



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK



Task-Management- Framework on Smart-Phone

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009/2010

Autor(en): Patrick Boos, Markus Kolb
Betreuer: Thomas Letsch
Gegenleser: Markus Stolze

Inhaltsverzeichnis

- 1 Aufgabenstellung
- 2 Erklärung über die eigenständige Arbeit
- 3 Vereinbarung über Urheber- und Nutzungsrechte
- 4 Abstract / Kurzfassung – AB
- 5 Management Summary / Kurzbeschreibung – MS
- 6 Poster
- 7 Technischer Bericht – TB
- 8 Persönliche Berichte – PBMK & PBPB
- 9 Glossar – GL
- 10 Literaturverzeichnis – LV
- 11 Projektplan – PP
- 12 Anforderungsspezifikation – AS
- 13 Domainanalyse – DA
- 14 Software Architecture Document – SAD
- 15 Design Dokumentation – DD
- 16 Testdokumentation – TD
- 17 Developer Guide – DG
- 18 Zeiterfassung – ZE

Inhaltsverzeichnis

1	Aufgabenstellung	
2	Erklärung über die eigenständige Arbeit	
3	Vereinbarung über Urheber- und Nutzungsrechte	
4	Abstract / Kurzfassung – AB	
5	Management Summary / Kurzbeschreibung – MS	
1	Ausgangslage	2
2	Vorgehen	2
3	Technologien	2
3.1	Java	2
3.2	Android	2
3.3	SQLite	2
4	Ergebnisse	3
5	Ausblick	3
6	Poster	
7	Technischer Bericht – TB	
1	Einführung und Übersicht	4
1.1	USB-Kommunikation	4
1.2	Funktionalitäten	4
1.3	Übrige Teile der Abgabe	4
2	Ergebnisse	4
2.1	Das Framework	4
2.1.1	Auf Android Smart-Phone und Computer lauffähig	5
2.1.2	Kommunikation durch XML-Pakete	5
2.2	Erkenntnisse	6
2.2.1	Kommunikation über USB	6
3	Schlussfolgerungen	6
3.1	Erreichte Ziele	6
3.2	Nicht erreichte Ziele und Unschönheiten	6
3.3	Ursachen	6
3.4	Vergleich mit anderen Lösungen	6
3.4.1	HTC Sync	6
3.4.2	Synchronisation über Internet	7
3.5	Ausblick	7

8	Persönliche Berichte – PBMK & PBPB	
9	Glossar – GL	
1	Einführung	3
1.1	Zweck.....	3
1.2	Gültigkeitsbereich.....	3
1.3	Definitionen und Abkürzungen.....	3
1.4	Referenzen.....	3
2	Glossar	4
10	Literaturverzeichnis – LV	
11	Projektplan – PP	
1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	Projekt Übersicht	5
2.1	Zweck und Ziel.....	5
2.2	Annahmen und Einschränkungen	5
3	Projektorganisation.....	6
3.1	Organisationsstruktur	6
3.2	Externe Schnittstellen.....	6
4	Management Abläufe.....	7
4.1	Projekt Kostenvoranschlag.....	7
4.2	Besprechungen.....	7
4.2.1	Mit Betreuer.....	7
4.2.2	Team-Besprechungen.....	7
4.3	Vorgegeben Dokumente bei der Studienarbeit	7
4.4	Projektplan	8
4.4.1	Zeitplan.....	8
4.4.2	Beschreibung der Dokument Inhalte.....	9
4.4.3	Meilensteine.....	11
4.4.4	Iterationsplanung.....	11
5	Risiko Management.....	13
6	Infrastruktur	14
7	Qualitätsmassnahmen	15
7.1	Teammeetings / Sitzungsprotokolle	15
7.1.1	Unit-Tests.....	15
7.2	Reviews.....	15
7.2.1	Code-Reviews	15
7.2.2	Dokumenten-Reviews	15
7.3	Logging	15
7.3.1	Levels	15

7.3.2	Dichtes Logging	16
8	Arbeitspakte	17
9	Anhang	22
9.1	Tabellenverzeichnis.....	22
12	Anforderungsspezifikation – AS	
1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich	4
1.3	Übersicht.....	4
2	Allgemeine Beschreibung	4
3	Synchronisations-Framework	5
3.1	Funktionale Anforderungen	5
3.1.1	Aktoren	5
3.1.2	Use Case-Diagramm	5
3.1.3	Use Cases brief	6
3.1.4	Zustandsdiagramm.....	7
3.1.5	Interfaces	8
3.2	Nichtfunktionale Anforderungen	12
3.2.1	Funktionalität	12
3.2.2	Erlernbarkeit	12
3.2.3	Zuverlässigkeit	12
3.2.4	Effizienz	12
3.2.5	Erweiterbarkeit.....	12
4	Task-Management Applikation	13
4.1	GUI Paper Prototype	14
4.1.1	Main Window (1)	15
4.1.2	New Task / Edit Task (2)	16
4.1.3	Filter (3)	16
4.1.4	Connect/Sync (4)	16
4.1.5	Verbindungs-Ansicht (5)	16
4.1.6	Settings	17
13	Domainanalyse – DA	
1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich	4
1.3	Definitionen und Abkürzungen	4
1.4	Referenzen.....	4
2	TaMaF	5
2.1	Domain Modell.....	5
2.1.1	Strukturdiagramm	5
2.1.2	Konzeptbeschreibung.....	6
2.2	System Sequenzdiagramme	8
2.2.1	UC1 Verbindung aufbauen	8
2.2.2	UC2 Verbindung beenden.....	8
2.2.3	UC3 Datei senden.....	9

2.2.4	UC4 Synchronisieren	10
2.2.5	UC5 Informationen über den Fortschritt	11
2.2.6	UC6 Listen Mode einschalten	12
2.2.7	UC7 Listen Mode beenden	12
2.2.8	UC8 Task hinzufügen/editieren	13
2.2.9	UC9 Task löschen	13
2.2.10	UC10 Einen kompletten Dump machen	13
2.2.11	UC11 Geräte auflisten	13
2.2.12	UC12 Gerät/Computer löschen	14
2.2.13	UC13 Verbundenes Gerät abfragen	14
2.3	Systemoperationen	15
3	TaMaF Demo auf Android Phone	17
3.1	Domain Modell	17
3.1.1	Konzeptbeschreibung	17
3.2	System Sequenzdiagramme	19

14 Software Architecture Document – SAD

1	Einführung	4
1.1	Zweck	4
1.2	Gültigkeitsbereich	4
1.3	Definitionen und Abkürzungen	4
1.4	Referenzen	4
3	Übersicht	5
3.1	Systemübersicht	5
4	Logical View TaMaF	6
4.1	Application	6
4.2	TaMaF	6
4.3	AndroidSyncFramework	6
4.4	CommunicationLayer	7
4.5	Persistence	7
5	Logical View Android	7
5.1	AndroidDemo	7
5.2	TaMaF	7
6	Deployment View	8
6.1	Technologien	8
7	Process View	8
7.1	Parallelitäten	8
8	Logging	8

15 Design Dokumentation – DD

1	Einführung	4
1.1	Zweck	4
1.2	Gültigkeitsbereich	4
1.3	Definitionen und Abkürzungen	4
1.4	Referenzen	4

2	TaMaF	5
2.1	package tamaf	5
2.1.1	class TaMaF	6
2.1.2	interface XferAgentTaMaF	6
2.1.3	class Task	6
2.2	package androidsync	6
2.2.1	interfaces	7
2.2.2	class AndroidSyncFramework	8
2.2.3	class Device	9
2.2.4	messages	10
2.3	package communication	14
2.3.1	Entscheid XML	15
2.4	package persistence	16
2.4.1	Implementation Proxy Pattern	16
2.4.2	Implementation Active Record Pattern	17
2.4.3	Zugriff auf die Datenbank auf dem PC	18
2.4.4	Zugriff auf die Datenbank auf dem Android	18
2.4.5	CommunicationDB	18
2.5	package utils	19
2.5.1	XmlCreatorHelper und XmlReaderHelper	19
2.5.2	SyncLogManager	19
3	Kernentscheide	20
3.1	Flexibles/Fixes Netzwerk	20
3.1.1	Fix	20
3.1.2	Flexibel	20
3.1.3	Entscheid für flexibles Netzwerk	21
4	Verbesserungen und Weiterentwicklungen	21
4.1	Verbesserungen	21
4.1.1	Listener an Stelle der onXYZ-Funktionen im XferAgent	21
4.1.2	HostId	21
4.1.3	onXferStateUpdate + XferStateUpdateHandler	22
4.1.4	deleteHostID	22
4.1.5	android.jar auf PC	22
4.1.6	FileTransfer mit Handy ist etwas langsam	22
4.2	Weiterentwicklungen	23
4.2.1	Security	23
16	Testdokumentation – TD	
1	Einführung	4
1.1	Zweck	4
1.2	Gültigkeitsbereich	4
1.3	Definitionen und Abkürzungen	4
1.4	Referenzen	4
2	Systemtest	5
2.1	Voraussetzung	5
2.2	Vorbereitung	5
2.3	Ablauf	5
2.4	Systemtest	5
3	Unit Tests	9

3.1	ch.hsr.sync.androidsync.tests	9
3.2	ch.hsr.sync.messages.tests	9
3.3	ch.hsr.sync.communication.tests	9
3.4	ch.hsr.sync.persistence.tests	9
3.5	ch.hsr.sync.tamaf.tests	9
3.6	ch.hsr.sync.utils.tests	9
4	Integrationstest	10
5	Funktionaler Test	10
6	Performance Test	10
7	Usability Test	10
17	Developer Guide – DG	
1	Einführung	4
1.1	Zweck	4
1.2	Gültigkeitsbereich	4
1.3	Definitionen und Abkürzungen	4
1.4	Referenzen	4
2	Einstieg	5
2.1	Was ist TaMaF (Task-Management-Framework)?	5
3	Grundlagen	5
3.1	TaMaF	5
3.2	XferAgentTaMaF	6
3.3	Technologien	6
4	Applikation auf PC	6
4.1	Build Path	6
4.2	Verbindung über USB	7
4.3	Android-Treiber	7
5	Applikation auf Android	7
5.1	Build Path	8
5.2	Permissions im AndroidManifest.xml	8
5.3	ActiveRecordManager auf Android setzen	8
5.4	ADB-Verbindungen zulassen	8
5.5	Emulator	9
6	Eigene nicht-Task-Objekte synchronisieren	9
6.1	AndroidSyncFramework	9
6.2	ISyncObject	9
7	Demo-Projekte	9
7.1	TaMaF Demo PC	9
7.2	TaMaF Demo Android	10
18	Zeiterfassung – ZE	

Task-Management-Framework on Smart-Phone

Studenten

- Boos Patrick
- Kolb Markus

Einführung

Android (<http://www.android.com>) ist ein Betriebssystem sowie auch eine Software-Plattform für mobile Geräte wie Smartphones, Mobiltelefone und Netbooks, welches von der Open Handset Alliance entwickelt wird.

Es soll auf Android ein Framework erstellt werden für die Erstellung von Task-Management-Applikationen (Verwaltung von Terminen, Aufgaben etc.).

Aufgabenstellung

Es sollen unter Berücksichtigung von Software-Engineering-Methoden ein Framework erstellt werden, welches die Erstellung von Android-basierten Applikationen im Bereich des Task-Managements mit speziellen Anforderungen erfüllt:

- grosse Datenbestände
- permanente Synchronisation mit einer PC-basierter Datenquelle resp. -senke.

Zusätzlich soll in Form eines Prototypen die Möglichkeiten für typische Darstellungen in Informationssystemen mit den vorhandenen GUI-Widgets gezeigt und damit gleichzeitig auch die Funktionstüchtigkeit des Frameworks bewiesen werden.

Technologien

- Android
- Java
- Eclipse
- Enterprise Architect

Generelles

- Die Vorgaben der Abteilung Informatik [1] sind einzuhalten.
- Die "Generelle Richtlinien für Studien- und Bachelorarbeiten" [2] sind einzuhalten.
- Mit dem CASE-Tool *Enterprise Architect* ist ein UML-Modell zu führen, welches synchron mit den *Programm-Sourcen* und der *Projekt-Dokumentation* ist.
- Ein Java-Entwickler muss mit der Projekt-Dokumentation in die Lage versetzt werden, die Applikation in Betrieb zu nehmen und weiter entwickeln zu können.

Termine

- Montag, 14.09.09 Beginn der Studienarbeit
- Freitag, 18.12.09 17:00 Uhr Abgabe der Studienarbeit

Betreuung

Thomas Letsch
tlletsch@hsr.ch
055 - 22 24 567 (HSR 5.204); 055 - 214 43 50 (Geschäft)

Referenzen

- [1] [www.hsr.ch>HSR-intern>Bachelor>Informatik>Allgemeine Infos Diplom-, Bachelor- und Studienarbeiten](http://www.hsr.ch>HSR-intern>Bachelor>Informatik>Allgemeine%20Infos%20Diplom-,%20Bachelor-und%20Studienarbeiten)
<https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html>
- [2] "Generelle Richtlinien für Studien- und Bachelorarbeiten"
(13.09.2009, Thomas Letsch)

Rapperswil, 13. September 2009



Thomas Letsch

Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

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

Ort, Datum:

Rapperswil, 16.12.2009

Name, Unterschrift:

Patrick Boos, 

Ort, Datum:

Rapperswil, 16.12.2009

Name, Unterschrift:

Markus Kolb, 

Vereinbarung über Urheber- und Nutzungsrechte

1. Gegenstand der Vereinbarung


Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit *Task-Management-Framework on Smart-Phone* von *Patrick Boos* und *Markus Kolb* unter der Betreuung von *Thomas Letsch* geregelt.

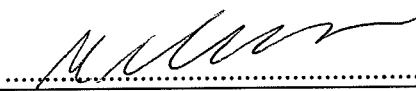
2. Urheberrecht

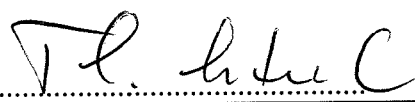
Die Urheberrechte stehen den Studenten zu.


3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von den Studenten, von der HSR wie vom Betreuer nach Abschluss der Arbeit frei verwendet und weiter entwickelt werden.

Rapperswil, den...19.11.09	
	Student Patrick Boos

Rapperswil, den...19.11.09	
	Student Markus Kolb

Rapperswil, den...12.11.09	
	Betreuer Thomas Letsch

Rapperswil, den...23.11.09	
	Studiengangleiter Abt. Informatik Prof. Dr. Lothar Müller

Abstract / Kurzfassung

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

1 Abstract / Kurzfassung

Abteilung	Informatik
Namen der Studenten	Patrick Boos, Markus Kolb
Studienjahr	HS 2009/2010
Titel der Studienarbeit	Task-Management-Framework on Smart-Phone
Examinatorin/Examinator	Thomas Letsch
Themengebiet	Software
Projektpartner	-
Institut	-

Ziel war es ein Framework zu erstellen, welches es ermöglicht, Tasks zwischen Android Handys und Computern zu synchronisieren. Solange zwei Geräte miteinander verbunden sind, soll es möglich sein, eine stetige Synchronisation aufrecht zu erhalten. Ebenfalls sollte eine Applikation für Android erstellt werden, welche die Nutzung des Frameworks im Stile eines Prototyps demonstriert. Die Synchronisation zwischen PC und Android Smart-Phone sollte über USB möglich sein.

Das Framework wurde unter Java geschrieben, da Java auf dem Computer und unter Android benutzt werden kann. Problematisch war, dass trotz Verwendung von Java einige Bereiche unter Android anders zu implementieren waren auf dem PC. So zum Beispiel die Datenhaltung und auch das Lesen und Erstellen von XML Dokumenten. Eine weitere Problematik bestand darin, dass die Tasks im Synchronisations-Netzwerk, bestehend aus mehreren Geräten, eindeutig identifizierbar sind.

Bei der Entwicklung der Software wurde als Software Engineering Vorgehensmodell der Rational Unified Process (RUP) eingesetzt. Das Ergebnis ist ein Framework, welches aus einer Java Archive Datei (.jar) besteht, welche für ein Projekt auf dem Computer, wie auch auf dem Android Betriebssystem verwendet werden kann. Die Synchronisationsnachrichten werden als XML Pakete über ein Socket übertragen, dies kann über die Medien USB, WLAN oder 3G, geschehen. Um über USB eine Verbindung zum Android Smart-Phone herzustellen, wurde das durch die Android Debug Bridge (ADB) zur Verfügung gestellte Port forwarding benutzt. Ebenfalls können Dateien über dasselbe Framework auf das andere Gerät transportiert werden. Das Framework limitiert den Programmierer nicht auf Task-Objekte die er synchronisieren kann. Unter Befolgung bestimmter Richtlinien kann der Programmierer eigene Objekte über das Framework synchronisieren, ohne sich über den eigentlichen Synchronisationsprozess Gedanken machen zu müssen.

Management Summary

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

1 Ausgangslage

Da die Mobilkommunikation heutzutage nicht mehr wegzudenken ist und bei Beginn der Arbeit noch kein Synchronisations-Framework für Applikationen im Bereich Task-Management auf Android existierte, wurde ein Task-Management-Framework für Android entwickelt. Auf dem entwickelten Framework sollen Entwickler die Möglichkeit haben, eine Task-Management-Applikation zu entwickeln, welche auf einem Android Smart-Phone oder einem Computer läuft. Die darin erfassten Tasks sollen ohne grossen Aufwand zwischen Computer und Smart-Phone über USB oder das Internet synchronisiert werden können. Teil der Aufgabe war es auch, mit dem Kunden gemeinsam die Anforderungsspezifikationen zu erarbeiten.

2 Vorgehen

Bei der Entwicklung des Android Frameworks wurde gemäss dem Vorgehensmodell „Rational Unified Process“ gearbeitet.

Erst wurde abgeklärt ob und wie eine Verbindung zum Smart-Phone über USB hergestellt werden kann. Dies stellte sich als machbar heraus, auch wenn es über einen Umweg geschieht.

Ein Problem bestand darin, dass die Tasks auf verschiedenen Geräten erfasst werden können und eindeutig identifiziert werden müssen. Dies ist nötig, damit man genau sagen kann welcher Task sich geändert hat.

Eine weitere Schwierigkeit war es, dass das Framework für Android und für den Computer geschrieben werden sollte. Es wurde versucht das Framework so zu schreiben, dass es ohne grössere Anpassung auf dem Computer, wie auch auf Android benutzt werden kann. Dies reduziert den Aufwand an der Entwicklung und Weiterentwicklung am Framework. Ebenfalls kann der Programmierer, welcher auf dem Framework eine Applikation entwickelt, egal ob auf dem Computer oder Android Smart-Phone, genau gleich mit dem Framework kommunizieren.

Eine wichtige Entscheidung während der Planung und Entwicklung war es, ob das Netzwerk der zu synchronisierenden Geräte flexibel gestaltbar sein soll. Wir haben zwei Lösungsansätze in Betracht gezogen. Die Lösung in welcher das Netzwerk bekannt sein muss, hätte das Framework vereinfacht, jedoch die Möglichkeiten sehr stark limitiert. So wäre eine Art Server-Applikation mit mehreren unterschiedlichen Clients nicht, oder nur sehr schwer, möglich gewesen. Aufgrund dessen, dass wir den Programmierer nicht einschränken wollten, haben wir uns für die zweite Variante entschieden und das Netzwerk flexibel implementiert.

3 Technologien

3.1 Java

Als Programmiersprache wurde Java verwendet. Sowohl auf der Computer- wie auch auf der Android-Seite. Grund dafür war, dass auf dem Android Smart-Phone Applikation unter Benützung dieser Programmiersprache geschrieben werden.

3.2 Android

Google hat mit Android ein offenes Betriebssystem für Handys bereitgestellt, welches auf Linux basiert. Im Jahr 2009 hat Android stark an Marktanteil zugenommen und ist somit eine lukrative Plattform für Mobile Applikationen. Die Schnittstellen für Programmierer sind gut dokumentiert und für Jeden zugänglich. Eine Applikation für Android wird unter Verwendung der Programmiersprache Java entwickelt.

3.3 SQLite

SQLite wird verwendet um die Daten abzuspeichern, welche im Framework anfallen. SQLite wird sowohl auf dem Computer wie auch auf dem Android Smart-Phone ver-

wendet. Das Abfragen der Daten verläuft auf dem Computer jedoch anders, als auf dem Android Smart-Phone.

4 Ergebnisse

Wie auch bei Android ist ein Developer Guide vorhanden, welcher es ermöglicht, sich schnell in die Thematik einzulesen. Zur Demonstration des Frameworks wurden zwei Demo Projekte entwickelt. Das Demoprojekt auf Android bringt ein kleines GUI mit sich. Das Demoprojekt auf dem Computer enthält ein Konsolen-Userinterface, mit welchem die Tasks verwaltet werden können.

Das Framework bietet dem Programmierer die Möglichkeit eine Task-Management-Applikation zu bauen ohne sich gross über die Synchronisation Gedanken zu machen. Die Applikation kann ohne Probleme die Funktionalität nutzen, sich mit einem anderen Gerät, Computer oder Android Smart-Phone, zu verbinden und zu synchronisieren. Ebenfalls kann in einen Listen-Mode geschaltet werden, welcher zur Folge hat, dass vorgenommene Änderungen sofort auf das verbundene Gerät synchronisiert werden. Man spricht hierbei von einer stetigen Synchronisation.

Über das Framework lassen sich auch Dateien zwischen den Geräten austauschen. Das Framework bietet auch die Möglichkeit, eigene nicht-Task-Daten zu synchronisieren. Dies erfordert nur geringen Mehraufwand vom Entwickler. So liessen sich auch Termine, Kontakte oder andere Daten synchronisieren.

5 Ausblick

Das Framework bietet sich nun an eine ausgereifere Applikation zu bauen, mit welcher Tasks oder auch andere Daten verwaltet und synchronisiert werden können. Die Sicherheit wurde nicht im Rahmen dieser Studienarbeit betrachtet und könnte bei Weiterentwicklungen implementiert werden. Da die Übertragung in Form von XML Paketen stattfindet, wäre es möglich auch unter anderen Technologien eine Gegenstelle für das Framework zu erstellen. Dies würde jedoch erfordern, dass die ganze Logik erneut geschrieben werden müsste.



Patrick Boos



Markus Kolb

Betreuer: Thomas Letsch

Projektpartner: -

Aufgabenstellung

- Synchronisationsframework für Tasks entwickeln
- Transfer von Files

Anforderungen

- Stetige Synchronisation
- Grosse Datenmenge

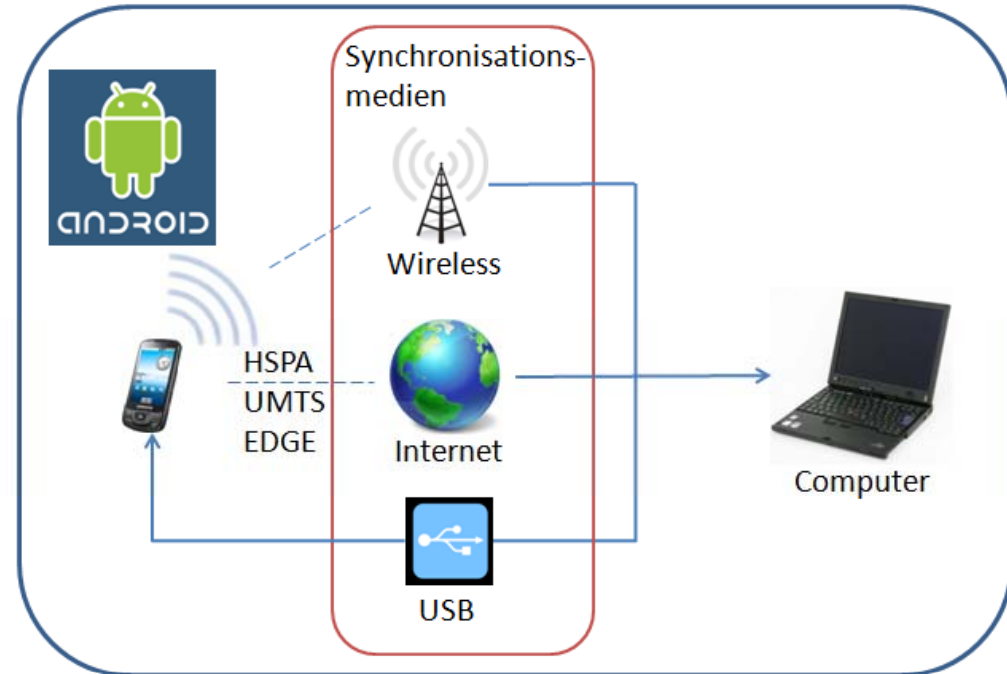
Technologien

- Android 1.6
- Java
- SQLite

Vorgehensmodell

- Rational Unified Process (RUP)

Systemübersicht



Ergebnis

- Synchronisationsframework
- Demo Projekt auf Android Handy und Computer
- Development Guide
- Eigene Nicht-Task-Objekte synchronisieren

Ausblick

- Weiterentwicklungen auf dem Framework

Technischer Bericht

Version 1.0

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

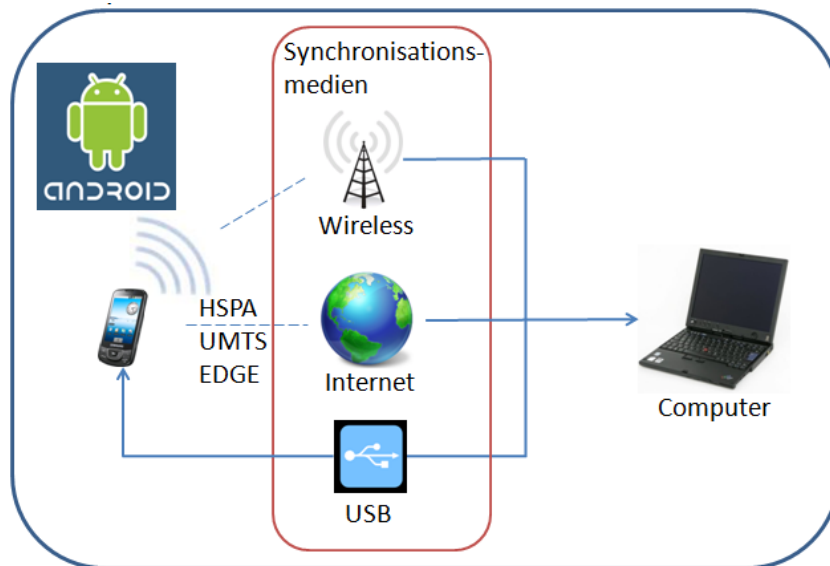
Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	14.12.2009	Erstellen des Technischen Berichts	MK
1.0rc02	In Bearbeitung	15.12.2009	Hinzufügen des Inhaltes	PB
1.0	Release	16.12.2009	Stabile Version 1.0	TEAM

Inhaltsverzeichnis

1	Einführung und Übersicht	4
1.1	USB-Kommunikation.....	4
1.2	Funktionalitäten.....	4
1.3	Übrige Teile der Abgabe	4
2	Ergebnisse.....	4
2.1	Das Framework.....	4
2.1.1	Auf Android Smart-Phone und Computer lauffähig	5
2.1.2	Kommunikation durch XML-Pakete	5
2.2	Erkenntnisse.....	6
2.2.1	Kommunikation über USB	6
3	Schlussfolgerungen	6
3.1	Erreichte Ziele	6
3.2	Nicht erreichte Ziele und Unschönheiten.....	6
3.3	Ursachen	6
3.4	Vergleich mit anderen Lösungen.....	6
3.4.1	HTC Sync	6
3.4.2	Synchronisation über Internet	7
3.5	Ausblick	7

1 Einführung und Übersicht

Da die Mobilkommunikation heutzutage nicht mehr wegzudenken ist, wurde ein Task-Management-Framework für Android entwickelt. Auf dem entwickelten Framework sollen Entwickler die Möglichkeit haben eine Task-Management-Applikation zu entwickeln welche auf einem Android Smart-Phone oder einem Computer läuft. Die darin erfassten Tasks sollen ohne grossen Aufwand zwischen Computer und Smart-Phone über USB oder das Internet synchronisiert werden können.



1.1 USB-Kommunikation

Am Beginn der Arbeit war noch unklar, ob und wie ein Austausch von Daten über USB möglich ist. Dies musste zu Beginn abgeklärt werden.

1.2 Funktionalitäten

Das Framework sollte die Funktionalität bieten Tasks auf Befehl/Knopfdruck zu synchronisieren und dann gegebenenfalls in einen Listen-Mode zu gehen, in welchem die ab dann veränderten/hinzugefügten/gelöschten Tasks automatisch und sofort synchronisiert werden.

Ausserdem sollte es möglich sein eine Datei von der einen Seite an die andere zu senden.

1.3 Übrige Teile der Abgabe

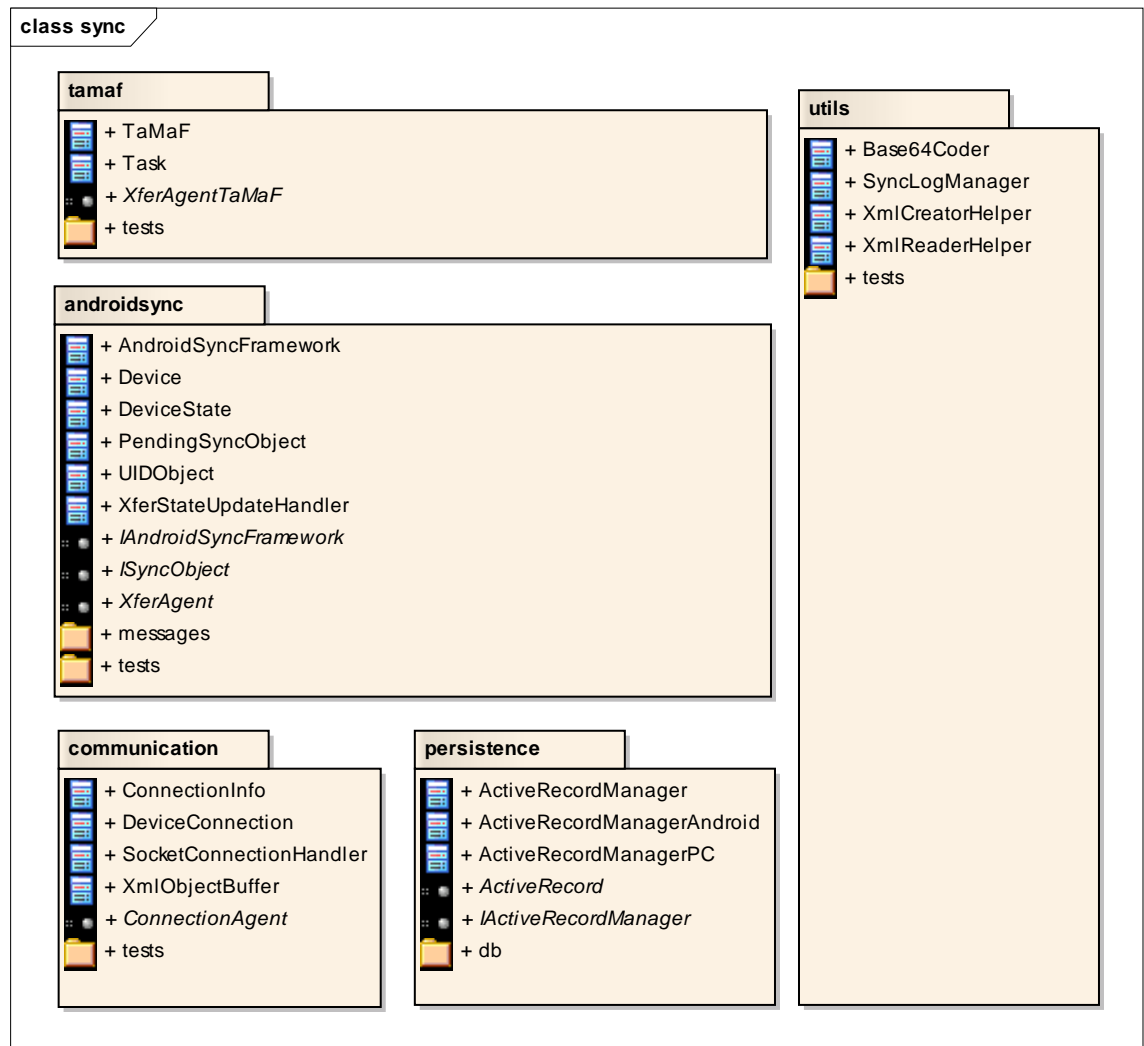
Die Abgabe besteht aus dem Framework selbst sowie je einem Demo-Projekt für den Computer und für das Android Smart-Phone.

Ein beigefügter Developer Guide soll Programmierern helfen auf dem Framework eine Applikation zu entwickeln.

2 Ergebnisse

2.1 Das Framework

Das erstellte Framework baut auf mehreren Layern auf. Der wichtigste Layer darin ist der AndroidSync-Layer in welchem die Logik der Synchronisation enthalten ist. Ta-MaF ist das Task-Management-Framework welches nur ein Adapter auf den AndroidSync-Layer ist. Dies kann in der folgenden Übersicht der Packages gesehen werden.



Der AndroidSync-Layer bietet die Funktionalität beliebige Objekte, welche bestimmte Richtlinien erfüllen, zu synchronisieren. Der TaMaF-Layer schränkt diese Funktionalität zur Vereinfachung ein und bietet die Task-Klasse an.

2.1.1 Auf Android Smart-Phone und Computer lauffähig

Dasselbe Framework läuft auf dem Android Smart-Phone sowie auf dem Computer. Damit dies erreicht werden konnte mussten Teile so geschrieben werden, dass es die auf beiden Systemen zur Verfügung stehenden Funktionen nutzt anstelle von einfacheren Funktionen, die nur auf einem System funktionieren.

