Fachhochschule Köln
Cologne University of Applied Sciences

Diplomarbeit

zur Erlangung des Diplomgrades
Diplom-Informatiker (FH)
in der Fachrichtung Allgemeine Informatik

Erstellung von Smartphone Applikationen zur Kommunikation mit Fertigungsmaschinen

Von: Georg Wolf
Matr-Nr.: 11052530
Erstprüfer: Prof. Friedbert Jochum
Zweitprüfer: ? (Poborski/Klocke)
vorgelegt: 7. Januar 2011

Zusammenfassung

Hier folgt eine kurze Zusammenfassung des Themas (ca. 5 Zeilen).

Abstract

Short description of the thesis (probably 5 lines).

Vorwort

Motivation (warum habe ich mich für dieses Thema entschieden)...

- Im Studium WPF Handyprogrammierung mit JavaME
- Interesse an Smartphoneprogrammierung
- Apps als boomendes Marketinginstrument erkannt

Danksagung

Zuerst möchte ich mich bei meinen Eltern bedanken, die mich mein ganzes Studium über unterstützt haben. Ohne Sie hätte ich es nicht geschafft überhaupt noch einmal zu studieren.

Ferner danke ich meinem Mentor Herrn Prof. Friedbert Jochum, der mich über den Zeitraum meiner Diplomarbeit betreute und mir immer wieder neue Anregungen und Aspekte aufzeigte, um meine Abschlussarbeit in die richtige Richtung zu lenken.

Nicht zu vergessen natürlich die Firma KHS GmbH, die mir die Möglichkeit gegeben hat meine Diplomarbeit in Ihrem Hause zu schreiben. Besonders zu erwähnen sind hier Herr Förster, mein direkter Ansprechpartner, sowie Herr Buchkremer und Frau Kholodenko, die immer ein offenes Ohr für mich hatten. (Fari?)

Dank gilt auch meinen geduldigen Korrekturlesern Oliver Pol und Maik “Schäpperts” E. für die investierte Mühe, Zeit und Rotstifte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

HMI - Human Machine Interaction
Visu-Station - ??? Visual Station?
Apps - ist die Abkürzung von "Applications". Gemeint sind zusätzliche Anwendungen für Smartphones
SDK - Software Developers/Development Kit
DVM - Dalvik Virtual Machine
MVC - Model-View-Controller
SSH -
Jailbreak -
IDE -
ADT -
MDA -
IRDA -
XOR -
Brute-Force -
MAC-Adresse -
OMG - Object Management Group
EAP-TLS -
EAP-TTLS -
PEAP -
LEAP -
SIMs -
RFC -
WEP -
WLAN -
WPA - Wi-Fi Protected Access
TKIP - Temporal Key Integrity Protocol
AES - Advanced Encryption Standard
MAC -
GSM -
BTS -
GPRS -

CI - Corporate Identity
GF - Geschäftsvorfall

1. Einleitung und Aufgabenstellung

- Kontext für den/die Leser herstellen
- Am besten schreiben, wenn die Arbeit im groben fertig ist

1.1. Zielsetzung und Beschreibung der Aufgabenstellung

Ziel dieser Arbeit ist es aufzuzeigen, ob die Kommunikation zwischen einer Fertigungsmaschine und einem Smartphone sinnvoll ist, und welche konkreten Vorteile dies bietet im Vergleich zu einer Feststation mit HMI.

Als Lösungsanbieter für Abfüll- und Verpackungsanlagen stellt KHS Fertigungsmaschinen für die Industrie her. Die Bedienung dieser Maschinen beim Kunden erfolgte in der Vergangenheit direkt an der Maschine über das Human Machine Interface (später als HMI bezeichnet).

Auf Grund der steigenden Verbreitung von Smartphones und der damit verbundenen Möglichkeiten soll nun im Rahmen dieser Diplomarbeit überprüft werden, inwiefern eine Kommunikation zwischen einer Smartphone Applikation und einer von KHS hergestellten Fertigungsmaschine sinnvoll sein kann. Es werden hier nicht nur die technischen Rahmenbedingungen geklärt, sondern auch wirtschaftliche Aspekte im Hinblick auf Verbreitung und Marketing aufgezeigt.

Dabei sollen verschiedene Modelle von derzeit gebräuchlichen Smartphones untersucht werden im Hinblich auf Einstiegsbeschränkungen zur Entwicklung und die Komplexität eine solche Anwendung zu erstellen. Auch Zusatzkosten spielen hier eine Rolle. Es sollen möglichst viele Smartphones bedient werden können. Dies bedeutet in der Entwicklung einen erheblichen Aufwand, da sich die einzelnen mobilen Endgeräte sehr unterscheiden. Über den Ansatz der “Model Driven Architecture” wird ein generisches Konzept für eine Applikation entwickelt und erläutert wie sich dieses im Anschluss für verschiedene Endgeräte umsetzen lässt. Mögliche Lösungsansätze für diese Probleme werden konkret aufgezeigt.

Weiterhin relevant stellen sich die Rahmenbedingungen der Smartphonehersteller dar. In dieser Arbeit wird dargelegt, welche speziellen Restriktionen sich nicht nur allgemein in

der Hardware befinden (Display, Prozessor, spezielle Funktionen der Geräte), sondern speziell auch vom Hersteller vorgegeben werden. Unter anderem die Voraussetzungen für die entsprechende Hardware entwickeln zu können. Bei den meisten Open-Source Betriebssystemen steht die Wahl der Entwicklungshardware und -umgebung relativ frei.

Ein konkretes Programm soll in Form eines Prototyps erst genau spezifiziert, dann mit Hilfe der Modellgetriebenen Softwareentwicklung (kurz MDA) ansatzweise auf möglichst viele mobile Endgeräte umsetzbar gemacht werden.

Es werden verschiedene Erwartungshaltungen formuliert im Hinblick auf die unterschiedlichen Eigenschaften der Endgeräte. Bei der Softwareerstellung muss vermehrt auf diese Restriktionen eingegangen werden. Ein kleines Display und diverse andere Eigenschaften der Smartphones müssen durch ein geeignetes Design des Userinterfaces abgedeckt werden.

Diese Diplomarbeit soll als Grundgerüst für zukünftige Entscheidungen und Entwicklungen im Bereich Apps für Smartphones dienen und einen konkreten Anhaltspunkt darüber geben, ob Entwicklung in dieser Sparte überhaupt sinnvoll erscheint.

1.2. Das Unternehmen KHS

Diese Diplomarbeit wird als Studie für die KHS GmbH erstellt.

> "KHS ist Lösungsanbieter technologisch innovativer und hochwertiger Abfüll- und Verpackungsanlagen für die Getränke- und Nahrungsmittelindustrie. Mit mehr als 5.500 Mitarbeitern weltweit realisiert der zentral von Dortmund aus geführte Konzern heute einen Jahresumsatz von fast einer Milliarde Euro. Neben Produktionsstätten in den USA, Mexiko, Brasilien, Indien und China ist KHS mit mehreren Werken in Deutschland vertreten.
>
> Das Unternehmen ist eine 100-prozentige Tochtergesellschaft der Salzgitter AG, einem der führenden Stahl- und Technologie-Konzerne Europas.
>
> Seit 2008 sind die Unternehmen KHS Corpoplast, KHS Plasmax, KHS Moldtec und KHS Asbofill in den Klöckner-Konzern integriert. Damit ist die KHS insbesondere im Bereich ganzheitlicher PET-Lösungen bestens aufgestellt. Durch neue Entwicklungen wie Trockenteil-Lösungen in Modulbauweise, universelle Füllsysteme, Hochleistungs-Etikettiertechnik oder moderne Kommunikations- und Diagnosetechniken festigt die KHS fortlaufend ihre Position als Innovator im Markt." [KHS10], letzter Abruf: 21.11.2010

2. Apps für Smartphones

2.1. Handykultur im heutigen Zeitalter

Entwickelt wurde das Mobiltelefon, um auch unterwegs und nicht nur an stationären Punkten Telefonate führen zu können. Doch bald war die Telefonfunktion nicht mehr die einzige bzw. relevanteste Funktion am Handy.

> "Zur Erfindung der SMS als "unbeabsichtigtes Nebenprodukt" des Mobiltelefonierens kam es im Jahr 1992 eher zufällig, als einige Entwickler der Firma Vodafone sich einige noch überschüssige Übertragungskapazität des Handy- Signalisierungskanals zunutze machten und über diesen eine Textnachricht mit dem Wortlaut "Merry Christmas" an das Handy eines Kollegen verschickten. Anschließend wurde der Versand von Kurzmitteilungen zunächst ausschließlich von den Netzbetreibern als Informationsdienst genutzt, um ihre Kunden über eingegangene Nachrichten auf der Sprachmailbox zu informieren, bevor der Short Message Service 1994 der Öffentlichkeit vorgestellt wurde und ab 1996 die ersten SMS-fähigen Mobiltelefone für den privaten Gebrauch den Siegeszug der SMS einläuteten." [Schmalenbach05], S. 3

Die Verbreitung des Handy nimmt weltweit immer mehr zu. Die von der ITU veröffentlichen Zahlen bewegen sich in unglaublichen Dimensionen: Von 6,1 Billionen weltweit verschickten SMS im Jahr 2010 beziehungsweise von 200.000 Kurzmitteilungen pro Sekunde ist die Rede. Das Handy wird für einen großen Teil der Weltbevölkerung immer wichtiger. Die ITU schätzt die Zahl der am Jahresende 2010 vorhandenen Handyverträge auf über 5 Milliarden. Dabei sind die Industrienationen führend in der Handynutzung. [ITU10], letzter Abruf: 30.11.2010

Mobiltelefone haben bereits einen Riesigen Markt abgedeckt und trotzdem steigt die Nachfrage nach ihnen weiter an. Sie sind für uns Telefon, Organizer und/oder Spielekonsole in einem und bieten uns viele verschiedene Funktionen an, die vor noch nicht allzu langer Zeit nur auf Desktoprechnern ausführbar waren. Im Schnitt werden Handys nach 12 bzw. 24 Monaten durch ein neues Modell ersetzt (Durchschnittliche Vertragslaufzeiten). Die Industrie überrascht uns immer wieder mit neuen Technologien und Innovationen innerhalb kürzester Zeit.

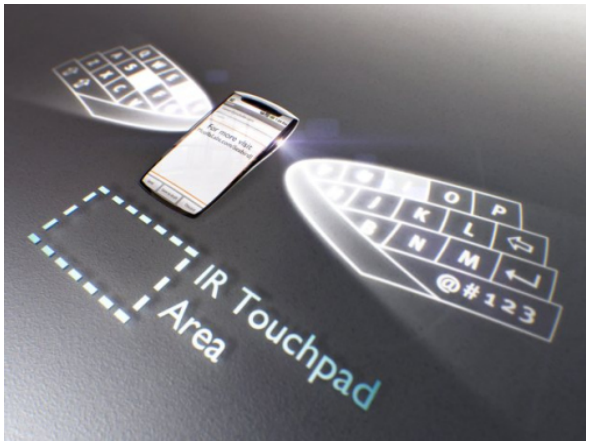

Abbildung 2.1.: Concept Phone 1

Abbildung 2.2.: Concept Phone 2

Abbildung 2.3.: Finger Whisper

Es gibt viele Konzepte von Mobiltelefonen und wie diese in der Zukunft aussehen könnten. Entwürfe mit den verschiedensten Materialien sollen Handys flexibler machen oder neue Formen von Design aufzeigen(Vgl. Abbildung 2.1 und 2.2).

Ein Interessanter Ansatz scheint es zum Beispiel zu sein, den Menschen quasi auch als "Hardware" in das Mobiltelefon zu integrieren (Vgl. Abbildung 2.3). Beim "Finger Whisper" beispielsweise erzeugt ein Terminal am Handgelenk aus Sprachinformationen Schallwellen, die über das Handgelenk weitergegeben. Diese Schallwellen können dann über den Zeigefinger ans Ohr gelangen, um so wieder zu verständlicher Sprache zu werden. [CHI99]

Die KHS möchte im Rahmen dieser neuen Technologien Anschluss finden, um moderne Konzepte in der Firma einzuführen und diese für die Produktion/Weiterentwicklung von Fertigungsmaschinen zu nutzen. Der Erfolg von modernen Mobiltelefonen und deren Verbreitung hat großes Potenzial, was die Wachstumsraten im Verkauf bestätigen. Dieser Markt soll im Rahmen der Möglichkeiten genutzt werden, um die KHS Linie entsprechend zu erweitern.

2.2. Definition App

Es sind viele Definitionen von “App” im Internet zu finden. Auch wenn das iPhone sehr viel zur Popularität von Apps beigetragen hat, hat Apple diese nicht erfunden. Nicht umsonst verzeichnet der Apple App Store, die zentrale Anlaufstelle für Anwendungen für das iPhone und den iPod Touch, bereits über sieben Milliarden Programm-Downloads. [Apple10], letzter Abruf 30.11.2010

“App” steht für Application, sprich die Anwendung und bedeutet erstmal auch nichts weiter als das. In Kombination von mobilem Endgerät (Smartphone) und einer darauf laufenden Anwendung spricht man daher vom “App”.

