der Bundeswehr
Universität München
FAKULTÄT FÜR INFORMATIK

Anwendungsentwicklung für Smartphones

Prof. Dr. Gunnar Teege
Oscar Bertin
Jörg Czanderle
Volker Eiseler
(Hrsg.)

Institut für Technische Informatik

Bericht 2012-01
Mai 2012

Werner-Heisenberg-Weg 39 • 85577 Neubiberg • Germany

Zum Inhalt

Die neue Handygeneration der Smartphones erfreut sich seit der Markteinführung des Apple IPhone 2007 ständig wachsender Beliebtheit bei den Nutzern. Nach einer aktuellen Meldung prognostiziert der Bitkom auf der Grundlage von Marktforschungsdaten, dass 2012 in Deutschland erstmals mehr Smartphones als klassische Mobiltelefone verkauft werden. Hinzu kommen die noch deutlich jüngeren Tablet-Geräte, die meist auf der gleichen Technologie basieren.

Aber auch bei Entwicklern erfreuen sich die neuen Plattformen großer Beliebtheit. Das Konzept der „Apps“, Anwendungen, die spezifische überschaubare Aufgaben erfüllen und online vertrieben werden, hat es jedem interessierten Entwickler ermöglicht, sich zu beteiligen. In gewisser Weise ähnelt dieser Ansatz den Systemanwendungen in den frühen Unix-Systemen, die ebenfalls einen „Werkzeugkasten“ aus kleinen Anwendungen zur Lösung spezifischer Aufgaben bildeten und auch heute noch Verwendung finden. Dieses Konzept unterscheidet sich deutlich von neueren großen monolithischen Anwendungspaketen, wie sie z.B. im Office-Bereich verwendet werden.

Aber nicht nur in dieser Hinsicht unterscheidet sich die Anwendungsentwicklung für Smartphones von der für Desktop-Systeme. Die Unterschiede reichen von ungewohnten Programmiersprachen (z.B. Objective-C für IPhones) über neue, dem spezifischen Verwendungskontext angepasste Formen der Benutzeroberfläche und über vielfältige Konnektivitätsformen bis hin zu neuen Deployment-Verfahren. Dies alles stellt die Entwickler vor viele neue Herausforderungen, bei denen Erfahrungen mit klassischer Software-Entwicklung nur bedingt hilfreich sind.

Die aktuelle Situation ist darüber hinaus von einer starken Fragmentierung der Systemplattformen geprägt. Obwohl die Geräte technisch nahezu identische Funktionen besitzen, unterscheiden sich die System-Schnittstellen zwischen den wichtigsten Plattformen wie iOS, Android, Windows Phone 7 und Blackberry so stark, dass weder die Software noch die Erfahrungen der Entwickler übertragbar sind. Dies ist von den Herstellern bewusst gewollt und typisch für junge Technologien bei denen ein starker Verdrängungswettbewerb statt findet. Aus diesem Grund sind die meisten existierenden Informationen zu Entwicklungskonzepten und -umgebungen spezifisch auf eine bestimmte Plattform ausgerichtet.

Vor diesem Hintergrund fand im Herbst 2011 an der Universität der Bundeswehr ein Seminar über die Anwendungsentwicklung für Smartphones statt, das auf hohe Resonanz bei den Studierenden stieß. Das Ziel war es, einen fundierten Überblick über die relevanten Aspekte zu schaffen und insbesondere die Ansätze für die verschiedenen Plattformen zu vergleichen und einzuordnen.

Dabei wurde auch das Thema behandelt, plattformübergreifende Anwendungen zu entwickeln. Hier ist ein Trend zu beobachten, der das unter dem Begriff HTML5 zusammengefasste Bündel neuerer Web-Technologien einsetzt, um Apps zu entwickeln, die auf allen Systemplattformen ausführbar sind. Es wird interessant sein zu beobachten, wie weit sich die entsprechende Entwicklung bei Desktop-Systemen, die mit der Einführung von Java vor fast 20 Jahren begann, im Bereich der Smartphones wiederholen wird.

Gunnar Teege
im Februar 2012

Inhaltsverzeichnis

1 Konzepte und Funktionalitäten **7**

Marcus Eckert

2 GUI und Benutzerinteraktionen **29**

Alexander Reeb

3 Anwendungsentwicklung für Smartphones - Sicherheitskonzepte **89**

Robert Hebestreit

4 Fernverwaltung und -administration **115**

Stephan Mehner

5 App-Stores **139**

Martin Hille

6 IDEs, Simulatoren, Zertifizierung **159**

Ronny Vogel

7 Universelle Entwicklung **183**

Christian Rolly

8 Smartphone-Apps auf Basis von HTML5 **209**

Christopher Rohde

9 3D-Grafik **225**

Christian Haupt

10 Augmented Reality **251**

Matthias Kuc

Acronyms **275**

Kapitel 1

Konzepte und Funktionalitäten

Marcus Eckert

Thema dieser Seminararbeit ist es eine Grundübersicht über die vier großen Smartphonebetriebssysteme Apple iOS, Google Android, RIM BlackBerry OS und Microsoft Windows Phone zu geben. Dabei wird immer zunächst das Konzept des Betriebssystems vorgestellt, im Anschluss eine exemplarische Versionhistorie aufgezeigt um schließlich zum Schluss nochmals kurz auf die aktuelle Version des Betriebssystems einzugehen.
Im letzten Kapitel sollen dann nochmal alle vier Betriebssysteme miteinander verglichen werden hinsichtlich der Offenheit bzw. Geschlossenheit des Betriebssystems und Systems an sich, auf welche Zielgruppe das Betriebssystem ausgerichtet ist und wo gemeinsame und spezielle Funktionen liegen.
Kein Thema dieser Seminararbeit werden technische Details, im besonderen die grafische Benutzeroberfläche und die App-Stores sein, da diese in anderen Seminararbeiten behandelt werden.

Inhaltsverzeichnis

1.1 Apple iOS 9
1.1.1 Konzept 9
1.1.2 Versionsgeschichte von iOS 9
1.1.3 iOS 5 11
1.2 Google Android 13
1.2.1 Konzept 13
1.2.2 Versionsgeschichte von Android 14
1.2.3 Android 4.0 „Ice Cream Sandwich" 15
1.3 Research in Motion BlackBerry OS 16
1.3.1 Konzept 16
1.3.2 Versionsgeschichte des BlackBerry OS 16
1.3.3 BlackBerry 7 OS 18
1.4 Microsoft Windows Phone 19
1.4.1 Konzept 19
1.4.2 Versionsgeschichte von Windows Phone 19
1.4.3 Windows Phone 7.5 „Mango" 20
1.5 Vergleich 22
1.5.1 offenes/geschlossenes Betriebssystem 22
1.5.2 offenes/geschlossenes System 22
1.5.3 Zielgruppe 23
1.5.4 gemeinsame Funktionen 24
1.5.5 Spezielle Funktionen 24
1.5.6 Verbreitung 25

1.1 Apple iOS

Dieses Kapitel beschäftigt sich mit dem Betriebssystem iOS von Apple, dass vor allem dadurch Verbreitung erfährt, dass es nicht nur auf dem Apple eigenen Smartphone iPhone eingesetzt wird, sondern auch auf dem Tablet iPad und dem PDA iPod touch. Das Teilkapitel Konzept bezieht seine Informationen vor allem aus den Quellen [1], [2], [3], [4] und [5], welche Gleichzeitig die Grundlage für das dritte Teilkapitel bildet. Quellen für das zweite Teilkapitel waren vordergründig Kleinere Seiten und Artikel, da Apple selbst keine Versiongeschichte nachlesbar auf seinem Internetauftritt anbietet.

1.1.1 Konzept

Das iOS Betriebssysten von Apple zielt sowohl auf Endkonsumenten als auch auf Geschäftsleute und öffentliche Einrichtungen ab, dies zeigt sich darin, dass zwar zum einen Unterhaltungsprogramme gefördert werden, gleichzeitig aber auch eine Menge Verwaltungsprogramme von Haus aus auf dem Betriebssystem installiert sind bzw. nachinstalliert werden können. Gerade im Bereich iPad macht Apple gezielt Werbung, dass dieses auch im schulischen oder medizinischen Bereich eingesetzt werden kann.
Desweiteren gehört es zu Apples Philosophie, dass alle Geräte miteinander kommunuzieren können und sich synchronisieren, dazu gehören neben Produkten mit dem iOS auch die Macs, aber auch PCs, die iTunes installiert haben bzw. seit kurzem auch die „iCloud" von Apple. Apples iOS ist dabei ein geschlossenes Betriebssystem und an sich ein geschlossenes System, dies spiegelt sich unter anderem darin wieder, dass es Entwicklern nur möglich ist für das iOS Programme zu entwickeln, wenn diese unter anderem ein Geheimhaltungsabkommen unterzeichnen und einen Mitgleidsbeitrag entrichten. Diese starke Kontrolle hat Für und Wider, so meinen die Einen, die Nichtzulassung fremder Programme sei Zensur, während andere wieder hervorheben, dass Apple durch diese starke Kontrolle auch die Sicherheit vor Schadsoftware erhöht, was ein entscheidendes Kaufkriterium für Kunden sein kann.
Durch die starke Konrolle ist es schwierig für Nutzer an ihrem Gerät herumzubasteln ,was viele als Bevormundung betrachten. Zudem ist es Apple möglich Programme vom iPhone des Nutzer zu löschen. Apple gibt an, dass diese Funktion darauf abzielt Schadsoftware direkt eliminieren zu können. Ein weiterer Kritikpunkt ist, dass sich vorinstallierte Apps nur mittels eines Jailbreaks entfernen lassen, es geht sogar dann nicht, wenn das Programm gar keinen Einfluss auf die Funktionstüchtigkeit des Gerätes hat.
Apple steht auch in der Kritik, weil es keinen Flash Player für das iOS gibt. Hintergrund hierfür ist, dass Flash eine proprietäre Technik von Adobe ist und Apple daher keine Abspielprogramme entwickeln kann. Apple meinte dazu, dass Adobe nicht bereit sei,oder nicht in der Lage ist, eine Lösung für das iOS zu entwickeln, die funktioniert.

1.1.2 Versionsgeschichte von iOS

Das Betriebssystem iOS, bis zum Juni 2010 noch „iPhone OS" genannt, wurde im Januar 2007 erstmal in Verbindung mit dem damals erschienenem iPhone vorgestellt. Es war da-

mals im Prinzip nicht mehr als ein an das iPhone angepasstes „Mac OS X" Betriebsystem von Apple.
Als im Laufe der Zeit das iPhone OS auf immer weiteren Apple Produkten zum Einsatz kam, wie dem iPodtouch und dem iPad wurde das Betriebssystem von Apple in „iOS" umbenannt. Für die Umbenennung war es notwendig, dass Apple die Markenrechte an „iOS" von Ciso Systems kauft. Cisco bezeichnet mit dem Kürzel „IOS" bereits seit den 1980er Jahren ihr „Internetwork Operating System", welches in Switches und Routern zum Einsatz kommt. An dieser Stelle sei angemerkt, dass sogar der Name „iPhone" ursprünglich Cisco Systems gehörte und erst nach einem beigelegten Streit zwischen Apple und Cicso an Apple verkauft wurde. Hier nun eine kurze Versionübersich mit wichtigen

Abbildung 1.1: Hier ein Bild des ersten iPhones [6]

Neuerungen:

- iOS Version 2.x - Diese erschien mit dem iPhone3G im Juli 2008. Wichtigste Neuerung war die Einführung des „App Stores", welcher es Drittanbietern ermöglichte Programme für das iPhone und den iPodtouch zur Verfügung zu stellen.

- iOS Version 3.x - Im Juni 2009 erschien zusammen mit dem iPhone 3GS die dritte Generation des Apple Smartphone Betriebssystems, wichtige Neuerungen waren geforderte Funktionen wie „kopieren" und „einfügen" aber auch Dienste wie „MMS". Nicht alle Funktionen waren für das ursprüngliche iPhone verfügbar, es war aber möglich Systeme, die mit einer 2.x Version von iOS liefen, aufzuwerten.

- iOS Version 4.x - Die vierte Generation von iOS wurde im Juni 2010 veröffentlicht und war die erste Version, die nicht mehr allen Geräten zur Verfügung stand. Es ist auch die erste Veröffentlichung für die iPod-Benutzer nicht zahlen mussten. das iPhone 3G und der iPod touch der 2. Generation können nicht alle Funktionen der damals neuen Version nutzen, während das iPhone 4, iPhone 3GS, iPod touch der 3. und 4. Generation auf alle Funktionen zurückgreifen können. das ursprüngliche iPhone und der iPod touch der 1. Generation können iOS 4.0 und folgende Versionen nicht nutzen.

Die Kompatibilität für das iPad wirde mit der iOS Version 4.2.1. im November 2010 erreicht.
Wichtige Neuerungen waren verbesserte Funktionen für Geschäftsleute und die Möglichkeit, dass Apps von Drittanbietern nun Multitasking nutzen können. Ansonsten viele kleinere Verbesserungen der Programme.

- iOS Version 5.0 - Die iOS Version 5.0 erschien im Juni 2011 und ist mit den Geräten iPhone 3GS, 4 und 4S, iPod touch der 3. und 4. Generation und dem iPad und iPad 2 kompatibel.

1.1.3 iOS 5

iOS 5.0 ist wie bereits im vorherigen Kapitel angesprochen im Juni 2011 erschienen und besitzt eine Reihe neuer Funktionen für das iPhone. Vor allem wird die Verknüpfung mit der iCloud von Apple verbessert. In der 5. Version von iOS wurden einige Funktionen erweitern, wie z.B.:

- Erinnerungen - Hat unter anderem die Möglichkeit Ortsbezogen zu sein, so kann man an einen Einkauf benachrichtigt werden, sobald man in der Nähe eines Ladens ist
- Twitter - ab sofort direkt integriert, eine Verwendung war schon vor iOS 5 möglich
- Kamera - Verbesserung der Funktionalität, so z.B. eine einfachere Einstellung der Fokussierung und Belichtung
- Safari - Verbesserung des Webbrowsers, so ist es ab sofort möglich Tabs zu nutzen
- Erweiterung der Funktionen des Mail Dienstes
- Erweiterung der Funktionen des Kalenders
- Erweiterung der Funktionen des Game Centers
- Funktionen, die die Verwendung von iOS-Geräten durch Behinderte unterstützt

Aber es kamen auch neue Funktionen hinzu:

- neue Multitasking Gesten
- AirPlay Synchronisierung - Möglichkeit eine Bildschirmsynchronisation von iPhone und HD Fernseher drahtlos via Apple TV durchzuführen.
- Wi-Fi Sync - Möglichkeit, dass sich das iOS Gerät, wenn es aufgeladen wird, mittels einer Wi-Fi Verbindung automatisch synchronisiert
- Mitteilungszentrale - Funktion welche es erlaubt einfacher zuletzt eingetroffene Benachrichtigungen zu betrachten

- Ohne PC - ab sofort ist es möglich sein iOS Gerät unabhängig von einem Mac oder PC zu aktivieren und einzurichten, außerdem sind automatische Backups und Wiederherstellung über iCloud möglich
- iMessage - Messaging Dienst, mit dem Nachrichten verschlüsselt übertragen werdenn können
- Zeitungskiosk - Programm zur Verwaltung eigener Abonements für Magazine und Zeitungen

1.2 Google Android

In diesem Katiel soll der Schwerpunkt auf das Betriebsystem Android von Google gelegt werden, welches auf vielen heutigen Smartphones zum Einsatz kommt.
Als Hauptquellen für dieses Kapitel dienten mir [7] und [8]. Als weitere kleine Quellen nutzte ich [9] und [10].

1.2.1 Konzept

Googles Android ist ein Smartphone Betriebsystem und Software-Platform. Es kommt bei Smartphones und Tablets verschiedener Hersteller zum Einsatz, wie z.B. bei

- HTC
- Google
- LG
- Samsung und
- Motorola.

Es ist dabei ein offenes Betriebssystem und an sich offenes System, welches in erster Linie für Endkonsumenten entworfen wurde. Android baut auf dem Linux-Kernel 2.6 auf und steht damit in der Tradition von Linux ein offenes System zu sein, d.h. es nutzt Virtuelle Maschinen und bietet durch seine Offenheit praktisch jedem Nutzer die Möglichkeit eigene Programme für sein Androidbetriebssystem zu schreiben, er benötigt dazu lediglich einen Java-SDK und Android-SDK. Ebenfalls ein Punkt, der die Offenheit von Android verdeutlicht ist die Eigenschaft, dass die unterschiedlichen Smartphonehersteller die Bedienoberfläche von Android ihren Vorstellungen anpassen können und das auch tun. Als Nutzer kann man natürlich auch diese Oberflächen seinen Wünschen gemäß anpassen.
Die Quellcodes der Versionen 1 und 2 sind dabei frei verfügbar, außer der Kernel und einzelne von Google vorinstallierte Programme. Der Quellcode der Version 3 ist nicht veröffentlicht, aber für die aktuelle Version 4 ist eine Veröffentlichung bereits angekündigt.
Von vielen als kritisch beäugt wird die Tatsache, dass bis zur Version 1.5 von Android man ein Google-Konto brauchte um es zu nutzen, dieser Umstand wurde aber mit der Version 1.5 abgeschafft. Nicht abgeschafft wurde die Möglichkeit von Google jedwedes Programm von Smartphones, die Android nutzen, mittels Fernzugriff zu löschen. Als im Juni 2010 ein Sicherheitsexperte zeigen wollte, wie leicht es ist mittels Android Schadsoftware zu verbreiten wurde diese Funktion eingesetzt um das entsprechende Schadprogramm von Smartphones der Nutzer zu löschen. Ein weiterer Nachteil ist, dass durch die Fülle an Smartphoneherstellern, die Android nutzen, nicht immer sichergestellt ist, dass ein Update auf die aktuelle Androidversion möglich ist.

1.2.2 Versionsgeschichte von Android

Die erste Version von Android erschien im Oktober 2008, dem vorausgegangen war, dass Google 2005 das 2003 gegründete Unternehmen Android kaufte, von welchem kaum mehr bekannt war, als dass es Software für Mobiltelephone entwickelte. Desweiteren gab Google im November 2007 bekannt, gemeinsam mit der Open Handset Alliance, ein Konsortium von 80 Firmen, die das Ziel hat offene Standards für Mobilgeräte zu schaffen, ein Smartphone-Betriebssystem namens Android zu entwickeln. Das erste Smartphone, welches das Androidbetriebssystem nutzte war das HTC Dream. Mit dem Erscheinen dieses Smartphones war auch der Android Market direkt verfügbar. Bereits dieses erste

Abbildung 1.2: Hier ein Bild des ersten Android-Smartphones, das HTC Dream

Gerät konnte auf das Global Positioning System zugreifen und verfügt über einen Bewegungssensor, diese beiden Dinge gehörten bereits in der ersten Version von Android zum Standard. Die Versionen 1 und 2 von Android sind dabei Smartphonebetriebsysteme, während Version 3 ein Tabletbetriebsystem ist. Erst die Version 4 vereinigt diese beiden Stränge miteinander. Eine weitere typische Eigenschaft der Anroid-Versionen ist, dass diese neben der Versionsnummer einen Namen erhalten. Im folgenden seien die wichtigsten Android-Versionen bis zur Version 4.0 mit ihren wesentlichsten Neuerungen kurz vorgestellt.

- 1.5 „Cupkake" - Neu in dieser Version waren die Bildschirmtastatur, der automatische Wechsel zwischen Hoch- und querformat und die Aufnahme und Wiedergabe von Videos

- 1.6 „Donut" - Ab dieser Version waren VPNs konfigurierbar, es gab eine differenzierte Energieverbrauchssteuerung, Gestenerkennung und mehr als eine Bildschirmauflösung

- 2.0 und 2.1 „Eclair" - Neu waren die IPv6 Unterstützung, Informationen zu Signalstärken, Erweiterungen von Webkit und die Unterstützung von Microsoft Exchange.

- 2.2-2.2.2 „Froyo" - Der Linux-Kernel 2.6.32 wurde ab dieser Version genutzt, da er weniger Arbeitsspeicher benötigt. Gleichzeitig wurde mehr als die bisher 256MB Arbeitsspeicher genutzt. Desweiteren waren Apps ab sofort auf SD-Karten speicherbar.

- 2.3-2.3.7„Gingerbread" - Neben der Nutzung des Linuskernes 2.6.35.7 wurde die Funktionalität stark erweitert, so wurden ab sofort HTML5 Audio, Google TV, das Ext4-Dateisystem und Gyroskope unterstüzt. Außerdem gab es einen integrierten SIP-Client für VoIP und einen integrierten Downloadmanager. In kleineren Aktualisierungen kam eine Dual-Core-Unterstützung und eine verbesserte Bluetooth-Unterstützung hinzu.
- 3.0-3.2.1 „Honeycomb" - Das auf Tablets spezialisierte Betriebsystem verbesserte vor allem die Benutzerfreundlichkeit der Bedienoberfläche. Außerdem wurden einige Funktionen an die größeren Bildschirme der Tablets angepasst. Die Aktualisierungen dieser Version verbesserten das Betriebssystem immer mehr für die Tabletnutzung.

Schon die jetzige Planung sieht vor die Version 5.0 von Android den Namen „Jelly Bean" zu geben, diese ist bisher für 2012 geplant.