Beispiel ist hier die Persistence durch SQLite. Hier unterscheidet sich die Funktionalität vollkommen. Damit aber im selben Framework beides vorhanden ist, wurde hier das ActiveRecord-Pattern verknüpft mit dem Proxy-Pattern verwendet. Der Proxy weiss um welches System es sich handelt und ruft entsprechend die richtige Implementation auf.

2.1.2 Kommunikation durch XML-Pakete

Es musste herausgearbeitet werden, wie die Kommunikation geschehen soll. Es boten sich mehrere Möglichkeiten an. So hätte man das ganze Serialisieren können.

Es wurde entschieden die ganze Kommunikation in Form von XML Paketen zu gestalten. Daraus ergeben sich mehrere Vorteile und wenige Nachteile.

Vorteile:

- Möglichkeit unter anderen Programmiersprachen eine TaMaF-Implementation zu schreiben. Wenn die Objekte einfach serialisiert worden wären, wäre dies nicht, oder nur sehr schwer, möglich.
- Es lässt sich gut in Wireshark nachverfolgen, was genau ausgetauscht wurde.
- Klarer Einblick darin, was übertragen wird und wie es übertragen wird.

Nachteile:

- Die Nachrichten fallen grösser aus. Dies ist jedoch ein Nachteil, der in Kauf genommen wird. Da heute die Netzwerkverbindungen, auch über das Internet, grosse Bandbreiten anbieten wird dieser Nachteil nicht gross ins Gewicht fallen.

2.2 Erkenntnisse

2.2.1 Kommunikation über USB

Leider bietet das originale System von Android keine eingebaute Kommunikationsmöglichkeit über USB. Ob dies später noch eingebaut wird ist derzeit nicht klar, wird aber von einigen Entwicklern gewünscht.

Aus diesem Grunde musste auf Developer Tools zugegriffen werden, welche diese Möglichkeit bieten. Durch die ADB (Android Debug Bridge) ist es möglich einen Port vom lokalen Computer auf das Handy weiterzuleiten. So kann dann darüber eine TCP-Verbindung aufgebaut werden. Damit dies möglich ist müssen die Treiber für das Handy installiert sein, welche vom Android SDK geliefert werden.

3 Schlussfolgerungen

3.1 Erreichte Ziele

- Synchronisation von Task Objekten
- Synchronisation von eigenen Objekten
- Stetige Synchronisation (Listen-Mode)
- Übertragung einer Datei
- Demo-Projekte für Computer und Smart-Phone

3.2 Nicht erreichte Ziele und Unschönheiten

- Sinnvolle Generierung der HostID
- Unschönheiten im Interface

3.3 Ursachen

Hauptursache für die Nicht erreichten Ziele ist der Mangel an Zeit.

Die Unschönheiten des Interfaces kristallisierten sich erst gegen Ende des Projektes heraus. Grund dafür ist, dass der genaue Einsatz des Frameworks

3.4 Vergleich mit anderen Lösungen

Es gibt andere Lösungen, welche sich jedoch auf ihre Bereiche beschränken. Keine von den gefundenen Lösungen bietet es an eine stetige Synchronisation aufrecht zu erhalten (Listen-Mode).

3.4.1 HTC Sync

HTC hat eine Synchronisierung über USB im System eingebaut. Diese ist jedoch nicht standardmässig in Android vorhanden. Deswegen kann dies auch nur auf den entsprechenden Geräten von HTC genutzt werden. Somit sind andere Geräte davon ausgeschlossen.

3.4.2 Synchronisation über Internet

Es gibt mehrere Applikationen, welche Daten über das Internet synchronisieren. Als ein Beispiel ist hier die „Remember the milk“ Applikation, welche Tasks mit der „Remember the milk“ Website synchronisiert in beide Richtungen synchronisiert.

Keine dieser Lösungen bietet jedoch die direkte Synchronisation mit dem Computer über USB.

3.5 Ausblick

Das Framework kann nun benutzt werden um eine Applikation zu bauen, welche Daten verwaltet und mit einem anderen Gerät synchronisiert. Es ist jedoch empfehlenswert erst die im Design Dokument vorgeschlagenen Verbesserungen durchzuführen. Es ist zwar möglich ohne diese eine Applikation zu bauen, doch würde dies durch die Verbesserungen erleichtert.

Persönlicher Bericht

Patrick Boos

1 Einleitung

Die Studienarbeit war eine lehrreiche Zeit da ich viele der gelernten Thematiken anwenden konnte. Die Thematik interessierte mich bereits seit Anfang der Arbeit und ich konnte mich deshalb gut für die Arbeit begeistern.

2 Ablauf der Arbeit

Der Anfang des Projektes schien mir etwas langatmig gewesen zu sein. Es wurden viele Dokumente geschrieben. Ich bin selbst meistens kein begeisterter Dokumenteschreiber. Trotzdem muss ich eingestehen, dass ich dabei einiges gelernt habe. Vor allem der Grund hinter den Dokumenten schien mir besser klar zu werden als während des SE2-Projektes. Auch wenn es weniger Dokumente waren, waren es meiner Meinung nach noch zu viele Dokumente. So waren die ersten 3-4 Wochen hauptsächlich Dokumente. Nebenbei wurde sich die API von Android eingearbeitet und erste Tests durchgeführt ob und wie die Umsetzung der Arbeit möglich ist.

Es stellte sich als schwierig heraus gemeinsam mit Herr Letsch die gewünschte Funktionsweise des Frameworks und die Interfaces zu definieren. So wurde dafür sehr viel Zeit in Sitzungen investiert. Es schien danach eigentlich klar zu sein, wie die Interfaces auszusehen haben. Während der Entwicklung der Demo Projekte stellte sich dann jedoch erst gegen Schluss heraus, dass es teils Unschönheiten im Interface gibt. Es ist zwar möglich es so zu machen, jedoch unschön. Wegen nicht vorhandener Zeit um alles so zu ändern wie es gut wäre, wurde es so gelassen, wie abgesprochen. Gelernt habe ich dadurch einige Punkte, welche wichtig sind beim besprechen des Interfaces. Wir haben zwar die Funktionalitäten des Interfaces beachtet, aber etwas zu wenig, wie dieses genau von der Applikation genutzt werden wird.

3 Nachforschungen

Was ich in der Studienarbeit etwas vermisst habe war das einarbeiten in die Lektüre. Es schien kaum Zeit dafür vorhanden zu sein sich in die Thematik der Synchronisation einzulesen. So wäre es meiner Meinung nach gut gewesen, wenn man sich mehrere bestehende Lösungen auf anderen Systemen hätte ansehen können. So zum Beispiel den SyncML Standard. Jedoch fand ich es interessant und herausfordernd eine Möglichkeit zu finden, wie die Daten über USB ausgetauscht werden können.

4 Teamarbeit

Von Anfang des Projektes an war klar, dass mein Teamkollege etwas weniger Erfahrung im Programmieren hat. So durfte ich öfters auch meinem Teamkollegen helfen, was mir ebenfalls Erfahrung brachte.

Wegen der unterschiedlichen Wissensstände und der Unterschiedlichen Stundenpläne war die Zusammenarbeit etwas erschwert. Hilfreich war es dabei, dass wir ab der zweiten Hälfte des Semesters öfters bei meinem Teamkollegen zu Hause am Abend gemeinsam an der Arbeit gearbeitet haben und somit allfällige Fragen schnell beantwortet werden konnten.

Persönlicher Bericht

Markus Kolb

1 Einleitung

Da ich ein begeisterter Windows Mobile Benutzer bin, wollte ich mein Horizont im Bereich Mobilkommunikation erweitert. Android hat klar im Jahr 2009 an Marktanteil zugenommen, trotz gesammelten Erfahrungen, tendiere ich beim nächsten Kauf eines Handys wieder zu Windows Mobile.

2 Ablauf der Arbeit

Anfangs der Arbeit wurde viel Zeit in die Dokumente und Meetings gesteckt, was in meinem Erachten, gar nicht mal so schlecht war. Wir haben viele Problematiken in den Teammeetings ausgemerzt und hatten so auch dieselbe Vorstellung der Lösung.

Neben den Dokumenten haben wir uns in die Android Entwicklungsumgebung eingearbeitet, was meines Erachtens, zu wenig Zeit vorhanden war. Vor allem widmete ich mich der Persistenz beim Projekt, was viele Fragen aufgeworfen hat. Einer der Kernentscheidungen war, dass wir uns entschieden haben, das Netzwerk flexibel aufzubauen, was vermutlich das ganze Projekt ein wenig komplizierter gestaltete. Beim erarbeiten des Demo Projektes auf dem PC, unterstütze mich mein Partner, indem er mir das Command Pattern zeigte.

3 Nachforschungen

Nachforschungen wurden vor allem betrieben im Bereich Android. Jedoch war keine Zeit vorhanden, andere Sync-Frameworks zu untersuchen.

4 Teamarbeit

Da ich aus dem Server/Netzwerk Bereich komme, habe ich wenige Erfahrungen im Bereich Programmieren. Während der Arbeit konnte mein Partner mich in verschiedenen Bereichen unterstützen, vor allem wenn es um Entwurfsmuster (Pattern) ging.

Unsere Stundenpläne waren nicht wirklich perfekt aufeinander abgestimmt und so war es währenddem Semester teilweise schwierig sich miteinander abzusprechen. Da ich ein Gästezimmer zuhause zur Verfügung habe, habe ich ab zirka Mitte des Semesters jeweils am Dienstag meinem Projektpartner Asyl gewährt. So konnten wir ungestört am Projekt arbeiten, was uns stark vorangetrieben hat.

Glossar

Version 1.1

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	18.09.2009	Erstellen des Glossars	PB
1.0rc02	In Bearbeitung	30.09.2009	Begriff Listen-Mode hinzugefügt	PB
1.0rc03	In Bearbeitung	03.10.2009	Abkürzungen für die Dokumente hinzugefügt	MK
1.0rc04	In Bearbeitung	06.10.2009	Abkürzung TaMaF hinzugefügt	PB
1.0rc05	In Bearbeitung	09.10.2009	Anpassung laut Besprechung	PB
1.0rc06	In Bearbeitung	09.10.2009	Dry-Konzept Hinzugefügt	MK
1.0rc07	In Bearbeitung	12.10.2009	Socket hinzugefügt	PB
1.0rc08	In Bearbeitung	18.10.2009	Korrektur vorgenommen	MK
1.0	Release	04.10.2009	Stabile Version 1.0	TEAM
1.1rc01	In Bearbeitung	30.11.2009	Begriffe für das GUI Release hinzugefügt	PB
1.1rc02	In Bearbeitung	05.12.2009	ABC Anordnung hinzugefügt	MK
1.1rc03	In Bearbeitung	07.12.2009	DG – Developer Guide hinzugefügt	PB
1.1rc04	In Bearbeitung	15.12.2009	Hinzufügen von HostID und UID	PB
1.1rc05	In Bearbeitung	15.12.2009	Weitere Dokumentabkürzungen hinzugefügt	PB
1.1rc06	In Bearbeitung	16.12.2009	Hinzufügen von XML und Port forwarding	PB
1.1	Release	16.10.2009	Stabile Version 1.1	TEAM

Inhaltsverzeichnis

1	Einführung	3
1.1	Zweck.....	3
1.2	Gültigkeitsbereich.....	3
1.3	Definitionen und Abkürzungen.....	3
1.4	Referenzen.....	3
2	Glossar	4

1 Einführung

1.1 Zweck

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

Name	Kürzel
Markus Kolb	MK
Patrick Boos	PB

1.4 Referenzen

Referenz	Quelle

2 Glossar

Begriff	Beschreibung
AB	Dokument: Abstract / Kurzfassung
ADB	Android Debug Bridge http://developer.android.com/guide/developing/tools/adb.html Die ADB ist ein Tool welches mit der Android SDK kommt, welches eine direkte Kommunikation und Kontrolle des Androidgerätes erlaubt.
Android	Android ist ein Betriebssystem sowie auch eine Software-Plattform für mobile Geräte wie Smartphones, Mobiltelefone und Netbooks, die von der Open Handset Alliance entwickelt wird.
AS	Dokument: Anforderungsspezifikation
DA	Dokument: Domain Analyse
DD	Dokument: Design Dokumentation
Device	Als Device wird ein Gerät verstanden auf welchem ebenfalls TaMaF läuft und mit welchem sich die Applikation verbinden lässt.
DG	Dokument: Developer Guide
Download	Handy -> Computer
DRY- Konzept	„Don't repeat yourself“ Keinen doppelten Text erfassen
EA	Enterprise Architect
Eclipse	Entwicklungsumgebung http://www.eclipse.org
EEA	Dokument: Erklärung über eigenständige Arbeit
GL	Dokument: Glossar
HostID	Eindeutige Identifikations-Nummer eines Gerätes, auf welchem TaMaF läuft.
IV	Dokument: Inhaltsverzeichnis
Java	Programmiersprache Java
Listen-Mode	Im Listen-Mode sind die beiden Endpunkte miteinander verbunden und Änderungen werden sofort übertragen/synchronisiert. Dies heisst es besteht eine stetige Synchronisation.
LV	Dokument: Literaturverzeichnis
MK	Markus Kolb
MS	Dokument: Management Summary
PB	Patrick Boos
PBPB	Dokument: Persönlicher Bericht von Patrick Boos
PBMK	Dokument: Persönlicher Bericht von Markus Kolb

Port forwarding	Eine Funktionalität der ADB, welche einen lokalen Port auf das Handy, welches per USB angeschlossen ist, weiterleitet.
PP	Dokument: Projektplan
SAD	Dokument: Software Architecture Document
Socket	Das Socket.
SQLite	SQL database engine. http://www.sqlite.org/
Subtask	Ein Subtask ist ein Task, welcher zu einem Task gehört. Er ist einem anderen Task untergeordnet.
TaMaF	Abkürzung für Task-Management-Framework. Dies wirkt auch als Kürzel für das Projekt.
Task	Ein Task ist eine Aufgabe, welche es zu erledigen gilt. Diese werden in der Applikation welche auf TaMaF aufsetzt verwaltet.
Task-Management	Das Verwalten von Aufgaben, wie dies z.B. in Microsoft Outlook möglich ist.
TB	Dokument: Technischer Bericht
TD	Dokument: Testdokumentation
UID	Unique Identifier Dieser wird im Umfang dieses Projekts für Objekte benutzt, welche Synchronisiert werden. Jedes Objekt hat eine eindeutige UID. UID ist 64bit lang und besteht aus der HostID auf welchem das Objekt erstellt wurde und einer ansteigenden Nummer.
Upload	Computer -> Handy
XML	Extensible Markup Language ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Datensätze in Form von Textdaten. http://de.wikipedia.org/wiki/Extensible_Markup_Language Nachrichten werden in diesem Format über das Netzwerk versendet.
ZE	Dokument: Zeiterfassung

Literatur Verzeichnis

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Inhaltsverzeichnis

1	Verzeichnis	2
---	-------------------	---

1 Verzeichnis

Beschreibung
Active Record Pattern, Quelle: Vorlesung User Interface 1
Android Debug Bridge, Quelle: http://developer.android.com/guide/developing/tools/adb.html
Android Development, Quelle: http://developer.android.com/index.html
Android SQLite, Quelle: http://www.anddev.org/working_with_the_sqlite-database_-_cursors-t319.html

Projektplan

Version 1.3

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	15.09.2009	Erstellen des Projektplans	MK
1.0rc02	In Bearbeitung	18.09.2009	Projektorganisation	MK
1.0rc03	In Bearbeitung	21.09.2009	Management Abläufe	MK
1.0rc04	In Bearbeitung	21.09.2009	Risiko Management	PB
1.0rc05	In Bearbeitung	21.09.2009	Qualitätsmassnahmen geschrieben	MK
1.0rc06	Review	22.09.2009	Korrekturen Vorgenommen	PB
1.0rc07	In Bearbeitung	30.09.2009	Arbeitspakete definiert, Projektplan angepasst	MK
1.0rc08	Review	30.09.2009	Korrekturen gemäss Meeting	MK
1.0rc09	In Bearbeitung	04.10.2009	Weitere Korrekturen gemäss Meeting	MK
1.0rc10	In Bearbeitung	09.10.2009	Korrekturen gemäss Meeting vom 8.10	MK
1.0	Release	10.10.2009	Stabile Version 1.0	TEAM
1.1rc01	In Bearbeitung	16.10.2009	Update auf 1.1 & Korrekturen vorgenommen	MK
1.1rc02	In Bearbeitung	18.10.2009	Terminplanung bis Communication Release	MK
1.1rc03	In Bearbeitung	26.10.2009	Korrekturen gemäss Meeting vom 22.10	PB
1.1rc04	In Bearbeitung	28.10.2009	Korrekturen gemäss Meeting vom 22.10	MK
1.1rc05	In Bearbeitung	02.11.2009	Korrekturen Logging	PB
1.1rc06	In Bearbeitung	02.11.2009	Korrekturen gemäss Meeting vom 29.10	MK
1.1	Release	03.11.2009	Stabile Version 1.1	TEAM
1.1.1	Release	05.11.2009	Stabile Version 1.1.1	TEAM
1.2rc01	In Bearbeitung	30.11.2009	Anpassungen für GUI Release	PB
1.2rc02	In Bearbeitung	01.12.2009	Weitere Anpassung im Dokument	MK
1.2rc03	In Bearbeitung	05.12.2009	Korrekturen vorgenommen	MK
1.2rc04	In Bearbeitung	09.12.2009	Anpassung der Arbeitspakete	PB
1.2rc05	In Bearbeitung	15.12.2009	Kleine Korrekturen	PB
1.2	Release	15.12.2009	Stabile Version 1.2	TEAM
1.3	Release	16.12.2009	Stabile Version 1.3	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	Projekt Übersicht	5
2.1	Zweck und Ziel.....	5
2.2	Annahmen und Einschränkungen	5
3	Projektorganisation.....	6
3.1	Organisationsstruktur	6
3.2	Externe Schnittstellen.....	6
4	Management Abläufe.....	7
4.1	Projekt Kostenvoranschlag	7
4.2	Besprechungen.....	7
4.2.1	Mit Betreuer.....	7
4.2.2	Team-Besprechungen.....	7
4.3	Vorgegeben Dokumente bei der Studienarbeit	7
4.4	Projektplan	8
4.4.1	Zeitplan	8
4.4.2	Beschreibung der Dokument Inhalte.....	9
4.4.3	Meilensteine.....	11
4.4.4	Iterationsplanung.....	11
5	Risiko Management.....	13
6	Infrastruktur	14
7	Qualitätsmassnahmen	15
7.1	Teammeetings / Sitzungsprotokolle	15
7.1.1	Unit-Tests	15
7.2	Reviews.....	15
7.2.1	Code-Reviews	15
7.2.2	Dokumenten-Reviews	15
7.3	Logging	15
7.3.1	Levels	15
7.3.2	Dichtes Logging	16
8	Arbeitspakte	17
9	Anhang	22
9.1	Tabellenverzeichnis.....	22

1 Einführung

1.1 Zweck

Der Projektplan der Semesterarbeit wird in diesem Dokument beschrieben. Mit diesem Dokument soll gemäss Vorstellung des Projektleiters weitergearbeitet werden können, falls dieser ausfällt.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

Siehe Dokument Glossar.

1.4 Referenzen

Referenz	Quelle

2 Projekt Übersicht

Das Synchronisations-Framework für Smart-Phone bietet die Möglichkeit, weitere darauf aufbauende Applikationen zu synchronisieren. Das spezielle an diesem Framework ist, dass während dem der Computer mit dem Handy verbunden ist, automatisch die Updates vom Computer auf das Handy geladen werden. Somit muss man vor dem abhängen des Handys nicht noch erst Synchronisieren. Das Handy kann mit mehreren Computern synchronisiert werden.

Um die Demonstration zu bewerkstelligen, wird hier ein kleines darauf aufbauendes Task-Management Programm mit GUI entwickelt. In diesem Programm soll ersichtlich werden, was für Funktionen das Synchronisations-Framework bietet.

Zusätzlich zur Demonstration wird ein Development Guide zur Verfügung gestellt, in welchem beschrieben wird, wie weitere darauf aufbauende Applikationen entwickelt werden können. Wichtig am Development Guide ist, dass die Entwickler möglichst schnell und einfach, das Synchronisations-Framework nutzen können, um Weiterentwicklungen zu tätigen.

2.1 Zweck und Ziel

Das Ziel ist eine kundenspezifische Lösung zu entwickeln, welche ein Synchronisations-Framework zur Verfügung stellt.

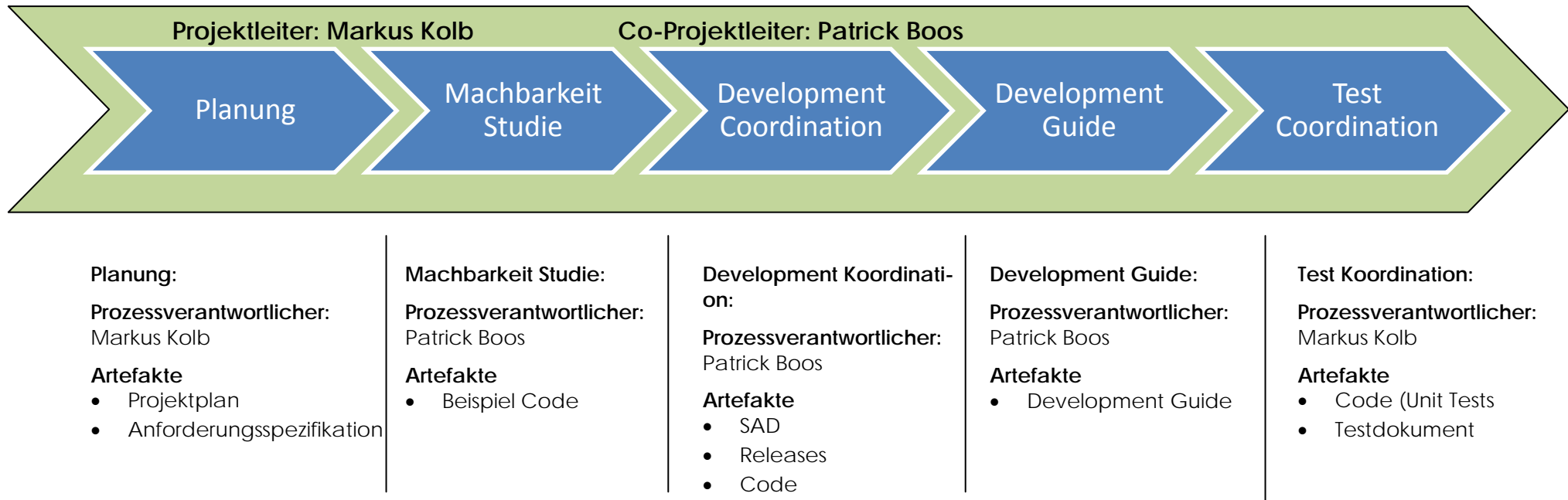
2.2 Annahmen und Einschränkungen

Die Soll-Arbeitszeit pro Projektmitarbeiter und Woche liegt bei ca. 17 Stunden. Bei unerwarteten Problemen und Aufwänden wird die Arbeitszeit soweit nötig ausgeweitet.

3 Projektorganisation

3.1 Organisationsstruktur

Ablauforganisation:



3.2 Externe Schnittstellen

Thomas Letsch

Betreuung, Beratung, Kontrolle & Kunde

4 Management Abläufe

4.1 Projekt Kostenvoranschlag

Der Umfang dieses Projekts beträgt pro Person 240 Stunden. Da das Projekt aus zwei Mitgliedern besteht, werden insgesamt ca. 480 Stunden aufgewendet. Das Projekt dauert vom 14. September 2009 bis 18. Dezember 2009.

4.2 Besprechungen

4.2.1 Mit Betreuer

Teilnehmer	MK, PB, TL
Datum/Zeit	Jeden Donnerstag, 17:10
Ort	HSR Rapperswil (Standard: Zimmer 5.207)

4.2.2 Team-Besprechungen

Teilnehmer	MK, PB
Datum/Zeit	Wöchentlich Dienstag 15:00 und Freitag 13:00
Ort	HSR Rapperswil Zimmer 1.258
Zweck	ToDo-Liste durchsehen/updates, anstehende Arbeit besprechen.

4.3 Vorgegeben Dokumente bei der Studienarbeit

Folgende Dokumente müssen bei jeder Studienarbeit erstellt werden:

Durch die Dokumente Abstract, Management Summary & Technischer Bericht wird das Dry-Konzept verletzt. Aufgrund der Vorgaben der Abteilung, wird dies bewusst gemacht.

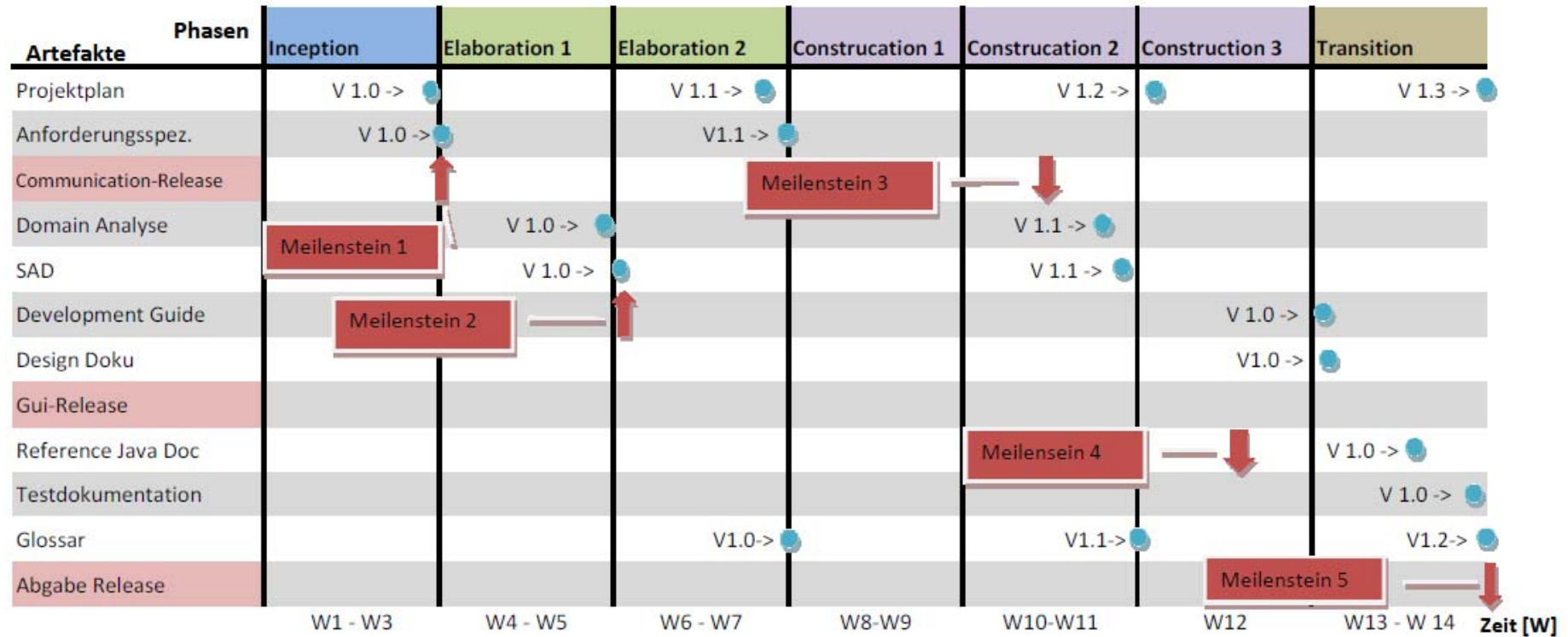
Name	Inhalt	Abgabe
Abstract	Zusammenfassung der Arbeit auf einer Seite, auf dem dafür vorgegebenen Formular	14.12.09
Management Summary	Das "Management Summary" richtet sich in der Praxis an die "Chefs des Chefs", d.h. an die Vorgesetzten des Auftraggebers (diese sind in der Regel keine Fachspezialisten)	14.12.09
Technischer Bericht	Umfasst eine Einleitung, Übersicht, Ergebnisse & Schlussfolgerungen	Am Ende
Persönliche Berichte	Persönliche Erfahrungen bei der Arbeit.	Am Ende
Literaturverzeichnis	Im Literaturverzeichnis sind alle verwendeten Quellen (Bücher, Publikationen) aufgelistet	Am Ende
Dokumente des Projekts	Enthält eine Übersicht über die übrigen Dokumente gemäss Aufgabenstellung bzw. Software-Engineering-Vorgehen.	Am Ende
A0-Poster	Enthält eine Übersicht über das Projekt.	14.12.09

Tabelle 1: Vorgegebene Dokumente

4.4 Projektplan

4.4.1 Zeitplan

In diesem Zeitplan ist zurzeit nur die Reihenfolge definiert.



4.4.2 Beschreibung der Dokument Inhalte

Dieses Kapitel beschreibt die Dokumentversionen und Inhalt, welche dem Kunden zur Verfügung gestellt werden.

Projektplan

Der Projektplan enthält die Anforderungen an das Projekt. Es enthält die Organisation des Projektes, das Risikomanagement und die Qualitätssicherungsmassnahmen.

Version	Inhalt
1.0	Der Projektplan umfasst die Planung bis zum Meilenstein 2. Alle weiteren Termine sind offen. (bis auf MS5)
1.1	Der Projektplan umfasst alles bis und mit Communication Release.
1.2	Enthält zeitlich alles bis und mit dem GUI Release.
1.3	Umfasst das ganze Projekt und sollte der letzte Projektplan Release sein.

Tabelle 2: Projektplan

Anforderungsspezifikation

Enthält die Anforderungen an das Produkt. Dabei werden funktionale und nicht funktionale Anforderungen unterschieden.

Version	Inhalt
1.0	Die Anforderungsspezifikationen enthält die Anforderungen an das Produkt bis zum Communication Release.
1.1	Weitere Anforderungsspezifikationen für das GUI werden in dieser Version erfasst.
1.2	Weitere Anforderungsspezifikationen für die Technischen Dokumente (Development Guide), werden in diesem Dokument niedergeschrieben.

Tabelle 3: Anforderungsspezifikation

Domain Analyse

Als Erstes wird durch ein Domain Modell eine erste Übersicht über die Problem Domain verschafft. Danach werden die Use Cases der Anforderungsspezifikation als Systemsequenzdiagramme dargestellt und dessen Systemoperationen genauer identifiziert.

Version	Inhalt
1.0	Das Domain Model wird gemäss Larman erfasst. Diese Version umfasst alles bis und mit Communication Release.
1.1	Die Domain Analyse wird ergänzt für den GUI Release.

Tabelle 4: Domain Analyse

SAD (Software Architecture Document)

Zu Beginn des Dokuments wird eine kurze Übersicht über das ganze System gegeben und die diversen architektonischen Entscheidungen & Konzepte erläutert.

Danach gliedert sich das Dokument ganz nach dem „N+1“ View Model gemäss Larman und ist in verschiedene Sichten unterteilt.

Version	Inhalt
1.0	Enthält eine Archidekturübersicht, die Unterteilung in die Subsysteme & die Schichten gemäss „N+1“ Viewmodel.
1.1	1.0 Release wird Ergänzt vor allem auch durch die GUI Entwicklungen.

Tabelle 5:: SAD

Design Dokumentation

Enthält die Informationen über das Design der Applikation. Ein wichtiger Bestandteil der Dokumentation sind die Design Entscheide.

Version	Inhalt
1.0	Das Dokument beschreibt das Design der Applikation. Pro Package ist ein Klassendiagramm vorhanden, wo sämtliche Elemente ersichtlich sind.

Tabelle 6: Design Dokumentation

Development Guide

Im Development Guide wird beschrieben, wie mit der Bestehenden Applikationen Weiterentwicklungen getätigt werden können.

Version	Inhalt
1.0	Einzige Version des Dokumentes.

Tabelle 7: Development Guide

Reference Java Doc

Wird aus den in-Code Kommentaren generiert. Soll als weitere Information zum Development Guide dienen.

Version	Inhalt
1.0	Beschreibung des Codes

Tabelle 8: Reference Java Doc

Glossar

Das Glossar umfasst alle Abkürzungen und Erklärungen von schweren Begriffen.

Version	Inhalt
1.0	Erste Version des Dokumentes
1.1	Glossar wird weiter ergänzt.
1.2	Glossar in seiner Schluss-Version.

Tabelle 9: Glossar

4.4.3 Meilensteine

In der folgenden Tabelle werden die einzelnen Meilensteine kurz beschrieben. Eine detaillierte Planung kann im Dokument 02_Projektplan\Zeitplan.xlsx nachgeschlagen werden.

Die Dokumente, welche zu welchem Meilenstein fertig sein sollen, können dem Zeitplan entnommen werden.

Bei jedem Meilenstein wird ein Release Paper abgegeben, indem spezielle Konfigurationen erläutert werden.

MS	Bezeichnung	Beschreibung	Datum
MS1	Projektplan & Anforderungsspezifikation Review	Das Projekt ist soweit geplant und die Anforderungen soweit abgesteckt, dass für das Framework die nötigen Informationen vorhanden sind. Bis und mit SAD ist geplant, die Reihenfolge ist definiert.	10.10.09
MS2	Domain Analyse & SAD Review	Die Entwicklung und Architektur für das Framework ist klar.	17.10.09
MS3	Communication-Release	Erster Prototyp, der die Kommunikation zwischen Handy und Computer zeigt. Framework sollte grundlegend fertig sein. Ebenfalls wurde für das GUI geplant.	18.11.09 07:00 Uhr
MS4	GUI-Release	Erster Prototyp mit GUI für Task-Management. Der Code ist synchron mit Enterprise Architect.	02.12.09 12:00 Uhr
MS5	Abgabe Release	Beinhaltet das Release einer funktionierenden Software, die die Synchronisation über das Framework demonstriert. Das Design des Frameworks wurde in dem dafür vorgesehenen Dokument dokumentiert. Ebenfalls wurde das Framework so dokumentiert, dass Entwickler, die darauf ihre Anwendungen bauen, die nötigen Informationen dazu haben.	18.12.09

Tabelle 10: Meilensteinübersicht

4.4.4 Iterationsplanung

Der Unified Process definiert vier Phasen. Diese vier Phasen werden in mehreren Iterationen durchlaufen. Die folgende Tabelle zeigt eine Übersicht über die einzelnen Iterationen und beschreibt diese mit einigen Stichworten.