Der Nutzen einer solchen lässt sich wie folgt kurz beschreiben: Eine App soll dem Anwender neue Funktionen in Form von Programmen, wie z.B. Spielen oder Anwendungen auf seinem mobilen Endgerät zur Verfügung stellen. Dabei gibt es Apps, die rein auf den Spaß-Faktor abzielen, andere wiederum können dem Nutzer schnell wertvolle Informationen bereitstellen wie Ausgehtipps mit Restaurant- und Veranstaltungsinformationen, Wetteraussichten, aktuelle Wechselkurse oder die günstigste Tankstelle.

Als Marketinginstrument wurden die Apps inzwischen auch von Unternehmen gezielt eingesetzt, um ihre Marke erlebbarer zu machen und emotional aufzuladen. So gibt es beispielsweise von Pizza Hut eine App, mit dem man sich mit wenigen Klicks seine Lieblingspizza zusammenstellen und bestellen kann. Der Vorteil: Man kann von unterwegs die Pizza bestellen und so Zeit sparen. Mit im App enthalten ist zudem der Pizza Hut Racer, ein Autorennen-Spiel - der beim (potentiellen) Kunden für jede Menge Spaß sorgen soll. So wird der Kunde an die Marke gebunden.

2.3. Funktionsumfang und Nutzen seitens der KHS

2.3.1. Zielsetzung

Nachfolgend werden spezielle Ziele, die die KHS GmbH mit der Erstellung einer Smartphone App erreichen möchete, vorgestellt.

2.3.2. Wirtschaftliche/Marketingtechnische Erwägungen

Dank der intuitiven Bedienkonzepte von Smartphones ist es möglich sehr benutzerfreundliche Umbegungen zu schaffen. Jeder, der bereits ein Smartphone bedient hat kennt die auf Touchscreens eingesetzten Techniken, um Symbole zu manipulieren. Somit werden intuitiv Oberflächen, die sich an diese Richtlinien der Smartphones halten, vom Benutzer

richtig erfasst und angenommen. Eine längere Schulungsmaßname, um mit der Bedienung arbeiten zu können entfällt logischerweise. [Clark10], S. 11ff

Ebenfalls für die Entwicklung auf mobilen Endgeräten spricht, dass viele Mitarbeiter von Firmen bzw. Kunden schon ein Smartphone besitzen. Somit entstehen keine bzw. nur geringe Zusatzkosten für die anzuschaffende Applikation beim Kunden. Dies macht den BLA unter Kostengesichtspunkten sehr attraktiv, da die Hardware für das Produkt schon vorhanden ist.

Durch die Unabhängigkeit zum Maschinenstandort wird die Flexibilität mit dem Umgang der Maschine erhöht. Beispielsweise sollen verschiedene Basisfunktionen auch über die Smartphone App zur Verfügung gestellt werden. Der Mitarbeiter muss also nicht direkt bei der Maschine sein, um diese zu bedienen oder zu warten. Dies verschafft dem Kunden einen Zeitvorteil, der eine Kostenersparnis nach sich zieht, weil Abläufe schneller erledigt werden können.

Eventuell lässt sich auch die Kommunikation der Mitarbeiter untereinander durch Smartphone Applikationen verbessern. Vorstellbar wäre hier, dass Nachrichten bzw. Protokolle über die App ausgetauscht werden können oder auch direkt die Nummer der zuständigen Hotline anwählbar ist, falls eine Rückmeldung der Maschine das erfordert.

Durch die Unterstützung von neuen Technologien, die dem Kunden Zeit- bzw. Kostenvorteile bieten werden von KHS hergestellte Maschinen deutschlich attraktiver als Maschinen von Mitbewerbern, die nicht über die gleiche Funktionalität verfügen.

Solche vergleichsweise einfach zu erstellenden Applikationen mit wenigen Grundfunktionen pro App (evtl. viele Verschiedene; hohe Skalierung) können beim Kunden neue Wünsche wecken. Die Möglichkeiten sind hier sehr weitreichend. Deshalb könnte es durch die Akzeptanz beim Kunden und die rasche Verbreitung von anfänglichen Apps zu einer neuen Produktlinie bei KHS führen, wenn die Nachfrage groß genug ist.

Corporate Identity (kurz CI) stellt den abgestimmten Einsatz von Verhalten, Kommunikation und Erscheinungsbild nach innen und außen dar. Basis ist das Unternehmensleitbild, welches mit Leben gefüllt wird. Die CI ist also die Persönlichkeit einer Organisation. Smartphone Applikationen haben mittlerweile einen relativ hohen Stellenwert in der modernen Gesellschaft erreicht. Mit einem gelungenen Einstieg in diesen Bereich wäre eine positive Außenreaktion und dadurch eine Ausweitung des Leitbildes der KHS möglich im Sinne eine Imagegewinns in der Öffentlichkeit.

2.3.3. Funktionsumfang

Konkrete Funktionen, die für Apps denkbar sind sehen wie folgt aus:

- Teaching: Bei der Inbetriebnahme einer Fertigungsmaschine soll es möglich sein ein Video zu streamen, welches erklärt wie die Maschine eingerichtet wird. Weiterhin sollen unterstützende Videos dem Nutzer durch Abruf ermöglichen, bei einfachen Fehlermeldungen der Maschine diese durch Anleitung zu beheben. Dazu werden kurze Videoausschnitte gezeigt, die die einzelnen Schritte für den Benutzer verständlich erklären. Durch Betätigung eines Buttons wird von Schritt zu Schritt gewechselt.
- Messaging: Es soll dem Nutzer der Applikation ermöglicht werden einfache Meldungen und/oder Notizen an die Visu-Station zu schicken bzw. Meldungen zu empfangen.
- Push Notification: Verschiedene Zustände der Maschine sollen dem Nutzer als Nachricht übermittelt werden können. Diese werden allerdings nicht vom Nutzer abgerufen sondern von der Maschine auf das Gerät "gepusht". Dabei sind Fehlermeldungen genauso denkbar wie andere Nachrichten, z.B. wenn eine Maschine einen vorher festgelegten Richtwert überschreitet.
- Monitoring: Die Produktion und der Durchlauf der Maschine soll am Gerät live angezeigt werden. Mögliche Datensätze sind hier Auslastung der Maschine, umbauzeiten, Dauer von Wartungsarbeiten, etc.
- Reporting: Aktuelle Statistiken der Produktionsdaten in Form von Reports sollen vom Nutzer abrufbar sein. So können Echtzeitdaten beispielsweise in Präsentationen oder bei Besprechungen verwendet werden (Anzeige auf dem Smartphone oder Versand per Email auf Anforderung denkbar).
- Control: Es soll durch den Benutzer möglich sein unkritische Funktionen der Maschine zu steuern (mögliche Fehler durch den Benutzer in der Bedienung müssen ausgeschlossen werden).
- Bug-Reporting: Es soll möglich sein Maschinenfehler an den Server zu schicken, um diese dem Support zur Verfügung stellen zu können. Dies soll durch eine manuelle Eingabe in einem Fehlerberichtsformular möglich sein. Zusätzlich denkbar ist es, dass mit der Kamera am Smartphone ein Foto des Defekts mit dem Bericht mitgesandt wird.
- View: Einzelaggregate mit Ihren verschiedenen Stati sollen abfragbar sein. Dabei handelt es sich um Aggregate der Maschine, die normalerweise nicht am Maschinendisplay auftauchen und für spezielle Operationen benötigt werden (Beispielsweise bei Inbetriebnahme oder Störungsbeseitigung).
- Spare Parts Catalogue: In einem über Smartphone aufrufbaren Ersatzteilkatalog soll es möglich sein, Teile der verwendeten Maschine nachzubestellen. Bei einem Fehler wird von der Maschine überprüft, ob ein Maschinenteil defekt ist. Falls dies zutrifft,

wird dem Benutzer vorgeschlagen, dieses nachzubestellen.

3. Grundlagen für die folgenden Kapitel

Für das Verständnis der nachfolgenden Kapitel werden kurz wichtige Verfahren benannt und Techniken beschrieben, die Anwendung finden. Weiterhin werden Hinweise zu weiterführender Literatur gegeben.

3.1. MDA

In der Vergangenheit der Softwareentwicklung zu beobachten war, dass wenn zu erstellende Software-Systeme auf ein nicht mehr zu beherrschendes Maß angewachsen waren, die Lösung darin bestand auf eine höhere semantische Ebene zu gehen (Abstraktionssprung der Programmiersprache). Ausgehend von der Tatsache, dass Maschinensprachen über Assembler, Prozedurale Sprachen durch Objektorientierte Sprachen abgelöst wurden lässt sich schlussfolgern, dass die Zukunft des Software-Engineerings in den Modellierungssprachen liegt. [Gruhn06], S. 14ff

3.1.1. Begriffsklärung

> "MDA ist ein Standard der Object Management Group (kurz OMG). Die OMG selbst wurde 1989 gegründet und ist ein offenes Konsortium aus Firmen weltweit. Die OMG erstellt herstellerneutrale Spezifikationen zur Verbesserung der Interoperatibilität und Portierbarkeit von Softwaresystemen und ist traditionell eine Plattform für Middleware- und Tool-Hersteller zur Synchronisation und Standardisierung ihrer Betätigungsfelder." [Stahl07], Seite 377

3.1.2. Arbeitsweise

MDA arbeitet mit mehreren Schichten:

- Computation Independent Model (CIM)
  Das CIM liefert eine Sicht auf das Gesamtsystem unabhängig davon, wie es implementiert werden soll. Es enthält die Anforderungen des Systems an die Umwelt.

- Platform Independent Model (PIM)
  Das PIM beschreibt die formale Struktur und die Funktionalität des Systems.
- Platform Specific Model (PSM)
  Durch Anreicherung des PIM mit plattform-abhängigen Informationen entsteht das PSM.
- Code
  Aus dem PSM wird der Quellcode für die Zielplattform generiert. Meist entsteht dabei noch kein ausführbarer Code, sondern lediglich ein Grundgerüst, das für die weitere händische Implementierung genutzt wird.

Transformationen bilden die Grundlage für die Überführung von einer Schicht in eine andere. Dabei unterscheidet man zwischen:

- Modell-zu-Modell Transformation
- Modell-zu-Code Transformation

Kernidee der MDA ist es also vom unabhängigen Modell über Transformationen bis hin zum eigentlichen Programmcode zu gelangen.

Ziele der MDA:

- Konservierung der Fachlichkeit
- Portierbarkeit
- Systemintegration und Interoperabilität
- Effiziente Softwareentwicklung
- Domänen-Orientierung

[Gruhn06], S. 21 ff

Mehr Informationen zur Model Driven Architecture findet man auf der Webseite der OMG[1] oder als Literatur [Gruhn06], in diesen detailiert auf die Softwareerstellung mit Hilfe von MDA (auch an Beispielen) eingegangen wird.

3.2. ISO/OSI-Modell

OSI Model Data unit Layer Function Host layers Data 7. Application Network process to application 6. Presentation Data representation, encryption and decryption, convert machine dependent data to machine independent data 5. Session Interhost communication

[1] http://www.omg.org

Tabelle 3.1.: OSI-Modell

OSI Model		
Data Unit(Einheit)	Layer	Funktion

Segments 4. Transport End-to-end connections and reliability,flow control Media layers Packet 3. Network Path determination and logical addressing Frame 2. Data Link Physical addressing Bit 1. Physical Media, signal and binary transmission

3.2.1. Layer

3.2.2. TCP/IP Header

4. Systemspezifikation

Das folgende Kapitel beschäftigt sich mit der Systemspezifikation für die zu entwicklende Smartphone App. Dabei wird ein Konzept erstellt, dass als Grundlage für den modellgetriebenen Ansatz dienen soll.

Da wir uns jetzt bei der konkreten Umsetzung der Software befinden ist es an der Zeit dem Projekt einen Namen zu geben.

```
YASA - Yet Another Smartphone Applicaction
```

Nachfolgend im Text wird mit `YASA` bezug auf den Prototypen genommen.

Ausserdem werden die verschiedenen Tools ausgewählt, die zur Modellierung von `YASA` dienen sollen. Dabei wird spezzielles Augenmerkt darauf gelegt, dass die grafischen Werkzeuge saubere XML Dateien generieren, die später für die (eventuell Model-zu-Model und)Model-zu-Code Transformation benutzt werden. Beispielhaft wird versucht aus den verschiedenen Diagrammarten (Klassendiagramm, Aktivitätendiagramm) Textbausteine zu generieren, um zu überprüfen, welche Programme überhaupt zur Generierung von Transformationen geeignet sind und welche nicht alle Diagrammtypischen Merkmale beherrschen. Eventuell wird versucht bestehende Programme zu erweitern im Hinblick auf die volle Unterstützung der verwendeten Diagrammtypen.

4.1. Geschäftsvorfall für die umzusetzende App

Ein realistischer Geschäftsvorfall (kurz GF) soll als Grundlage für die Umsetzung von `YASA` dienen.

4.1.1. Vorgaben

Es soll gezeigt werden, dass mit MDA die Umsetzung des GF technisch möglich ist. Ebenso sollen die Grenzen der Technologie erforscht werden. Dabei soll gezeigt werden, dass folgende Artefakte aus dem Modell generiert werden können:

- Persistenz

- Fachliche Logik
- Benutzer Frontend

4.1.2. Beschreibung des Geschäftsvorfalls

Dafür wurde eine spezielle Funktion aus Kapitel 2 ausgewählt, die nun umgesetzt wird:

> "Spare Parts Catalogue: In einem über Smartphone aufrufbaren Ersatzteilkatalog soll es möglich sein, Teile der verwendeten Maschine nachzubestellen. Bei einem Fehler wird von der Maschine überprüft, ob ein Maschinenteil defekt ist. Falls dies zutrifft, wird dem Benutzer vorgeschlagen, dieses nachzubestellen."