1.2.3 Android 4.0 „Ice Cream Sandwich"

Diese Version erschien im Oktober 2011. Die erste wesentliche neue Funktion dieser Version ist die Zusammenführung des Smartphonebetriebssystems 2.x und des Tabletbetriebssystems 3.x. Desweiteren gibt es Systemverbesserungen und neue Funktionen. Es wurde unter anderem verbessert:

- das Multitasking
- der Browser, sowie Googel-Mail, Galerie und Kalender-App
- Kopieren-Funktion
- Fehlerkorrektur mit der Tastatur

Komplett neu dagegen ist

- Entsperren mittels Gesichtserkennung
- Nutzung von Wi-Fi Direkt, ist eine WLAN-Verbindung zwischen 2 Geräten, ohne dass ein AP vorhanden sein muss
- NFC für Kontakte, Daten und Links mit „Android Beam"
- Neues Galerie-Layout

Als technische Voraussetzung ist lediglich angegeben, dass das Smartphone bzw. Tablet über 256MB Arbeitsspeicher und einen „anständigen" Grafikprozessor verfügen muss. Die Entwickler von Android 4.0 beabsichtigen es aber auch für schwächere Smartphones verfügbar zu machen, dann aber natürlich mit eingeschränken Funktionen.

1.3 Research in Motion BlackBerry OS

In diesem Kapitel soll das Smartphonebetriebssystem BlackBerry OS behandelt werden, dass vor allem im geschäftlichen Bereich und dem amerikanischen Markt Verwendung findet.
Hauptquelle für diese Kapitel war eindeutig [11], desweiteren waren [12], [13] und [14] von Relevanz.

1.3.1 Konzept

Das BlackBerry OS ist ein geschlossenes Betriebssystem und geschlossenes System, welches ausschließlich für Smartphones ausgelegt ist. Es ist dabei weniger auf den „normalen" Endkonsumenten ausgerichtet, sondern mehr in die Richtung der Geschäftswelt, dies wird dadurch deutlich, dass der Email-Dienst besonders stark unterstützt wird, genauso wie Anwendungen, die die Firmenverwaltung unterstützen. So bietet BlackBerry sogar direkt den Firmen an, mit ihnen zusammen Unternehmenslösungen zu finden, was Datensynchronisierung und Einführung von BlackBerry-Geräten angeht. Aber auch Privatkunden haben natürlich die Möglichkeit diese Smartphones bzw. dieses Smartphonebetriebssystem zu nutzen, da das BlackBerry OS zwar im Schwerpunkt auf Firmen ausgerichtet ist, aber trotzdem über die üblichen Smartphonefunktionen wie Telephonie, SMS, Kalender, Adressbuch etc. verfügt. Eine der Vorteile dieses Betriebssystem ist, dass eine automatische Synchronisation der Daten stattfindet, entweder über Firmeninterne Server mit BlackBerrysoftware oder über „öffentliche" BlackBerryserver. Typisch für die BlackBerry Smartphones ist eine QWERTZ-Tastatur und ein Trackball bzw. bei neueren Modellen ein Trackpad, welche die Einhandbedienung erleichtern. Bei den Smartphones der neueren Generation kommen auch Touchpads zum Einsatz. Um Programme für das BlackBerry OS zu schreiben benötigt man lediglich Java und HTML5+ WebWorks.
Eines der wesentlichsten Probleme bei BlackBerry ist der Datenschutz, dies kommt durch die starke Ausrichtung auf den amerikanischen Markt. Im speziellen geht es um die Technik der Emailkommunikation. In den USA ist es üblich die Mails der Mitarbeiter zu überwachen, da die Firmen verantworlich für die Inhalte ihrer Leute sind. Diese Überwachungsmöglichkeit ist natürlich bei uns in Europa und besonders in Deutschland Datenschutzrechtlich bedenklich, da es durch diese Funktion möglich ist, dass Dritte an Informationen kommen, die für sie nicht bestimmt sind. Deswegen wird prinzipiell beim Einsatz von BlackBerrys empfohlen, niemals private Nachrichten oder ähnliches zu senden.
Ebenfalls ein Nachteil des BlackBerry OS ist, dass die Verkäufer der Smartphones entscheiden, ob und wann eine neuere Version des Betriebssystems an die Kunden weitergeben wird.

1.3.2 Versionsgeschichte des BlackBerry OS

Im Januar 1999 erschien die erste Version des BlackBerry OS mit dem BlackBerry 580, zum damaligen Zeitpunkt konnte man aber noch nicht wirklich von einem Smartphone-

betriebssystem reden. Es war damals ein Betriebssystem für Pager.
Erst die Version 3.6 vom März 2002 war ein Smartphonebetriebssystem, für das „BlackBerry 5810 Wireless Handheld". Seitdem wurde das Betriebssystem kontinuierlich weiterentwickelt, jeweils passend zu den aktuellen BlackBerry Smartphones.
Die „BlackBerry App World" steht den BlackBerrynutzern seit April 2009 zur Verfügung, zur Benutzung wird allerdings eine BlackBerry-ID benötigt. Im folgenden die wichtigsten Versionen mit ihren Neuerungen:

- 4.3 - Zwei wesentliche Neuerungen waren zum einen die Möglichkeit Sprachmemos aufzunehmen und diese zu verschicken und zum anderen die Abspielmöglichkeit von Flash-Videos.
- 4.5 - Ab dieser Version war Audio- und Videostreaming möglich, desweiteren gab es nun HTML Email Unterstützung.
- 4.7 - Hier wurde vor allem die Benutzeroberfläche verbessert, so wurde die „kopieren und einfügen"-Funktion genauso wie der Webbroser in seiner Handhabung verinfacht.
- 5.0 - Wesentliche Verbesserung dieser Version war die Geschwindigkeit, so liefen alle Programme nun schneller und flüssiger.
- 6.0 - Neben einer neuen Benutzeroberfläche standen im Browser nun Multitasking, Tabs und Favoriten zur Verfügung. Desweiteren wurden neue Animationen eingeführt, genauso wie Pop-Up-Menüs und Multi-Touch-Unterstützung.

RIM beabsichtigt die Entwicklungslinie des BlackBerry OS mit BlackBerry 7 OS zu beenden. Folgen soll ein Betriebssystem, welches sowohl auf den BlackBerry Smartphones als auch um den Tablets zum Einsatz kommen soll. Der Name für dieses neue Betriebssystem wird „BlackBerry BBX" sein. Laut Medienberichten soll es sogar möglich sein nicht nur Programme der BlackBerry App World, sondern auch Programme des Android Market auf diesem Betriebssystem zu installieren.
Zum Abschluss noch eine kurze Übersich über die aktuell angebotenen Smartphones von BlackBerry

Abbildung 1.3: Hier eine Auswahl aktueller BlackBerrymodelle [11]

1.3.3 BlackBerry 7 OS

Die aktuelle, im August 2011 erchienene und wie im vorigen Kapitel bereits angedeutet, letzte Version des BlackBerry OS versucht vor allem verlorenen Boden auf dem Smartphone-Markt wiederzugewinnen.
Die 7. Version des Betriebssystem enthält Verbesserungen von Programmen wie:

- bessere Grafik
- schnellerer Brwoser

Genauso wie die Einführung neuer Funktionen wie:

- Augmented Reality - Beinhaltet beispielsweise die Möglichkeit, Informationen über Restaurants abzurufen, an denen man gerade vorbeiläuft
- NFC-Technologie - „Near Field Communication" entspricht im Grunde der Idee von RFIDs, nur dass es schneller und platzsparender ist
- HD-Videounterstützung
- Speech-to-Text Funktion

Die Liste wirkt zugegebener Maßen recht kurz, dies liegt aber vor allem daran, dass BlackBerry bereits vor der Veröffentlichung dieser Aktualisierung den Entschluss gefasst hatte seine Bemühungen auf das neue Betriebssystem „BlackBerry BBX" zu konzentrieren.

1.4 Microsoft Windows Phone

Windows Phone ist das Smartphone Betriebssystem von Microsoft und soll in diesem Kapitel beleuchtet werden. Meine wichtigsten Quellen waren dabei [15], [16] und [17].

1.4.1 Konzept

Windows Phone ist ein offenes Betriebssystem, aber ein geschlossenes System, d.h. dass es zwar auf vielen Smartphones eingesetzt werden kann, aber der Quellcode nicht offen ist. Es war Anfangs vor allem auf Geschäftsleute ausgerichtet, während mit Windwos Phone 7 der Versuch gestartet wurde auch Endkonsumenten anzusprechen, es verfügt demzufolge über die typischen Smartphone-Funktionen wie Telephonie, Kalender, Adressbuch etc.. Smartphonehersteller, die Windows Phone auf ihren Smartphones anbieten, sind z.B.

- HTC
- LG
- Samsung

Im Februar 2011 hat Microsoft einen Vertrag mit Nokia geschlossen, der Windows Phone zum Hauptbetriebssystem der Nokia-Smartphones machen soll, trotzdem soll das Betriebssystem auch auf Smartphones anderer Hersteller zum Einsatz kommen. Für die Nutzung des Windows Market-Places ist eine Windows Live-ID erforderlich.
Kritikpunkte an Windows Phone sind z.B., dass keine direkte Synchronisation mit Outlook via USB oder Bluetooth möglich ist, außerdem wird der interne Speicher der Smartphones von PCs nicht als Massenspeicher erkannt.

1.4.2 Versionsgeschichte von Windows Phone

Windows Phone heißt bis Windows Phone 7 „Windows Mobile", wobei Die Windows Mobile Versionen Weiterentwicklungen waren während Windwos Phone 7 eine komplette Neuentwicklung war, es gibt auch kein Update von Windows Mobile auf Windows Phone 7. Die erste Version von Windows Mobile erschien 2002 mit „Windows Mobile 2002". Es basierte auf Windows CE 3.0 und wurde gezielt für Tastaturlose Pocket PCs mit QVGA Auflösung entworfen, es war auch das einzige Microsoft-Betriebssystem für eingebettete Systeme.
Der Windows Phone Marketplace erschien zusammen mit Windows Phone 7 im Oktober 2010. Im folgenden seien ein paar der letzten Windows Mobile Versionen des Smartphonebetriebssystem mit Neuerungen, die sie brachten, aufgelistet:

- Windwos Mobile 6.0 - Wichtigste Neuerungen waren IP-Telephonie, HTML-Unterstützung für Mails und eine integrierte Updatefunktion mit einer Optimierung für den Abgleich mit Windows Vista.

- Windwos Mobile 6.5-6.5.3 - Ab dieser Version gab es kinetisches scrollen, Multitouchsupport, sowie eine anpassbare Benutzeroberfläche.
- Windwos Phone 7 - Neu war vor allem das Konzept, eher auf Endkonsumenten ausgerichtet zu sein. So wurde besonders auf eine intuitive Benutzeroberfläche, welche Multitouch- und Gestensteuerung unterstützt, Wert gelegt. Gleichzeitig fiel aber die anpassbare Benutzeroberfläche weg.

1.4.3 Windows Phone 7.5 „Mango"

Das Windows Phone 7.5, dessen Code- und Entwicklungsname „Mange" ist, bringt natürlich wieder ein paar verbesserte Funktionen und auch Neue.
Verbesserte Funktionen sind:

- Internetexplorer 9 mit HTML5
- verbesserte Emailsuche
- Kommunikation mit einem Kontakt nun zentral und gebündelt dargestellt
- Suchmaschine Bing wurde verbessert
- neue Sprachbefehle

Neue Funktionen dagegen sind:

- Unterstützung von Dual-Cores
- SMS Diktat
- individuelle Klingeltöne sowie sonstige Töne abstellbar
- ausschaltbare Kamera
- Unterstützung einer Frontkamera für Videotelephonie
- Multitasking an sich und für Drittanbieterprogramme
- Keine Stimmenbefehle mehr möglich, wenn die Sperre im Gerät aktiviert ist.
- Augmented Reality

Um Windows Phone 7.5 auf seinem Smartphone nutzen zu können stellt es eine Reihe von Anforderungen an das System, diese wären:

- 256MB Arbeitsspeicher
- 8GB Festplattenspeicher

- Kompass
- GPS-Empfänger
- Beschleunigungssensor
- Lagersensor
- Multi-Touchscreen
- mind. 5 Megapixel-Kamera
- Mobilfunkschnittstelle
- WLAN-Schnittstelle
- Prozessor mit 1GHz und DirectX9-Unterstützung

Desweiteren sind nur 3 Gehäuseformen möglich:

- reines Touchscreen-Mobiltelephon
- Touchscreen-Mobiltelephon mit ausklappbarer QWERTZ-Tastatur
- klassische Riegelform mit Touchscreen

Abbildung 1.4: Die zulässigen Gehäuseformen für Windwos Phone 7.5

Es ist sehr auffällig, dass Microsoft von den Smartphones auf denen Windows Phone 7.5 laufen soll hohe Anforderungen stellt. Dies liegt aber vor allem daran, dass alle Möglichkeiten an Funktionen ausgeschöpt werden sollen und es keine Einschränkungen in der Verwendung geben soll, die durch eventuelle Minderleistung des Smartphones entstehen könnten.

1.5 Vergleich

Dieses Kaptiel bringt im wesentlichen keine neuen Informationen sondern vergleicht die vier vorgestellten Betriebssystem hinsichtlich Offenheit/Geschlossenheit des Betriebssystems und Systems, welche Zielgruppen die Betriebssysteme haben und was gemeinsame und spezielle Funktionen sind und fasst die Informationen zusammen. Im letzten Kapitel wird dann noch auf die aktuelle Verbreitung der vier Betriebssysteme eingegangen und bringt damit Informationen, die in den bisherigen Kapiteln noch nicht genannt wurden.

1.5.1 offenes/geschlossenes Betriebssystem

Offene und geschlossene Betriebssysteme unterscheiden sich dadurch, dass ein offenes Betriebssystem dafür gemacht ist auf vielen unterschiedlichen Geräten zu laufen, während ein geschlossenes Betriebssystem nur für ein oder wenige bestimmte Geräte vorgesehen ist. Zwar kann es durchaus möglich sein ein geschlossenes Betriebssystem auch auf anderen Geräten zum laufen zu bringen, es ist aber im Allgemeinen mit Leistungseinbußen zu rechnen.
In diesem Kontext wird klar, dass Apples iOS und das BlackBerry OS eindeutig geschlossene Betriebssysteme sind, da sie nur für die Firmeneigenen Smartphones bzw. bei Apple auch iPodtouch und iPad vorgesehen sind.
Dem stehen ganz klar Googles Android und Microsofts Windwos Phone gegenüber, wobei es auch hier nochmal Unterschiede gibt.
Während Googles Android nur wenig Anforderungen an das Gerät stellt und ggf. einfach weniger Funktionen bietet, falls das Gerät zu schwach ist oder bestimmte Sensoren nicht verbaut sind (siehe Vrgl. Kapitel 2) geht Microsoft hier einen anderen Weg. Microsoft sagt ganz klar, welche Funktionen, Sensoren und Leistungsparameter vorhanden sein müssen, dass Microsoft einer offiziellen Verwendung seines Windows Phone 7 Betriebssystems auf dem Smartphone zustimmt. Natürlich ist es möglich es auf anderen Smartphones zu installieren, aber will es ein Anbieter von „Haus aus" auf einem seiner Smartphones anbieten, so muss die lange Liste der Anforderungen von Microsoft erfüllt sein.

1.5.2 offenes/geschlossenes System

Ein durchaus verbreiteterer Begriff für „offenes System" ist „open Source", was soviel bedeutet, dass der Quellcode, oder zumindest große Teile davon, der Öffentlichkeit zugänglich sind und damit auch von Privatleuten ohne weiteres an die eigenen Bedürfnisse anpassbar. Vorteile solcher Systeme sind ganz klar, dass jeder nachvollziehen kann, wie die Programme arbeiten und evtl. Hintertüren im System erkennbar sind. Außerdem ist es für jedermann leichter Veränderungen am System vorzunehmen bzw. sogar als Firmen-Externer auf Lücken und Fehler hinzuweisen bzw. das Programm zu verbessern.
Dem gegenüber stehen die geschlossenen Systeme, die dadurch gekennzeichnet sind, dass quasi kein Quellcode der Öffentlichkeit zur Verfügung steht. Hier kann als Vorteil betrachtet werden, dass nur wenige das System wirklich kennen und es ausnutzen können, andererseits kann man nie Sicher sein, was die Entwickler alles für versteckte Funktionen

eingebaut haben, da diese durch die Geschlossenheit wesentlich schwerer ausfindig zu machen sind.
Vor diesem Hintergrund wird schnell klar, dass das einzige wirklich als offen zu bezeichnende System Googles Android ist, da hier der Großteil des Quellcodes bereits veröffentlicht ist, oder es, wie bei der aktuellen Version 4.0 „Ice Cream Sandwich", bereits geplant ist den Quellcode zu veröffentlichen.
Ganz klar auf der anderen Seite stehen Apples iOS, BlackBerrys OS und auch Microsofts Windwos Phone, wobei Apples die größten Anstrengungen unternimmt, dass keine Informationen über den Quellcode nach außen dringen bzw. allgemein seine Nutzer und Entwickler sehr stark einschränkt, was eigenständige Abwandlungen und Änderungen am System betrifft.

1.5.3 Zielgruppe

Im wesentlichen unterscheidet man, ob Firmen als Hauptzielgruppen Geschäftsleute oder den Endkonsumenten haben. Der wesentliche Unterschied liegt dabei darin, inwiefern bestimmte Funktionen unterstützt werden. Bei Geschäftsleuten steht prinzipiell der E-Maildienst im Vordergrund, da dies in Firmen einer der Hauptkommunikationswege ist. Gleichzeitig ist es förderlich, wenn die Smartphonehersteller Unternehmensangepasste Lösungen für die Verwendung ihrer Smartphones bieten, sei es einen eigenen Server, oder auf die Firma angepasste Verwaltungsprogramme. Im Gegensatz dazu dienen Smartphones für den Endkonsumenten vor allem als Unterhaltungsmedium und Selbstverwalter, demzufolge sind Funktionen wichtig, die Spiele ermöglichen, Messenger-Dienste bereitstellen, Audio und Videowiedergabe unterstützen, sowie Erinnerungs- und Termin-Funktionen, wobei die beiden zuletzt Genannten auch für die Geschäftsleute eine große Bedeutung haben.
Unter Berücksichtigung dieser Punkte wird schnell klar, dass BlackBerry eindeutig die Geschäftsleute als Zielgruppe hat, BlackBerry selbst steht für einen hervorragenden E-Maildienst, auch wenn gerade dieser in letzter Zeit ein wenig in Verruf geraten ist, und bietet Firmen die vorhin beschriebenen Unternehmenslösungen genauso wie Firmenverwaltungsprogramme.
Android ist dagegen ganz klar auf den Endkonsumenten ausgerichtet, von Anfang an waren bei diesem Betriebssystem Unterhaltungsmöglichkeiten aller Art vorgesehen. Auch sind einige Smartphones mit Android-Betriebssystem im niedrigen Preissegment angesiedelt um besonders Privatleute anzusprechen, die nicht einen ganzen Haufen Geld in lediglich ein Telephon investieren wollen.
Ein „Zwischending" bilden iOS und Windwos Phone, bei Windwos Phone aber erst seit der Version 7, zuvor war auch hier die Hauptzielgruppe die der Geschäftsleute. Diese beiden Bestriebssysteme bieten neben den Funktionen für Geschäftleute, ausgeprägter E-Maildienst und Firmenverwaltungsprogramme, auch die typischen Unterhaltungsfunktionen, die vor allem der Privatkunde nutzt.

1.5.4 gemeinsame Funktionen

Heuzutage haben alle Smartphones, unabhängig des genauen Betriebssystem Funktionen, die bei jedem vorkommen, dabei haben sie aber häufig unterschiedliche Namen obwohl die Funktionalität nahezu identisch ist. Hier ein paar Beispiele:

- Kommunikation - dazu zählen neben Telephonieren, SMS und MMS auch Instantmessenger und Videotelephonie
- E-Mail-Programm - vor allem für Geschäftsleute eine der elementarsten Funktionen
- Browser - ganz klar eine der meistgenutzten Funktionen in unserer Gesellschaft, wo quasi überall Internet verfügbar ist
- Adressbuch - eine Funktion, die schon bei klassischen Mobiltelephonen zur Verfügung stand
- Photoapparat und Videos aufnehmen - die Kamera ist in heutigen Smartphones auch mit vielen anderen Funktionen verknüpft, wie Videotelephonie
- Musik und Videos anschauen - vor allem die Funktion des Musik abspielens ist eine gern genutzte Funktion
- Erinnerungen und Notizzettel - Eine Funktion die auch dadurch unterstützt wird, dass man diese Erinnerungen auf anderen Geräten synchronisieren kann
- Uhr - Ein Klassiker, der manche Leute veranlasst keine Armbanduhr mehr zu tragen
- Rechner - gemeint ist damit ein kleiner Taschenrechner
- Multitasking - hat vor allem in den letzteren Versionen einzug gehalten, da der Nutzer dies heutzutage von seinem Rechner zuhause gewohnt ist

1.5.5 Spezielle Funktionen

Wie bereits im letzten Teilkapitel erwähnt, sind viele Funktionen sehr ähnlich bzw. identisch. Die Unterschiede der Funktionen sind meist nur minimal und temporär. So gibt es Versionabhängig meist ein Betriebssystem welches in einem Bereich Vorreiter ist, aber nach einer gewissen Zeit ziehen die anderen Betriebssysteme nach. Als Beispiel dafür kann genannt werden, dass iOS als erstes für Smartphones als auch Tablets verfügbar war. Android hat mit seiner Version 3.x ebenfalls ein Tabletbetriebssystem zur Verfügung gestellt bzw. mit seiner Version 4.0 „Ice Cream Sandwich" dafür gesorgt, dass es nun wie bei Apple eine Betriebssystemversion gibt, welche für Smartphones und Tablets nutzbar ist. BlackBerry und Microsoft planen dies für ihre nächsten Versionen ihrer Smartphonebetriebssysteme.
Ein anderes Beispiel ist, dass Android in seiner aktuellen Version als einziges die Möglichkeit bietet eine Entsperrung des Smartphones mittels Gesichtserkennung zu ermöglichen. Es bleibt abzuwarten wie lange es dauert, bis die anderen Betriebssysteme diese Funktion

bei sich ergänzen.
Ebenfalls ein Umstand, der in diese Beschreibung passt, ist das fast gleichzeitige auftauchen der „NFC"-Funktion in allen Betriebssystemen, siehe dazu die letzten vier Kapitel. Zu erklären ist das natürlich ganz einfach damit, dass kein Betriebssystem den anderen hinterher sein möchte.