Iteration	Beschreibung	Ende	Dauer [W]
Inception 1	Projektplan & Anforderungsspezifikation der Version 1.0 wird geschrieben, zusätzlich wird getestet ob das Projekt realisierbar ist.	W03	3

Elaboration 1	In dieser Phase wird vor allem am SAD & Domain Analyse gearbeitet.	W05	2
Elaboration 2	Anforderungen für den GUI Release sind erfasst. Projektplan enthält Planung bis Communication Release.	W07	2
Construction 1	Der Fokus liegt auf dem Communication Release.	W09	2
Construction 2	Fertigstellung des Communication Release (W10).	W11	2
Construction 3	Fertigstellung des GUI Release. (W11 - W12)	W12	1
Transition 1	Verfassen der folgenden Dokumente: Design Dokument, Development Guide und der vorgegebenen Dokumente gemäss Studienarbeit. Zudem wird intensiv getestet und eine kleine Testdokumentation geschrieben.	W14	2

Tabelle 11: Übersicht der Iterationen

5 Risiko Management

Nr.	Risikotitel	Risikobeschreibung	max. Schaden	Wsk.	gew. Schaden	Massnahmen	Vorausgehende Massnahmen
1	ADB nicht möglich	Verbindung über ADB stellt sich als nicht möglich/ausreichend heraus.	20h	20%	4h	Andere Möglichkeit zur Verbindung suchen.	
2	Komplexität Synchronisation	Komplexität Synchronisation ist schwerer als angenommen	20h	10%	2	Versuchen das Ganze zu vereinfachen.	Über Synchronisation Informationen beschaffen und lesen.
3	Ausfall Entwicklungsrechner	Ein Rechner, auf dem Entwickelt wird, fällt aus und kann zur Entwicklung nicht mehr benutzt werden.	8h	10%	0.8h	Auf Laptop/HSR-Rechner umsteigen. Reparatur/Ersatz beantragen.	Alle vier Stunden einchecken.
4	Kein Android-Handy	Ungerootetes Android Handy konnte nicht zur Verfügung gestellt werden.	10h	30%	3h	Personen mit einem solchen Handy anfragen, ob es für Tests benutzt werden kann.	

Tabelle 12: Risiko Management

6 Infrastruktur

Hardware	Software	Organisatorisches
Persönliche Notebooks	Subversion	HSR Räumlichkeiten
Computer HSR	Eclipse	
Linux-Server von PB	Microsoft Office	
Druckinfrastruktur HSR	Enterprise Architect	

Tabelle 13: Übersicht der genutzten Infrastruktur

7 Qualitätsmassnahmen

7.1 Teammeetings / Sitzungsprotokolle

Wie im Kapitel 4.2.2 beschrieben, organisiert das Team wöchentliche Meetings. Für das Meeting mit dem Betreuer muss ein Sitzungsprotokoll geschrieben werden, in dem alle Erkenntnisse und Entscheide aus dem Meeting sowie alle neu definierten Tasks protokolliert sind. So wird sichergestellt, dass alle Teammitglieder immer auf demselben Stand sind und auch alles mitbekommen, falls sie einmal nicht an einem Meeting teilnehmen können.

7.1.1 Unit-Tests

Jeder Entwickler ist aufgefordert an sinnvollen Stellen in seinem Source-Code fortlaufend Unit-Tests zu schreiben und gegen diese Tests zu entwickeln. Bei den Kernfunktionalitäten wird nach dem „Test-Driven-Development“ Verfahren gearbeitet. Bevor ein Entwickler seinen neuen Source Code ins SVN eincheckt, müssen alle seine Unit-Tests erfolgreich absolviert worden sein.

7.2 Reviews

7.2.1 Code-Reviews

Jeder Code muss mindestens einmal durch eine andere Person geprüft und wenn möglich verbessert werden. Der Revisor erstellt bei jedem Review ein Protokoll, wo er festhält, wann er welchen Code überarbeitet hat und welche Verbesserungen vorgenommen wurden. Zusätzlich muss jedes Review mit Datum, Revisor und Kommentar im Header der Source Datei erwähnt werden.

Der Revisor überprüft:

- Verständlichkeit des Codes (evtl. durch Kommentare ergänzen)
- Korrektheit des Codes
- Auftreten von „Code Smells“
- Effizienz von komplexen Algorithmen

7.2.2 Dokumenten-Reviews

Jedes geschriebene Dokument muss mindestens einmal von einer anderen Person durchgeschaut und wenn nötig verbessert werden. Der Revisor trägt sich in die „Revision History“ des entsprechenden Dokuments mit Datum und Kommentar ein.

Der Revisor überprüft:

- Rechtschreibung / Grammatik
- Einhalten des vorgegebenen Dokumenten Templates
- Konsistenz der verschiedenen Texte
- Vollständigkeit des Dokuments

7.3 Logging

Es wird ein Logging mit zweidimensionaler Laufzeit-Konfiguration eingesetzt. Dadurch kann eingestellt werden, welche Pakete mit welchem LogLevel geloggt werden sollen. Und ebenfalls wohin dies geloggt werden soll (per Handler).

7.3.1 Levels

Die folgenden Levels werden aus dem `java.util.logging` verwendet und über einen Logger von `java.util.logging` geloggt.

Die Levels, welche in diesem Projekt verwendet werden, sind jeweils fett und unterstrichen.

Level in Java	Einsatzgebiet
<u>Severe</u>	Wenn eine Situation auftritt, die Fehlerhaft ist. Daten können in einem inkonsistenten Zustand sein. Keine Garantie mehr, dass Daten richtig vorliegen. In dieser Situation können auch Daten verloren gegangen sein. Das Programm ist am weiterlaufen gehindert oder kann weiterlaufen, wobei dann jedoch unklar ist, was geschehen wird.
<u>Warning</u>	Eine Aktion oder ein Zustand wurde entdeckt, welche/r angesehen werden sollte, da dies zu einem Fehler führen könnte. Die Applikation kann jedoch noch weiter laufen.
<u>Info</u>	Eine Meldung einer normalen Aktion oder Events. Dies kann eine User-Operation sein. Beispiele dafür sind: „Verbindung hergestellt“, „Synchronisation gestartet“, „Synchronisation abgeschlossen“
Config	Wird nicht verwendet.
<u>Fine</u> <u>Finer</u> Finest	Trace oder Debug Nachrichten für Debugging und um die Performance zu überwachen. Meldungen wie z.B. „Task empfangen“.

Normalproduktionslevel: Info

7.3.2 Dichtes Logging

Bei dichtem Logging ist zuvor jeweils ein *if(logger.isLoggable(Level...))* zu machen, damit nicht unnötig die LogMeldung aufbereitet wird.

8 Arbeitspakete

1	Projekt Management	Verantwortlicher	125h
1.1	Meeting	MK	74h
Beschreibung	Wöchentliches Meeting, geschätzte Zeit 3 h pro Meeting		
Abhängigkeiten	-		
Risiken/Probleme	-		
1.2	Projektplan erstellen	MK	33h
Beschreibung	Projektorganisation festlegen, Meilensteine & Iterationen planen		
Abhängigkeiten	Meetings		
Risiken/Probleme	-		
1.3	Arbeitspakete definieren und planen	MK	4h
Beschreibung	Arbeitspakete unterteilen/verfeinern, Verantwortlichkeiten zuteilen		
Abhängigkeiten	[1.0] Projektplan		
Risiken/Probleme	-		
1.4	Infrastruktur Aufbau & Betrieb	PB	14h
Beschreibung	SVN Repository erstellen, Accounts erstellen		
Abhängigkeiten	-		
Risiken/Probleme	-		

Tabelle 14: Arbeitspakete der Disziplin Projekt Management

2	Requirements	Verantwortlicher	26h
2.1	Anforderungen definieren	MK	16h
Beschreibung	Definition der funktionalen und nicht funktionalen Anforderungen an das Produkt		
Abhängigkeiten	[1.0] Projektantrag		
Risiken/Probleme	-		
2.2	Use Cases	PB	10h
Beschreibung	aus funktionalen Anforderungen Use Cases erstellen		
Abhängigkeiten	-		
Risiken/Probleme	-		

Tabelle 15: Arbeitspakete der Disziplin Requirements

3	Analyse	Verantwortlicher	31h
3.1	Domainanalyse	PB	11h
Beschreibung	anhand Anforderungen die Domainanalyse erstellen		

Abhängigkeiten	[2.1] Anforderungen definieren		
Risiken/Probleme	-		
3.2	System Sequenzdiagramme	MK	12h
Beschreibung	Systemsequenzdiagramme der wichtigsten Interaktionen erstellen.		
Abhängigkeiten	-		
Risiken/Probleme	-		
3.3	System Contracts	PB	6h
Beschreibung	Dokumentieren der System Contracts		
Abhängigkeiten	[2.2] Use Cases		
Risiken/Probleme	-		
3.4	Validierung Analyse gemäss Requirements		2h
Beschreibung	Überprüfen, ob Analyse alle Requirements erfüllt		
Abhängigkeiten	Domainanalyse		
Risiken/Probleme	-		

Tabelle 16: Arbeitspakete der Disziplin Analyse

4	Design	Verantwortlicher	35h
4.1	Data Model	MK	2h
Beschreibung	Datenbankschema erstellen, Mappingstrategie beschreiben		
Abhängigkeiten	Domainmodel		
Risiken/Probleme	-		
4.2	Design Dokumentation	PB	10h
Beschreibung	Klassendiagramm, Objekt Interaktionsdiagramm, Package Diagramm etc. → Dokumente die das logische Design beschreiben		
Abhängigkeiten	[3.1] Domainanalyse		
Risiken/Probleme	-		
4.3	Software Architecture Document	MK	10h
Beschreibung	Das Software Architektur Dokument erstellen.		
Abhängigkeiten	[3.1] Domainanalyse		
Risiken/Probleme	-		
4.4	Paper Prototype	MK	3h
Beschreibung	Paperprototype erstellen		
Abhängigkeiten	[2.1] Anforderungen definieren		
Risiken/Probleme	-		
4.5	Developer Guide	PB	10h

Beschreibung	Developer Guide erstellen
Abhängigkeiten	[5.3] TaMaF
Risiken/Probleme	-

Tabelle 17: Arbeitspakete der Disziplin Design

5	Implementation	Verantwortlicher	161h
5.1	Persistence PC & Handy	MK	30h
Beschreibung	Speicherung der Daten		
Abhängigkeiten	-		
Risiken/Problemen	-		
5.2	Communication	PB	30h
Beschreibung	Verschicken der Daten		
Abhängigkeiten	-		
Risiken/Probleme	-		
5.3	TaMaF	PB	8h
Beschreibung	Task Objekte		
Abhängigkeiten	-		
Risiken/Probleme			
5.4	Android Syc Framework	PB	60h
Beschreibung	Alle Funktionen, Interfaces etc..		
Abhängigkeiten	-		
Risiken/Probleme	-		
5.5	Demo auf Smart-Phone	PB	25h
Beschreibung	Entwicklung einer Demo-Applikation auf dem Android Smart-Phone mit GUI.		
Abhängigkeiten	-		
Risiken/Probleme	-		
5.6	Demo auf Computer	MK	8h
Beschreibung	Entwicklung einer Demo-Applikation auf dem Computer.		
Abhängigkeiten	-		
Risiken/Probleme	-		

Tabelle 18: Arbeitspakete der Disziplin Implementation

6	Test	Verantwortlicher	56h
6.1	Unit Tests	MK	30h

Beschreibung	Erstellen der Unit Tests		
Abhängigkeiten	Hängt mit den jeweiligen Bereichen zusammen		
Risiken/Probleme			
6.2	Integration Tests	MK	2h
Beschreibung			
Abhängigkeiten			
Risiken/Probleme			
6.3	System Tests	MK	10h
Beschreibung	Erstellen der Systemtests		
Abhängigkeiten			
Risiken/Probleme			
6.4	Bugfixing	PB	14h
Beschreibung	Beheben von gefundenen Fehler bei Tests		
Abhängigkeiten	-		
Risiken/Probleme	Bei ungenauem Arbeiten werden mehr Fehler auftreten, wobei mehr Zeit in dieses Arbeitspaket investiert werden muss.		

Tabelle 19: Arbeitspakete der Disziplin Test

7	Deployment	Verantwortlicher	13h
7.1	Dokumente laut Anleitung SA	MK & PB	9h
Beschreibung	Die Dokumente Abstract, Management Summary, Poster, Technischer Bericht, Persönliche Berichte, Glossar, Stundenauswertung erstellen		
Abhängigkeiten	Projektabschluss		
Risiken/Probleme	-		
7.2	Enterprise Architects Synchron	PB	4h
Beschreibung	Beheben von gefundenen Fehler bei Tests		
Abhängigkeiten	-		
Risiken/Probleme	Bei ungenauem Arbeiten werden mehr Fehler auftreten, wobei mehr Zeit in dieses Arbeitspaket investiert werden muss.		

Tabelle 20: Arbeitspakete der Disziplin Deployment

8	Reserve	Verantwortlicher	33h
----------	----------------	------------------	-----

Tabelle 21: Risiko

9 Anhang

9.1 Tabellenverzeichnis

Tabelle 1: Referenzen.....	Fehler! Textmarke nicht definiert.
Tabelle 2: Vorgegebene Dokumente	7
Tabelle 3: Projektplan	9
Tabelle 4: Anforderungsspezifikation	9
Tabelle 5: Domain Analyse.....	9
Tabelle 6:: SAD.....	10
Tabelle 7: Design Dokumentation	10
Tabelle 8: Development Guide	10
Tabelle 9: Reference Java Doc.....	10
Tabelle 10: Glossar	10
Tabelle 11: Meilensteinübersicht	11
Tabelle 12: Übersicht der Iterationen.....	12
Tabelle 13: Risiko Management	13
Tabelle 14: Übersicht der genutzten Infrastruktur.....	14
Tabelle 15: Arbeitspakete der Disziplin Projekt Management	17
Tabelle 16: Arbeitspakete der Disziplin Requirements.....	17
Tabelle 17: Arbeitspakete der Disziplin Analyse	18
Tabelle 18: Arbeitspakete der Disziplin Design	18
Tabelle 19: Arbeitspakete der Disziplin Implementation	19
Tabelle 20: Arbeitspakete der Disziplin Test	20
Tabelle 21: Arbeitspakete der Disziplin Deployment.....	21
Tabelle 22: Risiko	21

Anforderungsspezifikation

Version 1.1

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	18.09.2009	Erstellen der Anforderungsspezifikation	PB
1.0rc02	Review	22.09.2009	Verschiedene Korrekturen	MK
1.0rc03	In Bearbeitung	29.09.2009	Nicht funktionale Anforderungen hinzugefügt	PB
1.0rc04	In Bearbeitung	30.09.2009	Use Cases und Zustandsdiagramm hinzugefügt.	PB
1.0rc05	In Bearbeitung	02.10.2009	Änderungen laut Sitzung vorgenommen.	PB
1.0rc06	Review	03.10.2009	Einige Korrekturen vorgenommen	MK
1.0rc07	In Bearbeitung	06.10.2009	Hinzufügen der Informationen über das Interface zu TaMaF	PB
1.0rc08	In Bearbeitung	06.10.2009	Anpassungen am Interface	PB
1.0rc09	Review	07.10.2009	Review OK	MK
1.0rc10	In Bearbeitung	09.10.2009	Anpassungen gemäss Besprechung	PB
1.0	Release	10.10.2009	Stabile Version 1.0	TEAM
1.1rc01	In Bearbeitung	19.10.2009	Korrekturen	PB
1.1rc02	In Bearbeitung	03.11.2009	Interface XferAgent um getHostID erweitert	PB
1.1rc03	In Bearbeitung	04.11.2009	GUI Prototype für Smart-Phone hinzugefügt	PB
1.1rc04	In Bearbeitung	06.11.2009	Anpassungen gemäss Besprechung	PB
1.1	Release	06.11.2009	Stabile Version 1.1	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Übersicht.....	4
2	Allgemeine Beschreibung	4
3	Synchronisations-Framework	5
3.1	Funktionale Anforderungen	5
3.1.1	Aktoren	5
3.1.2	Use Case-Diagramm	5
3.1.3	Use Cases brief	6
3.1.4	Zustandsdiagramm.....	7
3.1.5	Interfaces	8
3.2	Nichtfunktionale Anforderungen	12
3.2.1	Funktionalität	12
3.2.2	Erlernbarkeit	12
3.2.3	Zuverlässigkeit	12
3.2.4	Effizienz	12
3.2.5	Erweiterbarkeit.....	12
4	Task-Management Applikation	13

1 Einführung

1.1 Zweck

Der Inhalt dieses Dokumentes ist die Aufnahme der Anforderung das Produkt Task-Management-Framework on Smart-Phone.

1.2 Gültigkeitsbereich

Dieses Dokument ist für die gesamte Projektdauer.

1.3 Übersicht

Enthält die generellen Anforderungen an das Projekt sowie Anforderungen an das Framework sowie für die Applikation (GUI).

2 Allgemeine Beschreibung

Für eine Übersicht über das Projekt bitte im Projektplan nachsehen.

3 Synchronisations-Framework

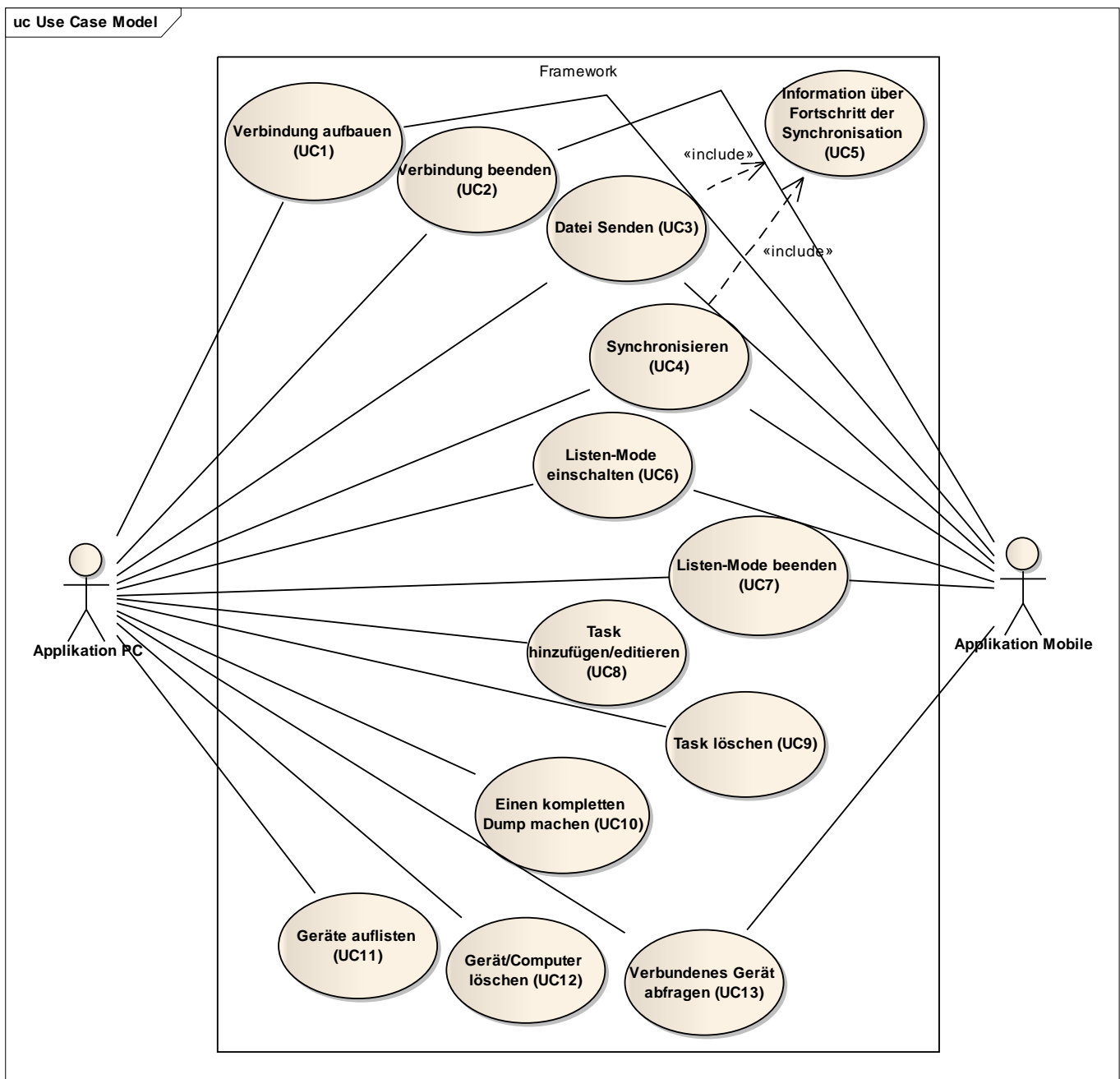
3.1 Funktionale Anforderungen

3.1.1 Akteure

Der Akteur ist die auf dem Framework aufbauende Applikation.

3.1.2 Use Case-Diagramm

Hier wird die Kommunikation von links nach rechts angesehen. Obwohl der Akteur Applikation Mobile ebenfalls alle Use Cases nutzen kann, wird hier gezeigt, welche Use Cases zur Applikation Mobile weiter gehen.



3.1.3 Use Cases brief

3.1.3.1 Verbindung aufbauen (UC1)

Es wird eine Verbindung zum Gerät/Computer aufgebaut werden.

3.1.3.2 Verbindung beenden (UC2)

Das Beenden der Verbindung wird auf Knopfdruck oder durch Abhängen der Verbindung (Kabel, Netzwerk) getätigt.

3.1.3.3 Datei Senden (UC3)

Unter Angabe von Quelle und Ziel wird eine Datei versendet.

3.1.3.4 Synchronisieren (UC4)

Die geänderten/hinzugefügten/gelöschten Tasks werden ausgetauscht.

3.1.3.5 Information über Fortschritt der Synchronisation (UC5)

Das Framework informiert darüber, wie weit die Synchronisation abgeschlossen ist und wie lange diese noch in etwa andauern wird.

3.1.3.6 Listen-Mode einschalten (UC6)

Der Listen-Mode soll eingeschalten werden.

3.1.3.7 Listen-Mode beenden (UC7)

Der Listen-Mode soll beendet werden. (Listen-Mode Beschreibung im Glossar)

3.1.3.8 Task hinzufügen/editieren (UC8)

Es wird ein Task hinzugefügt oder geändert.

3.1.3.9 Task löschen (UC9)

Ein Task wird gelöscht.

3.1.3.10 Einen kompletten Dump machen (UC10)

Es sollen alle Daten (nicht nur geänderte) von der anderen Seite abgefragt werden (Dump). Dies sollte in beide Richtungen möglich sein. Von Handy zu Computer sowie von Computer zu Handy.

3.1.3.11 Geräte auflisten (UC11)

Alle Geräte, die dem Framework bekannt sind wollen abgefragt werden.

3.1.3.12 Gerät/Computer löschen (UC12)

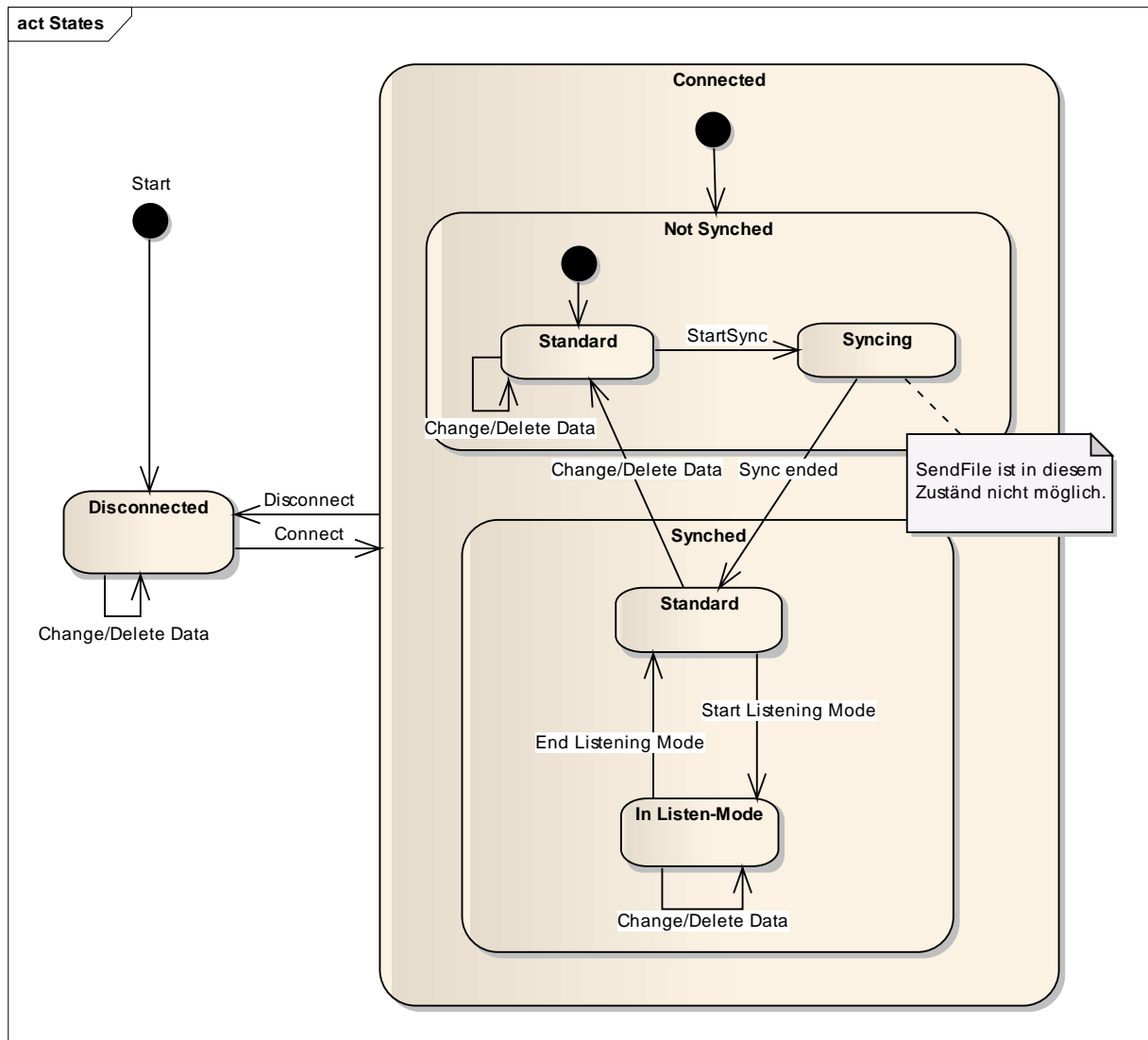
Ein Computer/Gerät wird entfernt und nicht mehr gemerkt.

3.1.3.13 Verbundenes Gerät abfragen (UC13)

Abfragen, mit welchem Gerät momentan eine Verbindung besteht.

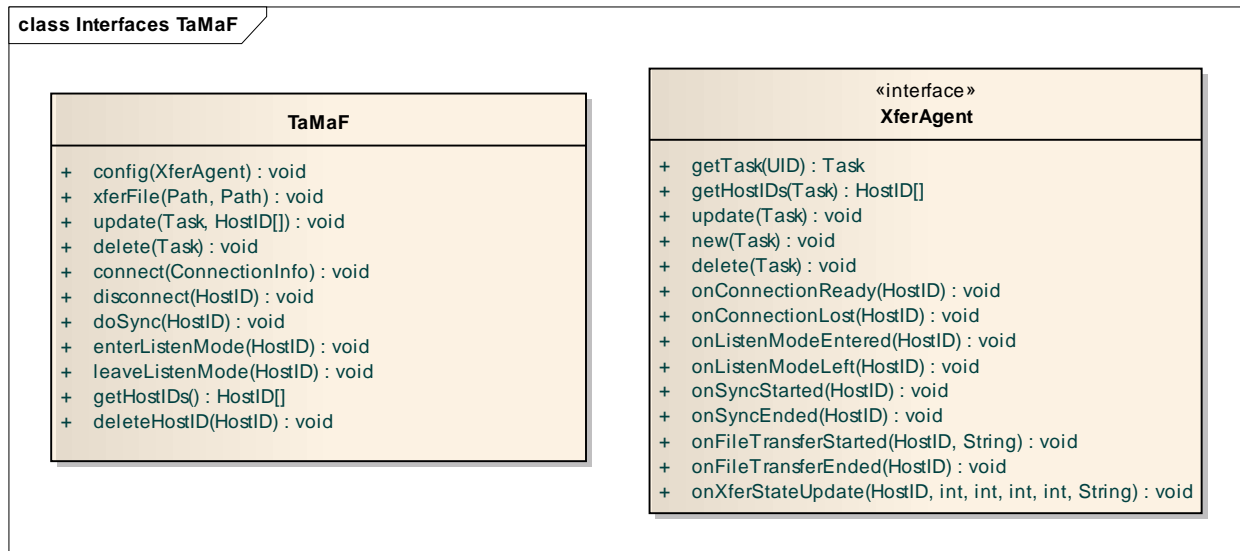
3.1.4 Zustandsdiagramm

Dieses Zustandsdiagramm dient dazu um zu zeigen, wann in den Listen-Mode gegangen werden kann und wann Dateien gesendet werden können. Es wird absichtlich nicht alles aufgeführt um auf das wichtige zu fokussieren.



3.1.5 Interfaces

Folgendes Diagramm zeigt die Interfaces, über welche die Applikation auf TaMaF zugreift und über welche es angibt, wie es bestimmte Informationen von untern erhalten möchte. Hier wird kurz beschrieben, wofür die Funktionen in den Interfaces stehen, und wie diese zu den Use Cases stehen.



3.1.5.1 TaMaF

3.1.5.1.1 config(XferAgent)

Dies wird einmal am Anfang beim Verbinden mit TaMaF von der Applikation aufgerufen und die notwendigen Informationen mitgegeben. Diese Informationen dienen TaMaF um zu wissen, wie es Informationen von der Applikation nachfragen kann und Informationen an diese weiterleiten kann.

Die durch config() angegebenen Informationen sind nötig für mehrere Use Cases.

3.1.5.1.2 xferFile(Path, Path)

Dies stellt die Funktion des UC3 dar.

3.1.5.1.3 update(Task, HostID[])

Dies stellt die Funktion des UC8 dar.

Mit HostID[] wird angegeben, mit welchen Geräten dieser Task Synchronisiert werden soll. Hiermit wird auch bezweckt, dass die Applikation sogenannte Gruppen machen kann (wie z.B. Privat und Geschäft). Wenn ein Task eine Gruppe ändert, wird der Task nochmals per update() mitgeteilt und in HostID[] dementsprechend die HostIDs gesetzt. TaMaF merkt dann selbst, bei mit welchem Gerät nicht mehr Synchronisiert werden soll (und sendet somit ein Delete dieses Tasks an das Gerät).

3.1.5.1.4 delete(Task)

Dies stellt die Funktion des UC9 dar.

3.1.5.1.5 connect(ConnectionInfo)

Dies stellt die Funktion des UC1 dar. Hierbei werden noch Informationen zur Verbindung angegeben.

3.1.5.1.6 disconnect(HostID)

Dies stellt die Funktion des UC2 dar.

Die HostID wird hier mitgegeben, damit TaMaF weiss, mit wem die Verbindung beendet werden soll. Mehrere gleichzeitige Verbindungen werden in diesem Projektumfang noch nicht möglich sein. Das Interface stellt dies jedoch bereits zur Verfügung, da es sehr gut möglich ist, dass man dies später noch implementieren wird. [Dieser Absatz wird in 3.1.5.1.7 - 3.1.5.1.9 und 3.1.5.2.* referenziert]

3.1.5.1.7 doSync(HostID)

Dies stellt die Funktion des UC4 dar. Diese Funktion ist blockierend und beendet sobald alle Tasks synchronisiert wurden.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.1.8 enterListenMode(HostID)

Dies stellt die Funktion des UC6 dar.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.1.9 leaveListenMode(HostID)

Dies stellt die Funktion des UC7 dar.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.1.10 getHostIDs()

Dies stellt die Funktion des UC11 dar.

Hierdurch kann die Applikation alle im TaMaF registrierten HostIDs (Geräte) erhalten. Dadurch kann die Applikation zwischen verschiedenen Geräten unterscheiden. Wird benötigt, damit update() richtig benutzt werden kann, da diese HostIDs da benötigt werden.

3.1.5.1.11 deleteHostID(HostID)

Dies stellt die Funktion des UC12 dar.

Eine gelöschte HostID wird nicht mehr gemerkt.

3.1.5.2 XferAgent

Wird von der Applikation implementiert und durch config() an das TaMaF gegeben. Es enthält Informationen, die TaMaF der Applikation Informationen weiterleiten kann.

3.1.5.2.1 getTask(UID)

Wird für UC4 und UC6 benötigt.

Über diese Funktion kann TaMaF die Daten eines Tasks erhalten. Dies wird benötigt, dass er dann diese über das Netzwerk senden kann.

3.1.5.2.2 *getHostIDs(Task)*

Über diese Funktion kann TaMaF herausfinden, mit welchem direkt verbundenen Gerät der Task synchronisiert werden soll. Dies wird in dem Fall benötigt, wenn ein Task während einer Synchronisation neu erhalten wird. In dem Fall muss noch nachgefragt werden, mit wem dieser Task synchronisiert werden soll.

3.1.5.2.3 *update(Task)*

Wird für UC4 und UC6 benötigt.

Wenn von einem anderen Gerät ein veränderter Task erhalten wird, wird TaMaF über diese Funktion den veränderten Task an die Applikation reichen.

3.1.5.2.4 *new(Task)*

Wird für UC4 und UC6 benötigt.