4.2. Anwendung auf die Anforderungsdefinition

Nach der Kurzbeschreibung des Anwendungsfalls folgt als nächstes die Dokumentierung der Einzelheiten. Zuerst folgt eine textuelle Beschreibung, die als Grundlage zur Erstellung des CIM dient.

Der Prototyp "Spare Parts Catalogue" soll folgende Funktionen unterstützen:

- Ersatzteilkatalog durchsuchen
- Ersatzteil bestellen
- Bestellung stornieren
- Bestellstatus abfragen
- Maschinenfehler melden

Anhand dieser Beschreibung kann ein Use-Case-Diagramm erstellt werden mit den unterstützten Funktionen seitens des Nutzers und des Systems (Abbildung 4.1).

4.2.1. Erstellen eines Computation Independent Models

Die Funktionen von YASA lassen sich im Detail wie folgt beschreiben:

Funktion	Ersatzteilkatalog durchsuchen
Kurzbeschreibung	Ein Kunde möchte den Ersatzteilkatalog durchblättern bzw. durchsuchen
Akteur	Benutzer
Vorbedingungen	Kundendaten bekannt

Teilhandlungen	Suchbegriff eingeben Ersatzteile blättern Ersatzteil auswählen (Detail ansehen)
Nachbedingungen	-
Funktion	Ersatzteil bestellen
Kurzbeschreibung	Ein Kunde möchte ein Ersatzteil bestellen, dessen Bezeichnung und Funktion ihm bereits bekannt ist
Akteur	Benutzer
Vorbedingungen	Ersatzteilkatalog durchsuchen/ Maschinenfehler melden
Teilhandlungen	Bestellvorgang einleiten Bestelldetails anzeigen Bestellung abschicken
Nachbedingungen	Bestellung wurde erfolgreich übermittelt
Funktion	Bestellung stornieren
Kurzbeschreibung	Ein Kunde möchte eine bereits ausgeführte Bestellung stornieren
Akteur	Benutzer
Vorbedingungen	Ersatzteil wurde bestellt
Teilhandlungen	Bestelldetails anzeigen Bestellung stornieren
Nachbedingung bei Erfolg	Bestellung wurde erfolgreich storniert
Nachbedingung bei Misserfolg	Bestellung konnte nicht storniert werden (evtl. wurde Bestellung schon versandt)
Funktion	Bestellstatus abfragen
Kurzbeschreibung	Ein Kunde möchte den Status seiner Bestellung abfragen (in Bearbeitung, versandt, nicht lieferbar, etc.)
Akteur	Benutzer
Vorbedingungen	Ersatzteil wurde bestellt
Teilhandlungen	Bestelldetails anzeigen Bestellstatus anzeigen
Nachbedingungen	-
Funktion	Maschinenfehler melden
Kurzbeschreibung	Die Fertigungsmaschine hat einen Fehler festgestellt und kann nicht fortfahren. Eine Systemanalyse stellt fest, dass ein Teil der Maschine kaputt ist und sendet dem Benutzer eine Fehlermeldung
Akteur	System
Vorbedingungen	Fehler im System

	Fehler kann ermittelt werden
Teilhandlungen	Systemmeldung mit Fehler erstellen passendes Ersatzteil ermitteln Systemmeldung an Benutzer senden
Nachbedingungen	Nutzer wurde über Systemfehler informiert und kann Ersatzteil bestellen

Als nächstes werden nun diese textuellen Beschreibungen möglichst genau in Diagramme umgewandelt. Zur Erfassung dieser auf grober Detailstufe wird als Darstellung das Aktivitätsdiagramm verwendet.

4.2.2. PIM

Um die dargestellten Funktionen weiter umzusetzen und ein Platform Independent Model zu generieren wird nun die Plattformunabhängige Basisarchitektur festgelegt.

Diese lässt sich wie folgt darstellen:

Tabelle 4.2.: Architektur von YASA-Anwendungen

Präsentation / View
Dialogfluss
Geschäftslogik/Services
Persistenz

Als oberste Schicht im Diagramm 4.2 finden wir die Präsentationsschicht. Diese ist zuständig für die Darstellung des Inhalts auf dem jeweiligen Smartphone in einer für den Benutzer verständliche Art und Weise.

Darunter befindet sich der Dialogfluss. Hier werden Kontrollflüsse der Anwendung definiert.

Sollen Daten im Rahmen der Geschäftslogik dauerhaft gespeichert und abrufbar sein, kümmert sich die Persistenzschicht darum.

Die festgelegten Schichten können durch folgende UML-Profile modelliert werden:

- Präsentation (View): Metamodell zur Modellierung der auf dem Smartphone darzustellenden Oberfläche in Form von Dialogmasken. Interaktionsmöglichkeiten des Nutzers mit der Oberfläche und den enthaltenen Schaltflächen wird hier festgelegt.
- Dialogfluss (Pageflow Profile): Profil zur Modellierung von verzweigten Dialogflüssen

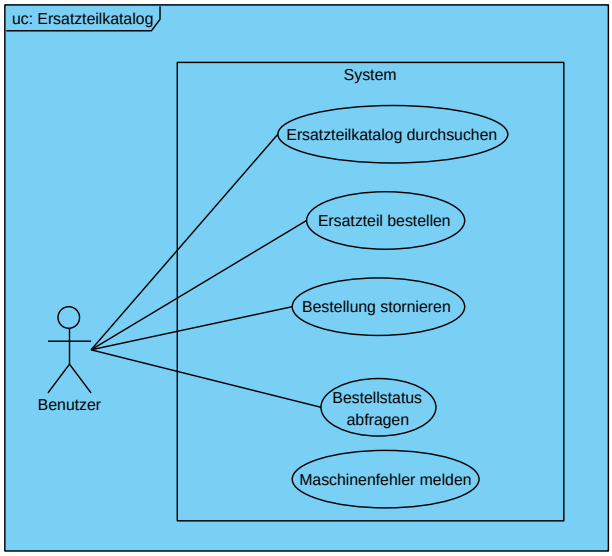

Abbildung 4.1.: Geschäftsanwendungsfälle Use-Case-Diagramm

- Persistenz (Persistence Profile): Modellierung des persistenten Charakters von Entitätsklassen

4.2.2.1. Modellierung PIM

Als Basisdiagramm soll uns das Aktivitätsdiagramm dienen, welches durch weitere Elemente erweitert wurde. Zum einen wäre dies das Metamodell View, das die Konstrukte zur Modellierung der Dialogmasken enthält (siehe Abbildung 4.3).

Als nächstes zu definieren ist das View Metamodell. Hier wird festgelegt, welche Elemente zur Dartellung verwendet werden und wie der Benutzer die dargestellten Elemente manipulieren darf (Abbildung 4.4). Basisbaustein ist der `View`. Ihm werden beliebig viele View-Elemente zugeordnet. Weiterhin im Modell enthalten sind `OutputElemente` für die übergebenden Modelldaten in grafischer Form und den `InputElementen`, die Werte des Pageflow-Kontextes durch Benutzereingaben aktualisieren. Daneben gibt es `CommandElemente`, an die ein `Outcome` gehängt werden kann(Kontrollflusssteuerung). Die Mitgeschickten Nutzdaten werden durch ein Element des Typs `ContextContainer` angehangen.

Nach der Definition der verschiedenen Modelle lassen sich diese auf unsere Spezifikation anwenden. Im konkreten Fall “Ersatzteilkatalog durchsuchen” können wir den in Abbil-

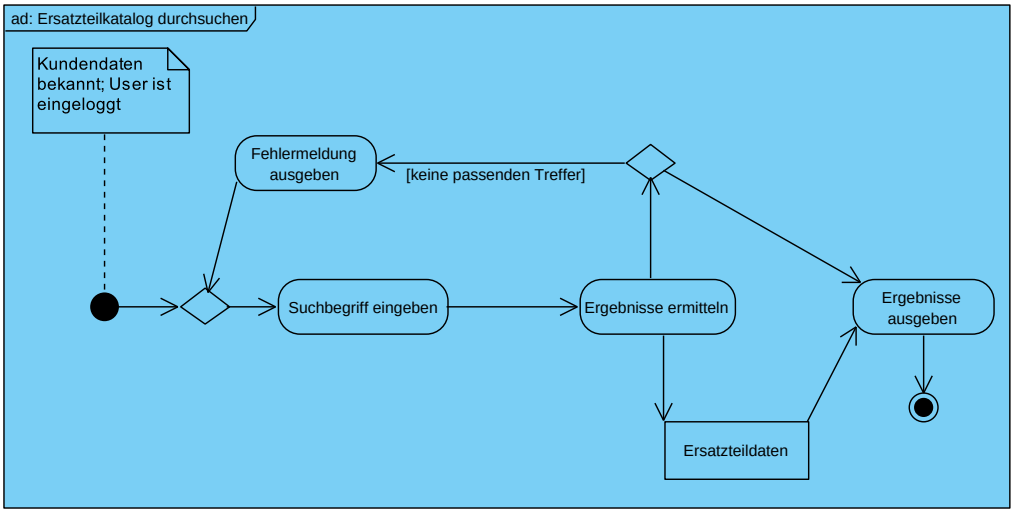

Abbildung 4.2.: Aktivitätsdiagramm Ersatzteilkatalog durchsuchen

dung 4.5 dargestellten Dialogfluss generieren.

TODO: Erklärung Fluss

Zur Beschreibung des Dialogflusses gehören ausserdem weitere Details, um die gestellten Anforderungen vollständig abzubilden. (evtl. Verweis Entwurfsskizze). So muss jeder von einem `ViewCall` aufgerufene `View` weiter detailliert und mit dem entsprechenden Datenmodell verbunden werden (Abbildung 4.6).

4.2.3. Vorhandene Modellierungs-/Transformationstools auf dem Markt

Verzichtet werden soll auf die Verwendung kommerziell-, proprietärer Software zugunsten offener Standards und frei verfügbarer Open-Source Technologien. Dabei sind auch Überlegungen zu den unterschiedlichen Lizenzen anzustellen.

Deswegen werden im Folgenden erst grafische Tools zur Modellierung der Diagramme untersucht und anschließend Transformationstools, die für die Überführung der einzelnen Modelle und schliesslich für den entstehenden Quellcode verantwortlich sind.

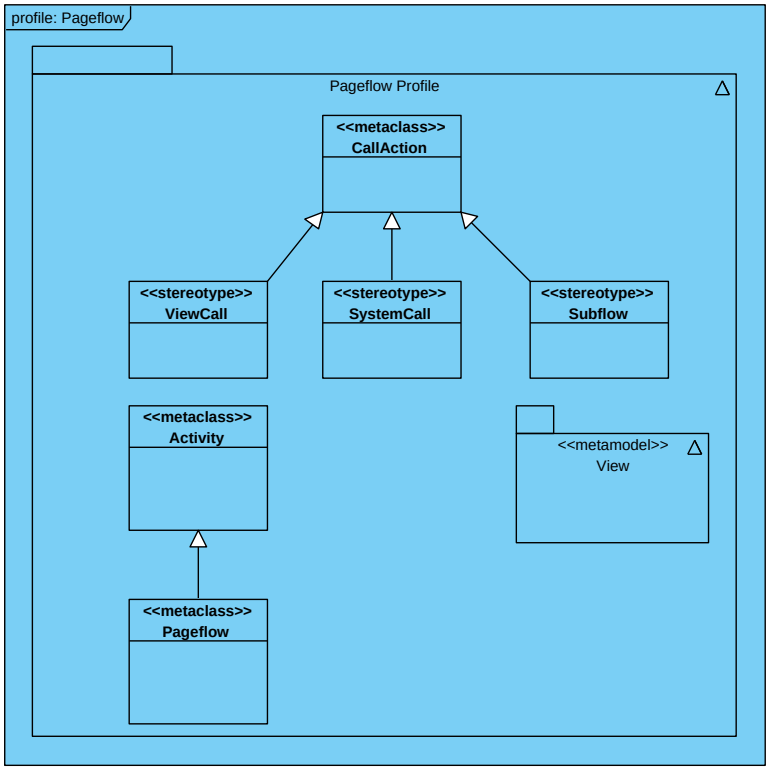

Abbildung 4.3.: Definition von Pageflow Profile

4.2.3.1. Visual Paradigm

Visual Paradigm ist sehr leicht und intuitiv zu erlernen. Die meisten Funktionen findet man auf anhieb, ohne auch nur einen Blick in die Dokumentation werfen zu müssen. Allerdings macht das arbeiten mit Visual Paradigm in Kombination mit einem Transformationstool nicht viel Sinn, da beim Export der Diagramme in XML Files Informationen verloren gehen. Beim Import in Acceleo beispielsweise fehlen die Namen bzw. Bezeichnungen der Diagramme so wie Zusammenhänge der einzelnen Diagrammarten, die jedoch vorher definiert wurden. Damit ist Visual Paradigm für den Zweck der Codegenerierung relativ unbrauchbar.