1.5.6 Verbreitung

In der beigefügten Statistik, der die Verkaufszahlen an Endkunden zugrunde liegen, kann man gut erkennen, dass der Anteil an Android-Smartphones stark gestiegen ist, während RIM mit seinem BlackBerry OS einen immer kleiner werdenden Marktanteil zu verzeichnen hat. Desweiteren kann man sehen dass der Anteil an iPhones mit dem iOS Betriebssystem immer mehr steigt. Microsofts Windows Phone spielt bisher keine große Rolle, auch ist ungewiss wie sich die Kooperation mit Nokia auf diese Entwicklung auswirkt, so ist eine Vermutung, dass Windows Phone den bisher noch durch Symbian vertretenen Anteil schluckt. Es ist aber eindeutig ein Trend hin zu Android bzw. auch iOS und weg von BlackBerry zu erkennen, ob BlackBerry diesen Trend zu wenden vermag ist ebenfalls fragwürdig.

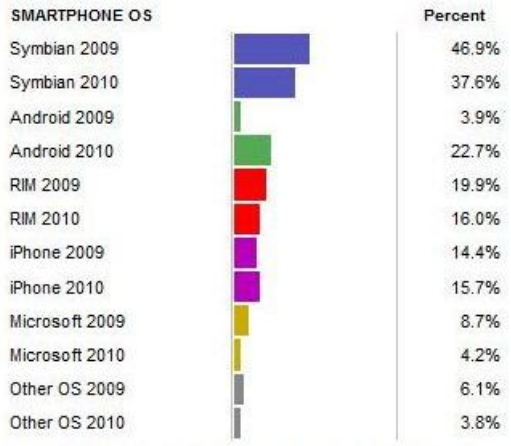

Abbildung 1.5: Verkaufszahlen von Smartphones 2009 und 2010 [18]

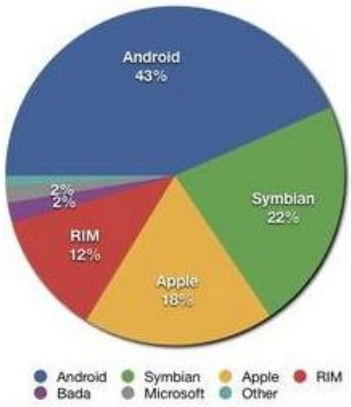

Abbildung 1.6: Verkaufszahlen von Smartphones im 2.Quartal 2011 [19]

Literaturverzeichnis

[1] https://www.eff.org/deeplinks/2010/03/\iphone-developer-program-license-agreement-al
Inhalt: begrenzte Nutzerrechte bei iPhonebenutzung, zuletzt abgerufen am 15.11.2011.

[2] http://gizmodo.com/5633721/apple-to-allow-other-iphone-development-tools-publishes-a
Inhalt: begrenzte Nutzerrechte bei iPhonebenutzung, zuletzt abgerufen am 15.11.2011.

[3] http://www.npr.org/templates/story/story.php?storyId=125561844
Inhalt: evtl. Sicherheitsvorteile des iPhones, zuletzt abgerufen am 15.11.2011.

[4] http://apple.slashdot.org/story/10/01/31/1657233/
Apples-Trend-Away-From-Tinkering
Inhalt: Bedenken bzgl. Apples Bedienkonzept, zuletzt abgerufen am 15.11.2011.

[5] http://www.apple.com/de/
Inhalt: Apples Internetauftritt, zuletzt abgerufen am 15.11.2011.

[6] http://support.apple.com/kb/HT3939?viewlocale=de_DE
Inhalt: iPhone-Identifikation, zuletzt abgerufen am 15.11.2011.

[7] http://www.android.com/
Inhalt: Android-Internetauftritt, zuletzt abgerufen am 15.11.2011.

[8] http://developer.android.com/index.html
Inhalt: Seite für Entwickler für Android, zuletzt abgerufen am 15.11.2011.

[9] http://www.businessweek.com/technology/content/aug2005/tc20050817_
0949_tc024.htm
Inhalt: Google kauft Android, zuletzt abgerufen am 15.11.2011.

[10] http://news.cnet.com/8301-17939_109-9810937-2.html
Inhalt: Google startet Smartphone-Entwicklung, zuletzt abgerufen am 15.11.2011.

[11] http://de.blackberry.com/
Inhalt: BlackBerry-Internetauftritt, zuletzt abgerufen am 15.11.2011.

[12] http://www.wienerzeitung.at/themen_channel/wz_digital/digital_news/
388480_London-Riots-Hacker-griffen-Blackberry-Homepage-an.html
Inhalt: Kritik am BlackBerry-System, zuletzt abgerufen am 15.11.2011.

[13] http://www.tagesschau.de/wirtschaft/blackberry110.html
Inhalt: Kritik am BlackBerry-System, zuletzt abgerufen am 15.11.2011.

[14] http://www.spiegel.de/netzwelt/web/0,1518,708673,00.html
Inhalt: Kritik am BlackBerry-System, zuletzt abgerufen am 15.11.2011.

[15] http://www.focus.de/digital/handy/handyvergleich/tid-20231/windows-phone-sieben-fehler-die-kritik-an-windows-phone-7-im-detail_aid_565551.html
Inhalt: Kritikpunkte bei Windwos Phone 7, zuletzt abgerufen am 15.11.2011.

[16] http://www.microsoft.com/windowsphone/de-DE/features/default.aspx
Inhalt: Microsoftseite über Windows Phone 7.5, zuletzt abgerufen am 15.11.2011.

[17] http://windowsteamblog.com/windows_phone/b/windowsphone/
Inhalt: Windwos Phone Blog, zuletzt abgerufen am 15.11.2011.

[18] http://www.mobilephonedevelopment.com/archives/1149
Inhalt: Verkaufszahlen der Smartphones und deren Betriebssysteme, zuletzt abgerufen am 15.11.2011.

[19] http://www.gartner.com/it/page.jsp?id=1764714
Inhalt: Verkaufszahlen der Smartphones und deren Betriebssysteme, zuletzt abgerufen am 15.11.2011.

Kapitel 2

GUI und Benutzerinteraktionen

Alexander Reeb

Diese Ausarbeitung befasst sich mit der Gestaltung der grafischen Benutzeroberflächen und des Bedienkonzeptes von Smartphones. Es werden die vier am weitesten verbreiteten Geräte bzw. Systeme untersucht, diese sind: Black Berry (RIM), Android (Google), Windows Phone (Microsoft) und iPhone (Apple).
Zu Beginn werden in Abschnitt 2.1 die Grundlagen für die folgenden Beobachtungen geschaffen. Es wird zunächst der Begriff GUI definiert und das Thema motiviert, d.h. warum ist die Gestaltung der GUI für mobile Geräte eine besondere Herausforderung und worin bestehen die Probleme bei dem Design von Anwendungen.
Nach den Grundlagen werden die jeweiligen Geräte: Black Berry (Abschnitt 2.2), Android (Unterabschnitt 3.4.1), Windows Phone (Abschnitt 2.4), iPhone (Abschnitt 2.5) einzeln untersucht.
Die Untersuchung ist jeweils gleich aufgebaut. So werden zunächst die allgemeinen Anforderungen der Hersteller an Anwendungen aufgeschlüsselt. Danach wird das Bedienkonzept untersucht, d.h. wie wird das Gerät bedient und welche Möglichkeiten der Interaktion bzw. welche Hardwaretasten hat das Gerät. Im Anschluss werden die Möglichkeiten zur Strukturierung der GUI untersucht und die einzelnen GUI Elemente vorgestellt. Es wird hierbei auch erläutert welche Anforderungen und Beschränkungen die Hersteller für die Verwendung der einzelnen Elemente machen.
Zum Abschluss jedes Gerätes, wird zusammengefasst welche Möglichkeiten dem Entwickler geboten werden und wie stark die Einschränkungen und Vorgaben sind. Ebenso werden die persönlichen Erfahrungen bei der Analyse der jeweiligen Guidelines dargestellt.
In Abschnitt 2.6 erfolgt der Vergleich der einzelnen Geräte auf Basis der zuvor angestellten Untersuchungen. Zunächst werden die Herausstellungsmerkmale der einzelnen Geräte aufgeführt und erläutert. Danach folgt eine Übersicht, welche Elemente bei den verschiedenen Geräten verwendet werden.
Zum Schluss folgt in Abschnitt 2.7 ein Fazit wie die Hersteller die Herausforderungen bei der Gestaltung der GUI gemeistert haben und eine Prognose wie sich die GUI in den nächsten Jahren entwickeln könnte.

Inhaltsverzeichnis

2.1 Grundlagen 31
2.1.1 Definition GUI 31
2.1.2 Motivation 32
2.2 BlackBerry (RIM) 33
2.2.1 Allgemeine Anforderungen 33
2.2.2 Bedienkonzept 34
2.2.3 GUI Strukturierung 36
2.2.4 GUI Elemente 38
2.2.5 Themes 42
2.2.6 Zusammenfassung 42
2.3 Android 43
2.3.1 Allgemeine Anforderungen 43
2.3.2 Bedienkonzept 45
2.3.3 GUI Strukturierung 46
2.3.4 GUI Elemente 48
2.3.5 Companion Widgets 53
2.3.6 Zusammenfassung 54
2.4 Windows Phone 55
2.4.1 Allgemeine Anforderungen 55
2.4.2 Bedienkonzept 57
2.4.3 GUI Strukturierung 59
2.4.4 GUI Elemente 63
2.4.5 Zusammenfassung 67
2.5 iPhone 68
2.5.1 Allgmeine Anforderungen 68
2.5.2 Bedienkonzept 70
2.5.3 GUI Strukturierung 73
2.5.4 GUI Elemente 76
2.5.5 Zusammenfassung 81
2.6 Vergleich 82
2.6.1 Besonderheiten 82
2.6.2 Gemeinsamkeiten 83
2.7 Fazit 84

2.1 Grundlagen

In diesem Abschnitt werden zunächst die Grundlagen für die weitere Ausarbeitung geschaffen. Zu Beginn wird der Begriff der GUI definiert. Dies bildet den Einstiegspunkt für die folgende Betrachtung der vier mobilen Geräte bzw. Betriebssysteme: Android, Black Berry, Windows Phone und iPhone, welche in dieser Arbeit auf die GUI Konzepte und Vorgaben der Hersteller untersucht werden. Im Anschluss folgt eine kurze Motivation für die Thematik der GUI Gestaltung und warum dies vor allem auf mobilen Geräten eine besondere Herausforderung darstellt.

2.1.1 Definition GUI

Es gibt bzgl. des Begriffs GUI viele Definitionen die jedoch im Kern alle die gleichen Punkte behandeln. Die folgende Definition deckt diese weitestgehend ab:
Die graphische Benutzeroberfläche (engl. „Graphical User Interface“, kurz GUI) ist ein Bestandteil von Betriebssystemen und Anwendungsprogrammen, die dem Benutzer eine leichtere Bedienung der Software ermöglichen soll. Im Gegensatz zur Kommandozeile kommt dabei nicht nur die Tastatur, sondern auch ein Zeigegerät wie eine Computermaus oder ein Touchscreen zum Einsatz. GUIs besitzen graphisch, metaphorische Elemente wie Fenster, Menüs und Symbole (Icons) und sind so für den Nutzer intuitiv bedienbar.[1][2]
Man sieht, dass die GUI als Mensch-Maschine Schnittstelle dient. Sie ist die Weiterentwicklung zu CLI („command line interface“) bei dem man durch Eingabe von Befehlen (Text) mit dem Computer interagiert, wie man dies z.B. von MS-DOS kennt. Ziel der GUI ist es, durch die Benutzung möglichst wirklichkeitsnaher grafischer und motorischer Metaphern dem Benutzer ein realitätsnahes Arbeitsumfeld darzubieten.[3] Durch die graphische Aufbereitung ergeben sich viele Vorteile für den Nutzer. Dies wird vor allem daran deutlich, dass die GUI ergonomische Anforderungen erfüllt, wie:

- **Entspanntes Arbeiten**
  - *Kontrolle beim Benutzer* statt beim Programm
  - Gedächtnis entlasten: *Browsing* statt Kommandos
  - Fehlertoleranz: *Undo*-Operationen
- **Menschengerechte Darstellung von**
  - Systemzustand: „direkt“ manipulierbar
  - Datenmengen: Ausschnitte in Fenstern, navigierbar
- **Verzahnte Abläufe unterstützt**
  - Daten *quasiparallel* in verschiedenen Werkzeugen

[4]

2.1.2 Motivation

Seit dem Durchbruch der GUI, Mitte der 80iger Jahre [4], wurde das Konzept der graphischen Benutzeroberfläche stetig weiter entwickelt und rückte immer mehr in den Vordergrund. So wurde die Art und Weise wie die Hersteller ihre GUI umsetzten zum Hauptverkaufsargument. Denn durch eine geeignete Darstellung von Informationen und eine einfachere Zugänglichkeit zu Funktionen, kann die Benutzerfreundlichkeit und somit auch die Produktivität gesteigert werden. Zunächst beschränkten sich die Interaktionsmöglichkeiten zwischen Mensch und Maschine auf Maus und Tastatur. Die Entwicklung führte zu den Oberflächen wie wir sie heute von Windows, Mac OS und weiteren Betriebssystemen kennen.
Dies änderte sich mit dem Beginn des Smartphone-Booms welcher unter anderem durch das iPhone im Jahre 2007 eingeläutet wurde. Die Hauptgründe für den Erfolg des iPhone waren seine neuartige GUI und das innovative Bedienkonzept. Als Reaktion auf den großen Erfolg von Apple, begannen nun auch andere Hersteller neue GUI und Bedienkonzepte zu entwickeln.
Die Herausforderungen und Probleme im Vergleich zu Desktop-Systemen liegen zwar auf der Hand, sind jedoch nicht trivial zu lösen. Ein Hauptproblem bei Smartphones ist die beschränkte Bildschirmgröße. Hier gilt es unter anderem eine Balance zwischen der Größe der GUI Elemente und der Lesbarkeit der Informationen zu finden. Eine weitere Aufgabe ist die geschickte Anordnung, um auf dem geringen Platz, eine maximale Anzahl von Steuerelementen und Funktionalitäten zu platzieren ohne den Nutzer zu überfordern oder zu verwirren. Hierbei spielt auch die graphische Gestaltung der einzelnen Elemente eine wichtige Rolle.
Aber nicht nur die Gestaltung der GUI Elemente musste überarbeitet, sondern auch das gesamte Bedienkonzept. Smartphones besitzen keine Maus und aufgrund der eingeschränkten Größe nur eine kleine bzw. überhaupt keine Tastatur. Dies ist neben der eingeschränkten Bildschirmgröße das zweite große Probleme bzw. Herausforderung die es für die Hersteller zu lösen galt. Das Fehlen der Eingabegeräte (Maus & Tastatur) löste man durch die Verwendung von berührungssensitiven Bildschirmen (Touchscreen) und daraus resultierenden Weiterentwicklung, der Erkennung von Gesten.
Die genannten Probleme stellen für die gesamte Industrie eine große Herausforderung dar. Denn wie die einzelnen Hersteller diese Aufgaben lösen und ihre GUI und ihr Bedienkonzept gestalten, ist ebenso wie in den 80igern auch heute ein Alleinstellungsmerkmal und spielt deshalb auch eine große Rolle bei der Kaufentscheidung der Kunden und somit für den Gesamterfolg des jeweiligen Gerätes. Hieran wird deutlich welchen Stellenwert die GUI, vor allem bei Smartphones, besitzt.
Die Hersteller haben für die oben aufgeführten Probleme verschiedene Lösungsansätze. Deshalb werden im Folgenden die Realisierungen der vier weitverbreitesten Systeme: iOS (Apple), Blackberry (RIM), Android (Google) und Windows Phone 7 (Microsoft) näher beleuchtet und miteinander verglichen.

2.2 BlackBerry (RIM)

Abbildung 2.1: Logo und Beispielgerät [5]

Der Hersteller RIM (Research In Motion) liefert umfassende Guidelines über die Konzeption, Gestaltung und das Bedienkonzept der Apps, welche für sein System entwickelt werden. In diesem Abschnitt werden die wichtigsten Punkte bzgl. der GUI und des Bedienkonzeptes behandelt und vorgestellt. Die getroffenen Aussagen stammen aus dem entsprechenden Dokument[5], welches die Guidelines vorgibt. Da kein Testsystem zur Verfügung stand, können nur die offiziellen Vorgaben behandelt werden und keine eigenen Erfahrungen bzw. Abweichungen von den Guidelines, z.B. durch OpenSource Projekte oder spezielle Realisierungen einzelner Apps, betrachtet werden.

2.2.1 Allgemeine Anforderungen

RIM beschreibt in seiner Guideline 3 Schlüssel Prinzipien für die Gestaltung von BlackBerry Apps:

- Informierend
- Vertrauensvoll
- Zugänglich

Informierend

Apps sollen den Anwender immer mit aktuellen Nachrichten und Informationen versorgen. Hierzu sollen Benachrichtigungen, neue Informationen und häufig verwendete Funktionalitäten einfach und schnell zu erreichen sein. Um dies zu realisieren müssen wichtige Informationen, wie z.B. der aktuelle Status leicht erkennbar sein, so dass der Nutzer nicht danach suchen muss. Hingegen sollen selten verwendete Elemente in Menüs und Optionen eingebettet werden. Daraus folgt, dass der Entwickler eine Balance der Informationsdichte zu finden hat.

Vertrauensvoll

Der Nutzer muss Vertrauen in die App haben, d.h. er muss zu jederzeit wissen was die Anwendung macht, z.B. durch die Anzeige des aktuellen Status bei der Bearbeitung einer vom Benutzer gewünschten Operation. Denn nur wenn sich ein Anwender bei der Nutzung einer App „wohl fühlt“ wird er diese weiter verwenden. Dies beinhaltet unter anderem das Verzeihen von Fehlern (durch Rückkängig- bzw. Wiederholen-Funktion), klare und eindeutige Informationen die den Nutzer bei der Durchführung einer Aufgabe unterstützen und die Vermeidung von „Sackgassen“ bzw. unübersichtliche Programmführung, welche den Anwender verwirren und er somit den Überblick verliert.

Zugänglich

Hierbei ist darauf zu achten, dass man die App so gestaltet, dass der Nutzer sich sofort zurecht findet und somit auch bereit ist die Anwendung gleich zu testen. Wichtige Faktoren für die Zugänglichkeit sind ein einfaches und klar strukturiertes Layout, ansprechendes und zugleich minimalistisches Design und reduzierte Komplexität. Ebenfalls muss die App so strukturiert sein, dass sie sowohl für erfahrene Benutzer geeignet ist und zugleich unerfahrene Nutzer nicht abschreckt. Dies soll durch die Verwendung von Standard UI Element erreicht werden. Eine ausführliche Beschreibung der einzelnen Komponenten erfolgt später.