Wenn von einem anderen Gerät ein neuer Task erhalten wird, wird TaMaF über diese Funktion den neuen Task an die Applikation reichen.

3.1.5.2.5 *delete(Task)*

Wird für UC4 und UC6 benötigt.

Wenn von einem anderen Gerät die Information kommt, dass ein Task gelöscht wurde, wird TaMaF über diese Funktion den gelöschten Task an die Applikation mitteilen.

3.1.5.2.6 *onConnectionReady(HostID)*

Dies stellt die Funktion des UC13 dar und gibt der Applikation die Information darüber, dass eine Verbindung hergestellt wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.7 *onConnectionLost(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass die Verbindung beendet wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.8 *onListenModeEntered(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass in den Listen-Mode gewechselt wurde. Dies kann durch die eigene Applikation oder von der anderen Seite ausgelöst worden sein.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.9 *onListenModeLeft(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass der Listen-Mode beendet wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.10 *onSyncStarted(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass die Synchronisation mit einem Gerät gestartet wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.11 *onSyncEnded(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass die Synchronisation mit einem Gerät beendet wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.12 *onFileTransferStarted(HostID, String)*

Hierdurch wird der Applikation mitgeteilt, dass die Übertragung einer Datei begonnen hat. String gibt dafür zusätzliche Informationen, diese Information ist der Zielpfad der Datei.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.13 *onFileTransferEnded(HostID)*

Hierdurch wird der Applikation mitgeteilt, dass die Übertragung einer Datei beendet wurde.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.1.5.2.14 *onXferStateUpdate(HostID, int, int, int, int, String)*

Während dem Senden einer Datei oder dem Synchronisieren, wird hierdurch über den Fortschritt informiert.

Die Parameter stehen für: Host mit dem der Austausch stattfindet, vergangene Zeit, totale Zeit, gesendete Datengröße, totale Datengröße, zusätzliche Nachricht.

Grund für HostID wird in 3.1.5.1.6 beschrieben.

3.2 Nichtfunktionale Anforderungen

3.2.1 Funktionalität

Das Framework soll als Grundlage für ein Programm dienen, das Daten mit einem anderen Programm, welches ebenfalls dieses Framework benutzt, synchronisieren kann.

Mehrere PCs und mehrere Handys

Die Synchronisation muss mit mehreren PCs möglich sein. (z.B. ein Geschäfts-Computer und ein Privat-Computer) Ebenfalls sollen mehrere Handys mit einem Computer synchronisiert werden. Dazu soll das Framework Auskunft darüber geben, welches Handy angeschlossen ist.

Parallelität

Priorität 1: Gleichzeitig kann nur eine Verbindung zu einem anderen Gerät bestehen.

Priorität 2: Es können gleichzeitig mehrere andere Geräte verbunden und synchronisiert werden.

3.2.2 Erlernbarkeit

Mit Hilfe des Development Guides soll ein Java-Programmierer innerhalb von wenigen Stunden verstehen können, wie er eine Applikation erstellen kann, welche das Framework nutzt.

3.2.3 Zuverlässigkeit

Falls die Synchronisation unterbrochen wird (z.B. durch Abschliessen des USB Kabels) sollen die Daten beim nächsten Mal ohne Probleme synchronisiert werden. Es soll durch diesen Unterbruch nicht in einen inkonsistenten Zustand gebracht werden.

3.2.4 Effizienz

Der Synchronisationsprozess soll mit einer Laufzeit von $O(n_{\text{changed}})$ laufen.

3.2.5 Erweiterbarkeit

Kommunikation über Socket

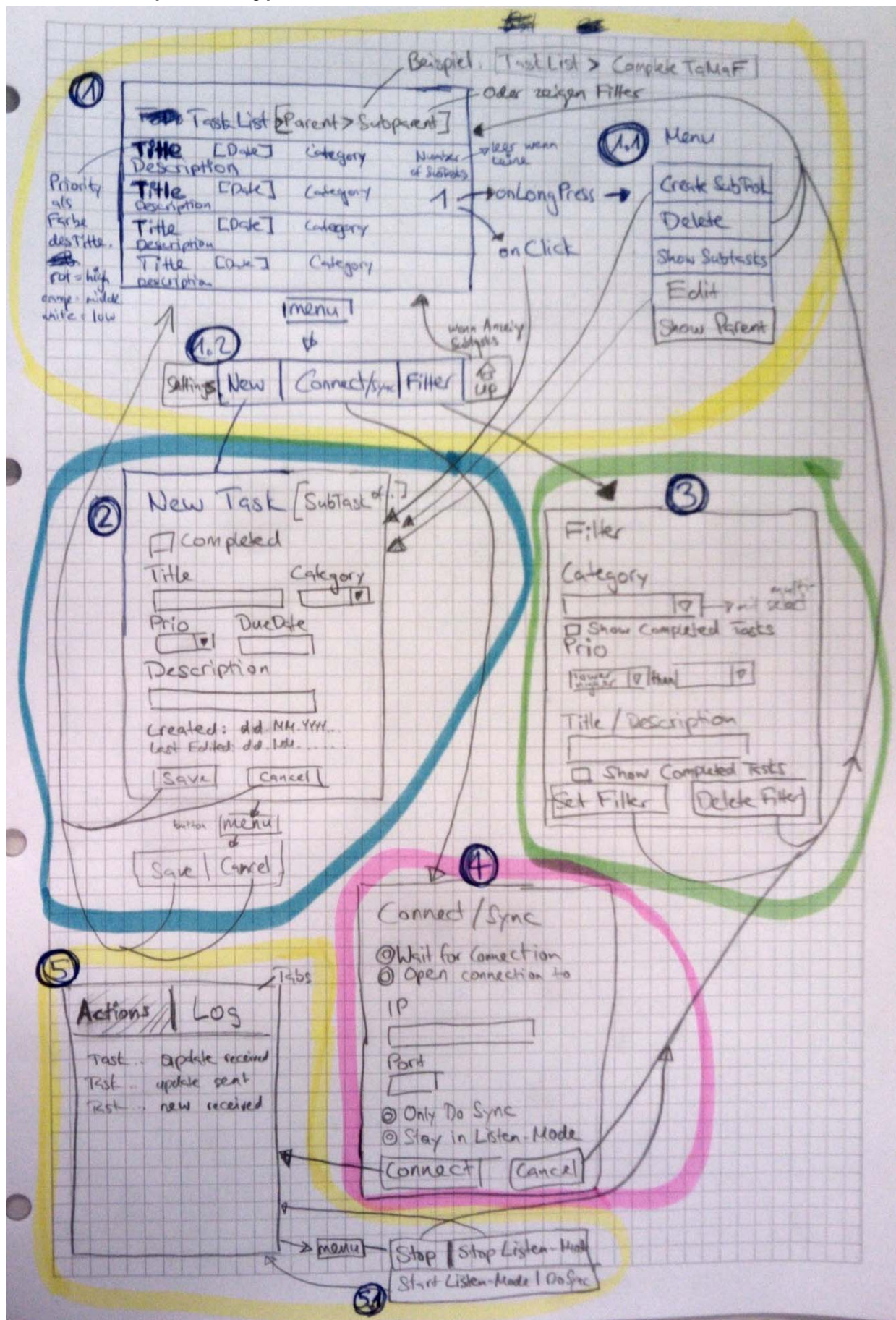
Dies damit man später auch in einer anderen Sprache dieses Socket nutzen kann. Dafür ist eine Dokumentation der Kommunikation notwendig. Diese Dokumentation wird Teil des Development Guides.

Änderung der Schnittstelle

Es soll möglich sein ohne zu grosse Änderungen die Synchronisation über andere Schnittstellen durchzuführen. So z.B. über das Netzwerk/Internet, USB oder Bluetooth.

4 Task-Management Applikation

4.1 GUI Paper Prototype



4.1.1 Main Window (1)

Diese Ansicht zeigt alle noch nicht „completed“ Top Level Tasks in einer Liste. Wenn ein Filter gesetzt wurde, werden diese angezeigt, welche den Filter erfüllen. In einem Filter werden auch SubTasks angezeigt welche farblich hinterlegt sind. Oberhalb wird gegebenenfalls angezeigt, ob es die SubTasks eines Tasks sind. Die Priority wird über die Farbe angezeigt. Rot = sehr wichtig, orange = wichtig, weiss=normal, hellblau = nicht so wichtig, blau = unwichtig.

Oberhalb sieht man die Navigation. Wenn man sich in einem SubTask befindet wird dies angezeigt. Ebenfalls wird angezeigt, falls ein Filter aktiv ist.

Die Sortierung ist nach DueDate und Priorität.

Rechts wird beim jeweiligen Task angezeigt ob und wie viele SubTasks vorhanden sind.

Wenn ein Task angeklickt wird erscheint die Ansicht (2) in welcher man den Task editieren kann.

Wenn man einen Task länger anklickt (gedrückt hält) erscheint das Task-Menu (1.1).

Wenn man auf dem Handy auf die Taste Menu klickt erscheint (1.2).

4.1.1.1 Task-Menu (1.1)

Menupunkt	Aktion
Create Subtask	Leitet das erstellen eines SubTasks ein. (2)
Delete	Löscht den Task und man kehrt zur vorherigen Ansicht zurück.
Show SubTasks	Zeigt eine Liste der SubTasks an (1)
Edit	Leitet zur Ansicht weiter um den Task zu editieren. (2)
Show Parent	Sichtbar, wenn es sich um einen SubTask handelt. Damit wird in der Hierarchie zur Ansicht des Parent (und dessen Nachbarn) gewechselt. (1)

4.1.1.2 Ansicht-Menu (1.2)

Menupunkt	Aktion
Settings	Zeigt eine Ansicht in der Einstellungen vorgenommen werden können. Z.B. welche Kategorien mit welchem Device synchronisiert werden.
New	Erstellen eines neuen Tasks (2)
Connect/Sync	Zeigt die Ansicht um eine Verbindung herzustellen (4)
Filter	Zeigt die Ansicht um einen Filter zu setzen (3)
UP	Wird nur angezeigt, wenn man sich in der Ansicht von SubTasks befindet. Dadurch kann wieder zum Parent navigiert werden.

4.1.2 New Task / Edit Task (2)

In dieser Ansicht wird ein neuer Task/SubTask erstellt oder ein bestehender Task editiert. Je nach dem was gemacht wird sieht die Ansicht etwas anders aus.

Wird ein SubTask erstellt/editiert wird angezeigt, von welchem Task dies der SubTask ist.

Über die Knöpfe am Ende oder über das Menu wird der Task gespeichert/erstellt oder es wird verworfen.

„Last Modified“ bedeutet wann der Benutzer bei diesem Task das letzte Mal auf „Save“ geklickt hat. Also ebenfalls wenn sich eigentlich nichts geändert hat. Da aber bewusst auf save gedrückt wurde wird diese Zeit geändert.

4.1.3 Filter (3)

Diese Ansicht lässt einen Filter erstellen, welcher die Liste der Tasks (1) nach bestimmten Kriterien filtert.

Dieser Filter kann erstellt werden und dann wieder gelöscht werden um die normale Ansicht zu haben.

Die Filter werden alle per UND-Verknüpfung zusammengeknüpft.

4.1.4 Connect/Sync (4)

Über diese Ansicht wird eine Verbindung mit dem Computer (oder einem anderen zu synchronisierendem Gerät) hergestellt.

Wenn man über das Netzwerk verbinden will mit einem Rechner, der auf eine Verbindung wartet, wählt man „Open Connection to“ und gibt IP und Port an.

Wenn man „Wait for Connection“ macht, wird nur der Port benötigt und dann gewartet bis ein anderes Gerät eine Verbindung zum Handy herstellt. Diese Option wird z.B. benutzt wenn über USB vom Computer her eine Verbindung mit dem Handy erstellt wird.

Die unteren Optionen („Only Do Sync“ und „Stay in Listen-Mode“) sagen aus, ob nach dem Verbinden nur ein Sync gemacht werden soll, oder ob gleich in den Listen-Mode gegangen werden soll.

Beim Klicken auf „Connect“ wird ein „Connecting...“-Dialog angezeigt, welcher blockiert, bis eine Verbindung hergestellt wurde. Wenn eine Verbindung hergestellt wurde wird die Ansicht (5) angezeigt.

4.1.5 Verbindungs-Ansicht (5)

Diese Ansicht zeigt den Stand der Verbindung an. Dabei gibt es Zwei Tabs, die verschiedene Ansichtsmöglichkeiten bieten.

Unter Actions wird nur generell angezeigt, was gemacht wird. Z.B. Task erhalten, Task gesendet, Verbindung erstellt, Verbindung beendet, Listen Mode gestartet, Synchronisation beendet, usw.

Unter Log wird eine Ausgabe des Log-Files gemacht. Ab welchem Level geloggt wird, kann per Settings definiert werden.

Über den Menu-Button des Handys erreicht man das Menu.

Bei Kommunikation mit dem verbundenen Gerät wird über ein aufflackern der LED informiert.

4.1.5.1 Menu (5.1)

Menupunkt	Aktion
Stop	Bricht die ganze Verbindung ab und kehrt zur Hauptansicht zurück (1)
Start/Stop Listen-Mode	Startet oder beendet den Listen-Mode.
Do Sync	Synchronisiert das Handy mit dem Verbundenen Gerät.

4.1.6 Settings

Einstellungen:

- Log Level
- Kategorien (optional)

Domainanalyse

Version 1.1

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	03.10.2009	Erstellen des Domainanalyse Dokuments	MK
1.0rc02	In Bearbeitung	04.10.2009	Erstellen des Domain Models	MK
1.0rc03	In Bearbeitung	06.10.2009	Überarbeitung des Domain Models	TEAM
1.0rc04	In Bearbeitung	06.10.2009	Ein Teil der SSD erstellen	PB
1.0rc05	In Bearbeitung	06.10.2009	Zweiter Teil der SSD erstellt & eingefügt	MK
1.0rc06	In Bearbeitung	07.10.2009	Teil von Konzeptbeschreibung	MK
1.0rc07	Review	07.10.2009	Verschiedene Korrekturen vorgenommen	PB
1.0rc08	In Bearbeitung	13.10.2009	Korrekturen gemäss Meeting vorgenommen	MK
1.0rc09	In Bearbeitung	14.10.2009	Systemoperationen	MK
1.0rc10	In Bearbeitung	20.10.2009	Korrekturen vorgenommen gemäss Meeting	MK
1.0rc11	In Bearbeitung	28.10.2009	Korrekturen vorgenommen gemäss Meeting	MK
1.0	Release	31.10.2009	Stabile Version 1.0	TEAM
1.0.1	Release	03.11.2009	Stabile Version 1.0.1	TEAM
1.1rc01	In Bearbeitung	30.11.2009	Domain Modell für TaMaF Demo Android	PB
1.1rc02	In Bearbeitung	07.12.2009	Korrekturen vorgenommen gemäss Meeting	PB
1.1	Release	07.12.2009	Stabile Version 1.1	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	TaMaF	5
2.1	Domain Modell.....	5
2.1.1	Strukturdiagramm	5
2.1.2	Konzeptbeschreibung.....	6
2.2	System Sequenzdiagramme	8
2.2.1	UC1 Verbindung aufbauen	8
2.2.2	UC2 Verbindung beenden.....	8
2.2.3	UC3 Datei senden.....	9
2.2.4	UC4 Synchronisieren.....	10
2.2.5	UC5 Informationen über den Fortschritt.....	11
2.2.6	UC6 Listen Mode einschalten	12
2.2.7	UC7 Listen Mode beenden	12
2.2.8	UC8 Task hinzufügen/editieren	13
2.2.9	UC9 Task löschen	13
2.2.10	UC10 Einen kompletten Dump machen.....	13
2.2.11	UC11 Geräte auflisten.....	13
2.2.12	UC12 Gerät/Computer löschen.....	14
2.2.13	UC13 Verbundenes Gerät abfragen.....	14
2.3	Systemoperationen	15
3	TaMaF Demo auf Android Phone	17
3.1	Domain Modell.....	17
3.1.1	Konzeptbeschreibung.....	17
3.2	System Sequenzdiagramme	19

1 Einführung

1.1 Zweck

In diesem Dokument wird die Problem Domain der Studienarbeit analysiert.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

Siehe Dokument Glossar.

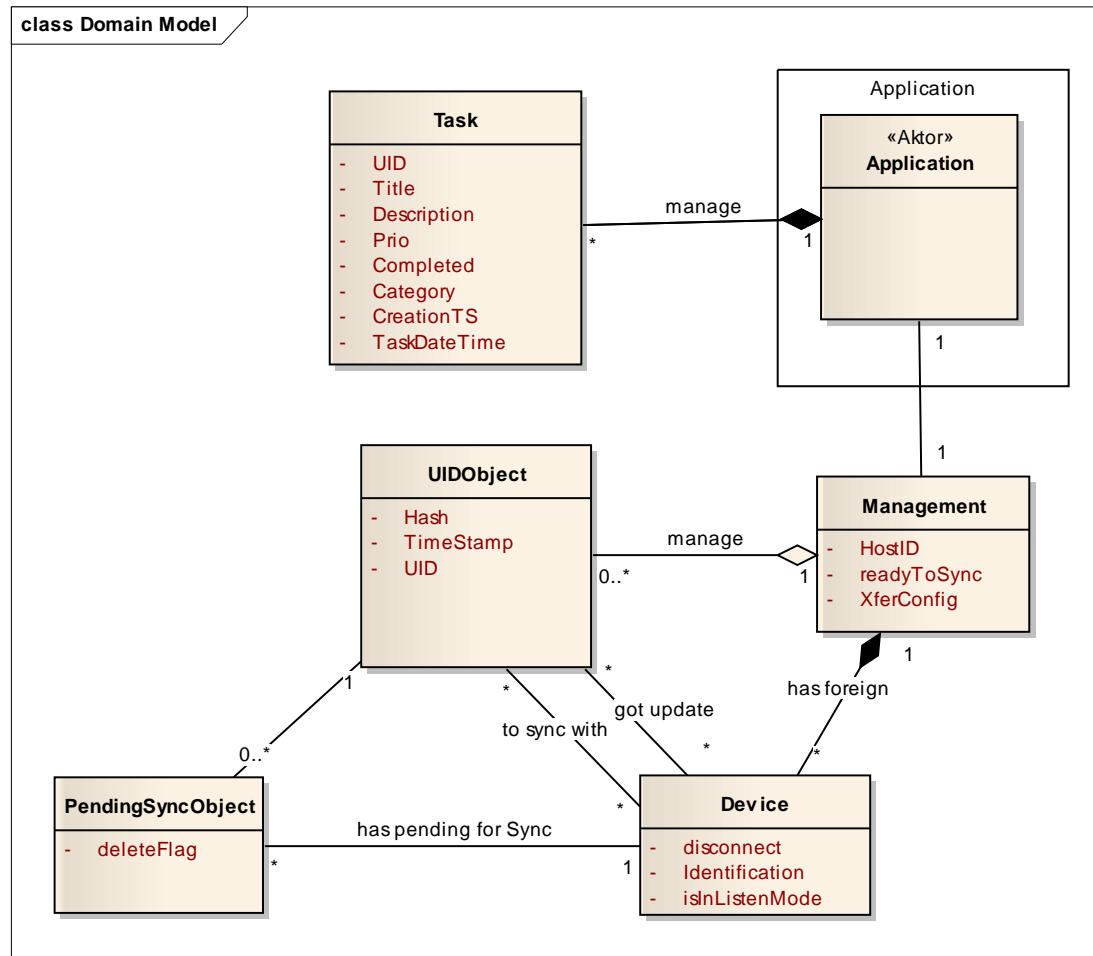
1.4 Referenzen

Referenz	Quelle

2 TaMaF

2.1 Domain Modell

2.1.1 Strukturdiagramm



2.1.2 Konzeptbeschreibung

Zu beachten ist, dass selbsterklärende Beziehungen nicht beschrieben werden.

1	Task	
Beschreibung	Ein Task steht jeweils für ein Objekt das Synchronisiert wird.	
Attribute	UID	Unique ID, welche vom Management zugewiesen wird.
	Title	Titel des Tasks
	Description	Beschreibung des Tasks
	Prio	Priorität des Tasks
	Completed	Gibt an, ob der Tasks abgeschlossen wurde
	Category	Ist ein String, welcher die Kategorie beschreibt.
	TaskDateTime	Wann der Task statt findet.
	CreationTS	Zeit an dem der Task erstellt wurde.
	LastEditedTS	Wann der Task zuletzt editiert wurde.
Beziehungen	Ein Task ist einer Applikation zugewiesen.	

2	Application	
Beschreibung	Das <i>Application Object</i> enthält alle gespeicherten Tasks.	
Attribute	-	-
Beziehungen	Die Application enthält mehrere Tasks & verwaltet diese. Die Application ist genau einem Management zugewiesen.	

3	Management	
Beschreibung	Übernimmt die Verwaltung der Objekte die Synchronisiert werden.	
Attribute	HostID	Die HostID des Gerätes
	readyToSync	Speicher ob eine Verbindung bereits besteht.
	XferConfig	Management hat eine Referenz auf das XFerConfig
Beziehungen	Das Management hat alle UIDObject referenziert. Zusätzlich hat das Management eins bis mehrere fremde Geräte, zu welchen die Referenzen auf die UIDObject gespeichert werden, welche noch nicht synchronisiert sind. Die Daten welche synchronisiert werden, werden an den Communication Agent gesendet.	

4	UIDObject	
---	-----------	--

Beschreibung	Ein UIDObject kann man vergleichen mit einem Task, nur dass hier nicht alle Informationen gespeichert werden, nur die Infos welche zum Synchronisieren benötigt werden.	
Attribute	Hash	Speichert den Hash über den Task
	TimeStamp	Speichert wann das letzte Mal eine Änderung vorgenommen wurde.
	UID	Ist dieselbe UID, welche einem Task zugeordnet wurde.
	WhoGotThisUpdate	Speicher welche Geräte das Update bereits haben.
Beziehungen	<p>Management verwaltet alle UIDObjecte. Zusätzlich hat ein Device Referenzen auf die UIDObjects, welche für die jeweiligen Geräte synchronisiert werden muss („to sync with“ Beziehung).</p> <p>Zusätzlich speichert das UIDObject, welche Geräte bereits das Update besitzen. („got update“ Beziehung)</p>	

5 Device

Beschreibung	Hierbei handelt es sich um ein Android Handy oder einem PC welches direkt verbunden wird (Nachbar-Gerät).	
Attribute	disconnect	Disconnect mit einem spezifischen Gerät.
	Identification	Identifiziert ein Gerät
	isInListenMode	Beschreibt ob das Gerät im Listen-Mode ist.
Beziehungen	Liste für Syncobjekte Referenziert, welche Synchronisiert werden müssen.	

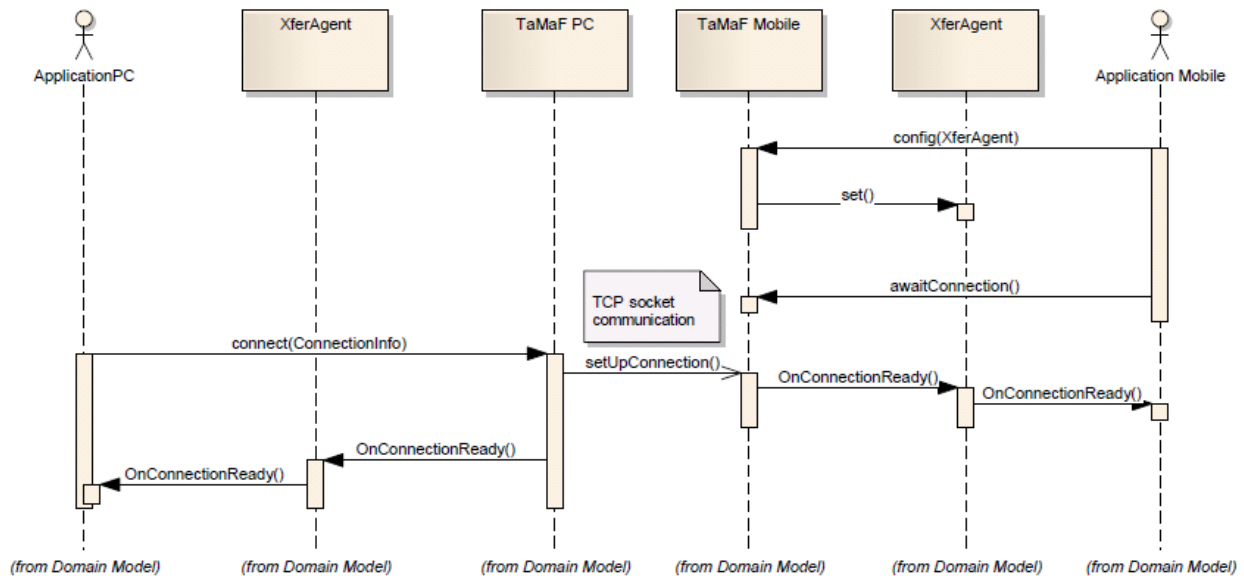
6 PendingSyncObject

Beschreibung	Falls ein Objekt gelöscht wird, wird dies im PendingSyncObject definiert.	
Attribute	deleteFlag	Es handelt sich hier um ein Boolean, ob die Referenz gelöscht werden kann.
Beziehungen	PendingSyncObject gehört genau zu einem Device und einem UIDObject.	

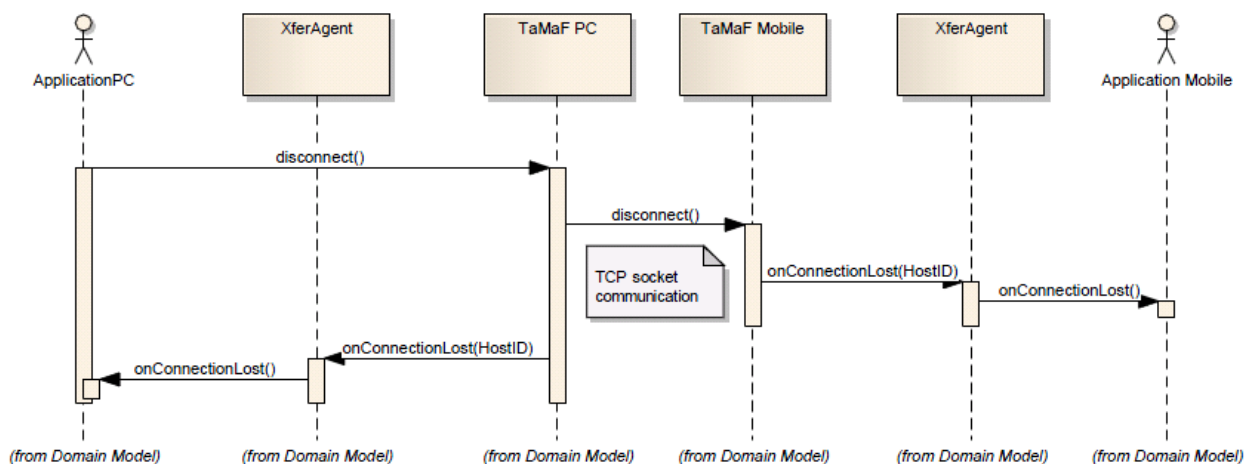
2.2 System Sequenzdiagramme

In den folgenden System Sequenzdiagramme wurden, die internen Abläufe zum besseren Verständnis teilweise dargestellt.