4.2.3.2. MagicDraw

Ein erster Versuch ein einfaches Klassendiagramm zu implementieren und in Code umzuwandeln klappte erstaunlicherweise recht gut. Allerdings ist die Bedienung von MagicDraw wesentlich komplexer und unübersichtlicher als bei Visual Paradigm. Die Einarbeitungszeit ist bei dieser Komplexität exponentiell größer. Daher verzögerte sich die Erstellung der Diagramme sehr. Was in Visual Paradigm in Stunden machbar war, dauerte hier einen ganzen Tag. Schlechte Navigation, unübersichtliche Menüs und unzureichende Dokumen-

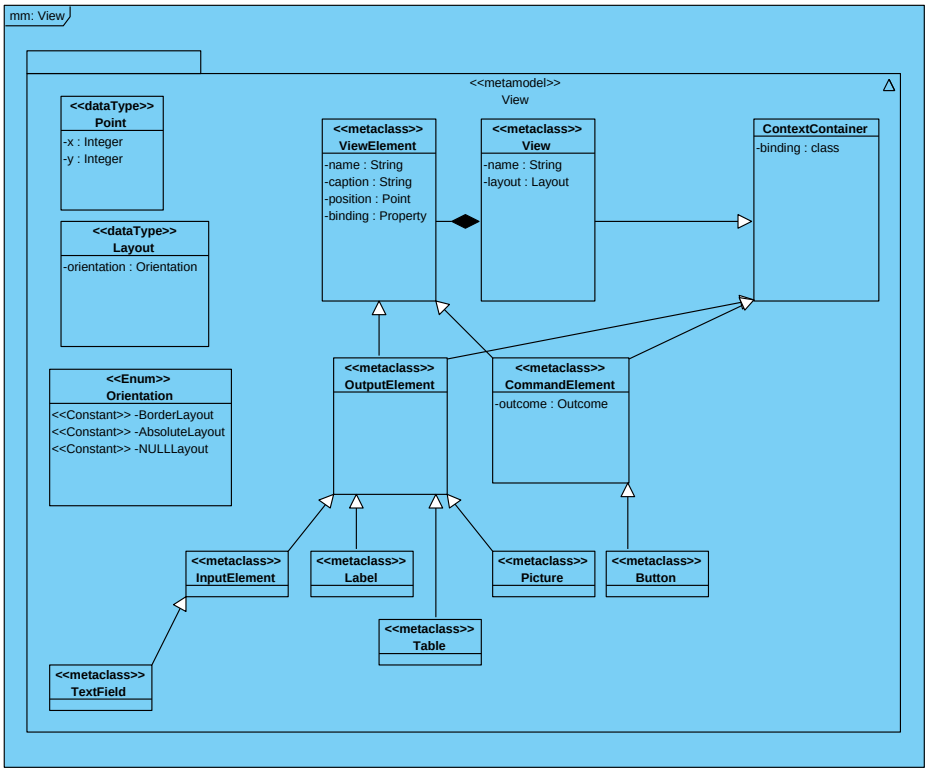

Abbildung 4.4.: Definition des Metamodells View

tation zur Software, aber dafür volle Kompatibilität zum XML Standard.

4.2.3.3. Acceleo

Einfache Klassendiagramme in Code umzuwandeln ist mit dem Tool kein Problem. Allerdings ist die Dokumentation in Bezug auf Diagrammarten ziemlich unvollständig. Das Durchlaufen von Aktivitätsdiagrammen ist prinzipiell anhand der Activities möglich. Allerdings konnte ich nicht herausfinden, wie man diese in der richtigen Reihenfolge darstellt. Acceleo wirft die Ergebnisse einfach so durcheinander, in der Reihenfolge wie sie im XML File des UML's generiert wurden. Aus einem Kontrollfluss heraus etwas zu erzeugen stellt sich also als ziemlich unmöglich dar.

4.2.3.4. oAW

4.2.4. Wahl der zu benutzenden Tools

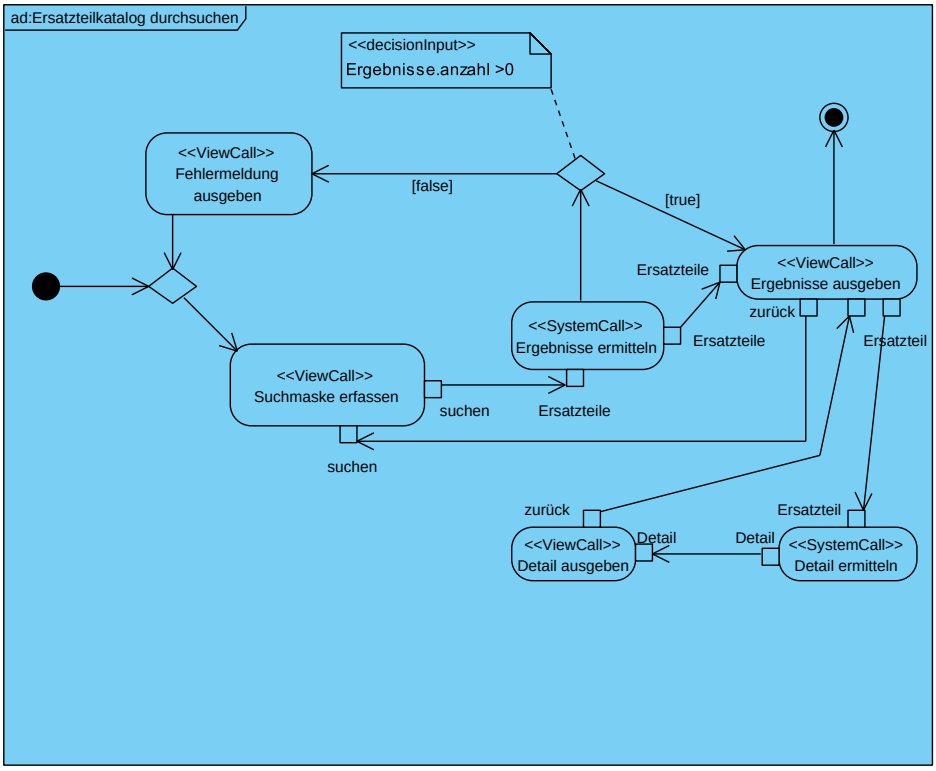

Abbildung 4.5.: Dialogfluss “Ersatzteilkatalog durchsuchen”

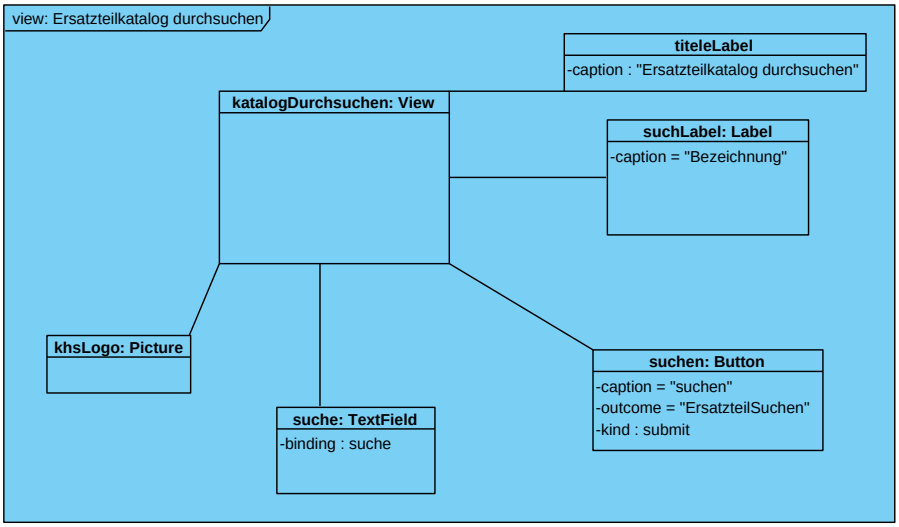

Abbildung 4.6.: Anwendung des View-Metamodells

5. Sicherheit in mobilen Netzen

5.1. Sicherheitsprobleme in mobilen LANs und WANs

> "Funknetze haben im Gegensatz zu leitungsgebunen Netzen zusätzliche Gefährdungspunkte, die zumeist aus den verwendeten Übertragungsprotokollen und der nur begrenzt kontrollierbaren Ausbreitung der Funkwellen ergeben." [Eren06], S. 259

5.1.1. Bluetooth

Bluetooth ermöglicht die drahtlose Datenübertragung und lässt sich dabei leichter konfigurieren als Beispielsweise WLAN. Es wird dabei kein direkter Sichtkontakt zwischen den beteiligen Geräten benötigt (vgl. IRDA).

In der Spezifikation von Bluetooth wird kein Verschlüsselungsalgorithmus vorgeschrieben, deswegen ist in der Standardkonfiguration vieler Hersteller die Verschlüsselung ausgeschaltet. Aber auch wenn diese aktiviert sein sollte, heisst dies nicht unbedingt, dass die Verbindung sicher ist. Der Verschlüsselungsalgorithmus bei Bluetooth baut auf einer XOR-Verknüpfung von Klartext und Schlüsselstrom auf. Ein Angreifer kann Teile der versendeten Nachricht herausfinden, weil der zur Kommunikation benutzte TCP/IP-Header eine bekannte Form hat (hinsichtlich Größe und Aufbau). Als Folge sinken mögliche Schlüsselkombinationen (Reduktion der effektiven Schlüssellänge von 128 Bit auf 84 Bit wegen des bekannten Headers). "Brute-Force"-Angriffe werden somit erleichtert.

Ein anderer Ansatzpunkt für Angreifer stellt der Zufallszahlengenerator dar. Zufallszahlen werden in einigen Sicherheitsfunktionen verwendet. In der Spezifikation des Bluetooth-Standards werden keine expliziten Anforderungen an diesen gestellt. Da verschiedene Hersteller wirtschaftlich günstige Algorithmen wählen, können teilweise durch die Bekanntheit dieser Algorithmen die Ergebnisse vorhergesagt werden.

Der bei einer Authentifizierung notwendige vierstellige Pin-Code stellt auch bei einer 128-Bit Verschlüsselung leider immer noch keine ausreichende Sicherheit zur Verfügung. Im Auslieferungszustand des Geräts ist dieser meist auf "0000" gesetzt. Wenn der Pin-Code vom Angreifer also richtig ermittelt werden kann, hat er die Möglichkeit die Kommunika-

tion von gepairten Geräten ungestört zu belauschen.

Alle Sicherheitsdienste bei Bluetooth sind in der Datenübertragungsschicht (Schicht 2 des ISO-OSI-Schichtenmodells) angesiedelt. Somit gibt es keine Ende-zu-Ende Sicherheit. Die Verbindung wird zwar verschlüsselt, aber es fehlt eine nahtlose und durchgehende Datenverschlüsselung zwischen den Endgeräten.

Das OBEX-Protokoll wird vor allem bei Bluetooth-Push-Diensten eingesetzt. Auch dieses hat Schwächen. Der Stack wird vom Zulieferer und nicht vom Hersteller implementiert. Somit kann es passieren, dass Bluetooh-Geräte, die den gleichen Stack und das gleiche virtuelle Dateisystem besitzen unauthorisierte Zugriffe aufs Filesystem (z.B. Telefonbuch) unbeabsichtigt offen legen.

5.1.2. GSM

GSM wurde auf Basis der bereits bestehenden Festnetztechnik entwickelt. Die Verschlüsselung der Daten wurde nur als Option definiert, da in einigen Ländern eine Verschlüsselung von staatlicher Seite untersagt ist. Auch wird auf den meisten Endgeräten nicht angezeigt, ob eine Verbindung verschlüsselt ist oder nicht. Ein weiterer Schwachpunt stellt der Umstand dar, dass Daten nur zwischen der BTS und dem Teilnehmer verschlüsselt werden. Das abhören von diesen Verbindungen ist an den diversen Schnittstellen der BTS zu anderen Netzkomponenten ohne großen Aufwand möglich. Der potentielle Angreifer verschafft sich physischen Zugang zu einem BTS und kann von dort die Kommunikation belauschen bzw. manipulieren.

Ebenfalls eine Schwachstelle ist der Kurznachrichtendienst SMS. Hier werden Nachrichten unverschlüsselt über den Signalisierungskanal übertragen. Anhand der Übertragungsfrequenz kann so ein Angreifer Nachrichten mitlesen.

5.1.3. GPRS

GPRS stellt eine Erweiterung des GSM-Netzes um paketorientierte Datenübertragung dar. Der Einstieg über das GPRS-Netz ins Internet wird als G_i-Schnittstelle bezeichnet. Während der Datenkommunikation über GPRS ist der Teilnehmer anfällig für im Netz verbreitete Gefahren und Angriffe. Falls der Datenverkehr unverschlüsselt stattfindet besteht die Gefahr, dass Daten durch einen Angreifer abgefangen und missbraucht werden. Häufig rechnen Mobilfunkbetreiber mit solchen Gefahren und platzieren eine Firewall an der G_i-Schnittstelle.

5.1.4. UMTS

UMTS stellt eine Erweiterung auf Basis der verbreiteten Technik dar. Damit gewährleistet ist die Abwärtskompatibilität zu GPRS/GSM.