2.2.2 Bedienkonzept

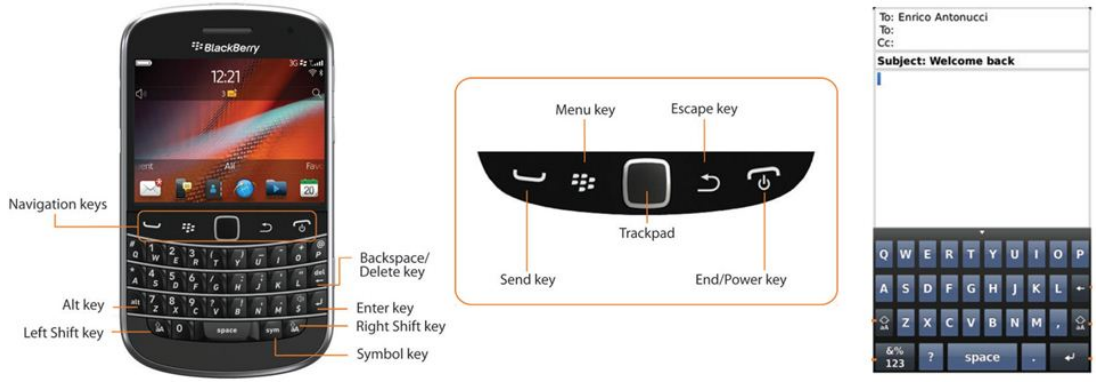

Abbildung 2.2: Bedienkonzept [5]

BlackBerry Geräte besitzen eine physische, eine virtuelle oder beide Tastaturen, speziell vorbelegte Navigationstasten und ein Trackpad. Neuere Modelle verfügen darüber hinaus noch über einen Touchscreen, über den das Gerät bedient werden kann (das BlackBerry Storm verfügt ausschließlich über einen Touchscreen). Der Hersteller verlangt, dass eine App sowohl über das Trackpad, als auch den Touchscreen ohne Einschränkungen bedienbar ist (Abwärtskompatibilität). Um zu verstehen wie man mit einem BlackBerry arbeitet und um die Bedienung mit Geräten anderer Hersteller zu vergleichen, folgt eine Aufzählung der möglichen Benutzerinteraktionen mit dem System bzw. welche Aktionen

das Gerät unterscheiden kann.
Trackpad:

- Einfacher Klick
- Klicken und Halten des Trackpads
- Klicken bei gedrückter Shift bzw. Alt Taste
- Bewegungen mit dem Finger auf dem Trackpad (wie Maussteuerung bei Notebooks)
- Wischgeste mit dem Finger über das Trackpad (Swipe)

Touchscreen:

- Einfache Berührung (Klicken)
- Doppelklicken
- Klick auf ein Objekt
- Klicken und Halten eines Objektes
- Vertikale Bewegung des Fingers
- Allgemeine Bewegung des Fingers
- Wischgeste über den Bildschirm
- Pinch-Gesten (2 Finger aufeinander zu bzw. auseinander Bewegen, häufig zum Zoomen verwendet

Darüber hinaus unterstützt das BlackBerry auch die Möglichkeit von Shortcuts, welche über die Tastatur aufgerufen werden können.

Navigationsleiste

Die Navigationstasten sind weitestgehend vorbelegt. So öffnet sich bei der Send-Taste die Telefon App (falls eine Nummer ausgewählt ist wird diese angerufen) bzw. sie dient zum Entgegennehmen eines Anrufes. Die Menü-Taste ist kontextsensitiv und öffnet das Menü zu dem aktuell sichtbaren Fenster der Applikation oder es kann durch gedrückt halten zwischen den Applikationen gewechselt werden. Mithilfe der End/Power-Taste können Gespräche beendet oder das System ausgeschaltet werden. Darüber hinaus kann sie auch zur Navigation verwendet werden. So bewirkt ein drücken der End/Power-Taste, dass man eine laufende App verlässt und zum Startbildschirm zurückkehrt (Die App wird dabei nicht beendet sondern nur pausiert). Die Escape-Taste dient, wie der Name vermuten lässt, zum Abbrechen von Aktionen oder zum Rückgängigmachen von Operationen (z.B. Zoomen), aber auch zur Navigation. Hier kann durch die Escape-Taste der vorherige Bildschirm oder Status wieder aufgerufen werden.

2.2.3 GUI Strukturierung

Homescreen und Title Bar

Abbildung 2.3: Homescreen [5]

Der Startbildschirm (Homescreen) bildet den zentralen Punkt eines BlackBerry. Hier werden im Banner die wichtigsten Informationen, wie Uhrzeit, Datum, Akkuzustand, Netzverbindung usw. angezeigt. Darüber hinaus befindet sich hier auch die Möglichkeit über eine zentrale Suche, applikationsübergreifend Informationen zu finden. Ebenso befinden sich im unteren Bereich die einzelnen Anwendungen die sich auf dem Gerät befinden. Das generelle Aussehen des Startbildschirms bzw. Banners kann nur durch die Änderung des Themes beeinflusst werden, welches später noch beschrieben werden. Einzelne Anwendungen haben keinen eigenen Banner, sie können jedoch Indikatoren definieren die unter der Uhrzeit erscheinen und über Ereignisse informieren (z.B. neue Nachrichten). Es besteht jedoch die Möglichkeit eine Title Bar am oberen Rand der Anwendung zu positionieren. Diese kann frei gestaltet werden und kann folgende Informationen enthalten:

- Anwendungsbild (Icon), beschreibenden Text, Uhrzeit
- Anwendungsinformationen wie z.B. neue Nachricht
- Verbindungsinformationen (Netz, Bluetooth, WLAN, UMTS)
- Batteriestatus
- Aktivitätsanzeige

Eine Title Bar könnte wie folgt aussehen:

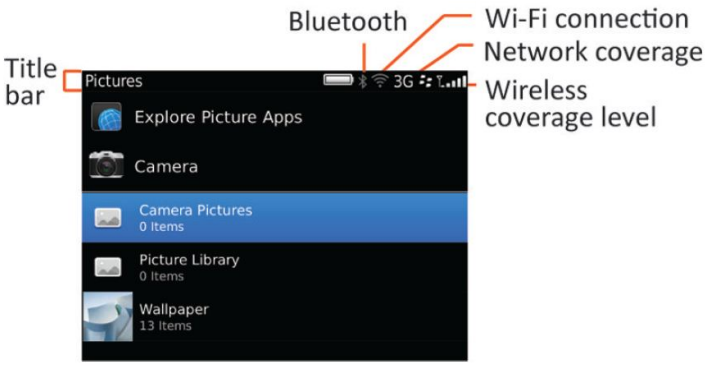

Abbildung 2.4: Beispiel einer title bar [5]

Panel Manager

Um Inhalte und Informationen zu gliedern und zu filtern, besteht die Möglichkeit einen Panel Manager zu verwenden. Hierdurch wird die Navigation für den Nutzer zwischen verschiedenen Inhalten vereinfacht, da er ohne Verlassen des aktuellen Kontexts zwischen verschiedenen Filtern, Ansichten und Funktionen wechseln kann.
Es gibt zwei Arten der Realisierung von Panel Managern. Zum einen die **Tab View**, in der am oberen Rand alle möglichen Inhalte angezeigt und ausgewählt werden können. Zum anderen die **Scrollable View**, hierbei hat man am oberen Rand eine Art Rad. Man kann fließend zwischen den einzelnen Inhalten wechseln oder durch Klicken einen bestimmten Filter für die aktuelle Auswahl bestimmen.

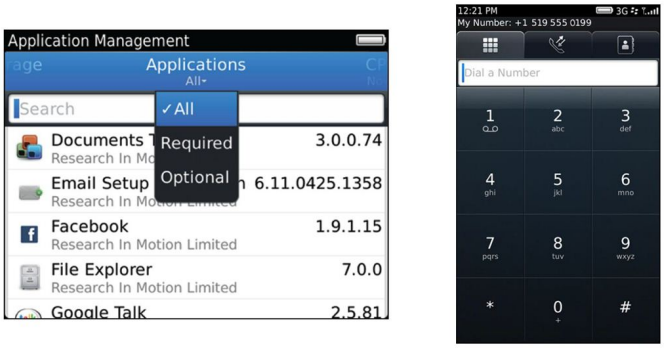

Abbildung 2.5: Beispiel für Tab View und Scrollable View [5]

2.2.4 GUI Elemente

Dialog Boxes

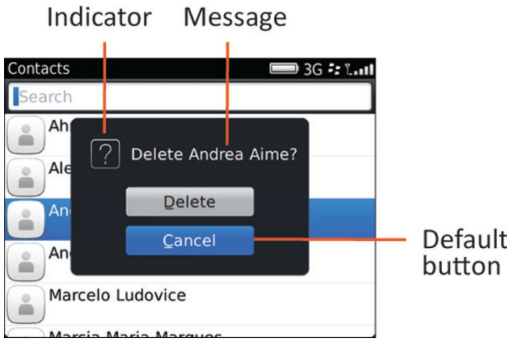

Abbildung 2.6: Aufbau der Dialog Boxes [5]

Die Dialog Boxes gibt es in verschiedenen Ausführungen, haben aber immer den gleichen Aufbau. Sie bestehen aus einem Indicator, welcher die Art der Dialog Box verdeutlichen soll. Des Weiteren aus einer Nachricht, die den User über ein Ereignis informiert und optionalen Buttons mit Entscheidungsmöglichkeiten des Nutzers. Sie werden dazu verwendet, notwendige Informationen vom Benutzer zu erlangen um eine Aufgabe zu beenden (z.B. „Wie ist der neue Dateiname?“) , den Anwender über den Status wichtiger Aktionen zu informieren (z.B. „Passwort wurde erfolgreich geändert.“) oder zu warnen wenn eine potentiell nicht annullierbare Operation durchgeführt werden soll (z.B. „Wollen Sie wirklich löschen?“).
Ein paar Beispiele:

Abbildung 2.7: Beispiele für dialog boxes [5]

Text fields

Textfelder werden dazu verwendet, Informationen vom Benutzer zu erlangen. Es gibt vordefinierte Arten von Textfeldern, welche sich dadurch unterscheiden, dass sie eine an die Eingabe angepasste Tastatur darstellen (z.B. zur Eingabe von Nummer oder Email Adressen) oder auf Eingaben unterschiedlich reagieren (z.B. Passwörter werden durch *** dargestellt oder bei Webadressen werden Leerzeichen automatisch durch Punkte ersetzt). Es gibt folgende Arten von Textfeldern, welche selbsterklärend sind:

- Email
- Date and Time
- Number
- Password
- Phone number
- Text
- Web address

Choice fields

Häufig ist es erforderlich, dass der Benutzer eine Auswahl zwischen verschiedenen Möglichkeiten trifft. Hierfür werden die sogenannten Choice fields bereitgestellt. Es gibt für nahezu jede Gelegenheit ein eigenes Choice field, welche sehr stark an das Aussehen von bekannten Desktop-Betriebssystemen erinnern und an diese angelehnt sind. Dies sind **check boxes** (Ja / Nein Auswahl), **drop-down lists** (Auswahl zwischen verschiedenen Werten) und **radio buttons** (Wahl zwischen mehreren Möglichkeiten von denen aber nur genau eine gewählt werden darf).

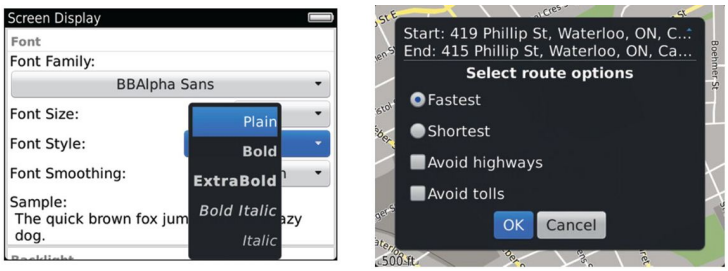

Abbildung 2.8: Beispiel für drop-down list, radio buttons und check boxes [5]

Darüber hinaus gibt es noch innovativere Möglichkeiten und Ansätze die wirklich neu und nicht nur einfache Kopien der bereits bekannten UI Elemente von Desktop Systemen sind. So gibt es die **Picker**, welche für häufig verwendete Auswahloperationen bereits eine für das Gerät vordefinierte Darstellung bieten.
Die Typen der Picker sind:

- File (Auswahl einer Datei, Browsen durch die Dateistruktur)
- Location (Positionswahl, aktuelle GPS Position oder zuvor verwendeter Ort)
- Date (Datumsauswahl)
- Time (Zeitauswahl)

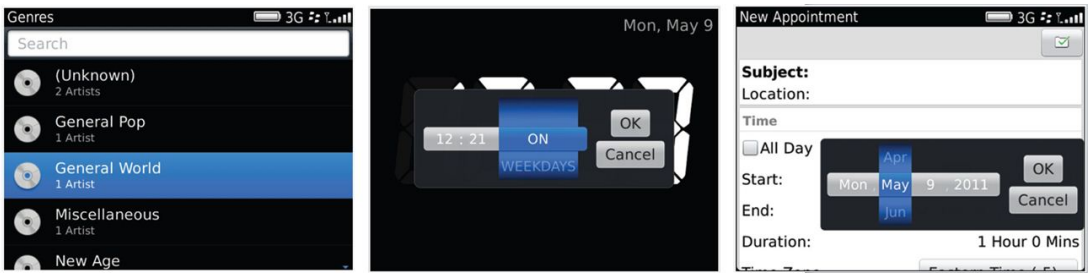

Abbildung 2.9: File, Time und Date Picker [5]

Diese Standardelemente können durch Einschränkungen angepasst werden, so dass z.B. bei der Auswahl eines Datums nur Monat und Jahr verfügbar sind.

Die letzte Art um Informationen darzustellen aus denen eine Auswahl getroffen werden soll ist die **tree view**. Wie der Name schon vermuten lässt, kann mit ihrer Hilfe Informationen in einer Baumstruktur dargestellt werden. Dieses UI Element bietet sich vor allem an um eine hierarchische Struktur nachzubilden (z.B. Dateistruktur).

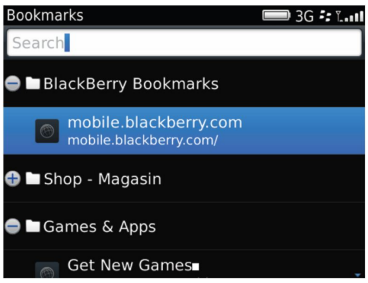

Abbildung 2.10: Beispiel einer tree view [5]

Neben den Pickern, bei denen die Auswahlmöglichkeiten durch den Typ des Pickers vordefiniert ist, gibt es auch die Möglichkeit die Auswahl zwischen frei wählbaren Werten zu realisieren. Hierfür werden die **Spin Boxes** bereitgestellt. Dieses UI Element soll vor allem für sequentielle Werte genutzt werden, das heißt, dass die Werte zwischen denen ausgewählt werden kann, in irgendeiner Art geordnet sind. Die Spin Boxes stellen somit eine schöne Alternative zu den drop-down lists da.

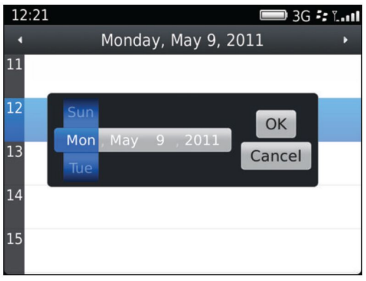

Abbildung 2.11: Beispiel einer spin box [5]

Lists und Tables

Häufig ist es erforderlich eine Menge von Informationen darzustellen. Um dies einfach strukturiert und übersichtlich für den Nutzer zu gestalten, stellt das Framework Listen und Tabellen (Lists und Tables) bereit. Dieses stellen eine gewisse Anzahl von Informationen in einer Liste dar. Sollten mehr Informationen bereitgestellt werden als auf eine „Seite“ passen, wird die Liste „scroll-fähig“ und man kann Sie nach oben bzw. nach oben bewegen um weiter Informationen zu sehen. Sollte nur Text dargestellt werden, handelt es sich lediglich um eine Liste. Um die Informationen für den Anwender aufzubereiten besteht jedoch auch die Möglichkeit in einen Eintrag der Liste Zusatzinformationen, wie Bilder usw. zu platzieren. Sollen Objekte in Reihen und Spalten sortiert werden können Tabellen verwendet werden. Um die Daten noch weiter aufzubereiten, können die Informationen innerhalb der Liste bzw. Tabelle gruppiert werden, z.B. nach Datum.

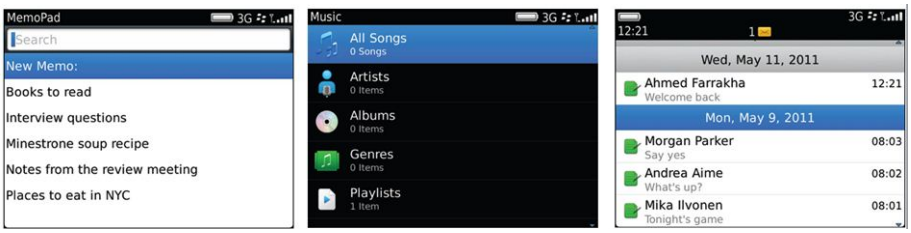

Abbildung 2.12: Listen und Tabellen [5]

Menüs und Toolbars

Um dem Benutzer Funktionen für das aktuell ausgewählte Objekt bereitzustellen gibt es Menüs. Diese sind kontextsensitiv und können jederzeit aufgerufen werden. Es gibt zwei Arten von Menüs, zum einen die an Desktop-Systeme angelehnten Full Menüs (inkl. Untermenüpunkten, sogenannten Submenüs) die immer am linken unteren Bildschirmrand ausgerichtet sind. Zum anderen gibt es Pop-up Menüs, diese bieten die Möglichkeit noch schneller auf spezielle Aktionen zuzugreifen. Pop-up Menüs gibt es mit drei, sechs oder maximal neun Elementen und sie erscheinen zentral auf dem Bildschirm.
Falls es Funktionen gibt, die häufig verwendet werden und jederzeit in der gesamten Anwendung oder in der aktuellen Ansicht verfügbar sein sollen, können diese über ein Icon in der Toolbar bereitgestellt werden. Die Toolbar befindet sich immer am unteren Rand. Es ist auch durchaus zulässig, dass Funktionen sowohl in der Toolbar als auch in einem Pop-Up-Menüs untergebracht werden. (z.B. Löschen einer Nachricht)

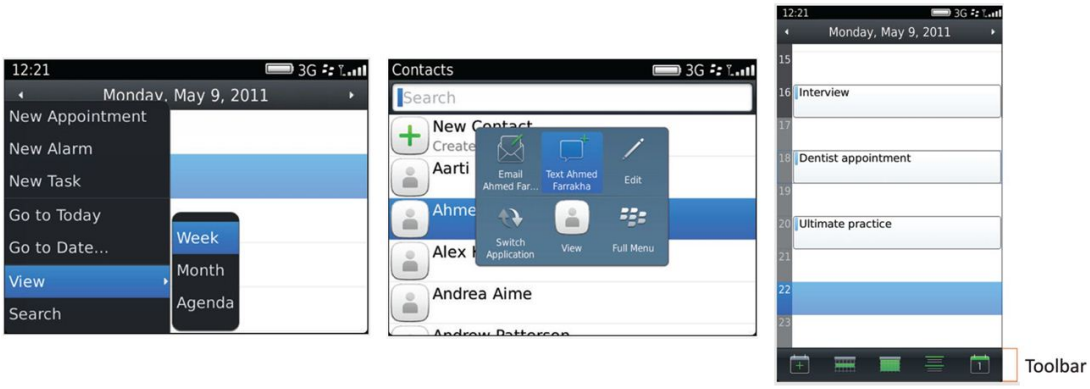

Abbildung 2.13: Beispiel für Menü, Pop-Up Menü und Toolbar [5]

2.2.5 Themes

Um das Look & Feel zu verändern, können Themes verwendet werden. Mit deren Hilfe können erweiterte Veränderungen an der Strukturierung des Starbildschirmes durchgeführt werden. Ebenso lassen sich einfache Veränderungen an den UI Elementen vornehmen, dies sind weitestgehend Änderungen an der Schriftart, Farbe, Hintergrundbildern und Icons. In dieser Ausarbeitung wurde lediglich das Standardtheme BlackBerry 7 behandelt. Der Nutzer hat jedoch die Möglichkeit zwischen vorinstallierten oder auch frei verfügbaren Themes zu wählen. Es können auch eigene Anpassungen durch den Anwender vorgenommen werden. Hierzu stellt RIM eine frei verfügbare Software (BlackBerry Theme Studio) bereit.

2.2.6 Zusammenfassung

Abschließend lässt sich festhalten, dass RIM bzgl. der Gestaltung von Apps eine ausführliche Guideline herausgegeben hat in der mit Hilfe von Beispielen und best practices dem Entwickler viele Hilfestellungen zum Design seiner Anwendung gegeben wird. In den Guidelines wird betont, dass die vom System bereitgestellten UI Elemente zu verwenden sind, da dadurch viele Vorteile entstehen. Es wird aber keine Aussage darüber getroffen wie und ob man eigene UI Elemente erzeugen kann bzw. darf. Auffällig bei der Betrachtung der einzelnen UI Elemente ist auch, dass viele davon stark an bereits bekannte Desktop-Systeme angelehnt sind. Dies hat zwar den Vorteil, dass der Anwender diese bereits kennt und der Entwickler bestehende Anwendung „einfacher“ portieren kann, bringt im Gegenzug aber auch Nachteilen mit sich. So sind diese altbekannten Steuerelemente vor allem für die Nutzung von Maus und Tastatur konzipiert und nicht für Touchscreens. Ein weiteres Problem ist die gewünschte Abwärtskompatibilität für BlackBerry Geräte ohne Touchscreen, denn so muss bei dem Design immer darüber nachgedacht werden wie man neuartige Konzepte die besonders für die Nutzung durch Touchscreens geeignet sind auch für ältere Systeme mit Trackpad umsetzt.

2.3 Android

Abbildung 2.14: Android Logo und Samsung Galaxy SII [7][6]

Bei Android handelt es sich um ein Betriebssystem für mobile Geräte wie z.B. Smartphones und Tablets. Der Hersteller Google verfolgt mit Android einen OpenSource Ansatz. Dies äußert sich unter anderem darin, dass nur minimalistische UI Guidelines existieren. Es gibt also keine genauen Vorschriften wie welches GUI Element einzusetzen ist.
Des Weiteren stand während der Erstellung dieser Arbeit nur für wenige Stunden ein Testgerät bereit, weshalb primär die Aussagen der Guidelines und das generelle Bedienkonzept untersucht wurden. Aufgrund des OpenSource Ansatzes gibt es natürlich die verschiedensten Abwandlungen der einzelnen Steuerelmente oder sogar ganz neue Entwicklungen, die hier jedoch nicht untersucht werden.
Als Basis dient zum einen die Developer Webseite von Android [13] und ebenso Präsentationen von Google [9], welche im Rahmen der Developer Conference (Google I/O) vorgestellt wurden.