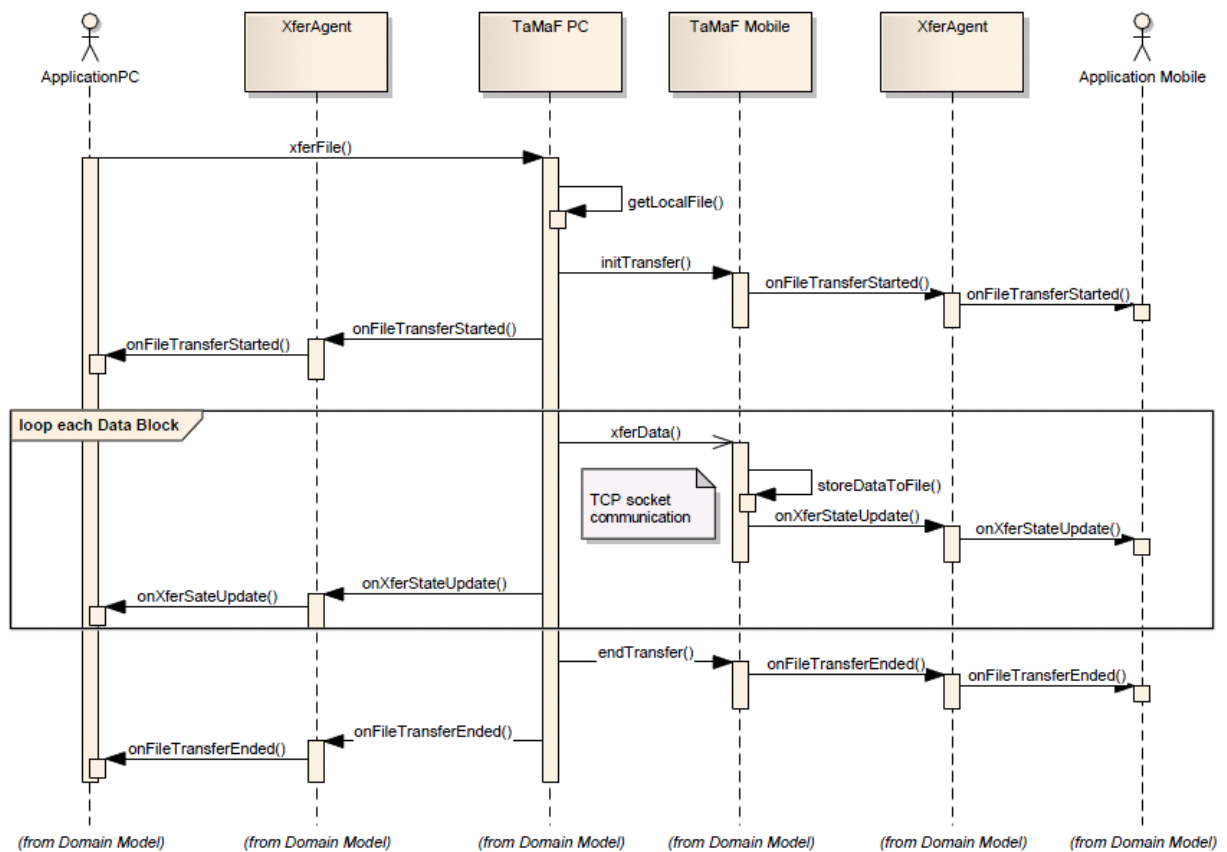
2.2.1 UC1 Verbindung aufbauen



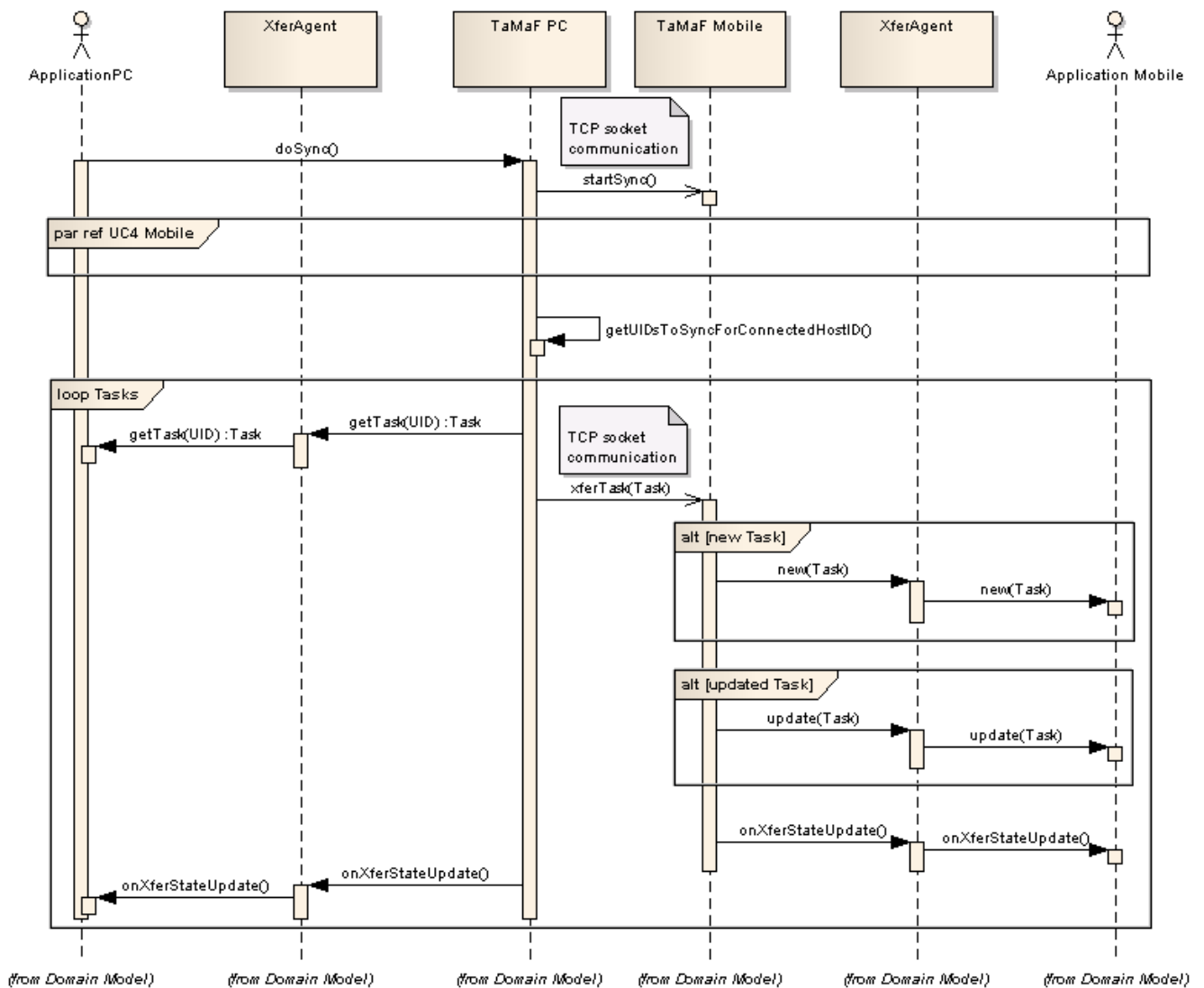
2.2.2 UC2 Verbindung beenden



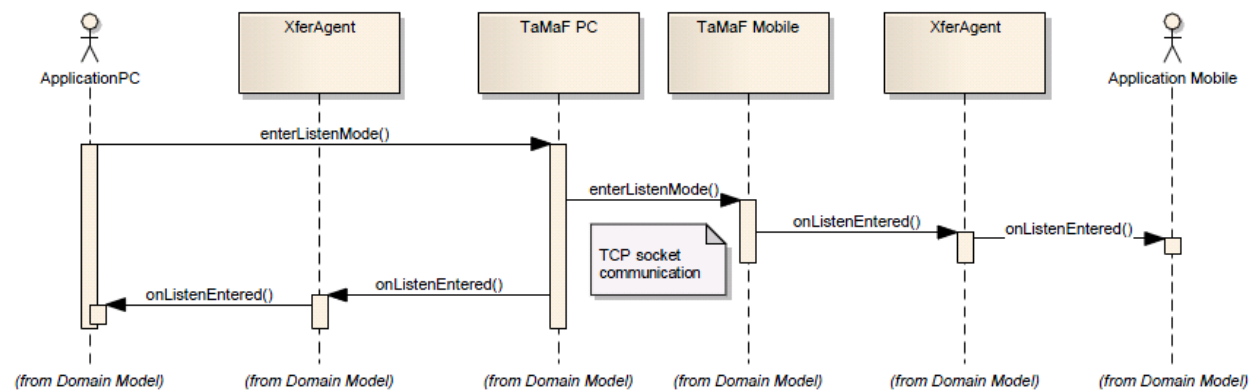
2.2.3 UC3 Datei senden



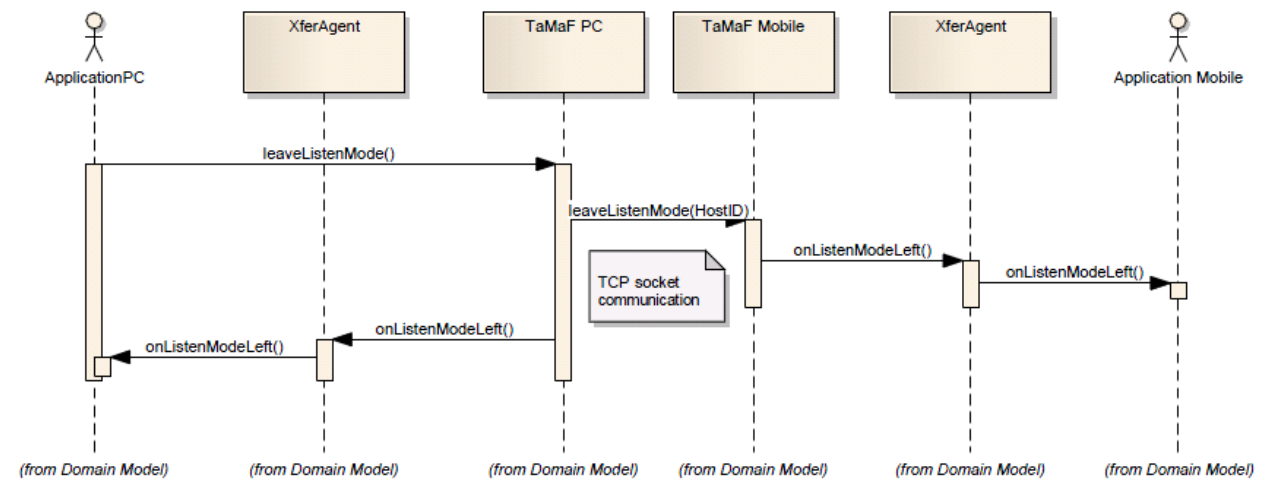
2.2.4 UC4 Synchronisieren



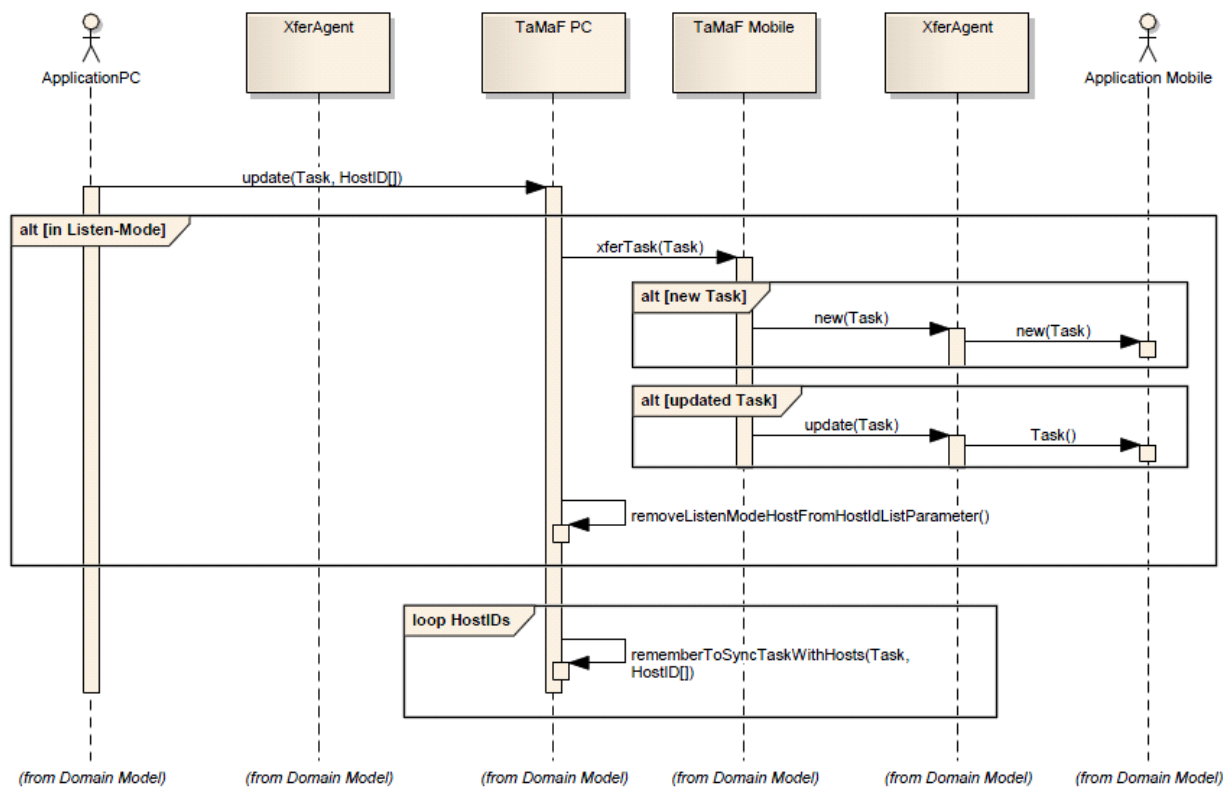
2.2.6 UC6 Listen Mode einschalten



2.2.7 UC7 Listen Mode beenden



2.2.8 UC8 Task hinzufügen/editieren



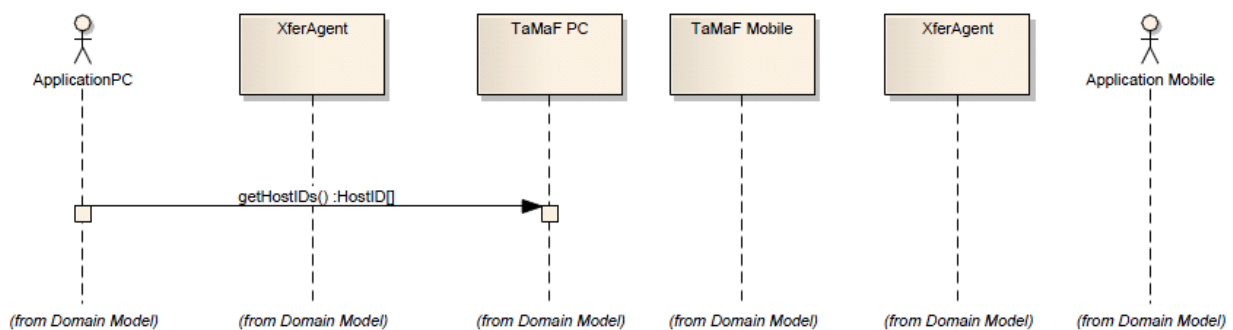
2.2.9 UC9 Task löschen

Im Prinzip gleich wie UC8 Task hinzufügen/editieren.

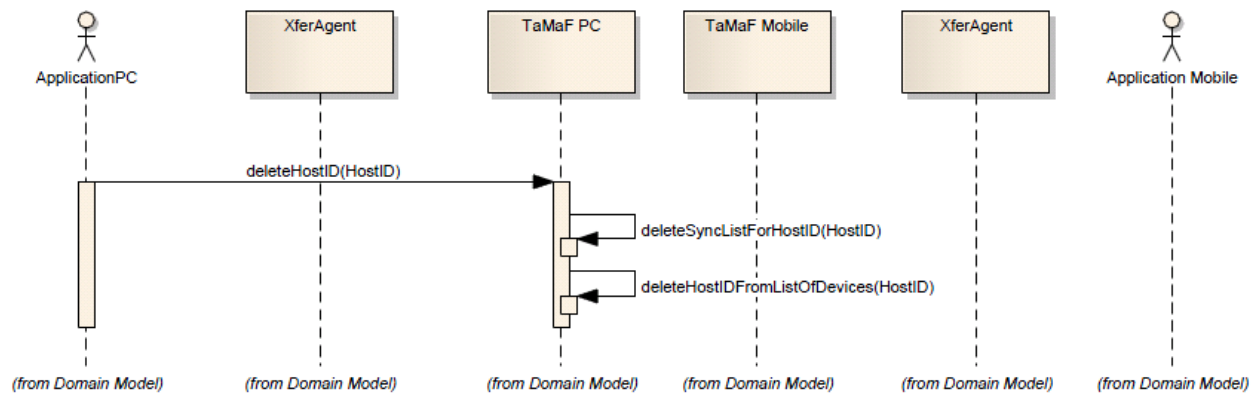
2.2.10 UC10 Einen kompletten Dump machen

Offen, wird eventuell nicht implementiert, da es der oberen Applikation überlassen wird.

2.2.11 UC11 Geräte auflisten



2.2.12 UC12 Gerät/Computer löschen



2.2.13 UC13 Verbundenes Gerät abfragen

Siehe UC1

Die Applikation wird informiert, sobald eine Connection besteht. Das Sync-Framework speichert die Informationen, ob ein Gerät bereit ist.

2.3 Systemoperationen

Folgend werden die wichtigsten Systemoperationen erklärt.

C01: config	
Operation	config(XferAgent)
Cross Reference	UC01: Verbindung aufbauen
Preconditions	<ul style="list-style-type: none"> Objekt XferAgent & Management existiert.
Postconditions	<ul style="list-style-type: none"> Referenz auf XferAgent ist gespeichert.

C02: awaitConnection	
Operation	awaitConnection()
Cross Reference	UC01: Verbindung aufbauen
Preconditions	<ul style="list-style-type: none"> Es wurde konfiguriert, über welche Schnittstelle synchronisiert werden soll.
Postconditions	-

C03: connect	
Operation	connect(ConnectionInfo)
Cross Reference	UC01: Verbindung aufbauen
Preconditions	<ul style="list-style-type: none"> Konfiguration wurde getätigt
Postconditions	<ul style="list-style-type: none"> Verbindung besteht

C04: onConnectionReady	
Operation	onConnectionReady()
Cross Reference	UC01: Verbindung aufbauen
Preconditions	<ul style="list-style-type: none"> Verbindung besteht Synchronisation kann gestartet werden
Postconditions	-

C05: disconnect	
Operation	disconnect()
Cross Reference	UC02: Verbindung beenden
Preconditions	<ul style="list-style-type: none"> Eine Verbindung besteht.
Postconditions	<ul style="list-style-type: none"> Die Verbindung wurde beendet. OnConnectionLost() wurde aufgerufen.

C06: onConnectionLost

Operation	onConnectionLost()
Cross Reference	UC02: Verbindung beenden
Preconditions	<ul style="list-style-type: none">• Verbindung wurde beendet.
Postconditions	<ul style="list-style-type: none">• Alle Actors sind informiert.

C07: onFileTransferStarted

Operation	onFileTransferStarted()
Cross Reference	UC3: Datei senden
Preconditions	<ul style="list-style-type: none">• File wurde übergeben.
Postconditions	<ul style="list-style-type: none">• File wurde an das zweite Gerät gesendet.

C08: getTask

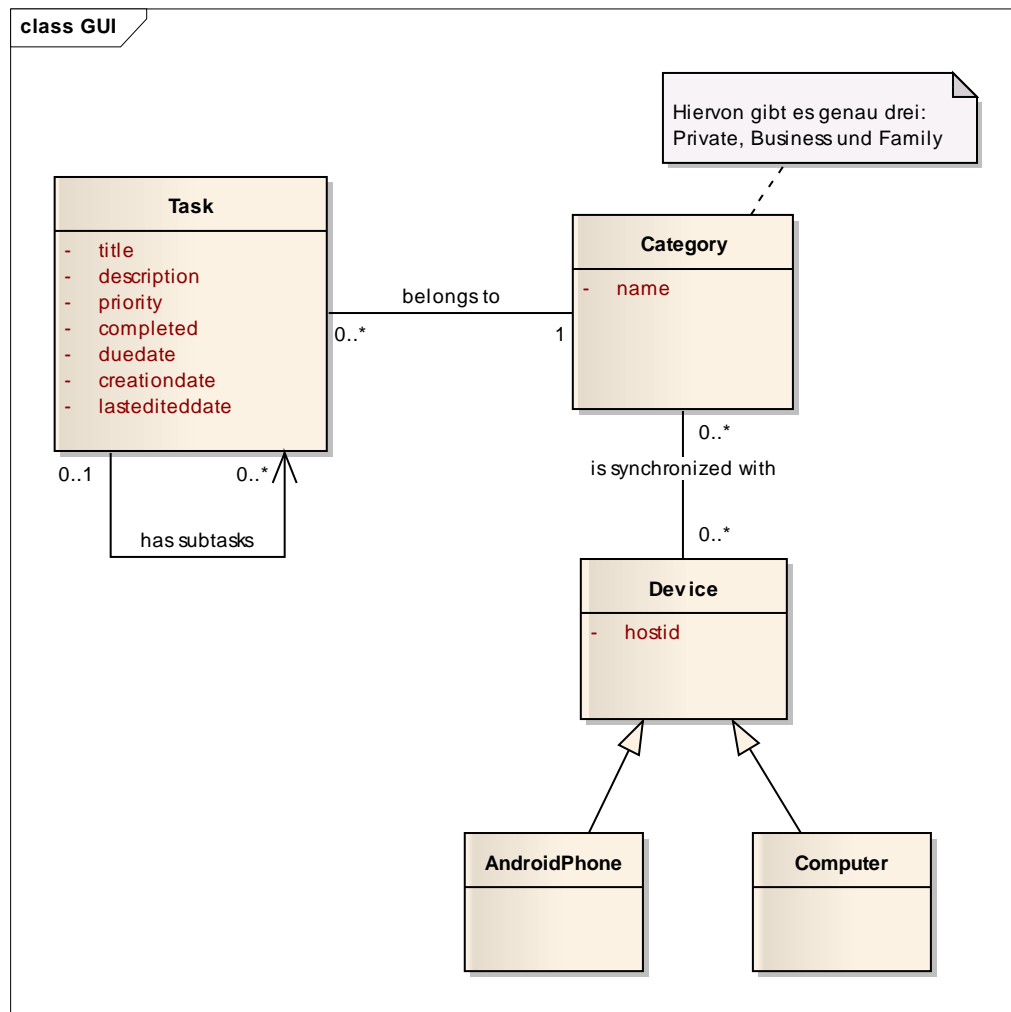
Operation	getTask(UID)
Cross Reference	UC4: Synchronisieren
Preconditions	<ul style="list-style-type: none">• Synchronisation wurde gestartet.• UID Objekt existiert.
Postconditions	<ul style="list-style-type: none">• Daten die transferiert werden müssen, werde zurückgegeben.

C09: enterListenMode

Operation	enterListenMode(UID)
Cross Reference	UC6: Listen Mode einschalten
Preconditions	<ul style="list-style-type: none">• Synchronisation wurde getätigt
Postconditions	<ul style="list-style-type: none">• Gerät wartet auf Empfang von neuen aktualisierten Daten.

3 TaMaF Demo auf Android Phone

3.1 Domain Modell



Dieses Domain Model sieht sehr ähnlich aus zu dem, welches in TaMaF vorhanden ist. Der einzige Unterschied ist, dass hier gezeigt wird wie die Synchronisation im GUI funktioniert. Diese funktioniert so, dass man für die drei Kategorien jeweils bestimmt mit welchen Devices diese synchronisiert wird. Ein Task gehört jeweils zu einer Kategorie und hat dadurch die Information, mit welchen Devices er synchronisiert wird.

In TaMaF ist nicht festgelegt, wie dies genau gemacht wird. Denn dies ist auch unterschiedlich zwischen den Programmen. So wäre z.B. auch möglich ein Programm zu machen, wo man sagt, dass mit einem Device nur Tasks der höchsten Priorität synchronisiert werden.

3.1.1 Konzeptbeschreibung

1	Task	
Beschreibung	Ein Task welcher erfasst wird und schlussendlich auch mit dem anderen Gerät synchronisiert werden soll.	
Attribute	title	Titel des Tasks.
	description	Zusätzliche Beschreibung des Tasks.

	priority	Priorität des Tasks. (zw. 1-5, wobei 5 das höchste ist)
	completed	Wenn gesetzt, ist der Task abgeschlossen.
	duedate	Datum, an welchem der Task erledigt sein sollte.
	creationdate	Datum, an welchem der Task erstellt wurde.
	lastediteddate	Datum, an welchem der Task zuletzt editiert wurde.
Beziehungen	Ein Task kann Subtasks haben. Ein Task kann ein Subtask eines anderen Tasks sein. Ein Task gehört genau einer Category an.	

2	Category	
Beschreibung	Kategorien, welchen ein Task zugeordnet werden kann. Diese sind fix: Private, Business und Family.	
Attribute	name	Name der Kategorie.
Beziehungen	Eine Kategorie kann mehrere Tasks enthalten. Eine Kategorie kann mit mehreren Devices synchronisiert werden.	

3	Device	
Beschreibung	Ist ein anderes Android Phone oder ein Computer mit welchem Tasks synchronisiert werden sollen.	
Attribute	hostid	Eindeutige Id des Devices.
Beziehungen	Ein Device kann mehrere Kategorien zugeordnet haben, welche damit synchronisiert werden. Ein Device kann ein AndroidPhone oder ein Computer (JavaVM) sein.	

4	AndroidPhone	
Beschreibung	Dies ist ein Smart Phone auf welchem das Android Betriebssystem läuft.	
Attribute	Siehe Device	
Beziehungen	Siehe Device	

3	Computer	
Beschreibung	Dies ist ein Computer, auf welchem eine Java Virtual Machine läuft, über welche das Task Programm welches dafür geschrieben wurde läuft.	
Attribute	Siehe Device	
Beziehungen	Siehe Device	

3.2 System Sequenzdiagramme

Werden für das GUI nicht erstellt, da diese praktisch gleich sind wie diejenigen, welche für die Analyse des TaMaF erstellt wurden.

Software Architecture Document

Version 1.1

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	04.10.2009	Erstellen des SAD	MK
1.0rc02	In Bearbeitung	07.10.2009	Logical View erstellt	PB
1.0rc03	In Bearbeitung	07.10.2009	Systemübersicht erstellt	MK
1.0rc04	In Bearbeitung	07.10.2009	Deployment View erstellt	PB
1.0rc05	Review	07.10.2009	Verschiedene Korrekturen vorgenommen	MK
1.0rc06	In Bearbeitung	12.10.2009	Arbeit an Deployment View	PB
1.0rc07	In Bearbeitung	13.10.2009	Kapitel Data View, Paralleliten geschrieben	PB
1.0rc08	In Bearbeitung	19.10.2009	Logging hinzugefügt	PB
1.0rc09	Review	21.10.2009	Überarbeitung der Parallelitäten	MK
1.0rc10	In Bearbeitung	26.10.2009	Bearbeitung gemäss Sitzung	PB
1.0rc11	In Bearbeitung	02.11.2009	Bearbeitung gemäss Sitzung	PB
1.0	Release	04.11.2009	Stabile Version 1.0	TEAM
1.1rc01	In Bearbeitung	01.12.2009	Hinzufügen des Inhaltes für GUI auf Android	PB
1.1	Release	06.12.2009	Stabile Version 1.1	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
3	Übersicht	5
3.1	Systemübersicht	5
4	Logical View TaMaF.....	6
4.1	Application	6
4.2	TaMaF	6
4.3	AndroidSyncFramework	6
4.4	CommunicationLayer	7
4.5	Persistence	7
5	Logical View Android	7
5.1	AndroidDemo.....	7
5.2	TaMaF	7
6	Deployment View	8
6.1	Technologien.....	8
7	Process View	8
7.1	Parallelitäten.....	8
8	Logging	8

1 Einführung

1.1 Zweck

Der Projektplan der Semesterarbeit wird in diesem Dokument beschrieben. Mit diesem Dokument soll gemäss Vorstellung des Projektleiters weitergearbeitet werden können, falls dieser ausfällt.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

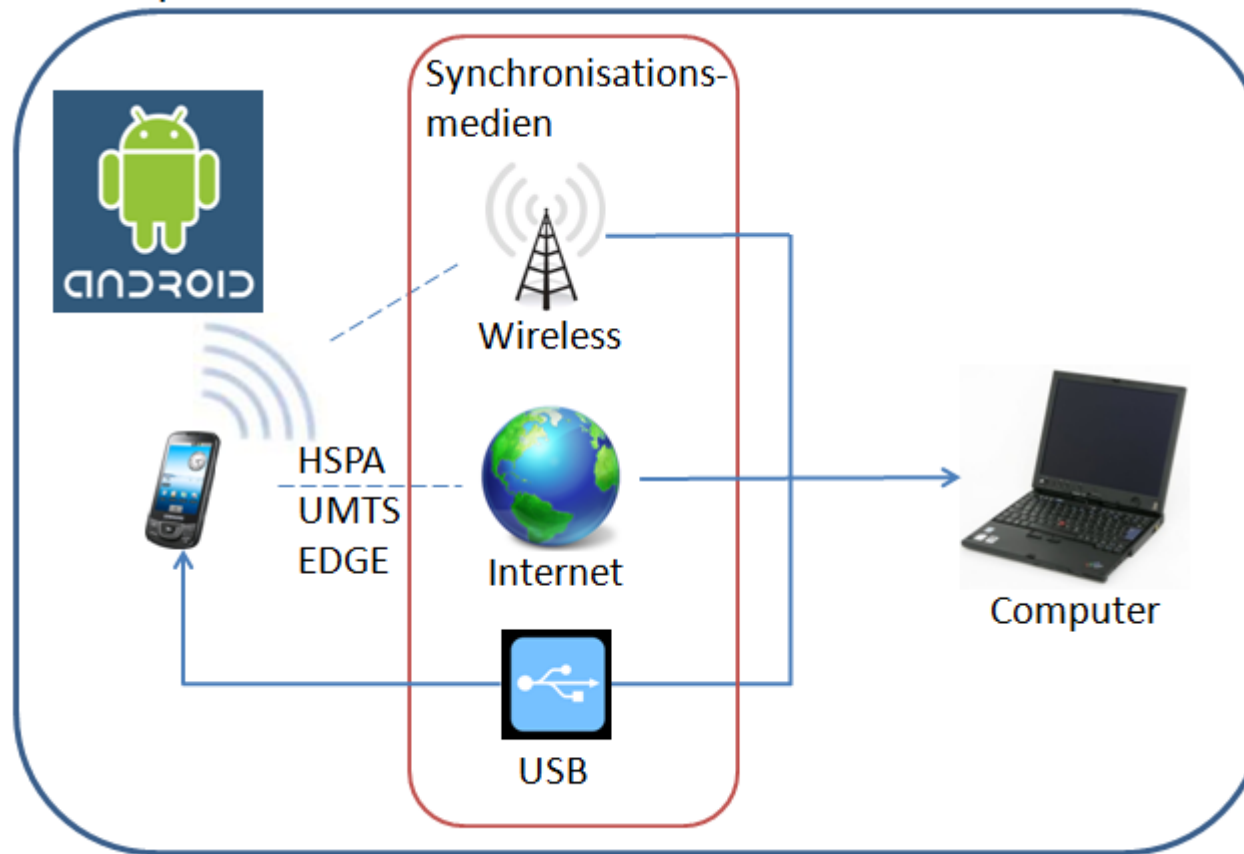
Siehe Dokument Glossar.

1.4 Referenzen

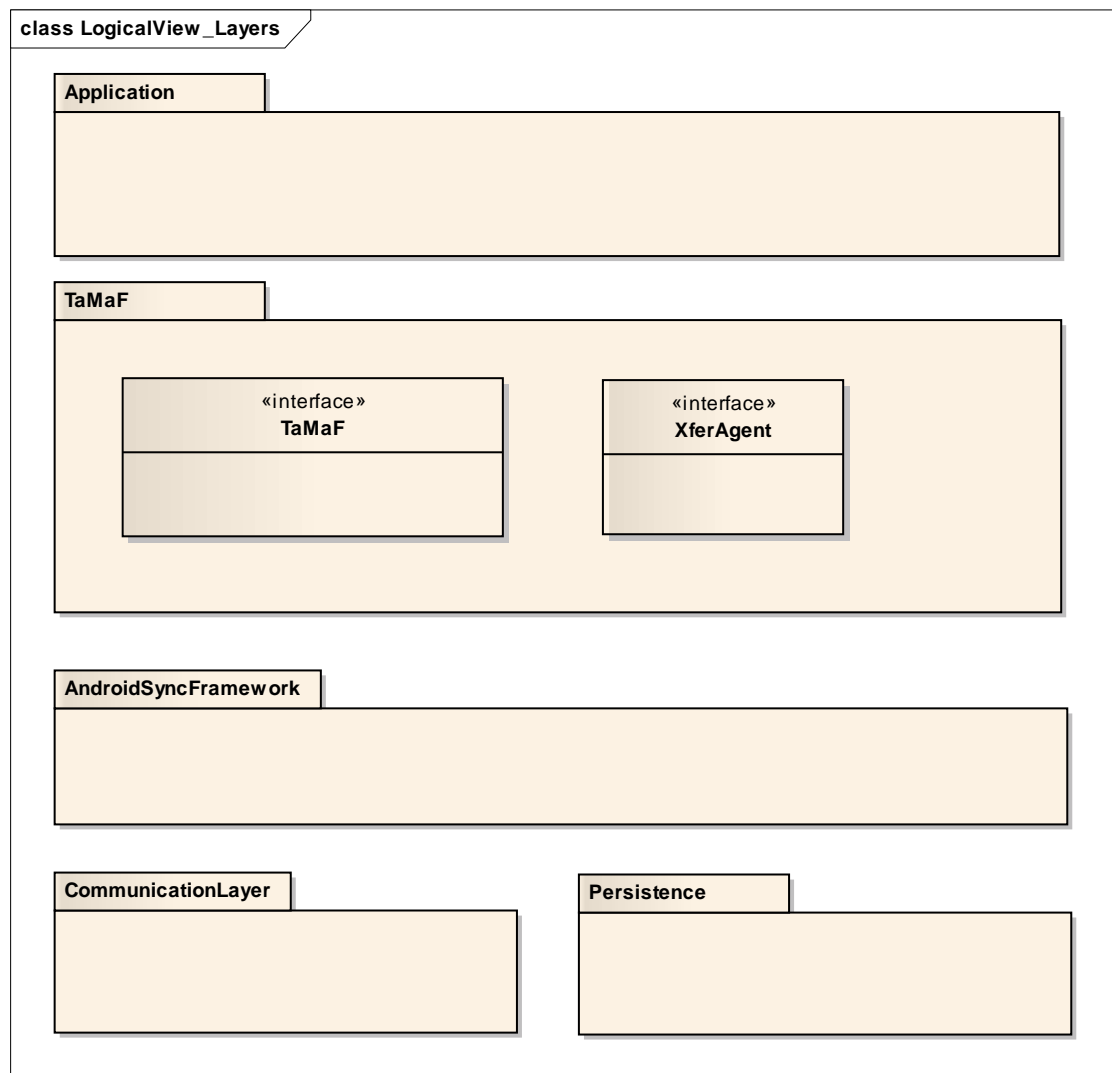
Referenz	Quelle

3 Übersicht

3.1 Systemübersicht



4 Logical View TaMaF



4.1 Application

Dies ist die Task-Management-Applikation, die auf dem TaMaF aufbaut. Es kommuniziert nur mit TaMaF und hat keinen direkten Zugriff auf die unteren Ebenen.

Die Applikation verwaltet und persistiert die Tasks selbst.

4.2 TaMaF

Der TaMaF Layer ist im Prinzipiellen ein sehr dünner Layer. Denn die meisten Funktionalitäten werden vom AndroidSyncFramework angeboten. So ist die eigentliche Funktion des TaMaF Layers als eine Art Wrapper des AndroidSyncFramework zu verstehen. Im Unterschied zum AndroidSyncFramework übergibt man dem TaMaF Tasks.

Über das TaMaF Interface greift die Applikation auf TaMaF zu und durch die Xfer-Config gibt die Applikation dem TaMaF weiter, wie dieser Informationen von oben erhalten oder Notifikationen senden kann.

4.3 AndroidSyncFramework

Dieser Layer übernimmt die eigentliche Funktion des Synchronisierens. Er verwaltet die Information, was mit wem synchronisiert wurde und was noch mit wem synchroni-

siert werden muss. Bei einer Synchronisation informiert dieser Layer den darauf aufbauenden Layer über neu/geänderte/gelöschte Tasks.

4.4 CommunicationLayer

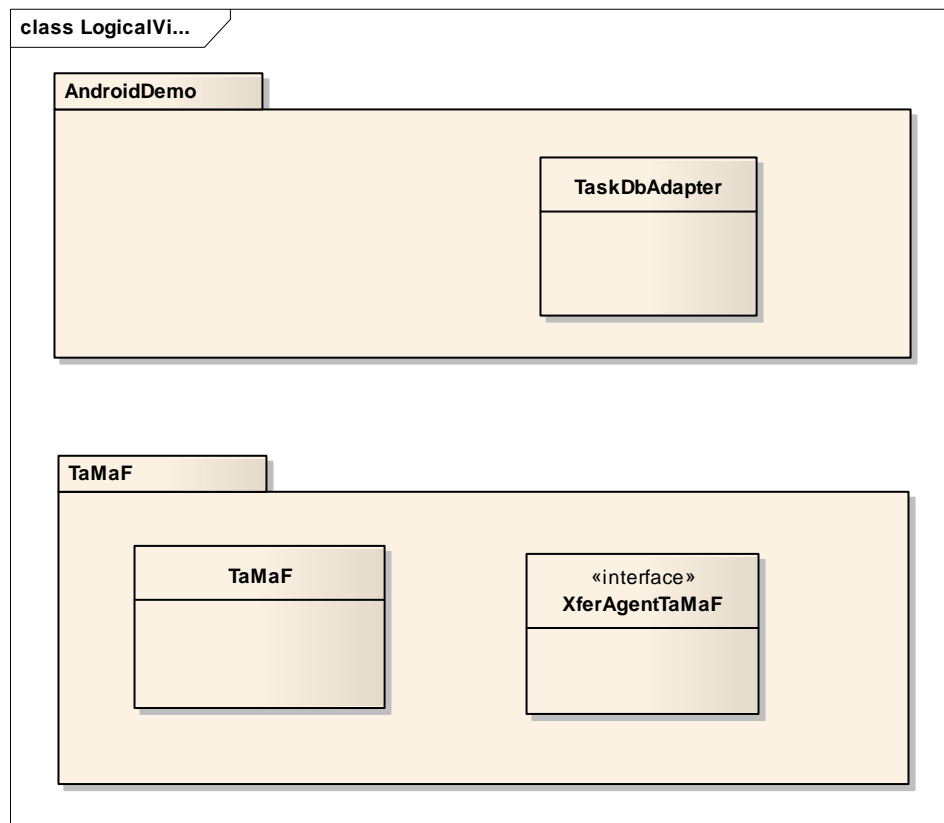
Der CommunicationLayer übernimmt die Kommunikation über eine Schnittstelle (z.B. Ethernet).

4.5 Persistence

Übernimmt die Speicherung der im AndroidSyncFramework gespeicherten Informationen. Somit ist auch nach einem Neustart des Handys oder des Programmes noch bekannt, was beim nächsten Synchronisierungsvorgang mit einem Gerät synchronisiert werden muss.

Die Persistence ist austauschbar, damit diese auf dem Handy anders sein kann als auf dem Computer. In diesem Projekt wird auf beiden Seiten mit SQLite gearbeitet. Dies aus dem Grunde, da darin ein Index erstellt wird, der schnelles suchen erlaubt. Ebenfalls die komfortable Abfragesprache SQL ist ein Grund für diese Entscheidung.