5.1.5. HSDPA

5.1.6. LTE

Neue Technik!!! noch nicht in Deutschland verfügbar

5.1.7. Wireless LAN

Die Mit der technischen Spezifikation verbundene Systemoffenheit zuverlässige und bequeme Einstiegspunkte ins ein Netzwerk zur Verfügung zu stellen enblößt leider auch abgeschottete private Netzwerke. Im Folgenden werden die verschiedenen Methoden der Verschlüsselung für Wireless LAN (kurz WLAN) erläutert und konkret auf deren Schwachstellen eingegangen.

5.1.7.1. WEP

Das WEP-Protokoll verwendet zur Absicherung der Verbindung den symmetrischen Algoritmus RC4.

> "Dieser ist ein Stromverschlüsselungsalgorithmus, der den Klartext über eine XOR-Operation mit einer Folge von Pseudozufallszahlen verknüpft." [Eren06], S. 287

Der WEP-Key wird entweder mit 40 Bit (WEP 64) oder 104 Bit (WEP 128) erzeugt.

Schlechte Implementierungen des Initialisierungsvektors (der zusammen mit dem WEP-Key dazu benutzt wird die Übertragung zu verschlüsseln) führen zu Kollisionen in den übertragenen Daten. Angreifer können diese Kollisionen erkennen und dadurch auf die unverschlüsselte Nachricht schliessen.

Bei WEP fehlt eine gegenseitige Authentisierung. Aufgrund der Einfachheit, eine Netzwerkkomponente (insbesondere MAC-Adresse) zu fälschen entsteht damit eine wesentliche Sicherheitslücke.

Nach RFC 1024 müssen alle IP- und ARP-Pakete stest mit einem "0xAA" beginnen. Dies kann von einem Angreifer ausgenutzt werden, um den WEP-Schlüssel zu ermitteln. Mit

relativ wenig Aufwand wird genug Chiffretext gesammelt. Um den Schlüssel ermitteln zu können, müssen die ersten Bystes des Klartextes bekannt sein. Aufgrund der Anforderung nach RFC haben Angreifer also leichtes Spiel.

5.1.7.2. WPA

Der grundlegende Unterschied zu WEP ist die Verwendung von TKIP als Verschlüsselungsprotokoll. Prinzipiell lässt sich WEP mittels Software- und Firmware-Updates auf WPA upgraden.

Um die Schwachstellen von WEP auszumerzen implementiert WPA dynamische Schlüssel für jedes versendete Paket. Dabei besitzt jeder Benutzer einen eigenen Schlüssel.

WPA ist anfällig für Wörterbuchattacken, da der Preshared Master Key direkt aus der Passphrase und der SSID abgeleitet wird. Für einen solchen Angriff reicht ein aufgezeichneter TKIP-Handshake aus. Allerdings benötigen die anschließenden Berechnungen einen aktuellen High-End-PC, da gerade einmal ca. 70 Passwörter pro Sekunde geprüft werden können. Falls also ein geeignetes starkes Passwort gewählt wurde geht von dieser Attacke eine nur minimale Gefahr aus.

5.1.7.3. WPA2

Da der bei WEP und WPA zugrundeliegende Algorithmus RC4 als gebrochen gilt, wurde bei WPA2 die AES Verschlüsselung eingesetzt. Dabei bleibt WPA2 noch zu WPA abwärtskompatibel.

Eine “Brute-Froce”-Attacke ist zwar rein theoretisch denkbar, aber sehr unwahrscheinlich, da die Wahrscheinlichkeit für eine identische Prüfsumme eines MAC bei ca. 1 : 1 Mio. liegt. Ausserdem wird bei einer Attacke ein 60 Sekunden andauernder Blackout (Unterbrechen der Verbindung) des Access Points verwendet, um etwaige Attacken abzublocken. Allerdings kann eine “Denial-of-Service”-Attacke so durchaus erfolgreich sein, wenn es der Angreifer nicht auf den Schlüssel abgesehen hat.

5.2. Denkbare Angriffsformen auf Mobiltelefone in einer Firmeninfrastruktur

- Ausspähen von Daten: Der Angreifer verschafft sich Zugang zu relevanten Daten, die auf dem jeweiligen Gerät gespeichert sind. Dazu gehören Kontaktlisten, E-Mails, SMS und andere vertrauliche Dokumente und Dateien.

- Nutzung eigener Dienste und Zugänge: Wird eine Authentifizierung lediglich beim Einschalten des Geräts verlangt, kann das Gerät im Verlustfall ganz leicht von Kriminellen wie ein Schlüssel für Dienste und Zugänge genutzt werden. Dadurch können Sicherheitsmechanismen, die Firmen-, Service- oder Datenstrukturen vor unbefugtem Zugang schützen, ausgehebelt werden.
- Manipulation der Software-Komponenten: Dabei können Angriffe auf Netzwerke in Verbindung mit der Synchronisation zwischen mobilem Gerät und dem Firmennetz erfolgen, um Informationen oder Daten direkt zu erhalten. Darüber hinaus bietet die Konfiguration von Proxies, die bei der Internetkommunikation genutzt werden, die Möglichkeit des Abhörens und Aufzeichnens, aber auch der Manipulation (Tracing, Capturing, Logging) der an das Gerät zurückgesendeten Informationen.
- Überwachung: Moderne Smartphones bieten neben Kommunikationsschnittstellen wie GPRS, 3G, WLAN und Bluetooth unlängst auch GPS-Module. Damit lassen sich raumbezogene Referenzinformationen ablegen, die das Erstellen eines genauen Bewegungsprofils ermöglichen. Dazu ist eine Manipulation des Geräts notwendig, die aber keines Diebstahls bedarf.

5.3. Sicherheitsmechanismen

Folgende Mechanismen sollen dazu beitragen, dass Firmenrelevante Daten nicht einfach von Dritten ausgespäht werden können.

5.3.1. WLAN sichern mit Radius

Beim WLAN-Einsatz in Unternehmen reicht die simple Authentifizierung über ein gemeinsames Passwort (Shared Secret) mit WPA-PSK nicht: Das Geheimnis ist bei großer Verbreitung zu schnell keines mehr. Spätestens wenn ein Kunde vorübergehend einen Zugang bekommen hat, muss man es ändern. Mit serverseitig zugeteilten Passwörtern erspart sich der Administrator viel Arbeit und Nachfragen seiner Nutzer.

Für solche Einsatzfälle ist die Spielart WPA Enterprise gedacht, bei der die WLAN-Basisstation Verbindungsanfragen von ihren Clients über das Protokoll IEEE 802.1x mit einem nachgelagerten Radius-Server aushandelt. Auf Linux-Systemen ist dazu das Open-Source-Paket Freeradius gängig. [Radius10], letzter Abruf 07.12.2010

Mit dem Extensible Authentication Protocol (EAP) unterstützt es verschiedene kryptografisch gesicherte Methoden (EAP-TLS/-TTLS, PEAP, LEAP), One-Time-Passworte und SIMs. Für die Authentifizierung sind Username/Passwort-Kombinationen oder Zertifikate gebräuchlich.

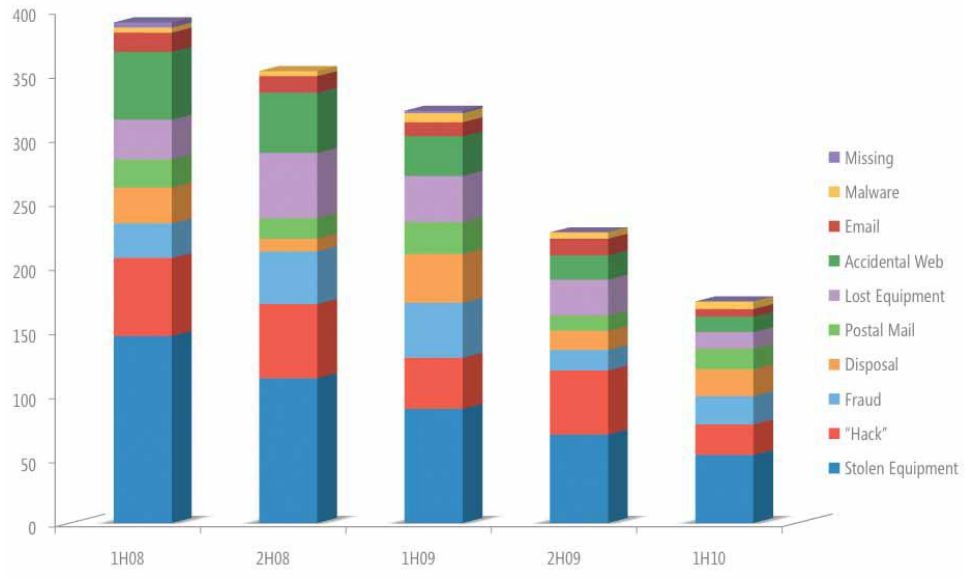

Abbildung 5.1.: Datenverluste in Unternehmen

Damit Daten nicht im Klartext durch die Luft beziehungsweise über die Leitung zwischen Basisstation und Radius-Server laufen, verschlüsselt Freeradius diese. Voraussetzung dafür ist, dass auf dem Client mindestens ein Stammzertifikat (Root CA Certificate) installiert ist, von dem das Zertifikat des Radius-Servers abgeleitet ist. Mit dem Stammzertifikat prüft der Client auch, dass er sich beim richtigen Radius-Server authentifiziert.

5.3.2. MDM

Wie man der Abbildung 5.1 entnehmen kann führt der Diebstahl von Endgeräten am häufigsten zu Sicherheitseinbrüchen in Systemen.

MDM (Mobile Device Management) bezeichnet die Möglichkeit von der Ferne aus auf ein Mobiltelefon zuzugreifen und dieses gegebenenfalls zu sperren bzw. Daten zu löschen oder andere Sicherheitsmechanismen auszulösen.

Die derzeitige Funktionspalette umfasst:

- auf dem Gerät enthaltene Daten sichern und wieder aufspielen (“Backup & Restore”)
- Software-Updates zentralisiert und drahtlos aufspielen, um Sicherheitslücken schnell zu schließen (“Update Over The Air”)
- ein gestohlenes oder verlorenes Gerät aus der Ferne sperren und seine Daten löschen (“Remote Lock & Wipe”) sowie per GPS verfolgen (“Mobile Tracking”)
- einzelnen Nutzern differenzierte Rechte zuteilen - vom Internet-Zugang bis zur Installation von Programmen (“Policy & Provisioning”)
- Statistiken erstellen, wie ein Smartphone genutzt wird und welche Kosten anfallen

("Logging & Accounting")

MDM-Lösungen müssen dabei den gesamten Lebenszyklus eines Smartphones begleiten. Zu Beginn bekommen die Geräte eine Identität zugewiesen, die mit den entsprechenden Zugangsberechtigungen und Funktionseinschränkungen verknüpft ist. Im Idealfall bekommt der Nutzer für sein Smartphone nur noch ein Passwort, mit dem er sich einmalig für die Erstkonfiguration anmelden kann. Während der normalen Nutzung im Alltag können die Firmen-Administratoren mittels MDM dann zentral Software-Updates verteilen, Geräte sperren oder durch Konfigurationsänderungen schnell auf Sicherheitslücken reagieren. Wird das Smartphone schließlich ausgemustert oder verlässt der Nutzer das Unternehmen, werden mittels MDM Berechtigungen gelöscht, Unternehmensdaten getilgt oder die Konfiguration auf ein neues Endgerät übertragen.

Der Markt für MDM-Lösungen ist derzeit noch übersichtlich. Statistiken über Marktanteile existieren nicht. Zu den großen drei, die immer wieder genannt werden, gehören jedoch die US-Unternehmen Mobile Iron, Good Technology und Sybase. Ihre Produkte unterscheiden sich dabei vor allem im Detail - während Good Technology großes Knowhow in Sales-Umgebungen verspricht, streicht man bei Mobile Iron die Handhabbarkeit unterschiedlicher Smartphone-Plattformen heraus. Sybase wirbt wiederum mit einer guten Anbindung an die hausinterne IT. Zumeist bezahlen Unternehmen einen Sockelbetrag plus eine Lizenzgebühr pro Nutzer. Mit insgesamt einigen Tausend Euro pro Jahr müssen die Firmen dabei rechnen. Anbieter wie Mobile Iron haben für Neukunden aber auch Lockofferten von nur vier Dollar pro Nutzer und Monat (plus Sockelbetrag) im Angebot.

Was die Sache noch komplizierter macht: In vielen Firmen werden längst nicht mehr nur reine Businessgeräte wie die Blackberrys des kanadischen Anbieters RIM eingesetzt. Apples iPhone beispielsweise hat mittlerweile ebenfalls einen Siegeszug in Unternehmen angetreten. Der traditionsreiche IT-Dienstleister Unisys etwa, der für Firmen und Regierungen auf der ganzen Welt arbeitet, benutzt iPhones, um Server zu überwachen, aber auch für höchst sicherheitskritische Anwendungen wie die Steuerung von Systemen zur Kameraüberwachung.

Ein weiterer kritischer Punkt sind private Anwendungen auf Firmen-Smartphones. Geräte wie das iPhone laden geradezu dazu ein, neben geschäftlichen Apps auch Spiele zu installieren oder Multimedia-Content zu konsumieren. Schließen lässt sich dieses Sicherheitsloch derzeit nur, indem Firmen das Aufspielen privater Inhalte ganz verbieten.