2.3.1 Allgemeine Anforderungen

Google gibt den Anwendungsprogrammierern generell keine Vorschriften, wie sie GUI Elemente zu nutzen haben oder ihre App gestalten sollten. Von der Developer Seite, gibt es jedoch eine Präsentation [14] in der einige Hilfestellungen in Form von Do's und Don'ts gegeben werden, welche beim Design einer App zu berücksichtigen sind.
Don'ts:

- Das User Interface nicht einfach von einer anderen Platform portieren (z.B. iOS)
- Nicht zuviele PopUp's bzw. Dialog Fenster
- Keine absoluten Werte für das Layout
- Vermeidung kleiner Textgröße

Do's:

- Verschieden Versionen einer App für alle möglichen Auflösungen
- Große Icons, bzw. UI Elemente die per Touchgeste angesteuert werden sollen
- Unterstützung von Trackball- und D-pad-Navigation (Steuerkreuz)
- Behandlung von Ausrichtungsänderungen (Gerät horizontal oder Vertikal)
- Nutzung von Themes und Styles um Redundanz zu vermeiden

Neben den Do's und Don'ts beschreibt Google noch fünf Prinzipien für ein gutes Interface. Diese Prinzipien beziehen sich jedoch nicht nur auf die Entwicklung für mobile Geräte, sondern sind generell für das Design von Anwendungen sehr hilfreich.

- **Konzentration auf den Nutzer**: Zielgruppenanalyse, d.h. wer soll die Anwendung nutzen und welche Vorkenntnisse, welches Alter, welche Geräte usw. werden die Anwender haben.
- **Mache die richtigen Dinge sichtbar**: häufig verwendete Operationen sollten sofort sichtbar und zugänglich sein; selten genutzte können in Menüs untergebracht werden.
- **Feedback**: der Nutzer muss zu jedem Zeitpunkt eine Rückmeldung zu einer durchgeführten Interaktion erhalten, hierbei ist darauf zu achten, dass die Rückmeldungen eindeutig sind.
- **Sei berechenbar**: für den Nutzer muss sofort erkennbar sein, was eine Funktion bedeutet und was sie bewirkt, dazu gehört auch, dass Interaktionsmöglickeiten als solche erkennbar sind.
- **Sei Fehlertolerant**: um den Anwender zum Testen anzuregen, muss die Anzahl der irreversiblen Operationen minimiert und wenn immer möglich eine Undo (Rückgängig-Funktion) angeboten werden.

2.3.2 Bedienkonzept

Funktionstasten

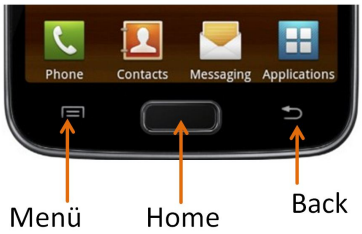

Abbildung 2.15: Funktionstasten [6]

Aktuelle Smartphone Modelle mit Android verfügen über einen Touchscreen, sowie drei Tasten. Zum einen den Homebutton mit dem es möglich ist, jederzeit zum Homescreen zurückzukehren. Des Weiteren gibt es eine Taste um das Options Menü ein- und wieder auszublenden. Dieses Menü enthält Funktionen die sich auf die aktuelle App bzw. Ansicht beziehen und ist unabhängig von eventuell markierten oder ausgewählten Objekten. Die dritte Taste, ist die Back (Zurück) Taste. Diese hat je nach aktuellem Kontext eine leicht unterschiedliche Funktion. Sie dient unter anderem als Abbrechen oder Zurück-Taste. Ist zum Beispiel ein Kontextmenü geöffnet kann dieses mit der Back Taste geschlossen werden oder befindet man sich auf dem Startbildschirm einer App wird diese geschlossen. Ihre Funktionsweise ist vergleichbar mit der ESC-Taste von Desktopsystemen.

Homescreen und Menü

Abbildung 2.16: Homescreen und Menü

Der Ausgangspunkt für die Bedienung bildet der Homescreen. Dieser kann weitestgehend frei gestaltetet werden. So befinden sich hier die vom Nutzer gewünschten Apps, die er schnellstmöglich erreichen möchte. Auch eine programmübergreifende Suchfunktion befindet sich am oberen Rand. Des Weiteren können hier auch Widgets, kleine Hilfsprogramme zu installierten Apps, platziert werden. Widgets werden später näher betrachtet. Mittels des Home/Menü Button am rechten unteren Rand kann zwischen dem Homescreen und einer Übersicht aller installierten Apps umgeschaltet werden. Befindet man sich im Menü kann man Apps auf den Homescreen verschieben oder löschen.

Benachrichtigungszentrale

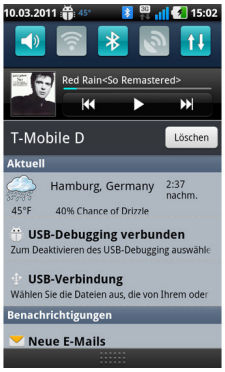

Abbildung 2.17: Benachritigungszentrale [8]

Um den Nutzer über aktuelle Informationen und Ereignisse zu unterrichten besteht die Möglichkeit durch eine vertikale Wischgeste die Benachrichtigungszentrale zu öffnen. Hier können am oberen Rand Dienste wie GPS, Bluetooth usw. aktiviert bzw. deaktiviert. Des Weiteren können im unteren Bereich Informationen von einzelnen Apps die z.B. im Hintergrund laufen, dargestellt und darauf zugegriffen werden.

2.3.3 GUI Strukturierung

Um verschiedene Ansichten und Funktionen übersichtlich darzustellen und den Nutzer nicht durch unnötig komplexe Menüs zu überfordern bietet Android verschiedene Möglichkeiten und Gestaltungsansätze. Die interessantesten sind das Dashboard, Tab Layout, Table Layout und die List View.

Dashboard

Abbildung 2.18: Beispiel eines Dashboards [9]

Bei der Verwendung des Dashboards werden auf dem Startbildschirm der App die wichtigsten Funktionalitäten dargestellt. Ebenso soll der Anwender sofort eine Übersicht über die Möglichkeiten der App erhalten, sowie auf eventuelle Neuerungen hingewiesen werden.

Ein weiterer Vorteil ist, dass dem Nutzer eine umständliche Navigation durch mehrere Untermenüs erspart bleibt. Als Gestaltungshinweise gibt Google vor, dass man sich auf drei bis sechs Auswahlmöglichkeiten beschränken soll; das Dashboard als Vollbild-Ansicht zu realisieren ist und die Auswahlmöglichkeiten aus verschiedenen Bereichen (Accountverwaltung, verschiedene Filter und Ansichten usw.) zu wählen sind. Das Dashboard findet unter anderem Verwendung in der Twitter- und Facebook App. Da diese jedoch ständig im Wandel sind, zeigt Abbildung 2.18 eine allgemeine Skizze für eine Dashboard Umsetzung.

Tab Layout

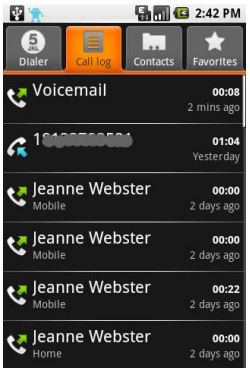

Abbildung 2.19: Beispiel eines Tab Layouts [10]

Häufig kommt es vor, dass man in einer App zwischen verschiedenen Ansichten bzw. dem Kontext wechseln muss. Um dies zu realisieren bietet sich die TabView an. Hier befinden sich am oberen Rand Auswahlmöglichkeiten welche die entsprechende Funktion bzw. Ansicht mit Hilfe eines Icons und Text beschreiben. Dies ist vergleichbar mit Lesezeichen bzw. Markierungen bei Karteikarten und Akten. Die Tab View findet unter anderem Verwendung in der Telefon-App von Android, hier kann zwischen Nummerneingabe, Kontakten, Verlauf (Anrufhistorie) und Favoriten gewechselt werden.

Table Layout

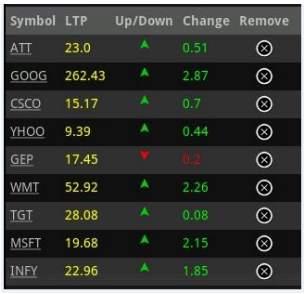

Abbildung 2.20: Beispiel für Table Layout [11]

Um die Informationen für den Anwender aufzubereiten ist es möglich, diese in Form einer Tabelle anzuordnen. Durch die tabellarische Anordnung ist sehr einfach viele Informationen konsistent darzustellen was zu einer besseren Lesbarkeit führt. Auch beim Table

Layout sei darauf hingewiesen, dass die Gestaltung völlig frei vorgenommen werden kann, d.h. der Inhalt der einzelnen Zellen ist nicht beschränkt.

List View

Abbildung 2.21: Beispiel einer List view [13] [12]

Um eine Menge von Elementen darzustellen bietet sich die List View an. Hierbei werden zum Beispiel verschiedene Elemente durch ihren Namen repräsentiert und in einer „scrollfähigen“ Liste dargestellt. Die List View unterstützt auch eine Art Filterfunktion, so kann man die Auswahl durch Eingabe eines Textes einschränken. Auch die List View kann derart abgeändert werden, dass sie fast nicht mehr als solche erkennbar ist und zum Beispiel aussieht wie ein Bücherregal.

2.3.4 GUI Elemente

Menüs

Bei Android gibt es drei Arten von Menüs: **Options-, Kontext- und Submenüs**.

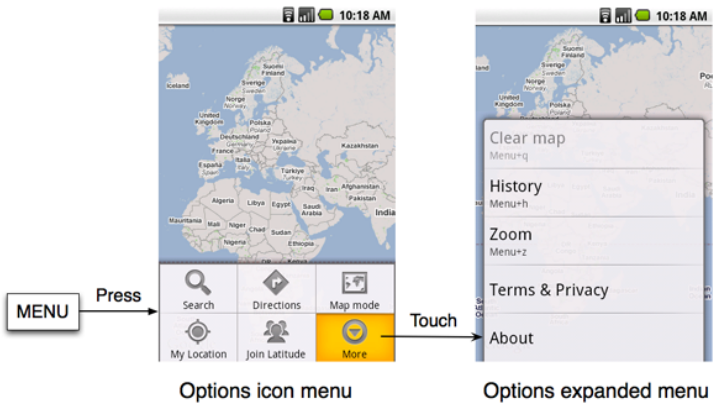

Abbildung 2.22: Aufbau von Optionsmenüs [13]

Das **Optionsmenü** erscheint wenn der Nutzer die Menütaste drückt. Es befindet sich dann am unteren Rand des Bildschirms und enthält nur Funktionen die Kontextunabhängig sind und benötigte Navigationselemente, d.h. globale Operationen. Dies ist zu

beachten, da bei Betätigen der Menütaste das eventuell ausgewählte Objekt vom System als „nicht mehr ausgewählt“ gesetzt wird. Der Zugriff auf die Elemente im Optionsmenü erfolgt in zwei Wegen, zum einen durch die Menütaste oder in neueren Android Versionen (3.0 oder höher) über die Action Bar, welche später näher betrachtet wird. Das Optionsmenü enthält sechs Elemente, wobei Android bei mehr als sechs Elementen automatisch ein „Mehr“-Element erzeugt, welches dann die weiteren Elemente in einem Menü darstellt. Bei Android 3.0 und höher werden die Elemente des Optionsmenüs automatisch in der Action Bar untergebracht. Als weitere Maßgabe von Google ist darauf zu achten, dass die Objekte in den Menüs nach ihrer Verwendungshäufigkeit zu ordnen sind, da die Menüs häufig „scroll-fähig“ sind und man somit dem Anwender unnötige Arbeit erspart.

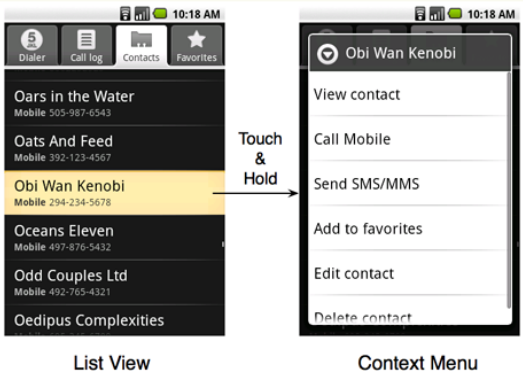

Abbildung 2.23: Aufbau Kontextmenüs [13]

Das **Kontextmenü** erscheint wenn der Nutzer auf ein Objekt drückt und dieses gedrückt hält. Dies ist vergleichbar mit dem Rechtsklick bei Desktopsystemen. Wie der Name vermuten lässt sind hier die kontextabhängigen Funktionalitäten untergebracht. Google gibt allgemeine Hinweise zur Verwendung von Kontextmenüs. So sollten Funktionen nicht ausschließlich in Kontextmenüs untergebracht werden. Ein Beispiel hierfür ist die Möglichkeit eine Telefonnummer direkt anzurufen. Es sollte zum einen über das Kontextmenü möglich sein die Nummer anzurufen, aber auch wenn der Nutzer den Namen auswählt und in einer anderen Ansicht Detailinformationen zu der ausgewählten Nummer / Person erhält. Dies ist wichtig da der Nutzer nicht ohne weiteres erkennen kann, dass ein Kontextmenü zur Verfügung steht. Ebenso sollte bei einem Kontextmenü immer erkenntlich sein auf welches Objekt sich die möglichen Operationen beziehen (z.B. als Titel der Dialog Box den Namen des Objektes).
Das **Submenü** ist eine eingebettete Liste von Funktionen zu einem anderen Menüpunkt, z.B. eines Optionsmenüs. Es unterscheidet sich vom Aussehen nicht von einem Kontextmenü, kann jedoch auch globale Funktionalität enthalten.

Action Bar

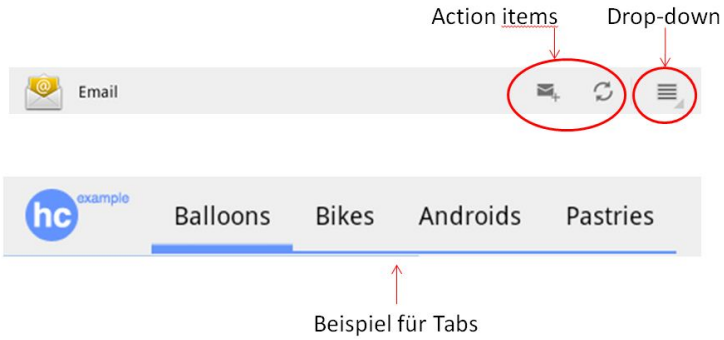

Abbildung 2.24: Beispiele für Action Bar [13]

Die Action Bar ist ein GUI Element welches die standardmäßige title bar ersetzt. Sie befindet sich am oberen Rand des Bildschirms und bringt einige Funktionen mit sich. Sie beinhaltet das Icon der aktuellen Anwendung, einen Text der die aktuelle Ansicht beschreibt und diverse Steuerelemente. Als Steuerelemente können verschiedenste Element platziert werden, es ist jedoch darauf zu achten, dass diese Elemente nicht kontextsensitiv sind. Es besteht die Möglichkeit Elemente aus dem Optionsmenü direkt als sogenannte action items in der Action Bar zu platzieren, wodurch ein schnellerer Zugriff ermöglicht wird. Elemente aus dem Optionsmenü welche keine action items sind, werden in einer drop-down Liste dargestellt, welche über den Menübutton der Action Bar geöffnet wird. Ebenso können Navigationselement (Tabs) platziert werden, um zum Beispiel schnell zu einer bestimmten Ansicht oder einem gewissen Kontext zu wechseln. Die Action Bar unterstützt auch frei gestaltbare Ansichten (Custom Views), sowie eine integrierte Suchleiste bzw. Suchfunktion.

Quick Actions

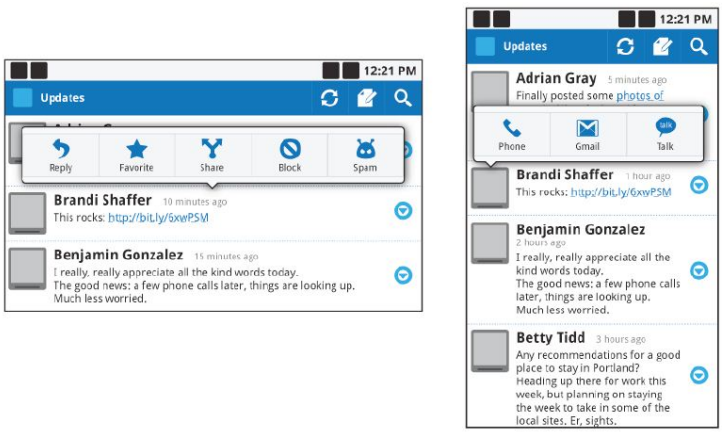

Abbildung 2.25: Beispiel für Quick Actions [9]

Eine Möglichkeit dem Nutzer kontextsensitive Funktionen bereitzustellen ohne den Arbeitsfluss zu behindern bieten die Quick Actions. Hierbei wird bei der Berührung eines

Objektes ein Menü eingeblendet, welches durch Icons und Text auf Interaktionsmöglichkeiten hinweist. Dies ist eine sehr intuitive und schnelle Art der Bedienung. Die Quick Actions motivieren den Nutzer die verschiedenen Funktionen der Anwendung zu entdecken, da sie ein positives Feedback liefern, wodurch auf eine gewisse Weise der Spieltrieb geweckt wird.
Als Hinweis gibt Google an, dass die Quick Actions nicht in Umgebungen mit Mehrfachauswahl genutzt werden sollen. Ebenso ist darauf zu achten, nur die wichtigsten und am häufigsten verwendeten Funktionen für das entsprechende Objekt bereitzustellen.

Dialogs

Abbildung 2.26: Beispiele für Dialogs [12] [13]

Um mit dem Nutzer zu interagieren gibt es Dialogs. Diese gibt es, wie auch bei anderen Systemen, in verschiedenen Varianten und für bestimmte Aktionen bereits vorgefertigt. Das Spezielle ist jedoch, dass diese bei Android frei gestaltet und modifiziert werden dürfen. Bei der Nutzung von Dialogs sollte man sich jedoch die Frage stellen ob diese unbedingt notwendig sind. Zum einen stören sie den Arbeitsfluss, zum anderen werden die meisten Funktionalitäten, einschließlich der search bar, deaktiviert. Bei Android gibt es vier Arten von Dialog Boxes, welche nach den eigenen Wünschen angepasst werden können:

- **AlertDialog**: Besitzt null, einen, zwei oder drei Schaltflächen (Buttons) und eine Liste von auswählbaren Elementen, z.B. check boxes und radio buttons.
- **ProgressDialog**: Eine Erweiterung des AlertDialog, mit einer Aktivitätsanzeige (drehendes Rad) und einem Fortschrittsbalken. Kann ebenso bis zu drei Buttons enthalten.
- **DatePickerDialog**: Standardelement zur Auswahl eines Datums
- **TimePickerDialog**: Standardelement zur Auswahl einer Zeit

Speziell für die Zeit- und Datumsauswahl gibt es bereits vorgefertigte GUI Element. Um zu veranschaulichen, dass in der Gestaltung der Elemente keine Beschränkungen herrschen ist hier ein Beispiel anhand des standardmäßigen Date Picker (Datumsauswahl) und einer modifizierten Variante.

Abbildung 2.27: Standard und modifiziertes UI Element [12]

Notifications (Benachrichtigungen)

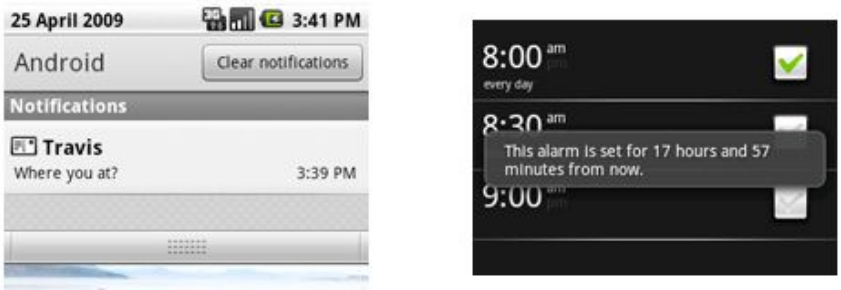

Abbildung 2.28: Toast und Status Bar Notificatiion [13]

Neben den zuvor erwähnten Dialogs gibt es noch weitere Möglichkeiten den Anwender über die Vorgänge und Ereignisse zu informieren. Hier unterscheidet man zwischen zwei Arten von Benachrichtigung, der **Toast Notification** und der **Status Bar Notification**.

Die **Toast Notification** ist eine Nachricht die im Vordergrund erscheint. Die aktuelle Anwendung wird hierbei nicht unterbrochen und bleibt sichtbar. Die Benachrichtigung ist nur so groß wie benötigt um die gewünschte Nachricht darzustellen. Sie wird automatisch ein- und wieder ausgeblendet und akzeptiert keine Interaktion. Eine Toast Notification kann auch von einem Hintergrunddienst (Background Service) ausgelöst werden und erscheint auch wenn die dazugehörige Anwendung nicht sichtbar ist. Sie sollte jedoch primär zur Information über durchgeführte Operationen wie: Datei gespeichert, Änderungen übernommen usw. genutzt werden, wenn man sicher ist, dass der Anwender auch auf den Bildschirm schaut.