5 Logical View Android



5.1 AndroidDemo

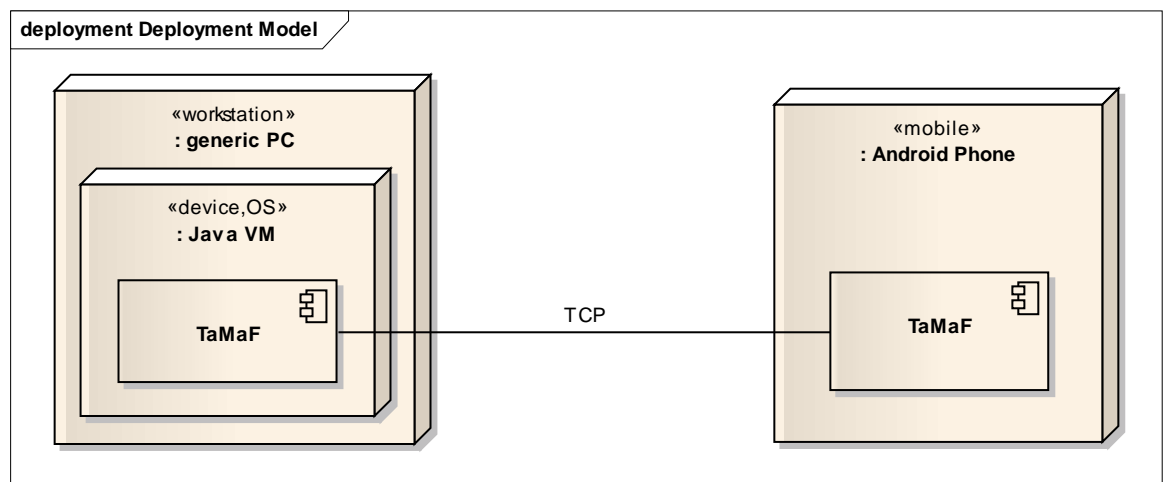
In diesem Layer befindet sich die ganze GUI-Applikation. Diese speichert und holt über den TaskDbAdapter Informationen zu den Tasks die erstellt werden. Die Applikation muss also ihre Tasks selbst verwalten.

Die Applikation greift auf die Funktionen des TaMaF zurück um die Synchronisation zu bewerkstelligen.

5.2 TaMaF

Dies stellt das in Kapitel 5 beschriebene TaMaF dar, worauf die Applikation aufbaut.

6 Deployment View



6.1 Technologien

Technologie	Version
Android	1.6
Java	1.6
SQLite	3.6.14.2 (mit JDBC v056) Auf dem Smart-Phone die von Android gelieferte Version.
Enterprise Architect	7.5.845

7 Process View

7.1 Parallelitäten

Folgend werden Parallelitäten in der Applikation betrachtet, welche daraufhin beschrieben werden:

Zwei Geräte, die sich miteinander synchronisieren stellen zwei separate Prozesse dar, die miteinander kommunizieren.

Das User Interface wird ebenfalls durch einen eigenen Thread laufen, damit keine Blockierungen im GUI auftreten. Was zu beachten ist, dass bei einer Blockierung bei Android ein Error auftreten würde ohne die Parallelisierung.

Im Design Dokument wird definitiv entschieden, ob für die TCP Verbindung (senden & empfangen) parallele Threads benötigt werden.

Gleichzeitige Verbindungen mit mehreren Geräten benötigen ebenfalls Parallele Prozesse.

8 Logging

Es wird über `java.util.logging` geloggt.

Android verwendet intern ein eigenes Logging System, bietet jedoch auch an über `java.util.logging` zu loggen. Dabei wird in Android ein Handler benutzt, der zum Loggen über die ADB genutzt wird. Dieser bildet die Levels von `java.util.logging` auf die

Levels von android.util.Log ab. Jedoch können die Handler auch ausgetauscht werden um z.B. in eine Datei zu loggen.

Das Mapping unter Android wird intern wie folgt gemacht:

Level in Java	Level in Android
Severe	Error
Warning	Warn
Info	Info
Config	Debug
Fine Finer Finest	Verbose

Da man java.util.logging somit auf dem Android Smart-Phone sowie in der Applikation auf dem Computer verwenden kann, wird dies verwendet.

Design Dokumentation

Version 1.0

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	30.11.2009	Erstellen des Design Dokuments	PB
1.0rc02	In Bearbeitung	03.12.2009	Kapitel Persistenz und Klassendiagramme wurden erstellt.	MK
1.0rc03	In Bearbeitung	08.12.2009	Kapitel zu tamaf, androidsync, communication und utils wurden verfasst.	PB
1.0rc04	In Bearbeitung	08.12.2009	Kapitel Persistenz wurde beschrieben	MK
1.0rc05	In Bearbeitung	08.12.2009	Weiteren Punkt unter Verbesserung hinzugefügt (android.jar)	PB
1.0rc06	In Bearbeitung	09.12.2009	Entscheide hinzugefügt (XML und Netzwerk-art)	PB
1.0rc07	In Bearbeitung	15.12.2009	Logging-Beschreibung wurde erweitert.	PB
1.0rc08	In Bearbeitung	16.12.2009	XML Pakete beschrieben	PB
1.0rc09	Review	16.12.2009	Korrekturen vorgenommen	PB
1.0rc10	Review	16.12.2009	Korrekturen vorgenommen	MK
1.0	Release	16.12.2009	Stabile Version 1.0	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	TaMaF	5
2.1	package tamaf	5
2.1.1	class TaMaF	6
2.1.2	interface XferAgentTaMaF	6
2.1.3	class Task	6
2.2	package androidsync	6
2.2.1	interfaces	7
2.2.2	class AndroidSyncFramework	8
2.2.3	class Device	9
2.2.4	messages	10
2.3	package communication.....	14
2.3.1	Entscheid XML	15
2.4	package persistence	16
2.4.1	Implementation Proxy Pattern.....	16
2.4.2	Implementation Active Record Pattern.....	17
2.4.3	Zugriff auf die Datenbank auf dem PC.....	18
2.4.4	Zugriff auf die Datenbank auf dem Android	18
2.4.5	CommunicationDB	18
2.5	package utils.....	19
2.5.1	XmlCreatorHelper und XmlReaderHelper	19
2.5.2	SyncLogManager	19
3	Kernentscheide	20
3.1	Flexibles/Fixes Netzwerk	20
3.1.1	Fix.....	20
3.1.2	Flexibel	20
3.1.3	Entscheid für flexibles Netzwerk.....	21
4	Verbesserungen und Weiterentwicklungen	21
4.1	Verbesserungen.....	21
4.1.1	Listener an Stelle der onXYZ-Funktionen im XferAgent	21
4.1.2	HostId	21
4.1.3	onXferStateUpdate + XferStateUpdateHandler	22
4.1.4	deleteHostID	22
4.1.5	android.jar auf PC	22
4.1.6	FileTransfer mit Handy ist etwas langsam.....	22
4.2	Weiterentwicklungen	23
4.2.1	Security	23

1 Einführung

1.1 Zweck

In der Design Dokumentation wird das Design von TaMaF sowie der GUI Applikation auf dem Android Phone beschrieben. Ebenfalls werden Design Entscheide beschrieben.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

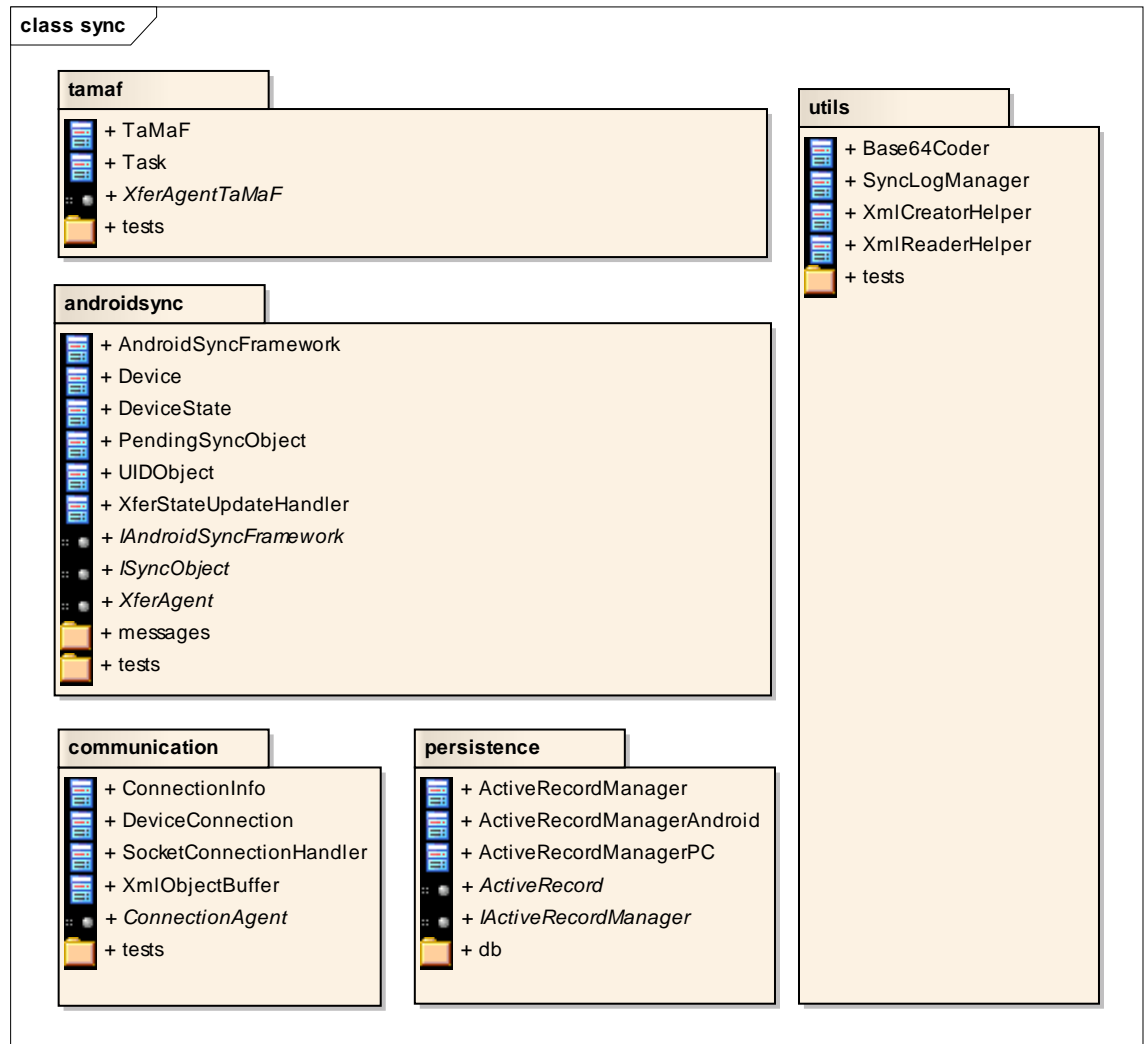
1.3 Definitionen und Abkürzungen

Siehe Dokument Glossar.

1.4 Referenzen

Referenz	Quelle

2 TaMaF

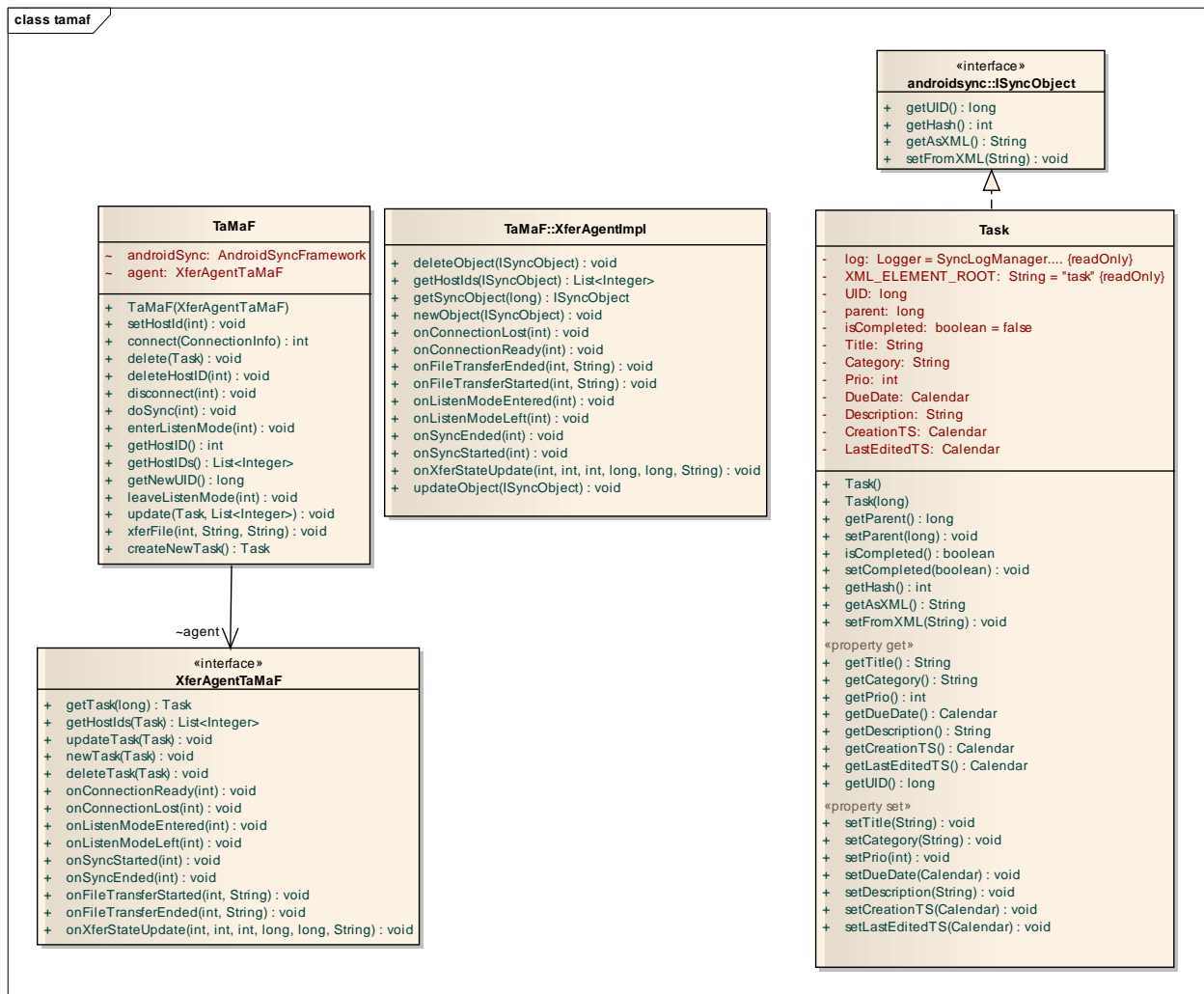


TaMaF basiert auf mehreren Layers welche zusammen das gewünschte Resultat erarbeiten. Das Package „utils“ bietet oft benutzte Utensilien, welche von mehreren Layern genutzt werden.

Die Layers werden im Folgenden genauer beschrieben.

2.1 package tamaf

Dieser Layer ist sehr klein und beinhaltet hauptsächlich das umwandeln von *ISyncObjects* in *Tasks*, damit das auf TaMaF aufbauende Programm, gleich mit *Tasks* arbeiten kann. Theoretisch könnte die Applikation auch gleich mit *AndroidSyncFramework* arbeiten. Der TaMaF-Layer macht die Arbeit mit *Tasks* jedoch komfortabler.



2.1.1 class TaMaF

Die Klasse TaMaF funktioniert als Adapter für das AndroidSyncFramework. Sie bietet fast dieselbe Funktionalität und leitet die Aufrufe entsprechend weiter.

Grösster Unterschied ist die neue Funktion `createNewTask()`, welche einen Task erstellt und diesen gleich mit einer neuen UID aus dem AndroidSyncFramework ausstattet.

Die Klasse enthält intern eine Implementation des XferAgent Interfaces, um die Meldungen vom AndroidSyncFramework über das XferAgentTaMaF weiter an die Applikation zu leiten.

2.1.2 interface XferAgentTaMaF

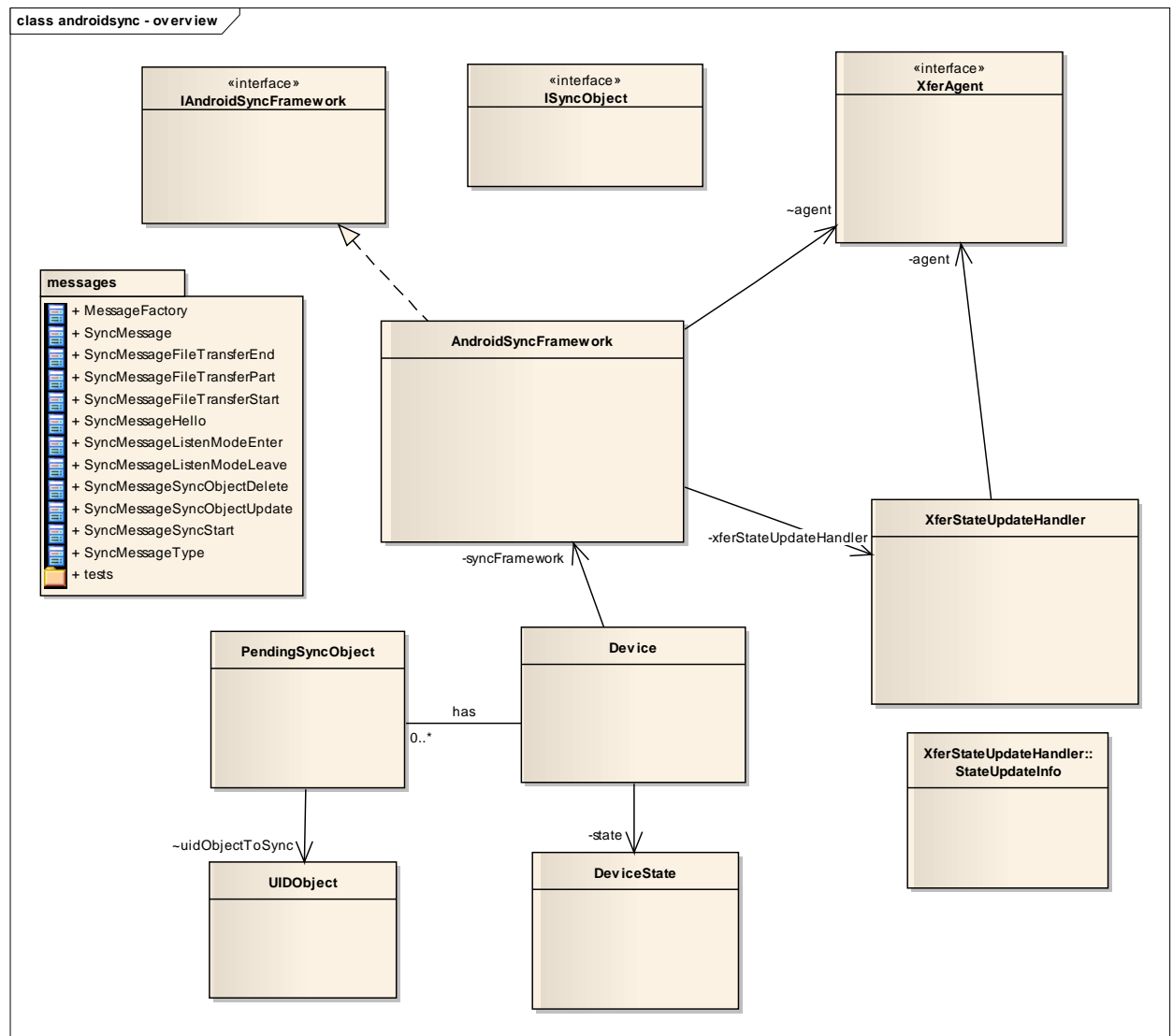
Dies ist fast gleichbedeutend wie XferAgent aus TaMaF. Jedoch sind die Funktionen auf Tasks ausgerichtet anstatt auf ISyncObjects.

2.1.3 class Task

Die Klasse Task implementiert das Interface ISyncObject, welches von androidsync vorgegeben ist. Dieses Interface muss implementiert werden, dass die Task Objekte synchronisiert werden können.

2.2 package androidsync

In diesem Layer steckt die Logik der Synchronisation.



2.2.1 interfaces

Die Interfaces sind die wichtigsten Zugriffspunkte für einen Layer, welcher auf androidsync aufbaut.

2.2.1.1 ISyncObject

Das ISyncObject ist das Interface, welches von den Objekten implementiert werden muss, welche synchronisiert werden wollen.

Folgend sind die wichtigen Eigenschaften, welche diese Objekte haben müssen:

- Konstruktor ohne Parameter
- besitzt eine UID (long)
- als XML darstellbar
- kann sich von einem XML setzen lassen
- bietet eine Hash-Funktion an

UID

Der Unique Identifier kurz UID, erhält man vom AndroidSyncFramework über getNewUID(). Damit wird ein Objekt im AndroidSyncFramework eindeutig identifiziert.

Diese UID muss das Objekt über die getUID()-Funktion bekannt geben.

getHash()

Der Hash sollte sich auch unterscheiden, wenn das Objekt nur geringfügig geändert wurde.

getAsXML() und setFromXML(...)

Das Objekt muss in der Lage sein, sich als XML zu repräsentieren. Wenn dem Objekt dieser XML-String über setFromXML() übergeben wird, muss es sich in dieses, im XML repräsentierte Objekt, verändern. Um dies zu erreichen werden im Package utils die Klassen XmlCreatorHelper und XmlReaderHelper zur Verfügung gestellt.

Grund für diese Notwendigkeit ist, dass das Objekt beim Synchronisieren in seiner XML-Repräsentation über die Verbindung zum anderen Gerät gesendet wird. Dort muss das Objekt mit dieser Information wiederhergestellt werden können. Dafür erstellt das AndroidSyncFramework erst ein Objekt über den Konstruktor ohne Parameter und übergibt dann die XML-Repräsentation an das Objekt per setFromXML(...).

2.2.1.2 IAndroidSyncFramework

Dieses Interface wird vom AndroidSyncFramework implementiert. Es handelt sich hierbei um eine Facade für das AndroidSyncFramework und bietet Funktionen an, welche von den darauf aufbauenden Layern benötigt werden.

Wenn von einem oberen Layer auf das AndroidSyncFramework zugegriffen wird, dann sollte dies vorzugsweise immer über dieses Interface geschehen.

«interface» IAndroidSyncFramework	
+	getHostID() : int
+	getNewUID() : long
+	config(agent :XferAgent) : void
+	xferFile(hostID :int, pathFrom :String, pathTo :String) : void
+	update(obj :ISyncObject, syncWith :List<Integer>) : void
+	delete(obj :ISyncObject) : void
+	connect(info :ConnectionInfo) : int
+	disconnect(hostID :int) : void
+	doSync(hostID :int) : void
+	enterListenMode(hostID :int) : void
+	leaveListenMode(hostID :int) : void
+	getHostIDs() : List<Integer>
+	deleteHostID(hostID :int) : void

2.2.1.3 XferAgent

Dieses Interface muss von dem auf androidsync aufbauenden Layer implementiert werden und per config() dem AndroidSyncFramework weitergegeben werden. Dies um die Kommunikation zwischen dem androidsync Layer mit dem oberen Layer sicherzustellen. Dazu gehören Meldungen, welche an den oberen Layer weitergeleitet werden sollen und Funktionen, welche es dem AndroidSyncFramework erlauben, das ISyncObject eines Objektes mit einer bestimmten UID zu erhalten. Weitere Informationen über die XferAgent Funktionen sind im Javadoc ersichtlich

«interface» XferAgent	
+	getSyncObject(UID :long) : ISyncObject
+	getHostIDs(obj :ISyncObject) : List<Integer>
+	updateObject(obj :ISyncObject) : void
+	newObject(obj :ISyncObject) : void
+	deleteObject(obj :ISyncObject) : void
+	onConnectionReady(hostID :int) : void
+	onConnectionLost(hostID :int) : void
+	onListenModeEntered(hostID :int) : void
+	onListenModeLeft(hostID :int) : void
+	onSyncStarted(hostID :int) : void
+	onSyncEnded(hostID :int) : void
+	onFileTransferStarted(hostID :int, message :String) : void
+	onFileTransferEnded(hostID :int, message :String) : void
+	onXferStateUpdate(hostID :int, passedTime :int, totalTime :int, sentSize :long, totalSize :long, message :String) : void

2.2.2 class AndroidSyncFramework

Die Klasse AndroidSyncFramework verwaltet die Devices und die zu synchronisierenden UIDs. Das eigentliche versenden und speichern ausstehender Updates wird jedoch in den Devices selbst gemacht.

Wenn ein Update oder ein Delete eines ISyncObject vom oberen Layer eingeht, wird dies überprüft und an die entsprechenden Devices weitergeleitet, welche dann die weitere Arbeit vornehmen. Ebenfalls behandelt das AndroidSyncFramework das Ein-

treffen eines Updates oder Delete welches von einem anderen Device gesendet wurde. Dies wird dem oberen Layer mitgeteilt und an die Devices weitergeleitet, welche ebenfalls über dieses Update/Delete informiert werden sollen.

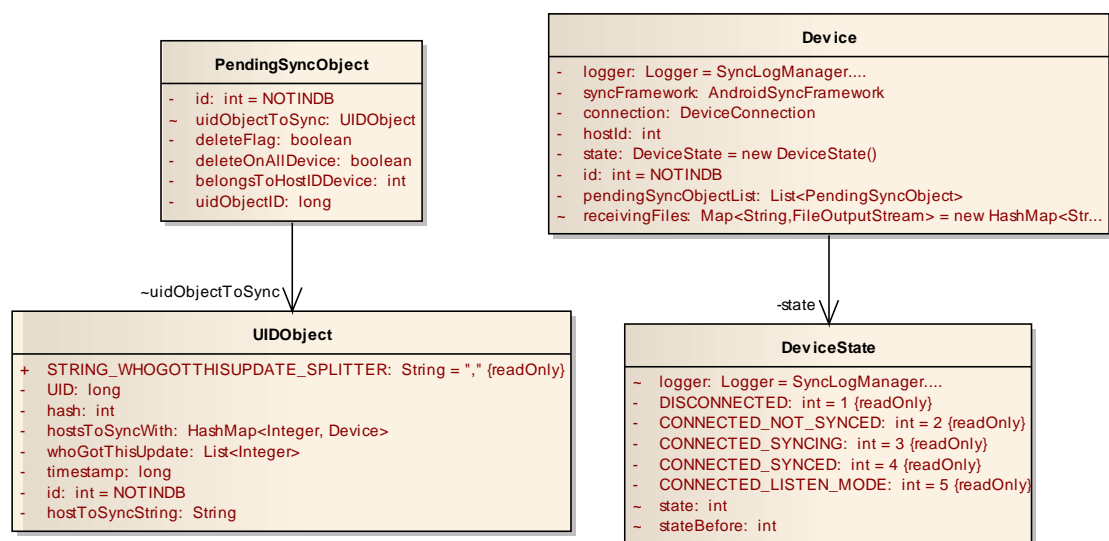
Informationen welche an den oberen Layer weitergereicht, oder von daher eingeholt werden sollen, werden über den XferAgent gesendet.

2.2.2.1 class XferStateUpdateHandler

Die Klasse XferStateUpdateHandler handhabt die Updates, welche an den oberen Layer über XferAgent weitergeleitet werden.

Die momentane Implementation verhält sich noch nicht 100%ig wie gewünscht. Deshalb gäbe es hier noch Verbesserungspotential, welches in dem Kapitel Verbesserungen noch genauer beschrieben wird.

2.2.3 class Device



Die Klasse Device stellt ein anderes Gerät dar, mit welchem eine Verbindung hergestellt werden kann oder bereits verbunden wurde.

Es enthält eine Liste von PendingSyncObjects (im Device als pendingSyncObjectList) was die Updates/Deletes repräsentiert, welche noch nicht an das Device gesendet werden konnten. Bei einer Synchronisation mit einem bestimmten Device wird diese Liste abgearbeitet und die entsprechenden Meldungen gesendet.

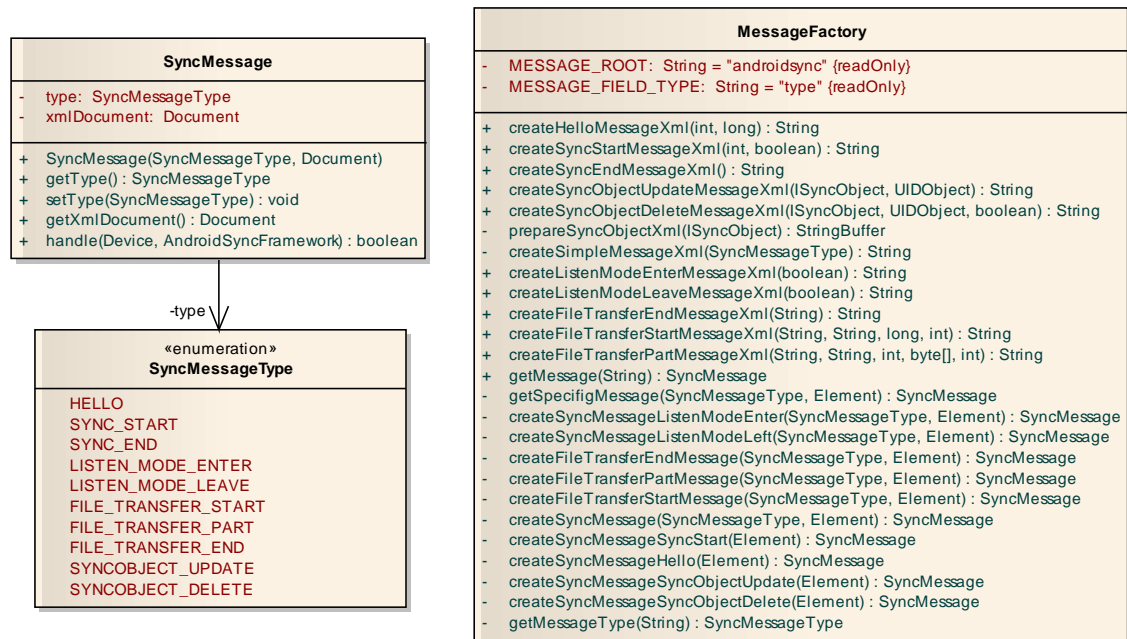
Ein PendingSyncObject gibt an, was mit dem Objekt geschieht (Update oder Delete) und bezieht sich auf ein UIDObject, welches ein ISyncObject repräsentiert. Es enthält jedoch keine Referenz auf dieses, da dies nur benötigt wird, wenn eine Meldung über eine Verbindung zu einem Gerät gesendet werden soll. Danach wird dieses Objekt über den XferAgent geholt.

2.2.3.1 class DeviceState

Die Klasse DeviceState repräsentiert den Status eines Devices. Diese gehen von „nicht verbunden“ (DISCONNECTED) bis zu „im Listen-Mode“ (CONNECTED_LISTEN_MODE).

Diese Klasse bietet die Möglichkeit einen Zustandswechsel durchzuführen. Ebenfalls wird dabei zurückgegeben, ob ein Wechsel von dem Momentanen in den neuen Zustand möglich war. So darf z.B. vom Zustand „nicht synchronisiert“ kein Übergang in den „im Listen-Mode“ Zustand geschehen.

2.2.4 messages



Beim oben ersichtlichen Bild werden die Messages dargestellt, welche über die Verbindung zu einem anderen Device übermittelt werden. Für jeden Typ von Message gibt es eine Ableitung von `SyncMessage`. Aus Platzgründen wurden diese jedoch oben weggelassen, können jedoch im EA angesehen werden.

Eine `SyncMessage` hat einen Typ, welcher ausgibt welche Art von Nachricht es ist und stellt eine Funktion `handle()` zur Verfügung, welche die Nachricht handhabt. Das heisst es wird das ausgeführt, was der Zweck der Nachricht ist. So wird bei einer `SYNC_START`-Message die Synchronisation gestartet.

Die `MessageFactory` stellt Funktionen zum erstellen der Messages zur Verfügung. Dies in beide Richtungen:

- Erstens für die Seite, welche die Nachricht versenden möchte. Dabei werden die öffentlichen `createABCMessageXml()`-Funktionen benutzt, welche den String zurückgeben der die Nachricht darstellt. Diese Nachricht kann dann mit dem Device ausgetauscht werden.
- Auf der anderen Seite erhält das Device über die Verbindung eine solche Nachricht als String. Um die ursprüngliche Nachricht als `SyncMessage`-Objekt zu erhalten, stellt die `MessageFactory` die Methode `getMessage(String)` zur Verfügung. Nun kann die Seite auf der Nachricht die `handle()`-Funktion aufrufen, wodurch die Aktion ausgeführt wird.

Durch das Namenskonzept wird implizit beschrieben, welche Message, was bewirkt. Jedoch wird folgend genauer auf die Struktur der versendeten XML-Messages eingegangen.

2.2.4.1 Grundstruktur XML

Als Beispiel für die Grundstruktur wird hier die `HELLO` Nachricht verwendet, welche beim Aufbau einer Verbindung versendet wird.

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="HELLO">
  <hello hostid="1234" time="1260966314917"/>
</androidsync>
```

Element / Attribut	Beschreibung
<androidsync>	Ist das Root-Element jeder Nachricht
type=""	Gibt den Typ der Nachricht an. Dieser Typ ist die String Repräsentation des Wertes aus dem enum SyncMessageType.
<hello>	Dies ist das Element der eigentlichen Nachricht. Name ist jeweils derselbe wie in TYPE angegeben, jedoch kleingeschrieben. Dieses Element kann weitere Attribute für die Nachricht und auch Unterelemente enthalten.

2.2.4.2 Message Hello

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="HELLO">
  <hello hostid="1234" time="1260966314917"/>
</androidsync>
```

Element / Attribut	Beschreibung
hostid=""	Gibt die HostID des Gerätes an, von welchem diese Nachricht kommt.
time=""	Gibt die momentane Uhrzeit des Gerätes an, von welchem diese Nachricht kommt. Hierbei wird getTimelnMillis() der Klasse Calendar verwendet.