5.4. Entscheidung für Übertragungstechnik

begründung durch schlussvolgerung aus sicherheitslücken und erläuterung sicherheitskonzept, damit die schwachstellen der technik nicht ausgenutzt oder nur zum teil ausgenutzt

werden können.

6. Mobile Betriebssysteme und Entwicklungsvoraussetzungen

Der Smartphonemarkt ist relativ groß und unübersichtlich. Im Nachfolgenden werden verschiedene Betriebssysteme angesprochen und verglichen, um eine Ausgangsbasis zu schaffen. Dabei wird verstärkt auf die derzeitigen Marktführer eingegangen (vgl. Abbildung 6.1).

Wie man der Grafik entnehmen kann sind dies vor Allem das von Nokia verbreitete Symbian, Google's Android und iOS von Apple. Symbian bleibt zwar laut Statistik wieterhin unangefochtener Marktführer, befindet sich allerdings auf dem absteigenden Ast. Android und iOS bauen ihren Marktanteil immer weiter aus und werden Symbian so wohl bald vom Thron stossten.

6.1. iOS

- Grundlagen und Bedienung

Company	3Q10 Units	3Q10 Market Share (%)	3Q09 Units	3Q09 Market Share (%)
Symbian	29,480.1	36.6	18,314.8	44.6
Android	20,500.0	25.5	1,424.5	3.5
iOS	13,484.4	16.7	7,040.4	17.1
Research In Motion	11,908.3	14.8	8,522.7	20.7
Microsoft Windows Mobile	2,247.9	2.8	3,259.9	7.9
Linux	1,697.1	2.1	1,918.5	4.7
Other OS	1,214.8	1.5	612.5	1.5
Total	**80,532.6**	**100.0**	**41,093.3**	**100.0**

Source: Gartner (November 2010)

Abbildung 6.1.: Smartphone - Marktanteile

Tabelle 6.1.: Die Architektur des iPhone OS

Eigene (native) Anwendungen
Cocoa Touch NSFFoundation, UIKit
Media (Quartz, Core Animation, Open GL, Core Audio, ...)
Core Services (Core Location, Adress Book, SQLite, CFNetwork, ...)
Core OS (I/O, Threads, Sockets, Bonjour, Key Chain, ...)

- Hinweis auf Legalität/Schwierigkeit

6.1.1. Architektur

Apple überlässt dem Entwickler beim iPhone nicht direkten Zugriff auf die Hardware, sondern über diverse Layer (Tabelle 6.1), die zwischen Hardware und der Applikation vermitteln sollen. Diese Layer beschreiben das iOS. [Koller10]

6.1.2. iOS Komponenten

Der “Cocoa Touch Layer” beinhaltet die am häufigsten von Entwicklern genutzten Frameworks zum Programmieren des iPhones. Er basiert auf der Standard Mac OSX Cocoa API und wurde an die Bedürfnisse des iPhones angepasst. Die wichtigsten Komponenten hieraus sind das “UIKit Framework” (Zuständig z.B. für das erstellen des Userinterfaces), der “Push Nocification Service” (Nachrichten werden eventgesteuert an den Benutzer weitergegeben), das “Message UI Framework” und das “Address Book UI Framework”.

Eine Schicht unter Cocoa befindet sich der “Media Layer”. Dieser ist dafür zuständig, dass das iPhone mit Audio-, Videodaten und (3D-)Grafik umgehen kann.

Mit dem “Core Service Layer” hat man beispielsweise die Möglichkeit auf die im System integrierte mySQL Datenbank zuzugreifen. Weiterhin lässt sich zum Beispiel mit dem “Data Framework” das MVC Prinzip für Applikationen umsetzen.

Der “Core OS Layer” liegt direkt auf der Hardware auf. Hier werden einige Dienste zur verfügung gestellt, wie zum Beispiel der Netzwerkzugriff (low level), der Zugriff auf externe Geräte oder Betriebssystemspezifische Dinge wie Speichermanagement oder wie mit Threads umzugehen ist.

Tabelle 6.2.: Apple Developer Programs

Apple Programmname	Kosten/Jahr
iOS Developer Program - Individual	99 $
iOS Developer Program - Company	99 $
iOS Enterprise Program	299 $
iOS Developer University Program	kostenlos

[1]

6.1.3. Sicherheit

Entwickelt wird für das iPhone in Objective C. Der Quellcode wird in Machinensprache übersetzt und läuft ohne zusätzliche Laufzeitumgebung. Apples Sicherheitskonzept sieht eine Isolierung der Apps und Prozesse mittels einer Mandatory Access Control vor.

In der Vergangenheit stellte sich immer wieder heraus, dass trotz Sandboxing Apps zumindest lesend auf Konfigurationsdateien zugreifen können. Das liegt unter Anderem auch an den Mac OS X ähnlichen Zugriffsregeln auf Basis von Regular Espressions. Es werden Zugriffsrechte generisch definiert und nicht auf einzelne Apps abgestimmt.

Allerdings existiert neben dem Softwareseitigen Sandboxing und dem Code Signing ausserdem ein hardwareseitiger Schutz. Der ARM-Prozessor des iPhones unterstützt eine Datenausführungsverhinderung (DEP). Diese soll die Auswirkungen von Buffer Overflows und Heap Overflows limitieren. [CT10_2], Seite 80ff

6.1.4. Entwicklung

Bei Apple muss zur Entwicklung ein Gerät aus dem eigenen Hause verwendet werden. Die XCode Tools, so wie sich die Entwicklungsumgebung nennt, sind nicht für Windows oder Linux erhältlich. Somit scheiden alle anderen Betriebssysteme erst einmal aus. Allerdings kann man sich nach der Registrierung als “Apple Developer” auf der Apple Entwicklerhomepage das iPhone SDK kostenlos herunterladen. [AppleDev10], letzter Abruf: 30.11.2010

Dafür erhält man mit dem iPhone SDK für die Xcode Tools eine komfortable Entwicklungsumgebung und einen iPhone Simulator zum Testen der selbstgeschriebenen Apps.

Um eine Eigenentwicklung allerdings tatsächlich auf dem Zielgerät laufen zu lassen, muss man sich bei Apple als Entwickler einkaufen. Dies ist in verschiedenen Variationen möglich (vgl. Tabelle 6.2).

Die für das iPhone eingesetzte Programmiersprache ist Objective C. Was in etwa der normalen Sprache C entprischt mit der Erweiterung auch objektorientiert programmieren zu können. Erlaubt wird allerdings auch normales C und C++.

[1] http://developer.apple.com/programs, letzter Abruf 24.11.2009

Jailbreak/Hackint0sh

Da Apple sehr restriktiv mit den Entwicklungsvoraussetzungen ist, haben sich "Hacker" zusammengetan und einen Weg gefunden, wie man diese Voraussetzungen umgehen kann.

Die erste Voraussetzung ist ein iPhone/iPad, dass "geJailbreakt" wurde. (siehe Anhang) Mittels eines SSH-Zugangs lassen sich nun auch Programme, die sich nicht im AppStore befinden auf das Gerät laden.

Aber nicht nur das Zielgerät für die App muss freigeschaltet werden. Prinzipiell sieht Apple vor, dass man nur mit hauseigener Hardware (z.B. Macbook) mit dem Betriebssystem SnowLeopard entwickeln kann. Da die neuesten Macbooks mittlerweile mit Intel Prozessoren bestückt werden, haben dies einige findige Hacker ausgenutzt und es geschafft das Betriebssystem auch auf herkömmlichen PC's mit Intel und sogar AMD Prozessoren lauffähig zu machen. (siehe Anhang).

6.2. Android

- Keine fest vorgeschriebene Hardware (SDK lauffähig unter Mac/Linux/Windows)
- Entwicklungsumgebung frei wählbar (Eclipse, möglicherweise auch Netbeans bzw. einfacher Texteditor...)
- Android Market muss nicht zwingend benutzt werden (Developer Tools auf dem Gerät muss dafür aktiviert sein). Es kann von jeder Quelle aus eine Anwendung auf das Gerät installiert werden.
- Anwendung einfach auf dem Handy installierbar oder auf dem Simulator zu Testen.

Seit dem 12. November 2007 ist eine Vorabversion des Android-SDK von Google verfügbar. Diese wurde sehr positiv von den Entwicklern angenommen und wird seitdem kontinuierlich erweitert und geupdatet.

Androidunterstützte Geräte sind vor allem im Mobilfunkbereich sehr gefragt. Allerdings gibt es auch viele Entwicklungen im Home Entertainment, sowie für Netbooks, Tablet-PC's oder für Festnetztelefone.

6.2.1. Architektur

Den Kern von Android bildet ein Linux-Kernel, der speziell auf geringen Energieverbrauch und effizientes Speichermanagement ausgelegt wurde (vgl. Restriktionen Smartphones <-> Desktop PC's).

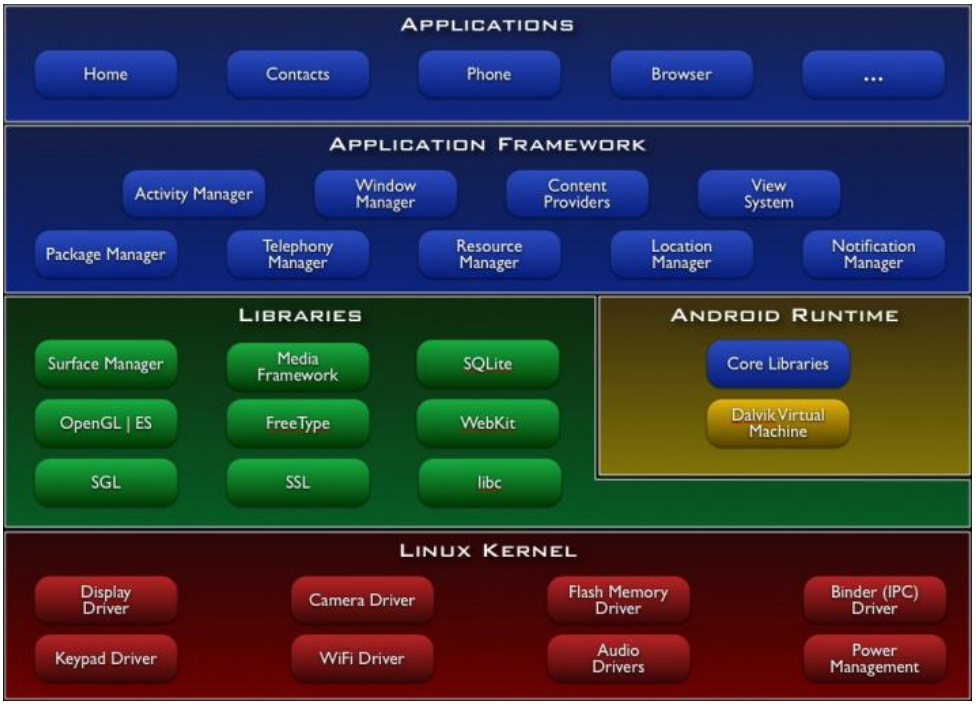

Abbildung 6.2.: Die Android-Systemarchitektur

Die Android-Laufzeitumgebung stellt die Dalvik Virtual Machine (im Folgenden DVM genannt) dar. Sie bildet das Herzstück der Plattform. Android lässt sich komfortabel und komplett in Java programmieren. Allerdings sollte man nicht den Fehler machen und meinen es handle sich bei der DVM um die reguläre Java Virtual Machine.

> "Die DVM und JVM sind grob gesehen sehr ähnlich, wobei sie sich aber in zwei Hinsichten grundlegend unterscheiden: Die DVM ist nicht als Stack-Maschine, sondern als Register-Maschine realisiert, und die Längen der Opcodes betragen bei der DVM zwei anstatt nur einem Byte. Eine auf Register basierte virtuelle Maschine holt ihre Bytecodes und Operanden aus (virtuellen) Registern. Dazu ist es natürlich erforderlich, dass die Operanden in Abhängigkeit des Opcodes in bestimmte Register geschrieben und von dort gelesen werden und nicht generell vom Stack geholt werden, wie es in einer Standard JVM geschieht."[Fokus09]

Der Vorteil daraus ist schnell ersichtlich. Bei den Registern handelt es sich nämlich quasi um Zwischenspeicher direkt im Mikroprozessor, die die Berechnungen über mehrere Zwischenergebnisse stark beschleunigen.

Da die Java-VM und der original Java-Bytecode lizenzrechtlich geschützt sind, bediente sich Google eines Tricks, diese aussen vorlassen zu können. Die DVM wird nicht explizit als Java-VM dargestellt und verarbeitet keinen Java-Bytecode, sondern Android-eigenen Dex-Bytecode, der nicht unter die Sun-Lizenz fällt. Es wird also gewohnterweise in Java programmiert und mit Hilfe des Java-Compilers von Sun Bytecode in Form von .class-

Dateien erzeugt. Das Android eigene dx-Tool liefert aus dieser Quelle den Dex-Bytecode, der in eine für Android-Geräte ausführbare .apk-Datei (sprich fertig ausführbare Anwendung) gepackt wird. Da die Programmierschnittstelle der Java SE von SUN bisher noch nicht patentrechtlich geschützt ist, werden hier somit keine Rechte verletzt.