Die **Status Bar Notification** fügt zum einen ein Symbol in die Systemstatusleiste hinzu (optional auch mit kurzem Text) und zum anderen eine ausführliche Nachricht in das Benachrichtigungsfenster (Benachrichtigungszentrale). Wenn der Benutzer die Nachricht

im Benachrichtigungsfenster auswählt kann eine Aktion ausgeführt werden, dies ist normalerweise das Öffnen der dazugehörigen Anwendung mit dem entsprechenden Kontext (z.B. Anzeige einer verpassten Nachricht). Diese Art der Benachrichtigung ist ideal, wenn die Anwendung im Hintergrund läuft und der Nutzer über Ereignisse informiert werden soll. Die Status Bar Notification hat den Vorteil, dass keine Nachricht bzw. kein Ereignis verloren geht, wenn der Nutzer gerade nicht auf den Bildschirm schaut und man kann zusätzlich auf neue Informationen durch Alarme hinweisen. Alarme können sein: Ton abspielen, Vibration und blinkende Lichter am Gerät.

Sonstige UI Elemente

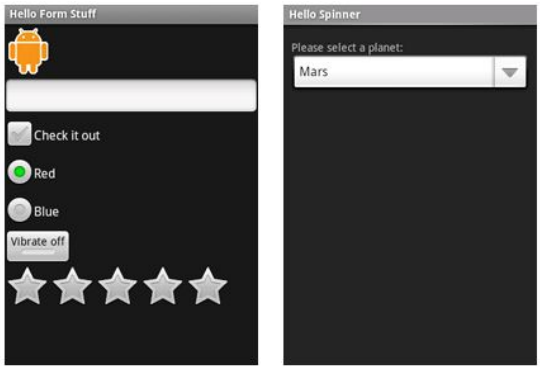

Abbildung 2.29: Übersicht der Standard UI Elemente [13]

Um im späteren Verlauf dieser Arbeit einen vollständigen Vergleich zwischen den einzelnen Systemen ziehen zu können sei erwähnt, dass Android auch über Standard UI Elemente wie man sie von Desktopsystemen kennt, verfügt. Dies sind zum Beispiel Check Boxes, Radio Buttons und Drop-down Listen.

2.3.5 Companion Widgets

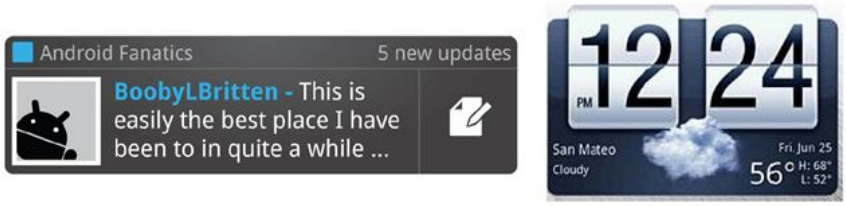

Abbildung 2.30: Beispiele für Companion Widgets [9] [12]

Ein interessanter innovativer Ansatz um alle Vorteile mobiler Geräte und der Apps zu nutzen stellen die Companion Widgets dar. Dies sind kleine Informationsfenster welche zu einer App gehören und auf dem Homescreen platziert werden können. Hierdurch kann der Nutzer ohne in die entsprechende App zu gehen, direkt die neusten Informationen abrufen. Diese Widgets können jedoch nicht nur Informationen anzeigen, sondern auch

Funktionen die direkte Auswirkung auf die App haben, bereitstellen. Dies trägt zur Freiheit des Nutzers bei, da dieser die Widgets nach Belieben platzieren, anordnen und wieder entfernen kann.
Google weist darauf hin, die Companion Widgets platzsparend zu gestalten und diese nicht nur als einfachen Programmaufruf zu nutzen, d.h. es sollen Informationen und Funktionen bereitgestellt werden und nicht einfach die dazugehörige App starten.

2.3.6 Zusammenfassung

Bei der Untersuchung der GUI von Android wurde deutlich, dass Google dem Entwickler / Designer völlig freie Hand lässt und die zentrale Aussage **„Guideline NOT Restriction“** sehr ernst nimmt. Dies bringt natürlich Vor- und Nachteile mit sich. Zu den Vorteilen zählt, dass man die GUI und auch vorgefertigte UI Elemente nach Belieben gestalten bzw. anpassen kann. Diese Freiheit kann jedoch sehr schnell zu negativen Ergebnissen führen, weil es sehr anspruchsvoll ist, ein intuitives, modern aussehendes und dabei immer noch effizient zu bedienendes UI Element, zu entwerfen. Auch äußert sich die Freiheit leider in den nur minimalistisch vorhandenen Guidelines. Einzelne Aussagen zu Gestaltungsrichtlinien und „best practices“ verstecken sich im allgemeinen Developer Guide, aber nicht in den eigentlich dafür vorgesehenen UI Guidelines. An dieser Stelle sei erwähnt, dass sich die Guidelines erst in der Entwicklung befinden und häufig erweitert werden. Die oben angestellten Betrachtungen beruhen auf dem Devoloper Guide bzw. UI Guidelines, Stand 19.10.11.
Google fordert, dass Apps nicht nur einfach von anderen Plattformen portiert und nachgebildet werden sollen, sondern man muss eine eigene „Android-Erfahrung“ schaffen. Dies ist eine völlig richtige Forderung, jedoch lässt Google den Entwickler bei der Umsetzung dieser Anforderung ziemlich allein. Es besteht meiner Meinung nach das Problem, dass der Entwickler auf sich selbst gestellt ist und er nur einen schweren Einstieg findet, wie man eine App für Android designt und worauf es zu achten gilt. Diesen Mißstand hat die Entwicklergemeinde erkannt und bereits eigene UI Guidelines entworfen, aber auch Google hat reagiert und arbeitet an seinen UI Guidelines.

2.4 Windows Phone

Abbildung 2.31: Logo und Beispielgerät [16] [15]

Windows Phone 7 ist das Betriebssystem für Smartphones vom Hersteller Microsoft. Als Grundlage für die folgenden Betrachtungen dient ein Beitrag im MSDN („Microsoft Developer Network“)[17]. Der genutzte Abschnitt ist: „User Experience Design Guidelines for Windows Phone“ , welcher über die Entwicklung und das Design von Anwendung auf Basis von Windows Phone informiert und Gestaltungsrichtlinien festlegt.
Während der Untersuchung stand kein Testgerät zur Verfügung, weshalb sich die getroffenen Aussagen auf die oben genannte Guideline beziehen.

2.4.1 Allgemeine Anforderungen

Microsoft stellt zu Anfang seiner Guidelines die Besonderheiten mobiler Geräte (Smartphones) heraus. Es wird darauf hingewiesen, dass man nicht bereits bekannte GUI Realisierungen von Desktop oder Browser basierten Anwendung übernehmen bzw. übertragen kann. Besonders die neuen Interaktionsmöglichkeiten über Touchscreens und der generell kleinere Bildschirm stellen eine besondere Herausforderung für das Design einer Anwendung dar. Ebenso ist die Mobilität ein wichtiger Punkt. Es gilt zu berücksichtigen, dass der Nutzer die Anwendung weitestgehend unterwegs und nicht zu Hause nutzt wo er evtl. noch Zugang zu einem Desktopsystem hat.
Um diese Aufgaben zu lösen hat Microsoft ein neues Look & Feel entworfen, welches intern den Codenamen „Metro“ trägt. Um das volle Potential der Windows Phone Plattform auszuschöpfen und Konsistenz zwischen allen Anwendungen zu erhalten, gibt es generelle Designprinzipien. Diese „Be“ (Sei) Prinzipien sollen eingehalten werden um das spezielle „Metro“ Look & Feel zu erzeugen.
Die Prinzipien sind:

- Be Modern: Das Design soll minimalistisch und frei von unnötigen Verzierungen sein. Bei Windows Phone stellt der Inhalt das Interface dar. Der Anwender erwartet, dass er direkt mit den angezeigten Objekten interagieren kann. Um eine klare Strukturierung zu erreichen, sollte genügend Freiraum zwischen den UI Elementen eingefügt werden.

- Be „On the Go“ Capable: Da die Anwendung weitestgehend unterwegs genutzt werden, ist auf eine einfache und schnelle Menüführung zu achten. Informationen müssen sofort erkennbar sein und die Anzahl der vom Benutzer benötigten Eingaben soll reduziert werden. Um dies zu erreichen können Eingabevorschläge, Filter, Favoriten usw. genutzt werden. Auch eine geeignete Präsentation trägt zur schnelleren Informationsaufnahme des Nutzers bei (z.B. Schriftgröße, Farbe usw.).

- Be clean: Es ist darauf zu achten, dass die Anwendung klar strukturiert ist. So sollen Touch-Elemente groß genug und nicht zu dicht beieinander platziert sein, denn dies führt schnell zu einem Fehlklick, was für den Nutzer sehr frustrierend ist. Ebenso gilt es darauf zu achten die Bedienung für jeden Nutzer intuitiv zu gestalten, auch für Nutzer mit größeren Fingern. Als Ziel sollte gelten, dass keine Anleitung für eine Anwendung benötigt wird, weil alle möglichen Funktionen klar erkennbar sind und durch ausprobieren entdeckt werden können.

- Be in Motion: Die Anwendung soll immer Feedback geben um den Nutzer zu informieren und auf Interaktionsmöglichkeiten hinzuweisen. Um die Illusion zu erzeugen, dass Objekte physisch manipuliert werden können, sollen diese auf möglichst viele verschiedene Interaktionen des Anwenders reagieren.

- Be Simple, Readable, and Minimalistic: Es gilt dem Nutzer nur die wichtigsten Informationen zu präsentieren und jegliche unnötigen Zusatzinformationen zu eliminieren. Dies gilt vor allem auch bei der Menüführung. Der Nutzer soll sich nicht in einer Hierarchie von Ansichten verlieren, so dass er nicht weiß was er gerade sieht oder wo er von der aktuellen Ansicht ausgehend hinspringen kann. Auch muss man sich von dem Gedanken: „Maximale Informationen in einer Ansicht“ verabschieden. Eine Hauptaufgabe bei der Gestaltung der UI ist das Filtern zwischen wichtigen Informationen in der Startansicht und dem Verbergen von Zusatzinformationen in Unteransichten.

- Be Consistent: Konsistenz innerhalb einer Anwendung und platfformübergreifend ist ein wichtiger Aspekt. Der Anwender ist mit der Bedienung und den allgemeinen Interaktionsmöglichkeiten wie: Navigation, Einstellungsänderung usw., bereits vertraut. Dadurch wird der Einstieg in eine neue Anwendung erleichtert und führt zu einer besseren Zufriedenheit beim Nutzer, was zum Erfolg der Anwendung beiträgt.

- Be Authentic: Eine Anwendung sollte die UI Element des Windows Phone nutzen und Inhalte nicht nur in einer Webansicht anzeigen, sondern an das allgemeine Bedienkonzept angepasst und aufbereitet werden. Erst hierdurch werden die Vorteile der mobilen Plattform deutlich und es entsteht ein Mehrwert für den Anwender. Die Verwendung der Standardelemente soll jedoch nicht bedeuten, dass die Elemente nicht angepasst werden dürfen oder sollen. Es ist sogar wichtig, die eigene Corporate Identity umzusetzen (z.B. Firmenlogos, Farben usw.), da dies zum Wiedererkennungswert beiträgt.

- Be Innovative: Erstellt man zu einer Desktop- oder Webanwendung eine Version für Windows Phone, gilt es nicht die einzelnen Funktionen oder das User Interface nachzubilden. Zum einen bietet das Windows Phone neue Möglichkeiten durch zusätzliche Sensoren wie: GPS, Kompass usw., welche in neue Funktionalitäten umgesetzt werden können (z.B. positionsbezogene Informationen oder Nachrichten mit einem Foto versehen). Zum anderen muss das User Interface an die Touchsteuerung und den kleineren Bildschirm angepasst werden, d.h. komplette Neugestaltung. Auch muss darüber nachgedacht werden, ob und welche Funktionalitäten der Anwender überhaupt unterwegs Nutzen möchte bzw. welche sinnvoll sind. Als Grundsatz gilt: „Neues Design statt Nachbildung“.

2.4.2 Bedienkonzept

Touchgesten

Die primäre Interaktion mit dem Gerät, erfolgt bei Windows Phone über den Touchscreen bzw. über Touchgesten die das Gerät erkennt. Windows Phone Geräte unterstützen hierbei sowohl Single-touch als auch Multi-touch-Gesten (Gesten mit einem oder mehreren Fingern). Während das zu Grunde liegende Framework bis zu zehn gleichzeitige Touchpunkte erkennt, unterstützen jedoch die meisten Touchscreen nicht mehr als vier gleichzeitige Eingaben.
Die verschiedenen Touchgesten die vom System unterschieden werden sind:

- **Single-Touch:**
  - Tap (einfaches Antippen)
  - Double Tap (doppeltes Antippen des gleichen Punktes)
  - Pan (Finger über den Bildschirm ziehen)
  - Flick (Schnelles, kurzes Streichen über den Bildschirm, „Anstoßen“)
  - Touch and Hold (Gedrückt halten)
- **Multi-Touch:**
  - Pinch (zwei Finger auf dem Bildschirm zusammenführen, „Zwicken“)
  - Stretch (zwei Finger auseinander bewegen)

Hardwaretasten

Geräte auf denen Windows Phone als Betriebssystem läuft, haben sechs Hardwaretasten von denen einige auch als Touchfläche in das Display integriert sein können. Die genaue Position der jeweiligen Tasten kann je nach Hersteller variieren. Es folgt ein Beispiel für eine mögliche Realisierung.

Abbildung 2.32: Hardwaretasten [17]

1. **Power/Sleep**: Ein- bzw. Ausschalten durch gedrückt halten oder in Standby-Modus versetzen.
2. **Lautstärke** + und -
3. **Kamera**: Kann zwischen halb- und ganz gedrückt unterscheiden. Bei vollständig gedrückt wird die Kamera Anwendung gestartet bzw. in der Kamera-Anwendung eine Aufnahme gemacht. Wird die Taste in der Kamera-Anwendung halb gedrückt, wird die auto-focus Funktion aktiviert (ähnlich zu „normalen Kameras“). Wird die Taste gedrückt und gehalten, während sich das Gerät im Stand-by Modus befindet, startet ebenfalls die Kamera-Anwendung.
4. **Back (Zurück)**: Dient dazu einen Schritt in der Navigationshierarchie zurück zu gehen, d.h. die aktuelle Ansicht zu verlassen und zur vorherigen zurückzukehren. Des Weiteren kann auch der aktuelle Kontext oder Funktion beendet werden, wie zum Beispiel: Menü schließen, Suchfunktion beenden, eingeblendete Tastatur schließen usw.
5. **Start**: Bringt den Nutzer zurück zur Start-Ansicht. Eine eventuell laufende Anwendung wird pausiert.
6. **Search (Suche)**: Öffnet die Bing Suche

Start

Abbildung 2.33: Startbildschirm und Übersicht aller Anwendungen [17] [18]

Den Ausgangspunkt bildet der Startbildschirm. Hiermit verfolgt Microsoft einen neuen Ansatz. Der Startbildschirm kann vom Nutzer frei gestaltet werden. Er besteht aus Kacheln, welche einen Schnellzugriff auf häufig verwendete Informationen, Funktionen und Anwendungen ermöglichen. Die Kacheln können zu installierten Anwendungen aktuelle Informationen darstellen (zum Beispiel: Neue Nachrichten), aber auch Kontakte, Bilder, Video und Audio Dateien können als Kachel auf dem Startbildschirm platziert werden. Neben dem Startbildschirm, den sich der Nutzer frei gestalten kann, besteht die Möglichkeit über den Pfeil nach rechts, eine Übersicht aller installierten Apps zu erhalten und auf die Systemeinstellungen zuzugreifen. Mittels der Hardwaretaste „Start“, kann der Nutzer jederzeit zum Startbildschirm zurückkehren, auch aus laufenden Anwendungen heraus.

2.4.3 GUI Strukturierung

Einer der wichtigsten Punkte bei der Entwicklung einer Anwendung ist das Design. Die Hauptaufgabe ist es, ein leicht zu nutzendes Interaktionsmodell zu schaffen, d.h. wie werden Informationen dargestellt und wie soll die Navigation innerhalb der Anwendung ablaufen. Die Umsetzung wird schwieriger je mehr Ansichten zu verwalten sind. Um den Programmablauf und die Navigation zu strukturieren und den Entwickler zu unterstützen bietet Microsoft verschiedene Konzepte zur Gestaltung einer Anwendung an. Diese Strukturierungsarten werden im Folgenden näher erläutert. Um die Konzepte zu verstehen wird zuvor noch kurz die Panorama- und Pivot Control erklärt.

Panorama Control und Pivot Control

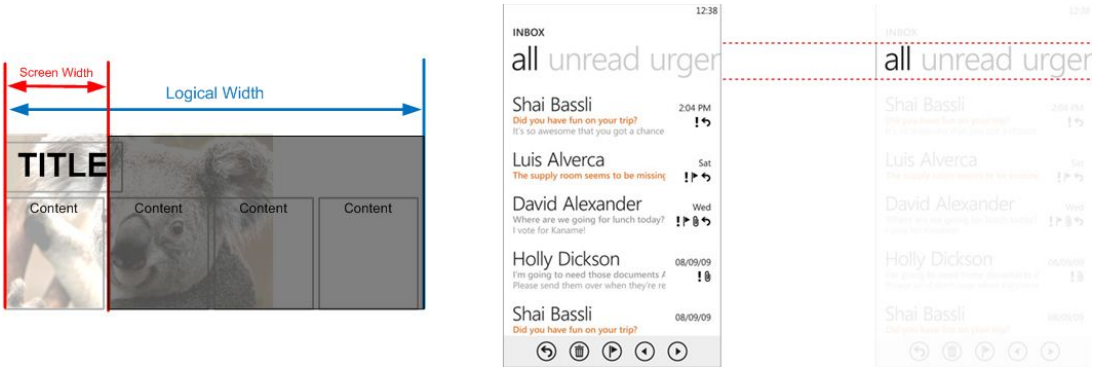

Abbildung 2.34: Konzept der Panorama und Pivot Control [17]

Die **Panoram Control** stellt verschiedene Ansichten/Bereiche horizontal nebeneinander dar. Hierbei sieht man schon einen kleinen Teil des nächsten Bereichs. Der Inhalt der einzelnen Bereiche kann frei gestaltet werden. Der Wechsel zwischen den einzelen Ansichten kann durch eine horizontale Wischgeste durchgeführt werden. Man kann „endlos “ navigieren, dass heißt nach dem letzten Bereich rechts folgt wieder die erste Ansicht.

Die **Pivot Control** ist mit den bereits bekannten Tabs vergleichbar. Man sieht am oberen Bildrand die möglichen Tabs zu denen man wechseln kann. Der Wechsel kann entweder durch eine Wischgeste oder durch Antippen eines anderen Eintrags durchgeführt werden.

Central Application Hub with Home Page Menu (Panorama or Pivot Control)

Bei diesem Konzept dient die Startseite als Hauptmenü, von dem aus in die verschiedenen Bereiche der Anwendung navigiert werden kann.
Beim Start der Anwendung befindet man sich auf Startseite mit dem Hauptmenü. Von hier hat man zwei Möglichkeiten sich weiter durch die Anwendung zu navigieren. So befinden sich auf der gleichen Navigationsebene wie das Hauptmenü, weitere Ansichten mit häufig genutzten Funktionalitäten und Informationen zwischen denen man über eine Wischgeste wechseln kann. Als Basis der höchsten Navigationsebene kann die Panorama- oder Pivot Control verwendet werden.
Zum anderen gelangt man über das Hauptmenü zu einer tieferen Schicht der Navigation. Die Navigationshierarchie bei Windows Phone ist vergleichbar mit einer Baumstruktur. So gelangt man über einen Menüpunkt oder durch Antippen eines Objektes einer Ansicht auf der höchsten Ebene, eine Stufe tiefer in der Struktur.
Durch die Zurück-Taste des Gerätes, gelangt man wieder eine Stufe höher in der Hierarchie. Eine mögliche Umsetzung zeigt die Music + Videos Anwendung, sowie ein Ausschnitt der Navigationshierarche in der folgenden Abbildung:

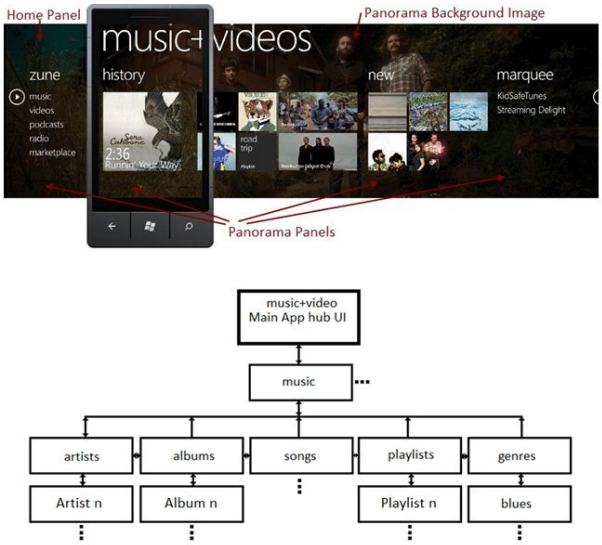

Abbildung 2.35: Music + Videos Anwendung und Navigationshierarchie [17]

In diesem Beispiel wurde die Panorama sicht als Grundlage genutzt. Es kann jedoch auch die Pivot Control (vergleichbar mit Tabs) verwendet werden. Auch bei der Gestaltung der Inhalte lässt Microsoft dem Entwickler freie Hand. So muss das Hauptmenü bzw. der Einstiegspunkt nicht zwangsweise der am weitesten links liegende „Ausschnitt“ sein. Ebenfalls müssen die Menüpunkte auf dem Hauptmenü keine Liste sein sondern können auch durch Bilder, Kacheln oder sonstige Elemente repräsentiert werden. Hier sind zwei Beispiele, welche die Möglichkeiten verdeutlichen.

