren einfacher als gedacht. Andere waren schwieriger als angenommen, darunter fallen Kalender und Kontakte. Einzig richtig eingeschätzt werden konnten die Einstellungen. Da diese jedoch nicht höchste Priorität hatten, wurde hier nur gerade eine Funktion umgesetzt.

Revisionshistorie

Version	Datum	Person	Änderung
1.0rc01	19.12.2010	dm	Dokument erstellt.
1.0rc02	20.12.2010	rr	Rechtschreibung angepasst.
1.0	20.12.2010	dm	Akzeptierte Version 1.0

Tabelle 3.1: Revisionshistorie

Abkürzungsverzeichnis

AnCoF Android Control Framework
API Application Programming Interface
DRY Don't repeat yourself
PC Personal Computer
SIM Subscriber Identity Module
SMS Short Message Service
TaMaF Task-Management-Framework on Smart-Phone
TDD Test Driven Development
UP Unified Process

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] LARMAN, Craig: *Applying UML and Patterns*. Prentice Hall, 2004. – ISBN 978–0–13–148906–2

Persönliche Berichte
«Android Control Framework»

Version 1.0

Daniela Meier (d2meier@hsr.ch) Ramona Rudnicki (rrudnick@hsr.ch)

20. Dezember 2010

Persönlicher Bericht Daniela Meier

Schon Wochen im Voraus freute ich mich auf die Studienarbeit. Nicht verwunderlich, denn nach den doch meist recht theoretischen Unterrichtsmodulen «plangte» ich richtig darauf, eine grössere Arbeit von Anfang bis zum Ende zu planen und umzusetzen. Erste Erfahrungen mit Android konnte ich bereits im vorhergehenden Semester im Modul Betriebssystemkonzepte sammeln, wo in einem Unterrichtsblock die Grundlagen der Android-Programmierung behandelt wurde.

Die intensive Betreuung in den ersten Wochen empfand ich als angenehm. Es wurde viel geplant, angepasst und spezifiziert. Das Dokumentieren, von vielen Informatikern als mühselig empfunden, war für mich spannend und hat mir geholfen, mich in unserem Projekt zurechtzufinden und mir einen Überblick zu verschaffen. Trotzdem freute ich mich riesig, als es darum ging, die ersten Prototypen zu implementieren. Schnell waren erste Erfolge da, Kalenderdaten konnten aus der Datenbank des Mobiltelefons gelesen und auch wieder hineingeschrieben werden. Doch bis zum fertigen Produkt war es noch ein langer Weg.

Es stellte sich schon bald heraus, das die Umsetzung gewisser Funktionalitäten, namentlich die Kalender- und Kontaktsynchronisation schwieriger als angenommen war. Ebenfalls der Verbindungsaufbau zwischen dem Mobiltelefon und dem PC war für mich komplex. Hier war ich froh, dass meine Teamkollegin mir diesen Aufbau immer wieder von neuem erklärte. Auch eine grosse Stütze war sie für mich beim Bugfixing, geduldig ist sie mit mir stundenlang durch den Code gegangen, bis jeweils die «Übeltäter» gefunden waren.

Rückblickend würde ich einen Teilbereich weniger implementieren und die so gewonnene Zeit einsetzen, um den Code zu überarbeiten und zu optimieren. Trotz der intensiven und auch anstrengenden Zeit, die Freude, mit der ich zu Beginn an die Arbeit gegangen bin, hat sich über die ganze Zeit erhalten.

Persönlicher Bericht Ramona Rudnicki

Vor der Studienarbeit hatte ich im Bereich Mobiltelefon-Programmierung erst Erfahrungen mit JavaME-Programmierung auf dem Betriebssystem «Symbian» gesammelt. Da ich dies damals als sehr mühselig und nicht intuitiv empfunden hatte, erhoffte ich vom neueren, moderneren Betriebssystem Android, dass diese Einschränkungen behoben wurden. Ich sollte nicht Enttäuscht werden.

Zu Beginn des Semesters wurde viel geplant und spezifiziert. Meiner Meinung nach etwas zu lange, da ich in den ersten Wochen nicht auf meine Soll-Stundenzahl kam. Diese habe ich natürlich in späteren Wochen wieder eingeholt bzw. sogar überholt. Sehr froh war ich in dieser Zeit um meine Teamkollegin, die mir sehr viel Arbeit an den Dokumenten abgenommen hat, da ich krankheitshalber etwas weniger leisten konnte. In Zukunft würde ich die Zeit neben dem Spezifizieren intensiver für das Aneignen von Wissen und Erstellen erster Prototypen nutzen.

Nach diesen ersten drei Wochen habe ich fast immer meine Sollzeit überschritten. Dies geschah vor allem aus Interesse und Ehrgeiz, doch noch alle Bereiche und Möglichkeiten zu implementieren und eine schön abgerundete Arbeit abzugeben. Obwohl gerade deshalb die letzten Wochen sehr intensiv und stressig wurden, bereue ich die investierte «Überzeit» nicht.

Während der Planung dachten wir, dass die Bereiche Telefonieren und Versenden von SMS im Gegensatz zu der Synchronisation von Kalender, Daten und Kontakten schwierig zu implementierenden sind. Wir wurden sehr schnell vom Gegenteil überrascht. Das Telefonieren stellte sich dank einem «Workaround» schnell als recht einfach heraus. Wohingegen sich der Kalender als sehr mühsam erwies, da er nicht von der öffentlichen API unterstützt wird und alle Funktionen in mühsamer Kleinarbeit in Internetforen und Blogs zusammengesucht und ausgetestet werden mussten.

In der zweiten Phase implementierten wir erstaunlich schnell und einfach die Funktionen um SMS zu versenden. Auch das Synchronisieren der Daten erwies sich einfacher als die Implementation der Kalendersynchronisation, unter anderem, da wir Funktionen der Vorgängerarbeit ausnutzen konnten. Die Kontakte zu synchronisieren war wieder ein etwas schwierigeres Thema, da die Kontakte komplex und dynamisch aufgebaut sind.

In Zukunft würde ich eher auf einen Bereich komplett verzichten. Dies deshalb, weil in mehreren Bereichen aus Zeitnot Einschränkungen gemacht werden mussten. Dieser Umstand hinterlässt etwas das Gefühl, die Arbeit nicht richtig und vollständig umgesetzt zu haben.