6.2.2. Android-Komponenten

Android stellt eine moderne Plattform für komponentenbasierte Anwendungsentwicklung dar. Hintergrund hiervon ist, dass für jede neue Applikation, die entwickelt werden soll das Rad nicht neu erfunden werden muss. Komplette Anwendungen oder Teile dieser sollen auch in neu entwickelten Anwendungen benutzbar sein. Erklären lässt sich dies an einem Beispiel:

Vorinstalliert auf einem Standard Android-Gerät findet man Anwendungen wie “Kontakte” oder “Telefon”. In diesen Anwendungen sind beispielsweise Telefonnummern und Kontaktdaten gespeichert. Möchte man nun die Daten aus den Anwendungen benutzen kann man über die Datenbank der entsprechenden App (vorhandene Berechtigung vorausgesetzt) direkt auf die benötigten Werte zugreifen.

Sauber gliedern sich die einzelnen Android-Komponenten gemäß dem in der Programmierung als Standard geltenden Model-View-Controller-Prinzips.

“Activity” übernimmt hier die Funktion des Views. Oberflächenelemente anzeigen, auslesen von Eingabefeldern und Benutzereingaben werden von den Activities behandelt. Man spricht hier von allen sichtbaren Bestandteilen einer Anwendung.

Für die Hintergrundprozesse ist im Android-Gerät der **“Service”** zuständig. Analog zum Model-View-Controller-Prinzip würde man hier vom Controller sprechen. Wichtig sind diese Services zum Beispiel, wenn Prozesse weiterlaufen sollen, obwohl die Anwendungen vom Benutzer schon geschlossen wurden (Beispiel Musik Player).

Das Model ist im Allgemeinen für die Verwaltung der darzustellenden Daten zuständig. Bei der Datenpersistenz in Android spricht man von **“Content Provider”**. Über Berechtigungen können seine Daten bestimmten Anwendungen zur Verfügung gestellt werden.

Zusätzlich zu den Standardkomponenten im MVC haben wir bei Android noch den **“Broadcast Receiver”**. Dieser empfängt Systemmeldungen z.B. über Störungen des Netzwerks. Anwendungen haben so die Möglichkeit, auf einen geänderten Systemzustand zu reagieren.

6.2.3. Sicherheit

Das Sandbox-Prinzip:

> "Android führt Anwendungen in der Sandbox aus. Eine Sandbox ist eine eingeschränkte Laufzeitumgebung, in der bestimmte Funktionen verboten sind." [BeckerPant10], S. 27

Eine Android-Anwendung besitzt aus der Sandbox heraus folgende Eigenschaften:

- eigener Prozess
- für einen eigenen Betriebssystem-User
- eigene DVM
- eigener Bereich im Hauptspeicher (Heap)
- eigener Bereich im Dateisystem

Wird eine Anwendung gestartet, läuft sie unter dem Userkonto in einem eigenen Prozess. Programme sind in Linux gegen den Zugriff von außen geschützt, und dadurch kann auch kein unerlaubter Zugriff auf den Speicher der DVM erfolgen, die in diesem Prozess läuft.

6.2.4. Entwicklung

Prinzipiell ist man bei Android weder an Hardware noch Software zum entwickeln gebunden. Empfohlen wird allerdings die Eclipse IDE mit dem ADT Plugin.[ANDEV10], letzter Abruf 01.12.2010
Das Android SDK kann dann entweder von Hand über die Konsole zum erzeugen eines Projekts benutzt werden oder eben komfortabel über die IDE. Anwendungen für die Androidplattform werden ausnahmslos in Java geschrieben.

6.3. Windows Phone 7

Relativ neu auf dem Markt ist das Windows Phone 7 (Verkaufsstart in Deutschland: 3. November 2010). [Heise10], letzter Abruf 30.11.2010
Damit will Microsoft sein leicht angestaubtes Mobilbetriebssystem auf den neuesten Stand bringen und zu iOS und Android aufholen. Erste Testberichte zeigen, dass Windows Phone 7 bis jetzt ein zweischneidiges Schwert ist.

Microsoft ist es gelungen eine einfach zu bedienende Oberfläche zu erstellen, die sich von iOS und Apple abhebt. Den signifikanten Unterschied stellen die "Live Tiles" dar. Diese sind in verschiednen Größen auf dem Home Bildschirm darstellbarer Inhalt, der quasi live

mit Informationen aktualisiert wird.

Ausserdem stellt Microsoft klare Anforderungen an die Hardware für sein Betriebssystem. Windows Phone 7 benötigt einen Prozessor mit mindestens 1 GHZ Taktfrequenz und einen internen Speicher von 8 GB. Dementsprechend flüssig und ohne Verzögerung laufen auch die meisten Anwendungen auf den Geräten, die den hohen Auflagen genügen.

Programmiert wird hier in Silverlight beziehungsweise XNA. Silverlight ist aus dem WPF (Windows Presentation Framework) heraus entstanden und auf verschiedenen Plattformen und Browsern lauffähig. Das XNA Game Studio wurde in der Vergangenheit für die Entwicklung von Spielen auf dem PC bzw. der XBOX 360 benutzt.

Zumindest die Entwicklung mit dem XNA Game Studio ist kostenlos und wird als Add On für das Visual Studio 2010 Express verwendet. Als Entwicklungsplattform ist Microsoft also ähnlich strikt wie Apple angesiedelt und lässt dies nur auf dem eigenen Betriebssystem zu.

6.4. Symbian

Entwickelt wird auf Symbian mit dem Qt SDK, welches für Windows, Linux oder Mac OS X verfügbar ist. Alternativ lässt sich auch der Eclipse-Ableger Carbide verwenden. Programmiert wird in C++. In Nokias Ovi Store kann die eigene Applikation vertrieben werden. Entwickler können sich in diesem für einmalig 50 Euro anmelden.

Wie bereits erwähnt befindet sich das Betriebssystem Symbian auf dem absteigenden Ast. Der Ansatz von Nokia das Betriebssystem seit dem 24. Juni 2008 unter der Open-Source-Lizenz zu veröffentlichen führte leider nicht mehr dazu, dass es wesentlich populärer wurde. Zu lange haben Nokia & Co. gewartet.

Nur noch bis zum 17. Dezember 2010 werden die Internetsites von Symbian mit Hilfen, Artikeln, FAQs, Tipps, Tricks und Historie online sein. Damit beendet Nokia seine Web-Präsenz für die Open Source-Entwicklung. Die Daten sollen, so die Symbian Foundation, aber auch weiterhin erhältlich sein. Das ist offiziell das sichtbare Ende der Symbian-Unabhängigkeit und der Neustart unter der Ägide des Mutterkonzerns Nokia. Nokia gab bekannt, dass “aufgrund der Marktentwicklung die Weiterführung des Projekts nicht mehr sinnvoll” sei. [Symbian10], letzter Aufruf: 02.12.2010

In Zukunft sollen Mobiltelefone von Nokia mit MeeGo angetrieben werden, einem offenen Linux-System, das seine Wurzeln einerseits in Nokias Maemo und andererseits in dem von Intel initiierten Moblin hat. [CT10], Seite 93

6.5. BlackBerry OS

Für Blackberrys gibt es vergleichsweise wenige Apps. Dies liegt sicher daran, dass diese Geräte häufig dienstlich genutzt werden und viele Firmen die Installation von Dritt-Anwendungen einschränken und/oder verbieten.

Research In Motion (kurz RIM) setzt auf das im Vergleich zu aktuellen Java-Versionen abgespeckte Java ME. Dies entspricht ungefähr dem Stand von Java SE 1.3, allerdings mit zusätzlichen Bibliotheken für die Entwicklung von Software bei Mobiltelefonen.

Empfohlen wird die Entwicklung mit Eclipse und einem entsprechenden Plug-in. Testen lässt sich die selbstgeschriebene Applikation mit einem Simulator für fast jedes Gerät von RIM. Allerdings ist der Zugriff auf viele APIs nur mit einer digitalen Signatur mögliche. Diese lässt sich RIM einmalig mit 20 $ finanzieren. Möchte man seine selbstgeschriebene Applikation im RIM eigenen Appstore vertreiben werden noch einmal 200 $ fällig.

6.6. HP webOS

Palm setzt mit seinem Betriebssystem auf Webstandards. Apps werden mit JavaScript, HTML5 und CSS geschrieben. Auch hier findet das MVC Prinzip einzug. Views werden als normale HTML-Dateien erstellt (mit Palm-proprietären Attributen und Styles). Controller werden mit Hilfe von JavaScript-Klassen dargestellt mit definierten Einstiegspunkten und Callbacks. Für die Persistenzschicht zum dauerhaften Speichern der Daten stehen HTML5-konforme APIs und SQLite-Datenbanken zur Verfügung.

Das WebOS-SDK iust für Windows, Mac OS X und Linux verfügbar. Zusätzlich bietet Palm ein Eclipse-Plug-in zum einfachen Packen und starten der Anwendung aus der IDE sowie der freien Virtualisierungssoftware VirtualBox basierenden Emulator. Alternativ kann mittels “Ares” im Webbrowser entwickelt werden. Dieser besteht aus einer Projekt-Verwaltung, einem JavaScript-Editor und einem WYSIWYG-Editor.

Fertige Anwendungen werden nicht signiert und liegen im Quellcode vor. Dies ist nicht ganz unkritisch, da diese leicht entpackt und verändert werden können, zum Beispiel zum aushebeln von Sicherheitsmechanismen, indem einfach die spezielle JavaScript-Zeile auskommentiert wird.

Die zuvor festgelegte Gebühr in höhe von 99 $ zur Teilnahme am Entwicklerprogramm hat Palm ausgesetzt. Von daher entstehen im Moment keine zusätzlichen Kosten bei der Entwicklung bzw. beim Publishing im Market.

6.7. WebApp

Eine schnell umzusetzende Applikation für alle verfügbaren Betriebssysteme im Mobilbereich wäre eine Browserbasierte Lösung. Diese kann natürlich für mobile Geräte im Hinblick auf Darstellung und Benutzbarkeit optimiert werden.

Die Entwicklung würde sehr schnell von statten gehen, da die benutzte Technik weit verbreitet ist und gängige Browser im Mobilbereich die Standards der Webprogrammierung unterstützen.

Leider ist es dabei aber nicht möglich auf die gerätespezifischen Eigenschaften zuzugreifen. Es gibt zwar Ansätze wie beispielsweise Sencha Touch, in der sich das Userinterface den entsprechenden Geräten anpassen lässt, aber Kamera, GPS & Co. sind aus Sicherheitsgründen nicht über ein Webinterface ansprechbar (Einzig Lokalisierungsdienste wie Google bieten hier eine Ausnahme). So ist die Entwicklung einer WebApp doch relativ beschränkt und nur für Anwendungen zu benutzen, die nicht auf die Hardware des Geräts zugreifen müssen.

Eine gute Kombination aus WebApp und nativen Apps bieten Cross-Platform Development Tools, welche im Folgenden beschrieben werden.

6.8. Cross-Platform Development Tools

Unter Cross-Platform development versteht man im Allgemeinen ein Framework, das es ermöglicht Applikationen auf mehreren Endgeräten zu deployen. Im Vordergrund steht nicht die Performance der Anwendung, sondern die Kompatibilität mit verschiedenen Betriebssystemen und deren “look and feel” so nahe wie möglich zu kommen. [CT10], S. 96

Zur Zeit befinden sich die Frameworks PhoneGap[2], Titanium[3] und Rhodes[4] (evtl. + Sencha Touch) sich auf dem Markt. All diese basieren auf Webtechniken und zielen speziell auf Webentwickler ab, da diese ihre Fähigkeiten direkt für die Smartphone Apps Welt benutzen können (Vgl. Tabelle 6.3).

PhoneGap und Titanium werden mit Hilfe von HTML, CSS und JavaScript programmiert. Beide benutzen JavaScript APIs, um auf die nativen Gerätefunktionen des Smartphones zugreifen zu können (Beispielsweise GPS, Accelerometer, Sound). Im Gegensatz dazu wird in Rhodes mit Ruby gearbeitet. Es bietet eine komplette Serverumgebung auf dem Gerät und dadurch Zugriff auf die nativen Funktionen.

[2] http://www.phonegap.com
[3] http://www.appcelerator.com/
[4] http://rhomobile.com/

PhoneGap und Titanium greifen über bestimmte JavaScript APIs auf die Funktionen der Smartphones zu. Die Logik (HTML, CSS, Javascript) wird allerdings in einem Browserfenster angezeigt. Durch die JavaScript APIs hat die "Web App" sozusagen Zugriff auf alle interessanten Funktionen des Handys, wie zum Beispiel die Geolocation, das Accelerometer, die Kamera, Kontakte, Datenbank oder das File System.

Prinzipiell kann jede Funktion, die das Smartphone SDK anbietet in die Javascript Welt überführt werden. Eine normale Web App, die im mobilen Browser des Handys läuft könnte zum Beispiel nicht auf diese Funktionen zugreifen (Sicherheitsaspekt).

Auf der Titanium Webseite wird damit geworben, dass dafür entwickelte Anwendungen in native Anwendungen für das zu entwickelnde System kompiliert werden können. Man darf das allerdings nicht falsch verstehen. Titanium kann kein HTML, CSS oder JavaScript in nativ geschriebene Anwendungen umwandeln. Diese werden einfach als Ressourcen zu der ausführbaren App hinzugefügt. Am besten stellt man es sich so vor, wie ein eingebettetes Bild. Wenn die App ausgeführt wird, werden diese Ressourcen in den "UIWebView" geladen und als JavaScript ausgeführt.