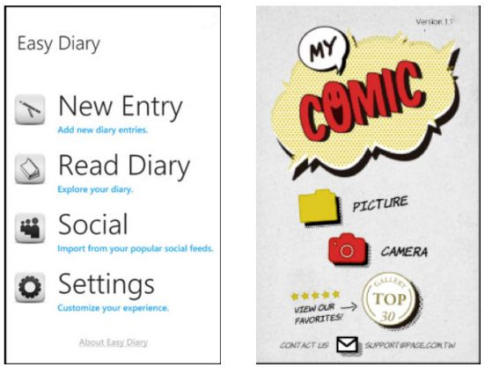

Abbildung 2.36: Frei gestaltbare Menüs [17]

Central Application Hub with Panel Areas (Panorama Control)

Abbildung 2.37: Konzept und Beispielumsetzung [17]

Bei diesem Konzept gibt es kein Hauptmenü sondern alle Ansichten befinden sich auf der gleichen Navigationsebene. Dies kann verwendet werden, wenn es keine Unterbereiche in der Anwendung gibt bzw. nicht benötigt werden. Alle Ansichten werden in einer Panorama-Ansicht (Panorama Control) horizontal nebeneinander angeordnet und der Nutzer kann durch eine Horizontale Wischgeste dazwischen wechseln.
Als Designhinweis gibt Microsoft an, dass die am weitesten links befindliche Ansicht als Startansicht bzw. Wiedereinstiegspunkt - wenn die Anwendung geschlossen wurde - dienen soll. Auch sollte man insgesamt nicht mehr als vier verschiedene Ansichten / Bereiche verwenden, da der Nutzer sonst schnell die Übersicht verliert. Bei mehr als vier Ansichten ist das zuvor genannte Designkonzept zu verwenden (Central Application Hub with Home Page Menu).

Application Tabs (Pivot Control)

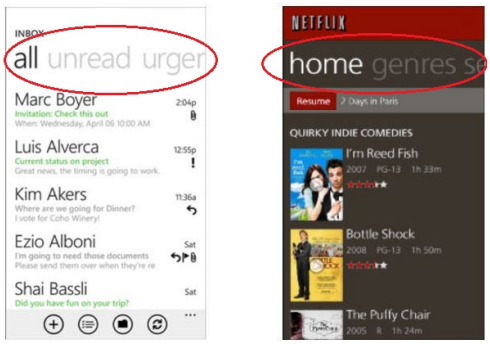

Abbildung 2.38: Pivot Control als Filter und Tabs [17]

Wie bereits bemerkt, ist die Pivot Control vergleichbar mit den Tabs von anderen Systemen. Sie soll verwendet werden, wenn es zu dem aktuellen Kontext bzw. Datensatz verschiedene Ansichten gibt. Sie dient als eine Art Filterfunktion und kann in andere Designkonzepte (z.B. Panorama Control) eingebettet werden, aber auch als Basis für die gesamte Anwendung dienen, wenn sich diese nur mit einem bestimmten Thema (z.B. Filme, Fußball, usw.) beschäftigt.
Eine Besonderheit bei der Darstellung ist, dass man am oberen Rand nur die nächsten zwei Tabs erkennen kann und nicht alle Möglichkeiten sofort sieht.

List with Details Drilldown

Abbildung 2.39: Beispiele für die Verwendung von Listen [17]

Häufig kommt es vor, dass man eine Vielzahl von Informationen darstellen muss. Hierfür bietet sich die Verwendung von Listen an. Diese Listen nehmen beliebig viele Listeneinträge auf und stellen dies vertikal untereinander dar. Sollen mehr Elemente dargestellt werden als auf die Anzeige passen, wird die Liste automatisch „scroll-fähig“. Durch Antippen eines Listeneintrages kann eine neue Ansicht mit Detailinformationen zu dem jeweiligen Eintrag angezeigt werden. Durch die Zurücktaste verlässt man die Detailansicht und kehrt zur Liste zurück. Um bei einer großen Liste schnell zu bestimmten Abschnitten zu springen, z.B. Namen die mit einem bestimmten Buchstaben beginnen, kann eine Sprungleiste am rechten Rand dargestellt werden. Listen können auch in die bereits genannten Gestaltungskonzepte bzw. UI Elemente (Panorama- bzw. Pivot Control) eingebettet werden.
Eine Besonderheit ist, dass die Listenelemente frei gestaltet werden können und keine Anforderung an Länge, Größe und Inhalt gestellt wird.

2.4.4 GUI Elemente

Application Bar & Application Bar Menu

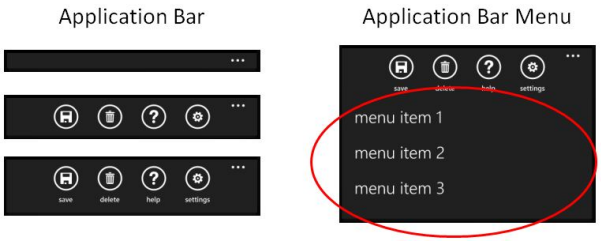

Abbildung 2.40: Arten der Application Bar und Beispiel eines Application Bar Menu [17]

Um den Nutzer schnellen Zugriff auf häufig verwendete Funktionen zu bieten, kann die Application Bar verwendet werden. Die bereitgestellten Funktionen können kontextsensitiv sein, müssen es jedoch nicht. Es handelt sich hierbei um eine Leiste, die sich immer über den Hardwaretasten befindet, d.h. bei vertikaler Ausrichtung am unteren Bildschirmrand

und bei horizontaler Ausrichtung links bzw. rechts am Bildrand. Die Application Bar kann minimiert dargestellt werden, wobei nur drei Punkte sichtbar sind welche als Platzhalter dienen und sich durch deren Antippen die Application Bar öffnen lässt. Des Weiteren gibt es eine Standarddarstellung, bei der die möglichen Funktionen durch Symbole dargestellt sind und über die drei Punkte können die Beschreibungen zu den Symbolen angezeigt werden. Die Application Bar kann maximal vier Symbole darstellen.
Zusätzlich zu den vier Symbolen in der Application Bar können noch weitere Funktionalitäten als Text im Application Bar Menu untergebracht werden. Dieses Menü erscheint bei „ausgeklappter“ Application Bar unter den Symbolen als eine Liste. Hier sind Funktionalitäten untergebracht die seltener verwendet werden oder nur schwer durch ein Symbol zu repräsentieren sind. Um die Konsistenz zu wahren werden die Texte automatisch nur als Kleinbuchstaben dargestellt. Das Application Bar Menu ist nicht hierarchisch und die Menüeinträge sind keine Untermenüs eines Symbols.
Microsoft gibt noch ein paar Hinweise für die Verwendung der Application Bar und des Application Bar Menu. Diese sind:

- Wenn möglich immer die Application Bar verwenden anstelle eines eigens kreierten Menüs.
- Verwendung der Standardfarben zur Vermeidung von Inkonsistenz und möglichen Problemen bei der Lesbarkeit.
- Kurze Texte für Menüeinträge, damit diese auch vollständig dargestellt werden können (maximal 14 bis 20 Zeichen).
- Nicht mehr als fünf Menüeinträge, da die Liste sonst nicht ausreicht und der Nutzer zum Scrollen gezwungen wird.

Status Bar

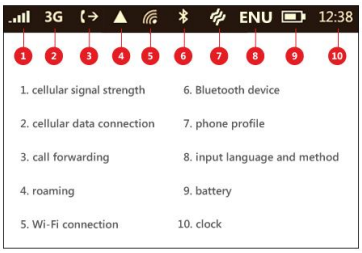

Abbildung 2.41: Status Bar Symbole [19]

Um den Nutzer jederzeit über den Systemzustand zu informieren wird die Status Bar verwendet. Sie befindet sich immer am oberen Bildschirmrand und ihr Inhalt kann nicht modifiziert werden. Es besteht jedoch die Möglichkeit sie auszublenden und nur wenn der Nutzer auf den oberen Bildschirmrand drückt, wird die Status Bar für ca. acht Sekunden eingeblendet. Standardmäßig wird nur die Systemzeit dauerhaft angezeigt und wenn der Nutzer auf den oberen Rand tippt werden alle anderen relevanten Symbole sichtbar. Auch bei wichtigen Änderungen, z.B. niedriger Ladezustand, blendet sich die Status Bar

ein. Die Status Bar enthält folgende Symbole, von links nach rechts geordnet: Signalstärke, Datenverbindung, Anrufumleitung, Roaming, WLAN Signalstärke, Bluetooth Status, Klingeltoneinstellung (Ton, Vibration, Stumm), Eingabestatus (Shift, Fn-Taste, Sprache) Ladezustand, Uhrzeit.

Benachrichtigungen

Um den Nutzer nicht nur über Änderungen des Systemzustands zu informieren, sondern auch auf neue Informationen durch Anwendungen hinweisen zu lassen, gibt es drei verschieden Arten von Benachrichtigungen: **Tile Notifications, Toast Notifications** und **Raw Notifications**.

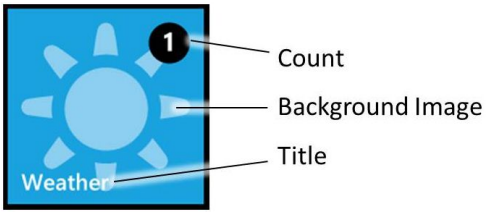

Abbildung 2.42: Aufbau der Tile Notification [17]

Die **Tile Notifications** sind Benachrichtigungen die den Nutzer nicht bei der Arbeit stören und nur zur Awareness (nur Hinweisen; Bewusstsein schaffen) eingesetzt werden. Sie erscheinen auf der Kachel der entsprechenden Anwendung auf dem Startbildschirm und bestehen meist nur aus einer Zahl die auf neue Informationen hinweist.

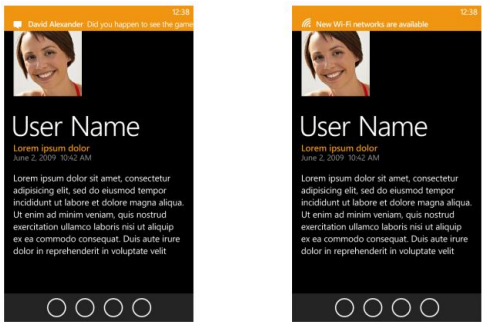

Abbildung 2.43: Beispiele einer Toast Notification [17]

Wichtige Änderungen die dem Anwender neue Möglichkeiten bieten (z.B. Neues WLAN gefunden) oder Informationen auf die der Nutzer reagieren kann (z.B. Neue SMS / eMail usw.), werden als **Toast Notifications** dargestellt. Diese Art der Benachrichtigung erscheint als ein farbiger Balken am oberen Rand und enthält ein verkleinertes Bild des Anwendungssymbols, einen fettgeschriebenen Titel und einem zusätzlichen Text. Ist der Text zu lang wird dieser automatisch gekürzt. Toast Notifications bleiben für zehn Sekunden sichtbar. Der Nutzer kann durch Antippen der Benachrichtigung zum jeweiligen Programm, welches diese Nachricht erzeugt hat, wechseln.

Abbildung 2.44: Beispiel einer Raw Notification [17]

Benachrichtigungen innerhalb einer Anwendung die eine Reaktion des Nutzers zwingend erfordern werden durch **Raw Notifications** dargestellt. Für die Darstellung und den Inhalt der Benachrichtigung ist allein die entsprechende Anwendung verantwortlich.

Sonstige UI Elemente

Abbildung 2.45: Beispiele für InkPresenter und Slider Control [17]

Auch Microsoft biete neben den bereits genannten UI Elementen weitere Standardelemente an, welche für bestimmt Aufgaben angepasst sind. So gibt es zum Beispiel für die Eingabe von Passwörtern die PasswordBox welche ein Textfeld enthält, bei dem die Eingabe durch * ersetzt wird oder fertige Picker zur Auswahl eines Datums. Auch gibt es Elemente die man bereits von Desktopsystemen kennt, sowie von den bereits untersuchten Betriebssystemen, beispielsweise RadioButtons und CheckBoxes.
Microsoft bietet aber auch neue Elemente wie den InkPresenter, welcher eine Fläche darstellt, auf der der Anwender mit seinen Fingern frei zeichnen kann. Ebenso gibt es für die Toucheingabe optimierte Steuerelemente wie die Slider Control, durch die der Anwender mit einem Schieberegler einen Wert auswählen kann.

2.4.5 Zusammenfassung

Microsoft versucht bei seinem mobilen Betriebssystem Windows Phone 7 einen Mittelweg zwischen Offenheit und Restriktion zu gehen. So existieren keine UI Guidelines, sondern das entsprechende Dokument heißt „User Experience Design Guidelines“. Was der Name allein schon vermuten lässt, hat sich auch durch die Untersuchung bestätigt. Microsoft legt einen großen Wert auf das Design und Konzept der Anwendung, d.h. im speziellen auf die Navigationsstruktur und den allgemeinen Ablauf bei der Bedienung. Es werden wenige Aussagen darüber getroffen, wie die einzelnen UI Komponenten zu nutzen sind oder welche Art der Verwendung verboten ist.
Generell erhält man den Eindruck, dass es sich um eine Art „Baukasten“ handelt. So lässt sich verallgemeinernd sagen, dass man die UI Element untereinander frei kombinieren und ineinander einbetten kann, jedoch bestehen nur wenige bzw. begrenzte Möglichkeiten die einzelnen Komponenten zu modifizieren.
Microsoft setzt bei seinem User Interface auf die Neuentwicklung mit dem Codenamen „Metro“, welche auch Einzug in Windows 8 halten soll. Hierbei weicht Microsoft auch teilweise von den aus anderen mobilen Betriebssystemen bekannten Konzepten ab und stellt eigene innovative Ansätze bereit. Als Beispiel hierfür seien die frei gestaltbare „Start“-Ansicht und die Panorama Control erwähnt.

2.5 iPhone

Abbildung 2.46: Apple Logo und iPhone 4 [20] [21]

Das iPhone von Hersteller Apple läutete das Zeitalter der Smartphones ein. Das Betriebssystem iOS, welches auch als Betriebssystem für das iPad dient, wurde seit dem Erscheinen stetig weiterentwickelt und liegt seit wenigen Wochen in der Version 5 vor.
Als Grundlage für die folgenden Betrachtungen dienen das entsprechende Dokument von Apple („iOS Human Interface Guidelines")[22], ein privates iPhone 3GS, sowie die eigene Erfahrung als Entwickler von Apps für iPhone und iPad.

2.5.1 Allgmeine Anforderungen

Auch von Apple gibt es allgemeine Anforderung die an Anwendungen gestellt werden. Speziell bei Apple sind diese interessant, da jede App bevor sie in den hauseigenen AppStore gelangt überprüft wird. Im Folgenden werden jedoch nur die allgemeinen Anforderungen näher beleuchtet, denn Apple behandelt diese Thematik sehr genau und widmet der Nutzererfahrung ein eigenes Kapitel. Es werden hierbei auch Thematiken beschrieben die in dieser Ausarbeitung zu weit führen würden, wie zum Beispiel die emotionale Bindung des Anwenders an das Gerät und die Nutzeroberfläche.

Für Apple stellt die Nutzeroberfläche (UI) eines der wichtigsten Elemente bei der Entwicklung von Anwendung für mobile Geräte dar. Denn die Oberfläche ist die Schnittstelle zum Anwender und dessen erster Kontakt mit einer Anwendung. So kann eine prinzipiell gute Anwendung mit umfangreicher Funktionalität und effizienten Algorithmen, an einer schlechten Menüführung oder unschönen UI Elementen scheitern. Um dies zu vermeiden stellt Apple sechs Prinzipien für eine erfolgreiche Oberfläche vor:

- **Aesthetic Integrity**: Hierin versteht Apple, dass eine Anwendung die in sie gestellten Erwartungen bezüglich des Erscheinungsbildes erfüllt. Zum Beispiel wird von einer Finanzanwendung erwartet, dass sie klar und einfach gestaltet ist und keine unnötigen Effekte und Verzierungen enthält, wohingegen bei Spielen genau das Gegenteil gilt.

- **Consistency**: Konsistenz bei der Gestaltung der Oberfläche hilft dem Nutzer sich schnell zurecht zu finden und bereits gemachte Erfahrungen bei der Bedienung wieder anzuwenden. Dies gilt zum einen für Programmnavigation und zum anderen auch für die einzelnen UI Elemente.

- **Direct Manipulation**: Wenn der Nutzer Objekte die er sieht direkt benutzen und mit ihnen interagieren kann, wird ihm die Bedienung erleichtert und er entwickelt ein Verständnis für die einzelnen Funktionen. Vor allem durch die Nutzung des Multi-Touch Bildschirms und der daraus resultierenden Möglichkeit Gesten zu erkennen und zu verwenden, wird die direkte Interaktion mit den UI Elementen unterstützt und man kann auf andere Eingabeelemente wie eine Hardwaretastatur verzichten.

- **Feedback**: Rückmeldung und Hinweise des Systems auf Benutzerinteraktionen sind äußerst wichtig und sollten sofort erfolgen. So liefern die Standard UI Elemente und Anwendungen ein gutes Beispiel hierfür. Auf jede Eingabe des Anwenders liefert das System eine Rückmeldung, hierbei ist es egal ob diese visuell oder akustisch ist. Aber auch bei jeder Operation die länger als ein paar Sekunden dauert, muss der Nutzer über die Vorgänge informiert werden, zum Beispiel durch eine Lade- oder Fortschrittsanzeige.

- **Metaphors**: Damit sich jeder Nutzer sofort in der virtuellen Welt zurecht findet, sollten Funktionen und Objekte auf dem Bildschirm reale Gegenstände als Vorbild haben. Ein gutes Beispiel hierfür sind Ordner. Anwender wissen, dass sie Objekte in Ordner ablegen und wiederfinden können. Wichtig ist es jedoch die Balance zwischen den realen Vorbildern und den virtuellen Möglichkeiten zu halten. So kann ein Anwender einen Softwareordner mit mehr Inhalt befüllen, als dies mit dem realen Vorbild möglich ist.

- **User Control**: Jede Aktion in einer Anwendung sollte vom Nutzer ausgehen und kontrolliert werden. Des Weiteren soll eine App den Anwender auf mögliche und sinnvolle Aktionen hinweisen, bzw. auch vor potentiell gefährlichen Konsequenzen warnen (z.B. Löschen?). Generell fühlt sich der Nutzer wohler, wenn sich ihm die Bedienung als einfach und sinnvoll darstellt und ihm die Möglichkeit bietet jederzeit Entscheidungen und Operationen zurückzunehmen bzw. abzubrechen.

Diese allgemeinen Anforderungen finden sich alle in den Standardanwendungen, wie Kalender und Mail, wieder. Sie sind alle sehr sinnvoll und müssen auch bei der Gestaltung eigener Apps berücksichtigt werden. Apple formuliert diese Anforderung noch sehr detailliert mit vielen Beispielen weiter aus. Dies würde jedoch an dieser Stelle zu weit führen und es sei auf den Abschnitt „User Experience Guidelines“ des Dokuments „Human Interface Guidelines“ verwiesen.

2.5.2 Bedienkonzept

Hardwaretasten

Ein Merkmal des iPhone ist, dass es mit minimalen Hardwaretasten auskommt. Um einen Überblick über das Gerät zu schaffen und da zum Beispiel der Home-Button für die weiteren Betrachtungen wichtig ist, folgt eine kleine Übersicht über die Tasten des iPhone.