2.2.4.3 Message SyncStart

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="SYNC_START">
  <sync_start isresponse="true" total="10"/>
</androidsync>
```

Element / Attribut	Beschreibung
isresponse=""	Gibt an, ob dies eine Antwort auf eine SyncStart Message ist. Falls "false", heisst dies, dass dieses Gerät das Synchronisieren initialisiert hat.
total=""	Gibt an wie viele Objekte das andere Gerät zu senden hat.

2.2.4.4 Message SyncEnd

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="SYNC_END">
  <sync_end/>
</androidsync>
```

Element / Attribut	Beschreibung
Keine Elemente vorhanden.	

2.2.4.5 Message SyncObjectUpdate

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="SYNCOBJECT_UPDATE">
  <syncobject_update ob-
jectclass="ch.hsr.sync.androidsync.messages.tests.TestSyncObject"
timestamp="1260966314995" uid="321" whogotupdate="1234">
    <testobject>
      <uid>321</uid>
      <field1>test</field1>
      <field2>1111</field2>
    </testobject>
  </syncobject_update>
</androidsync>
```

Element / Attribut	Beschreibung
objectclass=" "	Gibt die Klasse des ISyncObject an, welches hier gesendet wird.
timestamp=" "	Timestamp wann die letzte Änderung an diesem Objekt vorgenommen wurde. Dabei wird die Funktion getTimelnMillis() der Klasse Calendar verwendet.
uid=" "	Die UID des ISyncObject.
whogotthisupdate=" "	Gibt an, wer dieses Update bereits erhalten hat. Ist eine Liste von HostIDs, welche durch Komma getrennt wird.
<testobject>	Dies ist die Repräsentation des Objektes in XML. Dies ist der Rückgabewert der Methode getXML() des ISyncObject. Alles was unter dem <syncobject_update>-Element ist ist die Rückgabe des getXML(), jedoch ohne <?xml...?>.

2.2.4.6 Message SyncObjectDelete

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="SYNCOBJECT_DELETE">
  <syncobject_delete cascade="true" timestamp="1260966314995"
uid="321" whogotupdate="1234"/>
</androidsync>
```

Element / Attribut	Beschreibung
cascade=" "	Gibt an, ob dieser Delete weitergeleitet werden soll. true = weiterleiten false = nicht weiterleiten Grund hierfür ist, dass es sein kann, dass das Objekt nicht wirklich gelöscht wurde, sondern nur nicht mehr mit diesem Gerät synchronisiert wird und darum auf dem Gerät gelöscht werden soll.

timestamp=" "	Siehe timestamp bei SyncObjectUpdate (2.2.4.5)
uid=" "	Siehe uid bei SyncObjectUpdate (2.2.4.5)
whogotthisupdate=" "	Siehe whogotthisupdate bei SyncObjectUpdate (2.2.4.5)

2.2.4.7 Message ListenModeEnter

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="LISTEN_MODE_ENTER">
  <listen_mode_enter isresponse="true"/>
</androidsync>
```

Element / Attribut	Beschreibung
isresponse=" "	Siehe isresponse bei SyncStart (2.2.4.3)

2.2.4.8 Message ListenModeLeave

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="LISTEN_MODE_LEAVE">
  <listen_mode_leave isresponse="true"/>
</androidsync>
```

Element / Attribut	Beschreibung
isresponse=" "	Siehe isresponse bei SyncStart (2.2.4.3)

2.2.4.9 Message FileTransferStart

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="FILE_TRANSFER_START">
  <file_transfer_start file_from="C:\test\file\asdf.zip"
file_to="/sdcard/asdf.zip" parts="1" size="83648"/>
</androidsync>
```

Element / Attribut	Beschreibung
file_from=" "	Gibt den Pfad an, von welchem die Datei kopiert wird.
file_to=" "	Gibt die Zieldatei an.
parts=" "	Gibt an in wie vielen FileTransferPart Messages die Datei gesendet wird.
size=" "	Gibt die totale Grösse der Datei an.

2.2.4.10 Message FileTransferPart

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="FILE_TRANSFER_PART">
  <file_transfer_part file_from="C:\test\file\asdf.zip"
file_to="/sdcard/asdf.zip" nr="1" size="34">
    <data><![CDATA[QSSiKj9g5yoh6FDg0l/gXl866CFgPz0pXide/Khs9i7kJw==]]></data>
  </file_transfer_part>
```

```
</androidsync>
```

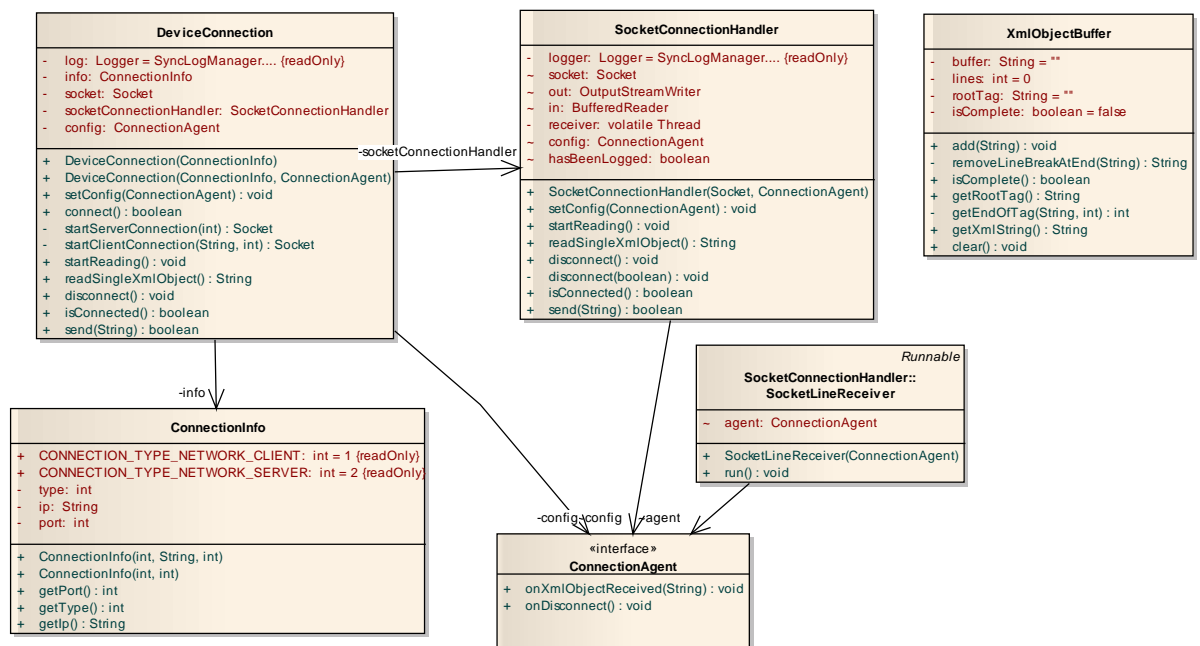
Element / Attribut	Beschreibung
file_from=" "	Siehe file_from bei FileTransferStart (2.2.4.9)
file_to=" "	Siehe file_to bei FileTransferStart (2.2.4.9)
nr=" "	Gibt an, die wievielte FileTransferPart Message dies ist.
size=" "	Gibt die Grösse dieses Teiles an.
<data>	Enthält als CDATA die Base64 encodierten Bytes der Datei.

2.2.4.11 Message FileTransferEnd

```
<?xml version="1.0" encoding="UTF-8"?>
<androidsync type="FILE_TRANSFER_END">
  <file_transfer_end file_to="/sdcard/asdf.zip"/>
</androidsync>
```

Element / Attribut	Beschreibung
file_to=" "	Siehe file_to bei FileTransferStart (2.2.4.9)

2.3 package communication



Wie der Name des Packages aussagt, behandelt dieser Layer die Kommunikation. Die Kommunikation verläuft über ein TCP-Socket.

Die ConnectionInfo wird der DeviceConnection gegeben, damit diese weiss, auf welche Art und mit wem die Verbindung hergestellt werden soll.

DeviceConnection erstellt dann die Verbindung mit der entsprechenden Gegenstelle (Device) und erstellt einen SocketConnectionHandler. Dieser wiederum ist für das Lesen und Schreiben auf dem Socket verantwortlich. Das Lesen erfolgt hier auf Basis von XML. Dem XmlObjectBuffer wird jeweils die empfangene Linie übergeben und

dieser merkt dann, ob das XML-Objekt fertig ist. Somit kann der SocketConnection-Handler dieses dann über den ConnectionAgent an die DeviceConnection weiterreichen. Die DeviceConnection reicht diese Information dann wieder über den ConnectionAgent an ein Device weiter.

Da das Lesen blockierend ist und es auch unschön ist, wenn immer wieder auf diese Funktion zugegriffen werden muss, wird dies in einen SocketLineReceiver ausgelagert sobald das aktive Lesen durch startReading() gestartet wird. Der SocketLineReceiver läuft in einem Thread, welcher dann stetig auf dem Socket liest und falls ein neues XML-Objekt erhalten wurde, wird dieses über den ConnectionAgent an die DeviceConnection weitergereicht.

2.3.1 Entscheid XML

Es wurde entschieden die ganze Kommunikation in Form von XML Paketen zu gestalten. Daraus ergeben sich mehrere Vorteile und wenige Nachteile.

Vorteile:

- Möglichkeit unter anderer Programmiersprache eine TaMaF-Implementation zu schreiben. Wenn wir die Objekte einfach serialisiert hätten, wäre dies nicht möglich.
- Es lässt sich gut in Wireshark nachverfolgen, was genau ausgetauscht wurde.
- Klarer Einblick und gute Steuerung, was und wie übertragen wird.

Nachteile:

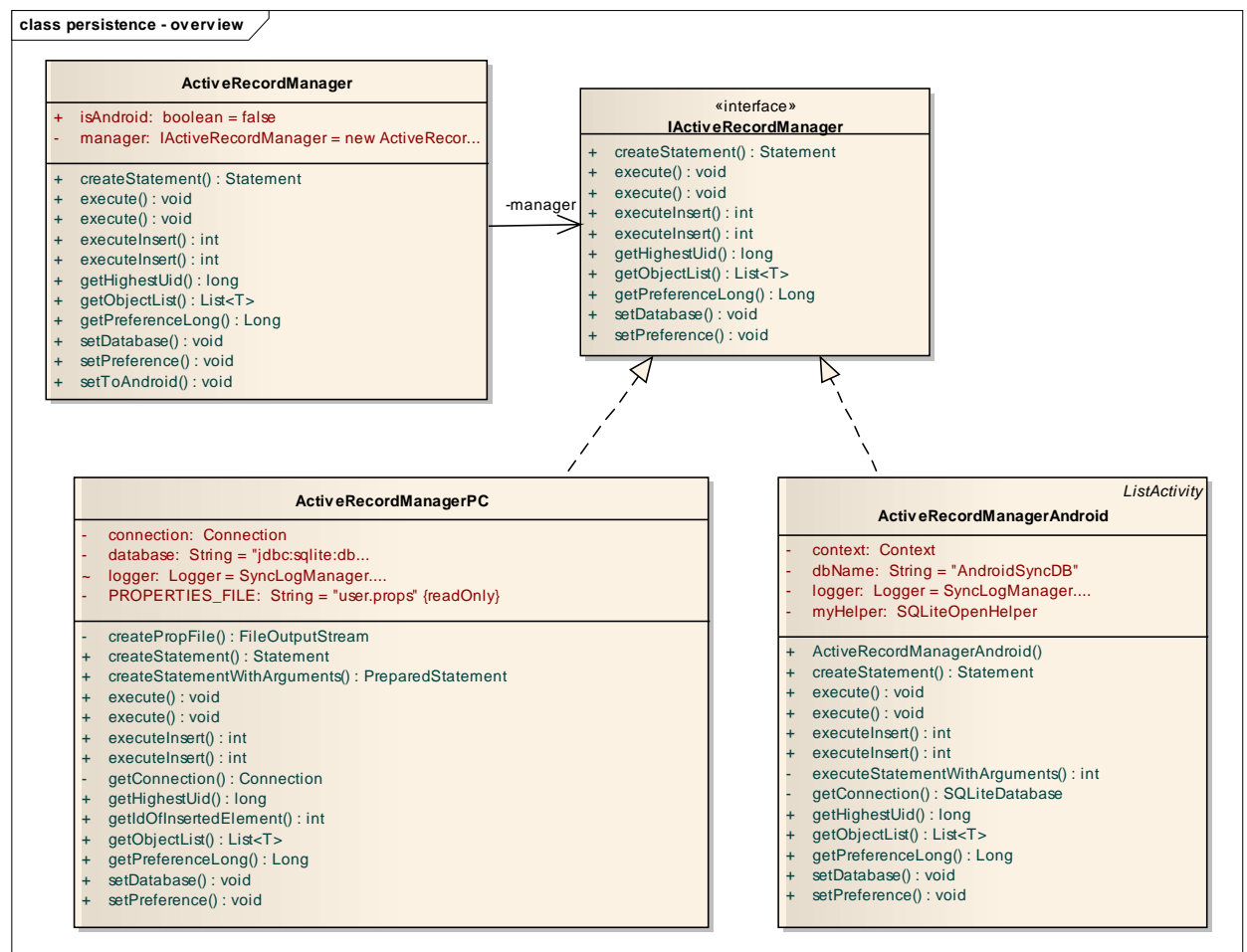
- Die Nachrichten fallen grösser aus. Dies ist jedoch ein Nachteil, der in Kauf genommen wird. Da heute die Netzwerkverbindungen, auch über das Internet, grosse Bandbreiten anbieten, wird dieser Nachteil nicht gross ins Gewicht fallen.

2.4 package persistence

Im Package Persistenz haben wir uns für zwei Pattern (Active Record Pattern & Proxy Pattern) entschieden. Für die Kommunikation und Interaktionen mit einer Datenbank ist das Active Record Pattern hinsichtlich der Konzeption und Ausführung sehr übersichtlich und einfach in der Anwendung.

2.4.1 Implementation Proxy Pattern

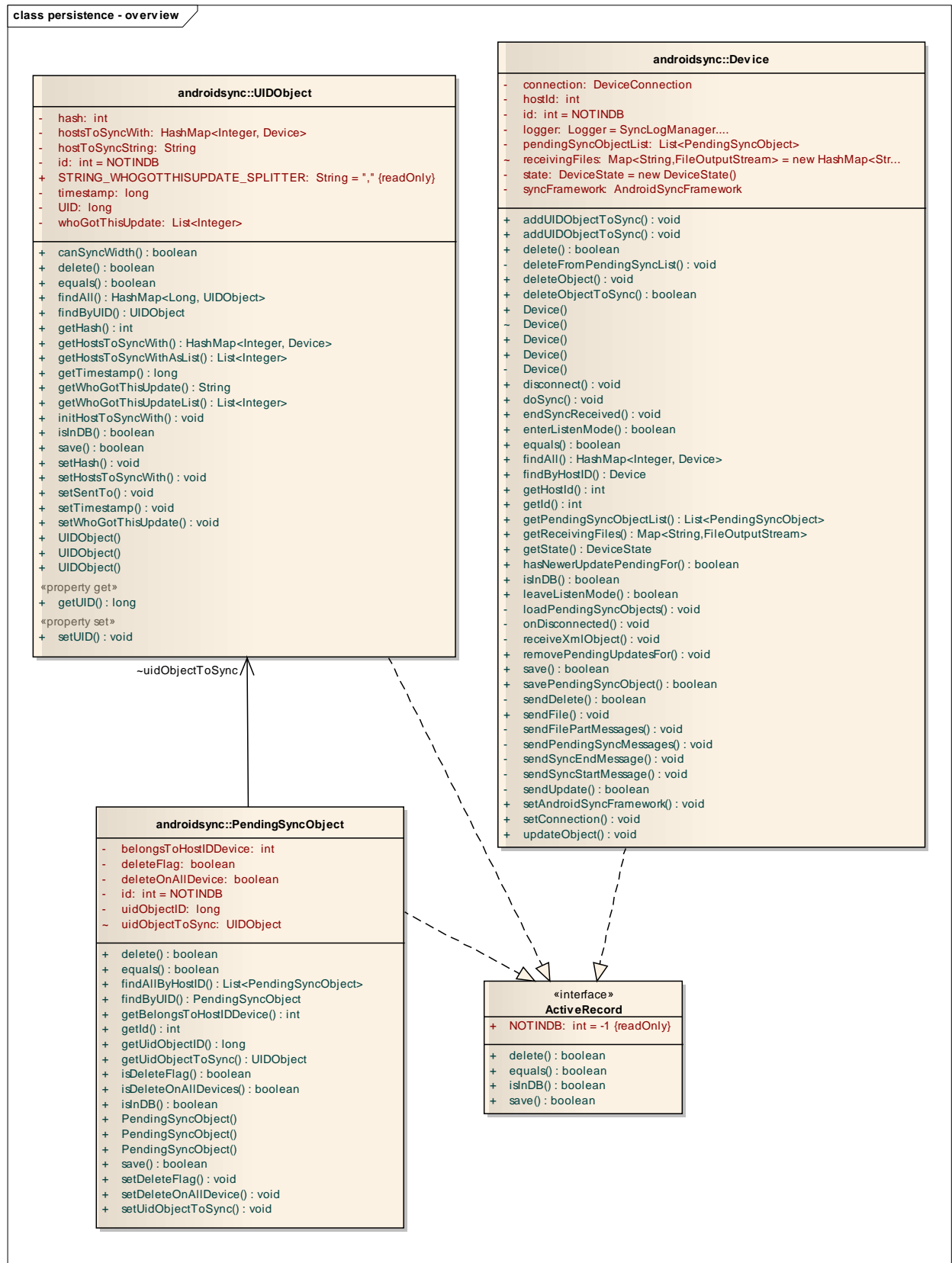
Das Proxy Pattern, welches zur Kategorie Strukturmuster gehört, ist im Persistence Layer eine optimale Lösung. Der Zugriff auf die Datenbank ist auf dem PC und Android unterschiedlich. Auf dem PC wird über die JDBC Library auf SQLite zugegriffen, jedoch bei Android werden Methoden zur Verfügung gestellt, auf welche später noch eingegangen werden. Das Ziel war, dass die TaMaF Library auf der PC Seite gleich bleibt wie auf der Android Seite. So bot sich das Proxy Pattern an, welches die schlussendliche Implementation hinter sich verbirgt. Falls es eine Android Applikation ist, wird über die Methode "setToAndroid()" im ActiveRecordManager eine Referenz auf die Klasse ActiveRecordManagerAndroid gesetzt. Standardmässig wird die Klasse ActiveRecordManagerPC verwendet. Die Klasse ActiveRecordManager speichert die Referenz auf das gewünschte Objekt (ActiveRecordManagerPC/ActiveRecordManagerAndroid).



2.4.2 Implementation Active Record Pattern

Die Klassen `ActiveRecordManagerPC` und `ActiveRecordManagerAndroid` sind der Kern des Active Record Pattern. Über statischen Zugriff auf die Klasse `ActiveRecordManager` werden die Operationen auf der Datenbank ausgeführt.

Das Active Record Interface wird an drei Orten implementiert `UIDObject`, `Device` und `PendingSyncObject`.



2.4.3 Zugriff auf die Datenbank auf dem PC

Der Zugriff auf die Datenbank wird über die JDBC Library getätigt.

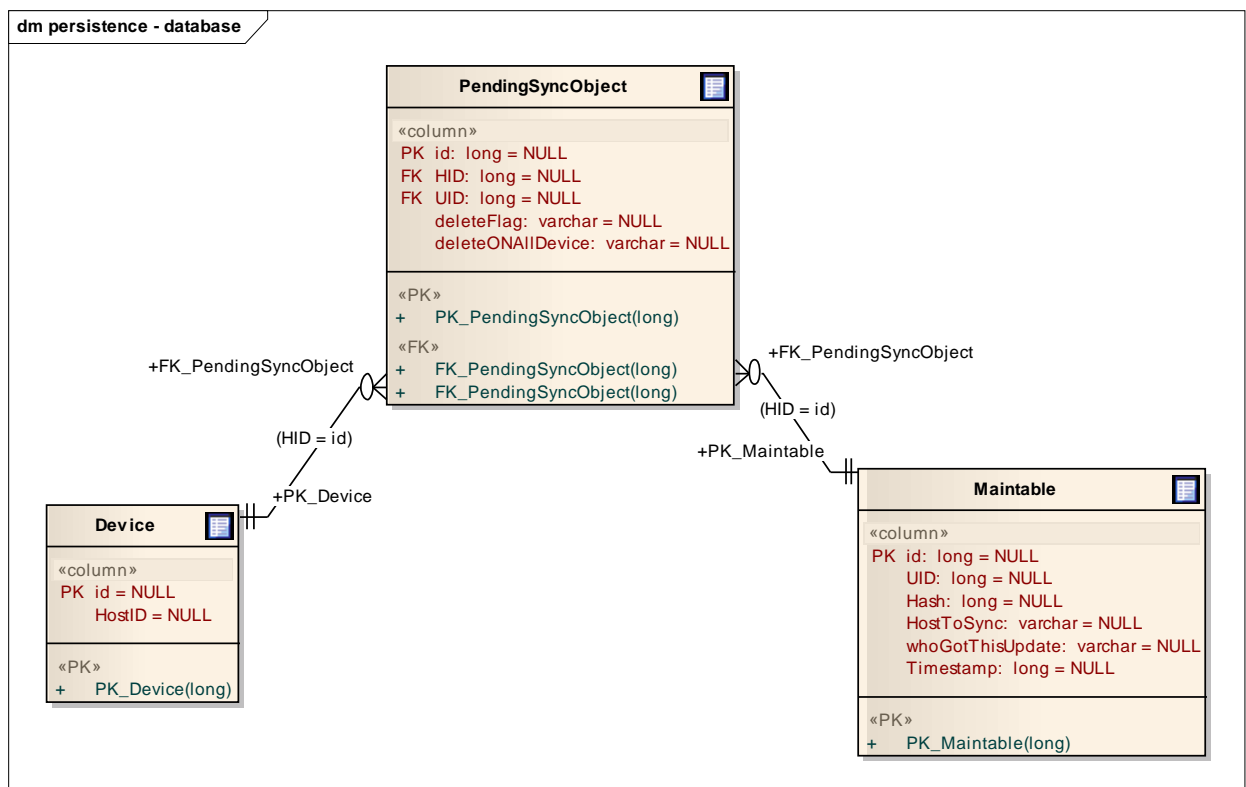
2.4.4 Zugriff auf die Datenbank auf dem Android

Bei Android ist der Zugriff ein wenig anders implementiert. Die SQLiteOpenHelper Klasse verwaltet die Datenbank Connection und Erzeugung. Die Methode onCreate wird aufgerufen, falls die Datenbank nicht existiert. Beim Erstellen der Objekte, welche in der Datenbank sind, muss von der Klasse UIDObject, Device und PendingSyncObject einen Konstruktor zur Verfügung gestellt werden, welcher das Cursor Objekt entgegen nimmt.

2.4.5 CommunicationDB

In der Maintable sind Informationen gespeichert, welche Objekte existieren und mit wem Sie Synchronisiert werden müssen. Zudem wird mit dem Timestamp festgehalten, welches die aktuellste Version des Objektes ist.

In PendingSyncObject wird gespeichert, welche Objekte aus der Maintable noch mit welchem Device synchronisiert werden müssen.



2.5 package utils

XmlCreatorHelper
+ XML_CDATA_END: String = ">" {readOnly} + XML_CDATA_START: String = "<![CDATA[" {readOnly} + XML_HEADER: String = "<?xml version=... {readOnly} + XML_TAB: String = " " {readOnly} + XML_LINE_BREAK: String = "\n" {readOnly}
+ createNewDocument() : Document + createRootElement(Document, String) : Element + documentToString(Document) : String + addVarToXML(Document, Element, String, String) : Element + addCDATAtoXML(Document, Element, String, byte[], int) : Node + addVarToXML(Document, Element, String, Calendar) : Element + addVarToXML(Document, Element, String, int) : Element + addVarToXML(Document, Element, String, long) : Element + addVarToXML(Document, Element, String, boolean) : Element + addSubElement(Element, String) : Element + addExistingElement(Document, Element, Element) : Element + getXml(Node) : String + getXml(Node, int) : String - appendSpacing(int, StringBuilder) : void - appendAttributes(StringBuilder, Element) : void - appendChildNodes(StringBuilder, Element, int) : boolean

XmlReaderHelper
+ getRootElement(String) : Element - replaceBadLineBreaksAndSpaces(String) : String + getSubElement(Element, String) : Element + readElementAsInteger(Element, String) : int + readElementAsBoolean(Element, String) : boolean + readElementAsLong(Element, String) : long + readElementAsString(Element, String) : String + readElementAsCalendar(Element, String) : Calendar + readElementAsCDATA(Element, String) : byte[] + readAttributeAsInteger(Element, String) : int + readAttributeAsString(Element, String) : String + readAttributeAsLong(Element, String) : long + readAttributeAsBoolean(Element, String) : boolean

SyncLogManager
~ globalLevel: Level = Level.INFO ~ ownHandlersSet: boolean = false
+ readConfigFromXML() : void + getClassLogger() : Logger + getLogger(Class<?>) : Logger - getLogger(String) : Logger - getCallerClass() : String + setRootLogLevel(Level) : void + setLogLevel(Level, String) : void + setLogHandlerLevel(Level) : void + setLogHandler(Logger) : void + deleteLogHandlers() : void + addLogHandler(Logger) : void

Base64Coder
- map1: char [] = new char[64] ~ i: int = 0 - map2: byte [] = new byte[128]
+ encodeString(String) : String + encode(byte[]) : char[] + encode(byte[], int) : char[] + decodeString(String) : String + decode(String) : byte[] + decode(char[]) : byte[] - Base64Coder()

Hierbei handelt es sich um Klassen, welche von den oberen Layers verwendet werden. Sie implementieren nicht Logik des Frameworks, sondern stellen global benötigte Funktionalitäten zur Verfügung.

2.5.1 XmlCreatorHelper und XmlReaderHelper

Diese bieten eine vereinfachte Umgangsform mit XML-Dokumenten.

Der XmlCreatorHelper bietet Funktionen, um ein XML-Dokument zu erstellen und es dann in einen String umzuwandeln.

Der XmlReaderHelper bietet an ein XML-Dokument von einem String einzulesen und die verschiedenen gesetzten Werte auszulesen.

2.5.2 SyncLogManager

Stellt statische Funktionen, welche für das Logging benutzt werden können, zur Verfügung.

Über diese Klasse SyncLogManager kann eine Applikation angegeben, was man wie geloggt haben möchte.

Der LogLevel auf einem Package kann über setLogLevel festgelegt werden. Die Logger unterhalb dieses Packages vererben die Konfiguration des oberen Loggers, falls keine eigene Einstellung für den Logger getätigt wurde.

Um einen Logger für das Logging in einer eigenen Klasse zu erhalten, bietet sich die Funktion `SyncLogManager.getLogger(Class<?>)` an. Dabei übergibt man die Klasse, für welche man den Logger erhalten möchte.

`readConfigFromXML()` wurde noch nicht implementiert.

2.5.2.1 *Handler*

Unter dem Computer wird standardmässig der Logger benutzt, welcher die Ausgabe auf den Standard-Error-Ausgabe-Kanal ausgibt. Auf Android wird der Standardmässige `AndroidHandler` verwendet, der das Logging auf LogCat weiterleitet, welches über ADB angesehen werden kann.

Der Handler kann durch `setLogHandler(...)` ersetzt werden. Falls man mehrere Handler gleichzeitig im Einsatz haben möchte, kann man durch `addLogHandler(...)` weitere Handler hinzufügen. Dabei muss man den Handler bereits vor dem Hinzufügen des LogLevels setzen.

3 Kernentscheide

3.1 Flexibles/Fixes Netzwerk

Früh stellte sich die Frage, ob man das Netzwerk der Geräte flexibel oder fix gestalten möchte. Folglich wird beschrieben, wie sich diese unterscheiden und welche Entscheidung getroffen wurde.

3.1.1 Fix

Unter fix versteht man, dass alle Geräte von Anfang an wissen müssen, welche anderen Geräte sich im ganzen Synchronisationsnetzwerk befinden.

Vorteile

- Einfachere Logik im Framework
 - o Nur Gerät, auf welchem das Objekt erstellt wird muss angeben, mit wem es synchronisiert werden soll. Die anderen Geräte wissen dadurch mit wem sie es weiter synchronisieren müssen.

Nachteile

- Jedes Gerät muss alle anderen Geräte kennen, die je synchronisiert werden
- Kommt ein neues Gerät hinzu, muss dies auf jedem Gerät eingetragen werden oder diese Information muss ebenfalls synchronisiert werden, was aber zu weiteren Problemen führen kann.

3.1.2 Flexibel

Flexibel bedeutet, dass ein Gerät von Anfang an keine anderen Geräte kennen muss. Wird mit einem anderen Gerät eine Verbindung aufgebaut wird dieses gemerkt. Ein Gerät kennt nur diejenigen Geräte, welche direkt mit ihm synchronisiert werden.

Vorteile

- Gute Skalierbarkeit
 - o Es können ohne Probleme neue Geräte hinzugefügt werden
- Ein Gerät muss nur seine Nachbarn kennen

Nachteile

- Kompliziertere Logik im Framework
 - o Jedes Gerät muss selbst entscheiden mit wem es das Objekt weiter synchronisiert.

3.1.3 Entscheid für flexibles Netzwerk

Es wurde der Entscheid gefällt die flexible Variante zu nehmen. Grund dafür ist, dass man den Programmierer nicht einengen möchte. Ebenfalls möchte man die Möglichkeit offenlassen einen Server auf TaMaF, oder AndroidSyncFramework, zu programmieren. Dies wäre mit der fixen Variante schwierig, wenn nicht sogar unmöglich, zu lösen.

4 Verbesserungen und Weiterentwicklungen

In diesem Kapitel werden mögliche Verbesserungen und Weiterentwicklungen erwähnt, für welche die Zeit, welche im Rahmen der Studienarbeit zur Verfügung steht, nicht ausreicht. Einige Verbesserungsmöglichkeiten wurden während oder gegen Ende der Studienarbeit entdeckt.

4.1 Verbesserungen

4.1.1 Listener an Stelle der onXYZ-Funktionen im XferAgent

Das AndroidSyncFramework, und auch das darauf aufbauende TaMaF, bietet das XferAgent-Interface, über welches geregelt wird, wie Informationen über Events vom AndroidSyncFramework an die Applikation gelangen.

Beim Entwickeln der GUI auf Android wurde herausgefunden, dass es mit Listeners etwas schöner gewesen wäre. Es funktioniert wie in der Momentanen Lösung umgesetzt, aber es ist nicht so flexibel und einfach handhabbar.

Die Verbesserung wäre, dass man einige Funktionen aus dem XferAgent heraus nimmt und anders umsetzt. Einige Funktionen würden jedoch im XferAgent bleiben. Diese sind: `getSyncObject(...)` und `getHostIds(...)`.

Die anderen würden durch Listeners ersetzt werden. Im AndroidSyncFramework (und im Interface) können dann Listeners für einen Event hinzugefügt werden. Als Beispiele:

- `addOnConnectionReadyListener(...)`
- `addOnXferStateUpdate(...)`
- `addOnUpdateObject(...)`
- `addOnDeleteObject(...)`

Als Parameter würde dann jeweils eine Implementation eines Interfaces mitgegeben werden, welche die gewünschte Aktion handhabt.

Nötig wäre dann auch, dass man diese wieder entfernen kann.

Positive Auswirkung wäre dann, dass sich mehrere Bereiche in der Applikation für einen Event als Listener registrieren können. Ein Beispiel wäre hier z.B. für das `updateObject`-Event, für welches sich da auf der einen Seite die Applikation registriert, die das dann auch speichert, und auch eine Art Log-Fenster, welches die aufgetretenen Aktionen auflistet.

4.1.2 HostId

Anfang der Arbeit war die Idee da, dass man die HostId jeweils für ein Gerät automatisch generiert und diese dann automatisch eindeutig ist. Gedacht war, dass sich diese auf der Mac-Adresse oder IMEI basiert. Jedoch fehlte die Zeit dies zu implementieren. Aus diesem Grund wird die HostId momentan zufällig gesetzt, falls sie nicht durch das Programm über die `setHostId()`-Methode gesetzt wurde.

Die Verbesserung wäre nun, dass wie anfänglich gedacht das Framework die HostId selbst generiert, jedoch nicht einfach zufällig. Dies müsste in der Funktion `initHostId()` angepasst werden. Die Funktion `setHostId()` könnte dann entfernt werden.

4.1.3 onXferStateUpdate + XferStateUpdateHandler

Dies wurde etwas unschön gelöst. Die Momentane Lösung funktioniert, aber es könnte schöner und verständlicher gelöst werden, was jedoch etwas Zeit benötigt.

Im Listen-Mode sind die XferStateUpdate-Meldungen ebenfalls vorhanden, aber sie sehen aus wie SyncUpdates.

So wäre es wahrscheinlich klug onXferStateUpdate aufzuteilen. Eines für das Synchronisieren, eines für den FileTransfer und noch eines für die Updates während dem man sich im Listen-Mode befindet. Dadurch würde die Anzahl der Parameter in den Funktionen kleiner werden und man könnte genauer unterscheiden, um was für eine Art Meldung es sich handelt.

Intern im Framework wurde dies momentan alles in einen Topf geworfen. Dadurch ist es sehr schwer diese Meldungen zu unterscheiden. So würde die obere Änderung auch benötigen, dass man intern einiges anpasst.

4.1.4 deleteHostID

Funktioniert zwar momentan wie gewünscht, jedoch gibt es danach immer noch Überreste der alten HostID. So in der „SyncWith“ Information eines UIDObjects. Dies ist nämlich ein String mit Kommas, in welchem sich die HostIDs der Devices befinden, welche damit Synchronisiert werden. Ebenfalls wird dies in der Datenbank so gespeichert. Problem ist nun, dass beim Löschen zwar der Device gelöscht wird, aber die HostID noch im SyncWith von UIDObjects ist, weil diese Einträge darin nicht so einfach von allen UIDObjects gelöscht werden können. Somit muss nun bei einem Update/Delete jeweils geschaut werden, ob diese HostID wirklich noch existiert.