Genau so wird es auch in PhoneGap gemacht. Hinsichtlich Architektur sind sich beide Frameworks sehr ähnlich. Ein wesentlicher Unterschied ist, dass PhoneGap nicht die nativen Userinterfacekomponenten in JavaScript zur Verfügung stellt. Das "Look and Feel" muss mittels CSS selbst aufbereitet werden. Im Gegensatz dazu bietet Titanium eine starke UI API, die mittels JavaScript dazu benutzt werden kann alle möglichen Arten von nativen UI Kontrollen zu erzeugen. Im direkten Vergleich sehen deswegen Titanium Apps eher wie direkt für das Smartphone geschriebene Apps aus. Allerdings unterstützt PhoneGap mehr verschiedene Betriebssysteme als Titanium. Die PhoneGap APIs sind allgemeiner gefasst und können so auf vielen verschiedenen Systemen wie z.B. iPhone, Android, Blackberry oder Symbian laufen. Titanium konzentriert sich auf das iPhone und den Android Markt.

Mit Hilfe des Rhodes Frameworks wird es Entwicklern gestattet native Apps zu erstellen mit der Möglichkeit HTML Templates zu erstellen und in Ruby zu programmieren. Apps, die mit Rhodes entwickelt werden haben die Performance und die Fülle von Apps, die nativ für das entsprechende System geschrieben wurden mit dem Vorteil, dass das Interface in HTML verfasst wird. Ausserdem sehr positiv fällt auf, dass Rhodes das MVC Prinzip verwendet und somit Logik und Anzeige strikt getrennt sind.

Entwickler mit Railserfahrung werden sicherlich schneller mit Rhodes zurechtkommen. Das Framework hat übrigens auch Grenzen. Diese manifestieren sich vor allem darin, dass es keinerlei Integration von Audio oder Video in den einzelnen Apps gibt. Rhodes bietet auch an, dass man Funktionen aus den nativen APIs des Systems aufrufen kann, wenn diese nicht im Framework unterstützt werden sollten. Jedoch sollte man sich nicht darauf verlassen, dass diese dann flüssig ablaufen.

Tabelle 6.3.: JavaScript-Frameworks für mobile Apps [5]

Name	PhoneGap	Titanium Developer	Rhodes
App-Typ	quasi-nativ	quasi-nativ	(quasi-)nativ
Hersteller	Nitobi	Appcelerator	Rhomobile
Betriebssysteme	alle(Eclipse)	Windows, Linux, Mac OSX	Windows, Mac OSX
Lizenz	MIT	Apache	MIT
Programmiersprache	JavaScript	JavaScript	Ruby
Plattformen			
Android	√	√	√
iPhone	√	√	√
WebOS	√	x	x
Symbian	√	x	√
Blackberry	√	√(nur Pro-Version)	√
Windows Mobile	√	x	√

Worin sich jedoch alle Produkte voneinander abheben ist die Entwicklerlizenz. PhoneGap ist Open-Source und kann frei genutzt werden. Titanium befindet sich noch im Betastadium, wird aber in Zukunft ein kommerzielles Produkt werden. Rhodes ist bereits kommerziell und kostet 500 $ pro Projekt. Die Summe muss vor Beginn der Entwicklung bezahlt werden.

[5] [CT10], S. 101

7. Implementierung und Test

7.1. Wahl des Betriebssystems

- tja... Entweder nur eins auswählen oder Android und iOS und Win7 Mobile (oder Webinterface...) -> ZU VIEL ARBEIT?
- Wahrscheinlich muss die Entwicklung für iOS ausgeschlossen werden, weil die entsprechende Entwicklungshardware fehlt. ODER: JAILBREAK -> Eclipse -> TESTEN LIVE AUF DEM GERÄT OHNE SIMULATOR (zumindest theoretisch auf diese Möglichkeit eingehen).
- gibt es evtl. freie iphone emulatoren? ->derzeitige Recherche-> NEIN
- Entwicklungsumgebung für Android und Win7 Mobile ist kostenlos.
- Annahne: Wahrscheinlich fällt die Wahl auf Android, da hier auch direkt auf einem Gerät getestet werden kann und nicht nur auf dem Simulator. Zu demonstrationszwecken ist dies wohl sinnvoller.

7.2. Codegenerierung

- Probleme während der Umsetzung?
- Erwartungshaltung?
- Umsetzung aus AndroMDA heraus

7.3. Tests White/Blackbox?

- Testfälle aufstellen und diese mit JUnit umsetzen(nur für Android)
- Alternatives Testframework für iPhone & Win7 Mobile?!?

8. Zusammenfassung und Ausblick

- 2-3 Seiten Zusammenfassung der Arbeit und Ausblick auf weitere Projekte für die Zununft
- Vgl. Erwartete Ergebnisse - Abweichungen
- marketingtechnische konzepte: kann das ergebnis im hinblick darauf kontrolliert werden? kundenzufriedenheit & co.? wie könnte eine mögliche evaluation erfolgen?

Die Erwartungshaltung für diese Diplomarbeit bestand darin in einer vorgegebenen Zeit zu evaluieren, ob der Einsatz von Smartphoneapps in der Kommunikation mit Maschinen lohnend ist oder nicht. Ausgehend vom Wissensstand zu Beginn dieser Arbeit wurde vermutet, dass die Umsetzung einer App durchaus möglich, wenn auch nicht einfach ist. Die größte Schwierigkeit im Vorfeld lag vor allem daran, die Darstellung der verschiedenen Funktionen für das Endgerät so zu optimieren, dass die Lesbarkeit der Information jederzeit gewährleistet sein sollte. Bei einer begrenzten Displaygröße vor allem bei Smartphones müssen Informationen klar und deutlich zu gewinnen sein.

Literaturverzeichnis

[1] [Clark10] Josh Clark: Tapworthy - Designing Great iPhone Apps, 1. Auflage, O'Reilly 2010

[2] [Broy10] Manfred Broy: CYBER-PHYSICAL SYSTEMS - Innovation durch softwareintensive eingebettete Systeme, 1. Auflage, Springer 2010

[3] [KHS10] o.V.: Unternehmen, In: KHS GmbH, Stand: 2010, URL: http://www.khs.com/de/unternehmen.html

[4] [Gartner10] Gartner: Technology Research & Business Leader Insight, In: Gartner, Inc., Stand: November 2010, URL: http://www.gartner.com/it/page.jsp?id=1466313 (letzter Abruf am 17.11.2010)

[5] [CHI99] Fukumoto Masaaki: Whisper - A Wristwatch Style Wearable Handset, o.V., USA 1999

[6] [Schmalenbach05] Jenni Schmalenbach: Das Phänomen Short Message Service (SMS) - Einführung in eine neue Kommunikationsform, 1. Auflage, Ludwig-Maximilians-Universität München 2005

[7] [BeckerPant10] Arno Becker, Marcus Pant: Android 2 - Grundlagen und Programmierung, 2. Auflage, dpunkt.verlag 2010

[8] [Petzold10] Charles Petzold: Programming Windows Phone 7, 1. Auflage, Microsoft Press 2010

[9] [ITU10] o.V.: International Telecommunication Union, Stand: 15. Februar 2010, URL: http://www.itu.int/net/pressoffice/press_releases/2010/06.aspx

[10] [Apple10] o.V.: Apple Special Event, Stand: 20. Oktober 2010, URL: http://events.apple.com.edgesuite.net/1010qwoeiuryfg/event/index.html

[11] [AppleDev10] o.V.: Apple Developer, Stand: 2010, URL: https://developer.apple.com

[12] [Fokus09] o.V.: Fokus Report, Fachhochschule Nordwestschweiz, 250. Auflage, Verlag: Fachhochschule Nordwestschweiz FHNW Institut für Mobile und Verteilte Systeme

[13] [Heise10] o.V.: Heise Online, Stand: 20.10.2010, URL: http://www.heise.de/newsticker/meldung/Windows-Phone-7-startet-bei-T-Mobile-

mit-Verzoegerung-1121952.html

[14] [CT10] o.V.: c't - magazin für computer technik, Heft 16, Heise Zeitschriften Verlag GmbH & Co. KG, Stand 19.07.2010

[15] [CT10_2] o.V.: c't - magazin für computer technik, Heft 20, Heise Zeitschriften Verlag GmbH & Co. KG, Stand 13.09.2010

[16] [ANDEV10] o.V.: Android Developers, Stand: 02.11.2010, URL: http://developer.android.com/guide/developing/other-ide.html

[17] [Symbian10] o.V.: Symbian Foundation, Stand: 28.11.2010, URL: http://developer.symbian.org/wiki/Symbian_Foundation_web_sites_to_shut_down

[18] [Koller10] Dr. Dirk Koller: iPhone-Apps entwickeln - Applikationen für iPhone, iPad und iPod touch programmieren, 1. Auflage, Franzis Verlag, Poing, 2010

[19] [Eren06] Evren Eren, Kai-Oliver Detken: Mobile Security - Risiken Mobiler Kommunikation und Lösungen zur Mobilen Sicherheit, 1. Auflage, Carl Hanser Verlag, München, 2006

[20] [Radius10] o.V.: freeRADIUS - The world's most popular RADIUS Server, Stand: 28.09.2010, URL: http://freeradius.org/

[21] [Stahl07] Thomas Stahl, Markus Völter, Sven Efftinge, Arno Haase: Modellgetriebene Softwareentwicklung - Techniken, Engineering, Management, 2. Auflage, dpunkt.verlag, Heidelberg, 2007

[22] [Gruhn06] Volker Gruhn, Daniel Pieper, Carsten Röttgers: Effektives Software-Engineering Mit UML2 und Eclipse, 1. Auflage, Springer Verlag, Berlin Heidelberg, 2006

A. Anhang

A.1. Jailbreak

WIKIPEDIA

Seit dem 27. Juli 2010 ist die Entsperrung in den USA legal, die Rechtslage in Deutschland ist bisher nicht eindeutig geklärt.

Im Juli 2007 beschrieb der Norweger Jon Lech Johansen in seinem Blog, wie man die Funktionen des iPhones auch ohne AT&T-Vertrag nutzen kann.

Im August 2007 berichtete das Technik-Blog Gizmodo, dass sich das iPhone mit Hilfe einer sogenannten "Turbo-SIM-Karte" auch in anderen GSM-Netzen betreiben lasse. Am 6. September 2007 veröffentlichte die Webseite des APC Magazine eine Zehn-Schritte-Anleitung zur Nutzung des iPhones in weltweit allen GSM-Netzen mit Hilfe einer Turbo-SIM-Karte.

Ebenfalls im August 2007 gelang es dem US-Amerikaner George Hotz, ohne den Umweg über eine Turbo-SIM-Karte die Beschränkung der Nutzung seines iPhones auf das AT&T-Netz aufzuheben. In seinem Blog beschrieb er dazu ein Verfahren, das komplizierte Eingriffe in die Hardware erfordert.

Im September 2007 stellte das iPhone Dev Team, eine freie Gruppe von Programmierern, eine Software-Netlock-Entsperrung für das iPhone frei im Internet zur Verfügung. Damit war das iPhone weltweit in allen GSM-Netzen auch für iPhone-Besitzer nutzbar, die den technischen Aufwand bislang verfügbarer komplizierter Hacks gescheut hatten.

Einen Tag nach Release des iPhones 3GS wurde ein Tool zur Entsperrung dieses Gerätes veröffentlicht.

Am 22. Juni 2009 hat das iPhone Dev-Team eine rein softwarebasierte SIM-Entsperrung namens ultrasn0w für alle iPhones ab Firmware 3.0 veröffentlicht, mit der die Geräte in allen Netzen betrieben werden können. Mit vielen Firmwarereleases kommen neue Versionen der Netzkontrollsoftware, wodurch neue Exploits gefunden werden müssen, um diese erneut freizuschalten. Bisher ist dies für jedes Baseband gelungen. Im Mai 2010 wurde das Tool Spirit veröffentlicht, das auch für das iPad mit iOS 3.2 funktioniert.

Am 1. August 2010 wurde ein webbasierter Jailbreak namens JailbreakMe veröffentlicht, mit dem auch das iPhone 4 geöffnet werden kann. Hierfür wurde ein Programmfehler bei der PDF-Darstellung im Programm Mobile Safari genutzt. Am Tag darauf warnte das BSI davor, mit dem iPhone PDFs zu öffnen, da auch ein schädlicher Code ausgeführt werden könnte. Eine Apple-Pressesprecherin verkündete, dass Apple den Fehler bereits behoben habe und schnellstmöglich ein Update herausbringen werde.

Apple gab daraufhin am 11. August 2010 ein Update (iOS 4.0.2) heraus, das zwei Sicherheitslücken schloss. Die eine Lücke betraf eine Systembibliothek ("Freetype") - durch die zweite erreichte das ausführende Programm System-Rechte am Gerät.

A.2. Hackint0sh

A.3. Model-View-Controller-Prinzip

Erhenwörtliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, den 7. Januar 2011

_ _ _ _ _ _ _ _ _ _ _ _ _

Georg Wolf