Man könnte dies schöner lösen, indem man z.B. eine neue Tabelle für das SyncWith macht und dann darin einfach alle UIDObjects mit HostIDs verbindet. Dann könnte man einfach auslesen, welche UIDObjects mit dieser HostID synchronisiert werden. Man könnte dann in den entsprechenden UIDObjects den String anpassen. Diesen String sollte man dann auch in eine Liste umwandeln.

4.1.5 android.jar auf PC

Momentan muss in einer PC-Applikation, welche auf TaMaF - oder Android-SyncFramework – aufbaut die android.jar in den Build-Path aufgenommen werden. Dies ist etwas unschön, jedoch nicht tragisch.

Grund dafür ist, dass der ActiveRecordManager unter Android einen Cursor an die Objekte weitergibt, welche sich dann laut Cursor initialisieren. Jedoch ist dieser Cursor im normalen Java nicht vorhanden. Auf dem PC läuft dies anstatt über den Cursor über ein ResultSet welches dem Konstruktor übergeben wird.

Problem ist hier, dass der Konstruktor über Reflection ausgelesen und aufgerufen wird. Dabei durchläuft er auch die anderen Konstruktoren und sieht den Cursor auch wenn er ihn schlussendlich nicht benötigt.

Lösungsmöglichkeit wäre, dass man dem Konstruktor anstelle eines Cursors eine Map übergibt, welche die von der Datenbank gelesenen Werte enthält. Dadurch würde der Cursor in keinem Konstruktor mehr auftauchen und man bräuchte die android.jar nicht mehr einzubinden.

4.1.6 FileTransfer mit Handy ist etwas langsam

Der FileTransfer läuft zwischen Handys sehr langsam ab. Bei unseren Tests dauerte es etwa 1-2 Minuten um eine 6MB Datei über USB zu übertragen. Über WiFi dauerte es nochmals etwas länger. Grund dafür ist, dass die Datei in XML übertragen wird. Dafür muss die Datei jeweils Base64-encodiert und auf der anderen Seite dekodiert wer-

den. Da dies auf dem Handy einiges länger braucht als auf dem Computer tritt hier die Verzögerung auf.

Als Vergleich dauert derselbe FileTransfer zwischen zwei Computer über WiFi über AndroidSync nur 2-5 Sekunden.

Wie dies genau verbessert werden könnte ist noch nicht klar. Ev. könnte man sich das enkodieren und dekodieren sparen, wenn man es nicht als XML übertragen würde. Dazu müsste man jedoch einiges im Framework ändern.

4.2 Weiterentwicklungen

4.2.1 Security

Momentan wird alles im Klartext übertragen. Wenn dies über die USB-Schnittstelle geschieht ist dies kein Problem. Wenn man das Handy jedoch über das Internet mit dem Computer zu Hause synchronisiert kann jeder, der die TCP-Pakete sieht genau sehen, was für Tasks/Objekte übertragen werden.

Nun könnte man sich überlegen, wie und wo man genau Security einbauen könnte.

Dabei gibt es verschiedene Bereiche die angesehen werden sollten:

- Nur mit berechtigten Systemen Synchronisieren (Authenticity)
- Übertragung verschlüsselt (Confidentiality)
- Übertragung vor Veränderung schützen (Integrity)

Testdokumentation

Version 1.0

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	01.12.2009	Erstellen der Testdokumentation	MK
1.0rc02	In Bearbeitung	01.12.2009	Systemtests durchgeführt	MK
1.0rc03	In Bearbeitung	14.12.2009	Unit Tests & Performance Test dokumentiert	PB
1.0rc04	In Bearbeitung	15.12.2009	Unit Tests dokumentiert	MK
1.0	Release	15.12.2009	Stabile Version 1.0	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	Systemtest.....	5
2.1	Voraussetzung	5
2.2	Vorbereitung	5
2.3	Ablauf	5
2.4	Systemtest	5
3	Unit Tests.....	9
3.1	ch.hsr.sync.androidsync	9
3.2	ch.hsr.sync.messages.....	9
3.3	ch.hsr.sync.communication.....	9
3.4	ch.hsr.sync.persistence	9
3.5	ch.hsr.sync.tamaf.....	9
3.6	ch.hsr.sync.utils	9
4	Integrationstest.....	10
5	Funktionaler Test	10
6	Performance Test	10
7	Usability Test	10

1 Einführung

1.1 Zweck

In der Systemtest Dokumentation befinden sich alle Test-Dokumentation. Dies beinhalten die Systemtests und die Usabilitytests.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

Siehe Dokument Glossar.

1.4 Referenzen

Referenz	Quelle

2 Systemtest

2.1 Voraussetzung

Der Benutzer verfügt über 1x Android Handy und einen Computer mit Java Installation. Um eine Verbindung aufzubauen, benötigt man ein USB Kabel oder Wireless.

2.2 Vorbereitung

Falls über USB die Verbindung aufgebaut wird, müssen folgende Dateien im Ordner vorhanden sein: adb.exe, AdbWinApi.dll & AdbWinUSBApi.dll

Falls über Wireless Daten synchronisiert werden, muss man die IP Adresse vom jeweiligen Zielsystem wissen.

2.3 Ablauf

Alle Use Case werden getestet und beschrieben.

2.4 Systemtest

Name	UC1 Verbindung aufbauen
Beschreibung	Zuerst wird eine Verbindung zum Handy erstellt.
Erwartetes Resultat	Im Logger ist erkennbar, dass eine Verbindung aufgebaut wurde.
Testdaten	
Resultat	OK
Kommentar	Über Wireless sowieso über USB kein Problem.
Testdatum	01.12.2009

Name	UC2 Verbindung beenden
Beschreibung	Sobald synchronisiert wurde, möchte man die Verbindung auch wieder beenden.
Erwartetes Resultat	Im Logger ist erkennbar, dass die Verbindung beendet wurde.
Testdaten	-
Resultat	OK
Kommentar	Über Wireless sowieso über USB kein Problem.
Testdatum	01.12.2009

Name	UC3 Datei senden
Beschreibung	Der Ursprung und Ziel Pfad der Datei muss angegeben, um eine Datei zu senden.
Erwartetes Resultat	Datei ist auf dem Zielsystem vorhanden.
Testdaten	Img86.jpg 1.5MB

Resultat	OK, ein wenig langsam
Kommentar	Datenübertragung ist ein wenig langsam. Der Grund liegt an der Performance des Handys, da die Bytes in die XML Norm konvertiert werden muss.
Testdatum	01.12.2009

Name	UC4 Synchronisieren
Beschreibung	Zuerst wird UC8 getestet, danach wird eine Verbindung aufgebaut und das Synchronisieren getestet.
Erwartetes Resultat	Tasks welche editiert/erstellt wurden, sind auf dem anderen Handy sauber synchronisiert.
Testdaten	Erstellte & Editierte Tasks
Resultat	OK
Kommentar	-
Testdatum	01.12.2009

Name	UC5 Information über Fortschritt der Synchronisation
Beschreibung	Der Progress ist ersichtlich beim Synchronisieren.
Erwartetes Resultat	Während der Synchronisation und des Kopieren eines Files, wird der Fortschritt angezeigt.
Testdaten	1 Byte File
Resultat	OK >> Progress with 1174963532 : 0/1 bytes. 0 seconds left. ([1174963532] SYNC RECEIVE) >> Progress with 1174963532 : 1/1 bytes. 0 seconds left. ([1174963532] SYNC RECEIVE)
Kommentar	Wird mit einem grossen File noch getestet.
Testdatum	01.12.2009

Name	UC6 Listen-Mode einschalten
Beschreibung	Der Listen Mode wird selber eingeschaltet. listenmode enter <hostid>
Erwartetes Resultat	Automatisch werden aktualisierte oder neu erstellte Task an den verbundenen Host gesendet.
Testdaten	-
Resultat	OK
Kommentar	-
Testdatum	01.12.2009

Name	UC7 Listen Mode beenden
Beschreibung	Nach dem beenden der Synchronisation beendet man den Listen Mode. listenmode enter <hostid>
Erwartetes Resultat	Der Listen Mode ist beendet.
Testdaten	-
Resultat	OK
Kommentar	-
Testdatum	01.12.2009

Name	UC8 Task hinzufügen/editieren
Beschreibung	Es werden Tasks erstellt und editiert.
Erwartetes Resultat	Task wurde korrekt geändert.
Testdaten	Beispieltask
Resultat	OK
Kommentar	Der Test funktioniert auf dem Handy sowieso auf PC einwandfrei.
Testdatum	01.12.2009

Name	UC9 Task löschen
Beschreibung	Tasks welche erstellt wurden, werden nun wieder gelöscht.
Erwartetes Resultat	Task sind nicht mehr vorhanden.
Testdaten	UC8 erstellter Task
Resultat	OK
Kommentar	Der Test funktioniert auf dem Handy sowieso auf PC einwandfrei.
Testdatum	01.12.2009

Name	UC10 Einen kompletten Dump machen
Beschreibung	Wird nicht realisiert, gemäss Sitzungsprotokoll 20091008.txt: - "wenn es mehr Probleme gibt, kann man darauf verzichten (selber wissen)". Es ist möglich, dies bei einer Weiterentwicklung zu implementieren.

Name	UC11 Geräte auflisten
------	-----------------------

Beschreibung	Alle Geräte, welche mit dem jeweiligen Gerät einmal verbunden werden aufgelistet.
Erwartetes Resultat	Alle Geräte werden aufgelistet
Testdaten	-
Resultat	OK
Kommentar	Auf PC als eigener Befehl vorhanden. Auf dem Android werden die bekannten Geräte in der Synchronisations-Konfiguration der Kategorien angezeigt.
Testdatum	01.12.2009

Name	UC12 Geräte löschen
Beschreibung	Geräte können entfernt werden.
Erwartetes Resultat	Gerät wird nicht mehr aufgelistet.
Testdaten	-
Resultat	OK
Kommentar	Nur auf PC Seite implementiert
Testdatum	01.12.2009

Name	UC13 Verbundenes Gerät abfragen
Beschreibung	Programm speichert welches Gerät ready ist.
Erwartetes Resultat	In der Konsole wird sauber ausgegeben, mit welchem Gerät man verbunden ist.
Testdaten	-
Resultat	OK >> Connection ready with 1174963532
Kommentar	-
Testdatum	01.12.2009

3 Unit Tests

3.1 ch.hsr.sync.androidsync.tests

Testbeschreibung	Resultat
Testet das AndroidSyncFramework. Testet dies Anhand indem es Zwei Geräte simuliert, welche mit einander kommunizieren und sich dann verbinden.	OK

3.2 ch.hsr.sync.messages.tests

Testbeschreibung	Resultat
Testet ob die Messages richtig zusammengesetzt werden. Auch wird getestet, ob diese generierten Messages dann als String wieder richtig interpretiert werden.	OK

3.3 ch.hsr.sync.communication.tests

Testbeschreibung	Resultat
Testet ob das Verhalten des XmlObjectBuffer korrekt ist. Testet den SocketConnectionHandler, welcher die Verbindung verwaltet.	OK

3.4 ch.hsr.sync.persistence.tests

Testbeschreibung	Resultat
Testet ob die Devices, UIDObjects und PendingSyncObjects gespeichert und wieder ausgelesen werden können.	OK

3.5 ch.hsr.sync.tamaf.tests

Testbeschreibung	Resultat
Testet ob der Erstellte Task die Kriterien erfüllt um als ISyncObject zu funktionieren. TaMaF selbst wurde nicht getestet, da es nur ein Adapter ist und die eigentliche Funktionalität bereits in ch.hsr.sync.androidsync.tests getestet wird.	OK

3.6 ch.hsr.sync.utils.tests

Testbeschreibung	Resultat
Testet die Hilfsklassen welche zur Erstellung und zum Auslesen von XML bereitgestellt werden.	OK

4 Integrationstest

Die Integrationstests wurden in Zusammenhang von Unittests und Systemtests durchgeführt.

5 Funktionaler Test

Anhand der Use Cases wurden mittels Systemtest, die Funktionen überprüft.

6 Performance Test

Name	Kopieren eines Files
Beschreibung	Testen der Dauer beim Senden einer Datei.
Testdatum	01.12.2009
Zeit	PC->Handy: 6 MB in 1-2 Minuten PC->PC: 6 MB in 2-5 Sekunden
Kommentar	Dauer von PC zu Handy ist etwas lange. Jedoch werden höchst wahrscheinlich meistens kleinere Dateien über das Framework gesendet. Aufgrund der Resultate scheint es am Handy zu liegen, welches weniger Leistung erbringen kann als der Computer und somit für die Base64-Codierung länger benötigt als der Computer. Aus diesem Grund kann man das Resultat so akzeptieren und es benötigt keine Verbesserung.

7 Usability Test

Auf den Usability Test wird verzichtet, da es hier um eine Demo Applikation für TaMaF handelt.

Developer Guide

Version 1.0

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

Revision				
Version	Status	Datum	Beschreibung/Änderung	Autor
1.0rc01	In Bearbeitung	07.12.2009	Erstellen des DG mit Inhalt	PB
1.0rc02	In Bearbeitung	15.12.2009	Hinzufügen von Informationen zum Emulator	PB
1.0rc03	In Bearbeitung	15.12.2009	Hinzufügen von Information bezüglich Handy-Treiber	PB
1.0rc04	In Bearbeitung	15.12.2009	Redundanz zu anderen Dokumenten entfernt	PB
1.0rc05	In Bearbeitung	16.12.2009	Redundanz zu anderen Dokumenten entfernt (ISyncObject)	PB
1.0rc06	Review	16.12.2009	Korrekturen vorgenommen	PB
1.0rc07	Review	17.12.2009	Korrekturen vorgenommen	MK
1.0	Release	16.12.2009	Stabile Version 1.0	TEAM

Inhaltsverzeichnis

1	Einführung	4
1.1	Zweck.....	4
1.2	Gültigkeitsbereich.....	4
1.3	Definitionen und Abkürzungen.....	4
1.4	Referenzen.....	4
2	Einstieg	5
2.1	Was ist TaMaF (Task-Management-Framework)?.....	5
3	Grundlagen	5
3.1	TaMaF	5
3.2	XferAgentTaMaF	6
3.3	Technologien.....	6
4	Applikation auf PC	6
4.1	Build Path	6
4.2	Verbindung über USB	7
4.3	Android-Treiber.....	7
5	Applikation auf Android.....	7
5.1	Build Path	8
5.2	Permissions im AndroidManifest.xml.....	8
5.3	ActiveRecordManager auf Android setzen	8
5.4	ADB-Verbindungen zulassen.....	8
5.5	Emulator	9
6	Eigene nicht-Task-Objekte synchronisieren	9
6.1	AndroidSyncFramework	9
6.2	ISyncObject	9
7	Demo-Projekte	9
7.1	TaMaF Demo PC	9
7.2	TaMaF Demo Android	10

1 Einführung

1.1 Zweck

Der Developer Guide richtet sich an Entwickler, welche eine Applikation auf TaMaF entwickeln. Er erklärt wie man die Funktionalität von TaMaF in eigenen Applikationen nutzen kann.

1.2 Gültigkeitsbereich

Die Gültigkeit dieses Dokuments gilt für die gesamte Projektdauer.

1.3 Definitionen und Abkürzungen

Siehe Dokument Glossar.

1.4 Referenzen

Referenz	Quelle

2 Einstieg

2.1 Was ist TaMaF (Task-Management-Framework)?

Siehe hierzu Kapitel 2 des Projektplanes und Kapitel 3 der Anforderungsspezifikation.

3 Grundlagen

3.1 TaMaF

TaMaF ist die Klasse, über welche jeglicher Zugriff stattfindet. Die meisten Funktionen sollten Selbsterklärend sein, wobei der Integer jeweils die HostID eines Gerätes ist mit dem die Aktion stattfinden soll. Deshalb wird im folgenden kurz auf die Funktionen eingegangen, welche etwas spezieller sind.

TaMaF(XferAgentTaMaF)

Der Konstruktor benötigt ein Objekt, welches das Interface XferAgentTaMaF implementiert. Dieses wird weiter unten kurz beschrieben.

connect(ConnectionInfo)

Es wird eine Verbindung aufgebaut mit einem Gerät, welches über die Verbindung, welche in ConnectionInfo angegeben ist, aufgebaut werden kann. Folgend ein Beispiel für eine ConnectionInfo:

```
TaMaF tamaf = new TaMaF(...);
ConnectionInfo c = new
ConnectionInfo(ConnectionInfo.CONNECTION_TYPE_NETWORK_SERVER, 12345);
Int connectedHostId = Tamaf.connect(c);
```

Hierbei wird auf Port 12345 auf eine eingehende Verbindung gewartet. Der Rückgabewert ist die verbundene Host-Id.

createNewTask()

Falls ein neuer Task erstellt wird, muss dies über createNewTask geschehen. Dies aus dem Grunde, dass intern eine ID gesetzt werden muss, welche von TaMaF selbst gesetzt wird. Falls ein Task einfach nur per „new Task()“ generiert wird, kann dieser nicht synchronisiert werden.

update(Task, List<Integer>)

Hiermit wird ein Task welcher editiert/erstellt wurde am TaMaF zur Synchronisation mitgeteilt. Die Liste ist eine Liste von HostIDs von Geräten, mit welchen dieser Task synchronisiert werden soll. Alle TaMaF bekannten HostIDs können per getHostIDs() erhalten werden.

Die Liste der HostIDs kann beim einen Mal anders sein, als bei einem anderen Mal. TaMaF merkt automatisch welche HostIDs neu hinzugekommen sind und welche seit dem letzten Mal entfernt wurden. An die gelöschten HostIDs wird dann ein delete gesendet, damit der Task von diesen entfernt wird.

Durch die Liste der HostIDs kann die Applikation selbst bestimmen, was mit wem synchronisiert werden soll. Die Applikation kann sich dafür Regeln aufstellen wie z.B. dass sie mit Gerät B nur die Tasks der Kategorie XYZ synchronisieren möchte oder dass nur alle Tasks mit hoher Priorität mit einem Gerät synchronisiert werden sollen.

xferFile(int, String, String)

TaMaF
+ connect(ConnectionInfo) : int
+ createNewTask() : Task
+ delete(Task) : void
+ deleteHostId(int) : void
+ disconnect(int) : void
+ doSync(int) : void
+ enterListenMode(int) : void
+ getHostId() : int
+ getHostIds() : List<Integer>
+ getNewUID() : long
+ leaveListenMode(int) : void
+ setHostId(int) : void
+ TaMaF(XferAgentTaMaF)
+ update(Task, List<Integer>) : void
+ xferFile(int, String, String) : void

Integer stellt die HostID des Ziel-Gerätes dar. Die beiden String sind die absoluten Pfadangaben für von und nach.

Auf einem Android Handy muss beim Zielort beachtet werden, dass nur nach /sdcard/... geschrieben werden kann und auch nur von da gelesen werden.

3.2 XferAgentTaMaF

Dieses Interface wird von der Applikation, welche auf TaMaF aufbaut implementiert um Nachrichten von TaMaF entgegenzunehmen und um TaMaF die Möglichkeit zu geben Informationen über Tasks von der Applikation zu erhalten.

onXYZ-Operationen

Werden von TaMaF jeweils aufgerufen, wenn diese Ereignisse auftreten.

newTask(Task)

Wird von TaMaF aufgerufen, um die Applikation über einen neuen Task, welcher empfangen wurde, zu informieren.

updateTask(Task)

Wird von TaMaF aufgerufen, um die Applikation über einen Task zu informieren, für welchen ein Update empfangen wurde. Der übergebene Task ist der neue Task. Der alte Task mit derselben UID ist zu ersetzen.

deleteTask(Task)

Wird von TaMaF aufgerufen, um die Applikation über einen Task zu informieren, welcher gelöscht werden soll.

getTask(long) :Task

Über diese Funktion fragt TaMaF die Applikation nach dem Task, welcher die übergebene UID hat.

getHostIds(Task): List<Integer>

Über diese Funktion fragt TaMaF die Applikation nach den HostIDs mit welchen der übergebene Task synchronisiert werden soll.

«interface» XferAgentTaMaF
+ deleteTask(Task) : void + getHostIds(Task) : List<Integer> + getTask(long) : Task + newTask(Task) : void + onConnectionLost(int) : void + onConnectionReady(int) : void + onFileTransferEnded(int, String) : void + onFileTransferStarted(int, String) : void + onListenModeEntered(int) : void + onListenModeLeft(int) : void + onSyncEnded(int) : void + onSyncStarted(int) : void + onXferStateUpdate(int, int, int, long, long, String) : void + updateTask(Task) : void

3.3 Technologien

Die verwendeten Technologien können in Kapitel 6.1 des Software Architecture Documents gefunden werden.

4 Applikation auf PC

Beim Entwickeln einer Applikation für den PC müssen ein paar Dinge beachtet werden. Diese werden folgend beschrieben.

Das „TaMaF Demo PC“-Projekt stellt ein Projekt dar, welches die Funktionalitäten des TaMaF demonstriert. Es kann als Beispiel verwendet werden um zu sehen, wie die verschiedenen Funktionen benutzt werden.

4.1 Build Path

Damit auf TaMaF zugegriffen und die Funktionen genutzt werden können müssen folgende Dateien in den Build Path des Projektes aufgenommen werden:

- android.jar
- sqllitejdbc-v056.jar
- TaMaF.jar

4.2 Verbindung über USB

Um eine Verbindung über USB zu einem Android Handy herstellen zu können, muss die „Port forwarding“-Funktionalität der ADB (Android Debug Bridge) genutzt werden. Über das Port forwarding wird ein lokaler Port auf einen Port auf dem Handy weiterleitet. So wird z.B. alles was an localhost:8080 gesendet wird auf dem Handy auf Port 8080 empfangen.

Dafür werden folgende Dateien in der Applikation benötigt:

- adb.exe
- AdbWinApi.dll
- AdbWinUsbApi.dll

Diese Dateien können im „tools“-Ordner des Android SDK gefunden werden.

Um aus der Applikation heraus das Port forwarding einzurichten kann folgender Befehl benutzt werden.

```
Runtime rt = Runtime.getRuntime();
try {
    System.out.println("Setting up adb...");
    String exec = "adb.exe forward tcp:8080 tcp:8080";
    Process p = rt.exec(exec);
    BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));
    String line;
    while((line = in.readLine())!=null){
        System.out.println(line);
    }
    p.destroy();
} catch (IOException e) {
    System.out.println("Failed to set up adb forward.");
}
```

In diesem Beispiel wird nun ein Forwarding von localhost:8080 auf das Handy unter Port 8080 eingerichtet. Dabei ist es auch möglich dies von localhost:61234 auf das Handy unter 8080 einzurichten. Dafür wäre der Befehl „adb.exe forward tcp:61234 tcp:8080“ zu verwenden.

So kann der Computer nun einen Socket auf localhost 8080 connecten lassen, während das Handy auf 8080 einen ServerSocket geöffnet hat und auf ein accept() wartet.

4.3 Android-Treiber

Damit das Android Smart-Phone auch als solches erkannt wird, müssen die Android Treiber auf dem Rechner installiert worden sein. Die Treiber sind im Android SDK im Ordner „usb_driver“ vorhanden. Diese Treiber könnten z.B. bei der Installation der PC-Applikation automatisch installiert werden.

5 Applikation auf Android

Um eine TaMaF-Applikation für Android zu entwickeln muss erst ein Android-Projekt erstellt werden. Wie dies geht kann unter <http://developer.android.com/> nachgelesen werden.

Wie die Funktionalität von TaMaF auf Android benutzt werden kann, wird im Projekt „TaMaF Demo Android“ demonstriert.

5.1 Build Path

In dem Android-Projekt muss folgende Library in den Build Path aufgenommen werden:

- TaMaF.jar

5.2 Permissions im AndroidManifest.xml

Im AndroidManifest.xml müssen der Applikation zwei Permissions zugeteilt werden. Diese sind nötig um eine Verbindung über das Netzwerk aufbauen zu können und um Dateien auf der SD-Karte zu schreiben.

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

5.3 ActiveRecordManager auf Android setzen

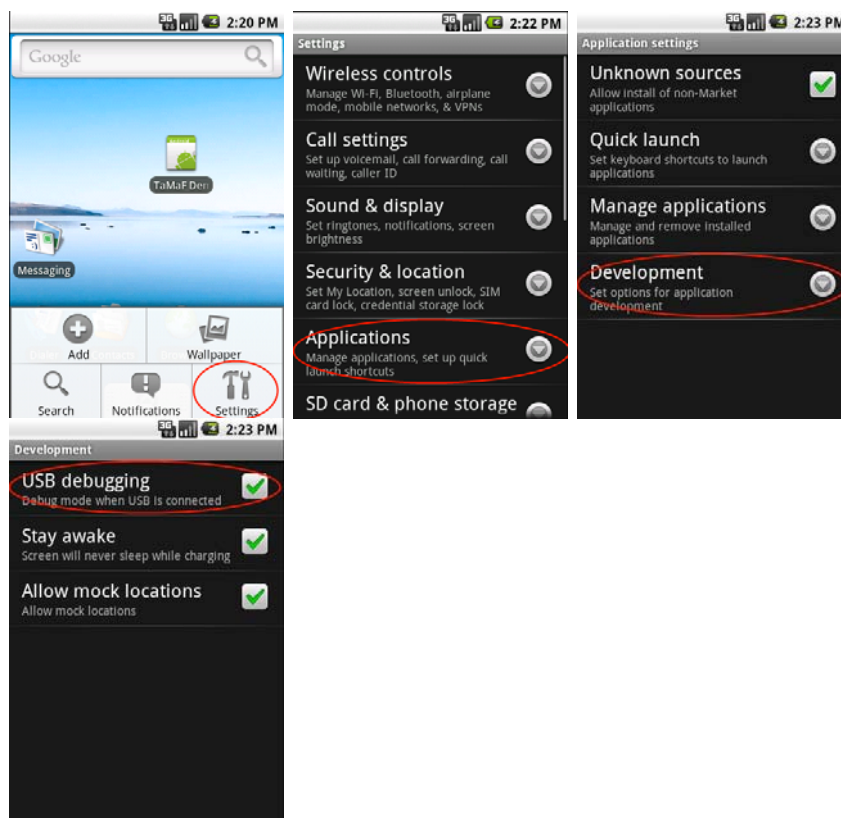
Damit die Persistence richtig läuft, muss dem ActiveRecordManager mitgeteilt werden, dass er unter Android läuft. Dies geschieht durch den folgenden Befehl in einer Activity:

```
ActiveRecordManager.setToAndroid(this);
```

„this“ ist hierbei die Activity welche ein Context ist. Denn der ActiveRecordManager muss den Context der Applikation kennen.

5.4 ADB-Verbindungen zulassen

Um die Verbindung über USB durch ADB zuzulassen, muss auf dem Handy „USB debugging“ aktiviert sein. Auf den unteren Bildern sieht man, wo diese Einstellung zu finden ist. Wenn „USB debugging“ nicht aktiviert ist, lässt sich das oben erwähnte „Port forwarding“ nicht einrichten.



5.5 Emulator

Der Emulator verhält sich gleich wie ein Android Smart-Phone, welches per USB an den Computer angeschlossen wurde. Dies heisst man kann ebenfalls per „adb forward ...“ ein Port forwarding einrichten, und dann über localhost auf den Emulator eine Verbindung herstellen.

6 Eigene nicht-Task-Objekte synchronisieren

Das unter TaMaF liegende AndroidSyncFramework erlaubt es auch eigene Objekte zu synchronisieren, welche nicht Tasks sind. Es wird einem dadurch fast uneingeschränkte Möglichkeiten geboten, was man synchronisieren möchte. Seien es Termine, Kontakte oder andere Objekte.

In diesem Kapitel wird kurz erklärt, was dafür gemacht werden muss. Viele Unterschiede gibt es dafür jedoch nicht.

6.1 AndroidSyncFramework

Da TaMaF diese Möglichkeit nicht bietet, muss auf den darunter liegenden Layer zurückgegriffen werden. Dieser bietet praktisch alle Funktionalitäten von TaMaF, wobei man hier jedoch nicht so stark eingeschränkt wird.

Aus diesem Grund werden nun alle Zugriffe nicht mehr über TaMaF getätigt, sondern über das AndroidSyncFramework. Nach der Instanzierung von AndroidSyncFramework ist es am besten, wenn nur noch über das Interface IAndroidSyncFramework darauf zugegriffen wird.

Beispiel:

```
IAndroidSyncFramework sync = new AndroidSyncFramework();  
sync.config(...); // hier wird die Konfiguration auf ein XferAgent gesetzt.  
sync.connect(...);
```

Unterschied ist hier, dass man beim Konstruktor von AndroidSyncFramework kein XferAgentTaMaF mehr angibt. Dies wird jedoch über die Funktion config(...) nach dem Aufruf des Konstruktors gemacht wie oben gezeigt. Das Interface XferAgent unterscheidet sich nur sehr geringfügig von XferAgentTaMaF, deswegen wird darauf nicht genauer eingegangen.

6.2 ISyncObject

Dies ist das Interface, welches von den Objekten implementiert werden muss, welche synchronisiert werden sollen.

Informationen zu den Methoden und zum Interface können in der Design Dokumentation im Kapitel 2.2.1.1 nachgelesen werden.

7 Demo-Projekte

Die Demo-Projekte sind als Demonstration für die Funktionalität von TaMaF bereitgestellt. Sie können auch eingesehen werden um ein Beispiel einer funktionierenden Applikation zu erhalten und zu sehen wie auf die Funktionalitäten des TaMaF zugegriffen wird.

7.1 TaMaF Demo PC

Dieses Projekt zeigt die Funktionalität von TaMaF Anhand eines Beispiels auf einem PC. Hierin kann man sehen wie die Funktionalitäten auf einem PC genutzt werden können.

Ebenfalls bietet diese Applikation die Möglichkeit eine Verbindung per USB mit dem Android Handy herzustellen.

7.2 TaMaF Demo Android

Dieses Projekt zeigt die Funktionalität von TaMaF Anhand eines Beispiels auf einem Android Handy. Hierin kann man sehen, wie die Funktionalitäten von TaMaF unter Android genutzt werden können.

Zeiterfassung

Projekt:

Task-Management-Framework on Smart-Phone

Projektmitglieder:

Patrick Boos

Markus Kolb

Betreuer:

Thomas Letsch

1 Zeiterfassung

Pakete	Geplant	Ist PB	Ist MK	Total	Differenz	Diff %
Projektmanagement	125.00 h	54.50 h	83.00 h	137.50 h	12.50 h	10.0%
Meeting	74.00 h	45.00 h	45.00 h	90.00 h	16.00 h	21.6%
Projektplan erstellen	33.00 h	5.00 h	28.00 h	33.00 h	0.00 h	0.0%
Arbeitspakete definieren und planen	4.00 h	0.50 h	3.00 h	3.50 h	-0.50 h	-12.5%
Infrastruktur Aufbau & Betrieb	14.00 h	4.00 h	7.00 h	11.00 h	-3.00 h	-21.4%
Requirements	26.00 h	6.25 h	8.00 h	14.25 h	-11.75 h	-45.2%
Anforderungen definieren	16.00 h	4.00 h	4.00 h	8.00 h	-8.00 h	-50.0%
Use Cases	10.00 h	2.25 h	4.00 h	6.25 h	-3.75 h	-37.5%
Analyse	31.00 h	6.00 h	20.00 h	26.00 h	-5.00 h	-16.1%
Domainanalyse	11.00 h	1.50 h	13.00 h	14.50 h	3.50 h	31.8%
System Sequenzdiagramme	12.00 h	4.50 h	2.00 h	6.50 h	-5.50 h	-45.8%
System Contracts	6.00 h		4.00 h	4.00 h	-2.00 h	-33.3%
Validierung Analyse gemäss Requirements	2.00 h		1.00 h	1.00 h	-1.00 h	-50.0%
Design	35.00 h	21.00 h	11.50 h	32.50 h	-2.50 h	-7.1%
Data Model	2.00 h		2.00 h	2.00 h	0.00 h	0.0%
Design Dokumentation	10.00 h	8.75 h	5.00 h	13.75 h	3.75 h	37.5%
Software Architecture Document	10.00 h	5.75 h	4.00 h	9.75 h	-0.25 h	-2.5%
Paper Prototype	3.00 h	2.50 h	0.50 h	3.00 h	0.00 h	0.0%
Developer Guide	10.00 h	4.00 h		4.00 h	-6.00 h	-60.0%
Implementation	161.00 h	113.00 h	78.50 h	191.50 h	30.50 h	18.9%
Persistence PC & Handy	30.00 h		38.50 h	38.50 h	8.50 h	28.3%
Communication	30.00 h	26.50 h		26.50 h	-3.50 h	-11.7%
TaMaF	8.00 h	5.00 h	1.00 h	6.00 h	-2.00 h	-25.0%
Android Sync Framework	60.00 h	47.00 h	26.00 h	73.00 h	13.00 h	21.7%
Demo auf Smart-Phone	25.00 h	30.50 h	2.00 h	32.50 h	7.50 h	30.0%
Demo auf Computer	8.00 h	4.00 h	11.00 h	15.00 h	7.00 h	87.5%
Test	56.00 h	31.00 h	30.00 h	61.00 h	5.00 h	8.9%
Unit Tests	30.00 h	13.50 h	13.00 h	26.50 h	-3.50 h	-11.7%
Integration Tests	2.00 h		2.00 h	2.00 h	0.00 h	0.0%
System Tests	10.00 h	0.50 h	5.00 h	5.50 h	-4.50 h	-45.0%
Bugfixing	14.00 h	17.00 h	10.00 h	27.00 h	13.00 h	92.9%
Deployment	13.00 h	13.00 h	11.00 h	24.00 h	11.00 h	84.6%
Dokumente laut Anleitung SA	9.00 h	11.00 h	10.00 h	21.00 h	12.00 h	133.3%
Enterprise Architect	4.00 h	2.00 h	1.00 h	3.00 h	-1.00 h	-25.0%

Reserve	33.00 h	6.50 h	0.00 h	6.50 h	-26.50 h	-80.3%
Total	480.00 h	251.25 h	242.00 h	486.75 h	13.25 h	2.8%