Abbildung 2.47: Hardwaretasten am Beispiel iPhone 4 [23]

1. An/Aus bzw. Standby
2. Stummschalten
3. Lautstärke + und -
4. Home-Button

Gesten

Das iPhone besitzt keine Hardwaretastatur und wird weitestgehend über den Touchscreen bedient. Das iPhone erkennt Single- und Multi-Touch-Gesten auf die der Anwendungsentwickler zugreifen kann um somit verschieden Funktionalitäten zu implementieren. Durch die Verwendung von Gesten wird dem Nutzer das Gefühl der direkten Manipulation von Objekten gegeben. Apple erwartet, dass die Standardgesten durch die Anwendungen unterstützt und genutzt werden, damit der Nutzer sich schnell zurecht findet und eine gewisse Konsistenz gewahrt bleibt.
Die Standardgesten sind:

- **Tap**: einfaches Antippen, ähnlich zu einem Mausklick
- **Drag**: das Ziehen von Objekten, zum Scrollen oder Bewegen von Elementen
- **Flick**: kurzes Anstoßen, zum Beispiel um schnell zu scrollen

- **Swipe**: Wischgeste, zum Beispiel über einen Tabelleneintrag wischen um die Löschfunktion aufzurufen
- **Double Tap**: doppeltes Antippen, zum rein- bzw. rauszoomen
- **Pinch open & close**: Zwickgeste, Daumen und Zeigefinger auseinander oder zusammen Bewegen, zum stufenlosen rein- und rauszoomen
- **Touch and hold**: Bildschirm Antippen und gedrückt halten, um bei editierbaren Text den Cursor aufzurufen
- **Shake**: Das Gerät Schütteln um eine Aktion abzubrechen oder rückgängig zu machen

Home-Screen

Abbildung 2.48: Suchfunktion und Beispiele für Ordner und Benachrichtigungen [24]

Den zentralen Ausgangspunkt bildet der Home-Screen. Hier befinden sich die installierten Apps in Form von Icons. Ebenso befinden sich am unteren Rand vier frei konfigurierbare Plätze für die am häufigsten genutzten Anwendungen. Standardmäßig befinden sich hier Telefon, Mail, Internet (Safari) und Musik. Der Home-Screen erweitert sich um weitere Ansichten, falls nicht genügend Platz für eine neue App und deren Icon besteht. Es können 4 x 4, also 16 Icons, auf einer einzelnen platziert werden. Die Anzahl der Ansichten wird am unteren Rand durch graue Punkte dargestellt. Damit man nicht zu schnell die Übersicht verliert, können die Appicons in Ordner zusammengefasst werden. In einem Ordner finden 12 Apps Platz. Es kann jedoch keine größere Ordnerstruktur generiert werden, da in einem Ordner keine weiteren Ordner erstellt oder hinzugefügt werden dürfen.
Die Anwendungssymbole erfüllen neben dem Starten und Löschen von Anwendungen noch einen weiteren Zweck: Eine Anwendung die sich im Hintergrund befindet, kann ihr Symbol auf dem Home-Screen mit einer Zahl versehen, um so zum Beispiel auf die Anzahl der neuen Nachrichten hinzuweisen.
Neben den Ansichten mit den auf dem Gerät installierten Anwendungen und den Ordnern befindet sich ganz links die Suchfunktion. Hiermit kann das Gerät, anwendungsübergreifend durchsucht werde. So kann man zum Beispiel nach einzelnen Anwendungen, Daten von Kontakten (Telefonnummer, Name, Adresse), eMails oder Musik suchen.

Home-Button

Abbildung 2.49: Multitasking Funktionen

Die zentrale Steuerungskomponente neben dem Touchscreen stellt der Home-Button dar. Über den Home-Button gelangt man jederzeit aus einer laufenden Anwendung zurück zum Home-Screen. Als weitere Funktionalität kann ab iOS 4, durch Doppeldrücken des Home-Button die Multitasking Funktion geöffnet werden. Hierbei öffnet sich am unteren Rand eine Leiste in der alle Programme aufgelistet sind, die sich im Hintergrund befinden. Durch drücken des Home-Button wird nämlich ein Programm nur in den Hintergrund gesetzt. Möchte man ein Programm vollständig schließen, kann dies nur durch Beenden im Multitasking Menü geschehen. Neben der Verwaltung der Anwendungen im Hintergrund kann hier auch die Systemlautstärke verändert oder auf die Musikfunktionen „Lied zurück oder vor“ zugegriffen werden. Ebenso lässt sich hier, das automatische Rotieren des Bildschirms, je nach Position des Geräts, abschalten. Zwischen den einzelnen Funktionen kann durch eine horizontale Wischgeste gewechselt werden.

Mitteilungszentrale

Abbildung 2.50: Beispiel für die Mitteilungszentrale [25]

Ein in iOS sehr neues Bedienelement ist die Mitteilungszentrale (Notificationscenter). Diese erschien erst vor wenigen Wochen mit dem Update von iOS auf die Version 5. Sie ähnelt sehr stark der Benachrichtigungszentrale von Android. Die Mitteilungszentrale kann jederzeit durch eine vertikale Wischgeste ein- und wieder ausgeblendet werden. Hier

können Programme die im Hintergrund laufen, für den Nutzer wichtige Informationen hinterlassen. Sobald es eine neue Nachricht gibt, wird der Nutzer durch eine kurzzeitige Einblendung am oberen Bildschirm, auch wenn eine andere Anwendung aktiv ist, darauf hingewiesen und kann durch Antippen der Nachricht zum entsprechenden Programm springen.
In der Mitteilungszentrale werden für Benachrichtigungen von Apps, das Icon und die jeweilige Nachricht angezeigt. Durch Antippen der Meldung kann man zu der jeweiligen Anwendung wechseln. Apple eigene Apps, wie Aktien oder Wetter, haben eine grafisch aufwändigere Gestaltung und man kann mit diesen auch interagieren. So kann man bei der Wetterapp durch eine horizontale Wischgeste zwischen Wochen- und Tagesvorhersage umschalten. Für Apps die nicht von Apple sind steht diese Möglichkeit noch nicht zur Verfügung, es ist jedoch sehr wahrscheinlich, dass es in Zukunft auch für „normale“ Entwickler möglich sein wird diese Funktion zu implementieren.
Die Mitteilungszentrale kann vom Nutzer frei gestaltet werden, d.h. der Nutzer legt in den Einstellungen fest welche Anwendungen, welche Art von Benachrichtigungen und wie viele Benachrichtigungen pro App angezeigt werden sollen.

2.5.3 GUI Strukturierung

Tab Bar Controller

Abbildung 2.51: Beispiele für die Tab Bar

Die Tab Bar erscheint am unteren Bildschirmrand als ein schwarze Fläche mit Schaltflächen. Eine Schaltfläche besteht aus einem Icon und einem kurzen Text. Durch Antippen einer Schaltfläche kann die aktuelle Ansicht gewechselt werden. Die aktuell ausgewählte Ansicht wird hervorgehoben in dem sie leuchtet. Die Tab Bar ist also für Apps geeignet um zum Beispiel eine Art Hauptmenü zu ersetzen bzw. um zwischen verschiedenen Aspekten einer Anwendung zu wechseln. So kann zum Beispiel in einer Nachrichtenanwendung zwischen den einzelnen Rubriken (Neues, Sport, Wirtschaft, usw.) gewechselt werden.
In der Tab Bar können maximal fünf Icons dargestellt werden. Sollte der Entwickler, der Tab Bar mehr als fünf Schaltflächen hinzufügen, erscheint automatisch als fünfte Schaltfläche das „Mehr“-Element. Durch Antippen von „Mehr“ werden alle weiteren Schaltflächen der Tab Bar in einer Table View dargestellt. Der Nutzer kann in dieser Übersicht, dann

frei entscheiden welche Schaltflächen bzw. Ansichten er auf den ersten vier Positionen haben möchte und kann auch die Reihenfolge frei wählen oder einfach die gewünschte Ansicht anzeigen. Diese Funktionalität bringt der Tab Bar Controller standardmäßig mit sich und ist nicht vom Entwickler zu implementieren.
Ein weitere Funktionalität der Tab Bar ist, dass die Anwendung über Neuigkeiten in einer speziellen Ansicht informieren kann. Dies geschieht über eine sogenannte Badge. Dies sind kleine rote Kreise mit weißen Zahlen die über dem jeweiligen Icon erscheinen. Ein Beispiel hierfür wäre eine App eines sozialen Netzwerks, welche mich während ich mir meine Freunde anschaue über neue Nachrichten informieren kann.
Als Richtlinie gibt Apple an, dass die Tab Bar nicht für kontextsensitive Funktionalitäten verwendet werden darf, sondern dafür die Toolbar zu nutzen ist. Ebenso soll, wenn immer dies möglich ist, ein Standardicon bevorzugt verwendet werden.

Navigation Controller

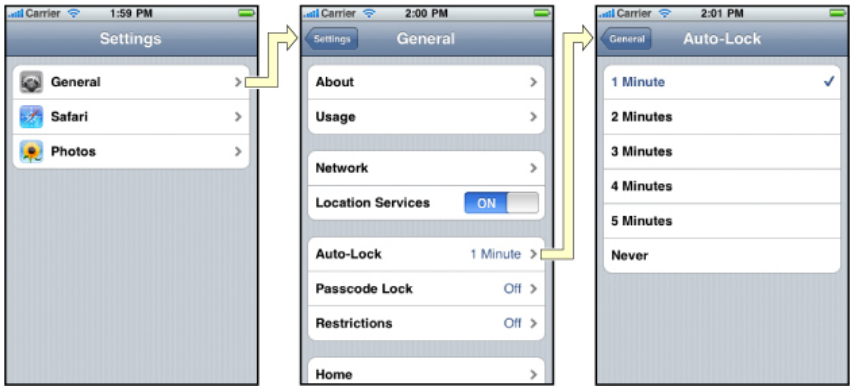

Abbildung 2.52: Funktionsweise des Navigation Controller [26]

Häufig ist es notwendig bei der Informationsdarstellung eine Hierarchie nachzubilden. Hierfür ist der Navigation Controller gedacht. Er unterstützt den Entwickler dadurch, dass er selbstständig am oberen Rand einen zurück Pfeil auf die vorherige Ansicht erzeugt und direkt die „Zurück"-Funktionalität vorhanden ist. Die Navigation Bar enthält den Zurückpfeil mit dem Titel der vorherigen Ansicht, den Titel der aktuellen Ansicht und einen Knopf (z.B. neu laden der aktuellen Ansicht). Sie erscheint immer am oberen Rand des Bildschirms und es kann zwischen verschiedenen vordefinierten Styles gewählt werden, zum Beispiel schwarz, halbdurchsichtig, usw.
Der Navigation Controller wird meistens dazu verwendet um detailliertere Informationen zum einem Objekt auszuwählen. Hierfür drückt der Nutzer zum Beispiel auf einen Personennamen und erhält eine Ansicht mit Anschrift, Bild und Telefonnummer. Über den Zurückpfeil kann der Anwender ganz einfach zur vorherigen Ansicht zurückkehren. Der Navigation Controller kann in den Tab Bar Controller eingebettet werden, um die Inhalte für den Nutzer besser darzustellen und die Navigation zu vereinfachen.
Von Apple ist es nicht gewünscht selbst erstellte Pfeile zu verwenden mit denen es möglich wäre mehrere Ebenen zurück zu springen. Es darf immer nur ein einfache Pfeil sein, der den Anwender eine Ebene in der Hierarchie zurück bringt.

Table View

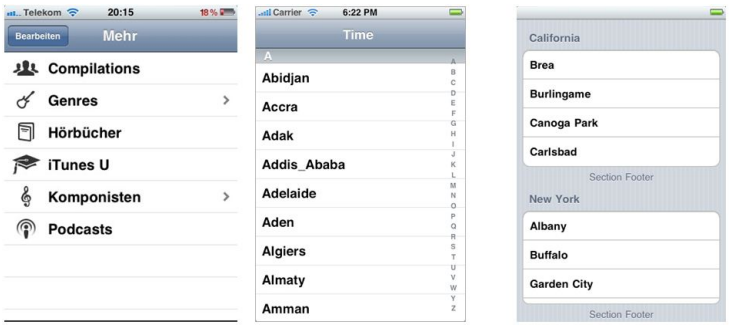

Abbildung 2.53: Table View Style Plain und Grouped [27][28]

Eines der am häufigsten verwendeten Elemente ist die Table View. Sie wird verwendet, um viele Informationen übersichtlich gegliedert darzustellen. Die Table View stellt die Informationen in Form einer Tabelle dar, die aus einzelnen Zellen besteht. Die Tabelle kann durch Überschriften in Sektionen unterteilt werden und wenn mehr Inhalte darzustellen sind, als auf den Bildschirm passen, wird die Tabelle direkt „scroll"-fähig.
Apple bietet zwei vordefinierte Arten von Table Views, dies sind:

- **Plain**: Die Zellen der Tabelle bedecken den Bildschirm vollständig und können durch Überschriftzellen in Sektionen getrennt werden. Der Zellenhintergrund ist standardmäßig weiß, darf aber verändert werden. Ebenso besteht die Möglichkeit am rechten Rand eine vertikale Liste einzublenden, mit der zu bestimmten Stellen der Tabelle gesprungen werden kann. Zum Beispiel bei einer alphabetisch sortierten Liste zum entsprechenden Buchstaben zu springen.
- **Grouped**: Unterteilt die Zellen in zusammenhängende Gruppen von Zellen, wobei die einzelnen Zellen nicht vollständig von links nach rechts gehen. Die Gruppen sind durch freien Raum voneinander getrennt und können noch mit einer Überschrift versehen werden.

Table View Cell

Neben der Art der Tabelle, kann auch bei der einzelnen Zelle zwischen verschiedenen vordefinierten Arten gewählt werden. Diese unterscheiden sich primär durch die Anzahl der darzustellenden Informationen. Die Zellen können auf der rechten Seite durch zusätzliche Elemente wie Haken für die Auswahl, Indikatoren für detaillierte Informationen (Pfeil rechts) usw. erweitert werden.
Es existieren vier verschieden Arten von Zellen:

- **Default**: Die Zelle verfügt über ein optionales kleines Bild (Thumbnail) links und einen schwarzen Text (Titel).
- **Subtitle**: Genau wie Default, jedoch zusätzlich einen Untertitel in kleinerer Schriftart und in grau.

- **Value 1**: Zelle besteht aus einem Titel in schwarzer Schrift auf der linken Seite und einem Untertitel in kleinerer blauer Schrift auf der rechten Seite.
- **Value 2**: Wie Value 1, jedoch sind die Seiten vertauscht, d.h. schwarzer Titel rechts und blauer Untertitel links

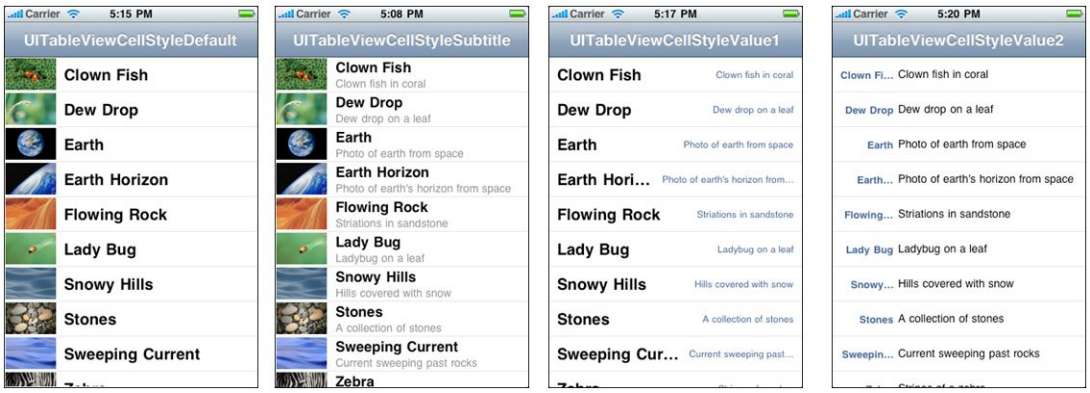

Abbildung 2.54: Standardtypen von Zellen [27]

Value 1 und Value 2 sind vor allem für die Darstellung von Attribut - Wertpaaren geeignet. Hier gilt, schwarzer Titel für den Wert und blauer Untertitel für den Attributnamen. Es können jedoch auch eigene Kreationen von Zellen erstellt werden, diese sind jedoch etwas anspruchsvoller zu erzeugen, da auf die Zellenhöhe, Bildgröße usw. selbst geachtet werden muss.

Bei der Verwendung von Table Views gibt es noch Hinweise und Richtlinien die es zu beachten gilt. So soll man es vermeiden verschiedene höhen der Zellen innerhalb einer Plain Tabelle zu verwenden. Ebenso gilt es, bei umfangreicheren Tabellen oder wenn die entsprechenden Informationen erst geladen werden müssen, darauf zu achten, dass man bereits erhaltenen Informationen darstellt und nicht erst auf alle Daten wartet. Des Weiteren sollte man bei einer langsamen Internetverbindung daran denken, den Nutzer über den Ladevorgang zu informieren um zu verdeutlichen, dass die Anwendung noch aktiv ist.

2.5.4 GUI Elemente

Status Bar

Die Status Bar ist eine Leiste mit gerätespezifischen Informationen, die für den Nutzer wichtig sind. Sie befindet sich immer am oberen Rand und die Angezeigten Informationen sind fest definiert, d.h. sie können auch durch den Entwickler nicht modifiziert werden. Der Entwickler hat jedoch die Möglichkeit die Status Bar komplett auszublenden, z.B. bei Spielen oder für Multimedia Anwendungen die im Vollbildmodus laufen. Ebenso kann aus drei vordefinierten Typen gewählt werden, dies sind: grau (Standard), schwarz und halbdurchsichtiges schwarz. In der Status Bar werden folgende Informationen dargestellt:

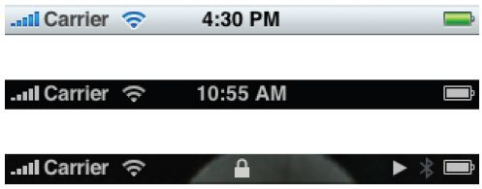

Abbildung 2.55: Die verschiedenen Arten der Status Bar [22]

- Signalstärke
- Flugzeugmodus (Zeigt an, dass alle Funkkomponenten deaktiviert sind)
- WLAN Qualität
- Edge oder 3G Verbindung
- Gesperrt (Eingabeschutz aktiviert)
- Uhrzeit
- Abspielsymbol (Dreieck nach rechts, zeigt an, dass die Musikwiedergabe aktiv ist)
- Alarm (Wird angezeigt wenn zum Beispiel ein Wecker aktiv ist)
- Bluetooth Symbol
- Batterieanzeige
- Freisprechgerät angeschlossen

Toolbar

Abbildung 2.56: Beispiel einer Toolbar [22]

Die Toolbar, soll verwendet werden um den Nutzer kontextsensitive Funktionalitäten bereit zu stellen. Sie befindet sich immer am unteren Rand des Bildschirms und es ist darauf zu achten, dass hier die wichtigsten Funktionen für ein Objekt untergebracht werden. Die Anzahl und das Aussehen der Symbole kann sich je nach ausgewähltem Objekt in der aktuellen Ansicht ändern. Wenn möglich sollen immer die vom System bereitgestellten Icons verwendet werden, um so die Konsistenz zu wahren. Ebenso soll man eine Mindestgröße von 44 x 44 Pixel als Touchbereich für eine Schaltfläche einhalten, weil der Nutzer sonst Schwierigkeiten hat das gewünschte Symbol zu treffen. Das Erscheinungsbild der Toolbar lässt sich dahingehend modifizieren, dass man die Hintergrundfarbe frei wählen kann und seit iOS 5 ist es sogar möglich ein Bild als Hintergrund zu wählen. Des Weiteren hat die Toolbar ein Attribut bekommen mit der man die Position zwischen Top, Bottom oder Any wählen kann. Dies wurde jedoch noch nicht in die entsprechende Guideline eingearbeitet.

Alert

Abbildung 2.57: Beispiel eines Alert [22]

Der Alarm wird dazu verwendet Benutzer über wichtige Ereignisse zu informieren, die das Verhalten der Anwendung oder sogar des ganzen Systems beeinflussen. Ein Alarm erscheint immer in der Mitte des Bildschirms und befindet sich immer im Vordergrund. Er informiert über wichtige Änderungen oder Ereignisse an die nicht zwingend durch den Nutzer ausgelöst werden müssen (z.B. niedriger Ladezustand usw.). Der Nutzer kann erst weiterarbeiten wenn der Alarm bestätigt wurde, denn solange er angezeigt wird, reagiert der Bildschirm nicht auf weitere Eingaben.
Ein Alarm besteht mindestens aus einer Schaltfläche, mit der er beendet werden kann, sowie einem Titel und einer kurzen Beschreibung des Alarms. Das Aussehen ist vom System vorgegeben und kann nicht durch Anwender oder den Entwickler verändert werden.
Apple gibt für die Verwendung noch einige Hinweise. So soll man kurze und prägnante Titel und Beschreibungen wählen, da zum Beispiel ein zu langer Beschreibungstext dazu führt, dass er gescrollt werden muss. Ebenso sollen die Alarme sehr sparsam verwendet werden, da sie die Interaktionen des Nutzers unterbrechen und erst abgearbeitet werden müssen. Soll der Anwender mehr als zwei Antwortmöglichkeiten haben, so schlägt Apple vor über die Verwendung eines Action Sheet nachzudenken.

Action Sheet

Abbildung 2.58: Beispiele für einen Action Sheet [22]

Der Action Sheet stellt dem Nutzer verschiedene Möglichkeiten zur Verfügung, wie er fortfahren möchte. Im Gegensatz zum Alert, wird der Action Sheet immer nur als Reaktion auf eine Benutzereingabe eingeblendet, enthält kontextsensitive Auswahlmöglichkeiten und bezieht sich nur auf die aktuelle Anwendung und nicht auf Systemereignisse.
Der Action Sheet wird von unten über die aktuelle Ansicht gelegt und der Hintergrund

wird ausgegraut. Der Nutzer muss eine der möglichen Aktionen auswählen bzw. Abbrechen wählen, da der Toucscreen sonst auf keine weiteren Interaktionen reagiert. Der Action Sheet wird zum Beispiel in der Safari-App verwendet, wo der Nutzer auswählen kann was er mit der aktuellen Webseite machen möchte (Lesezeichen, Drucken, Per eMail versenden usw.).
Nicht nur als Auswahlmöglichkeit zwischen verschiedenen Aktionen kann der Alert Sheet verwendet werden, sondern auch als Hinweis auf potentiell zerstörerische Operationen, wie zum Beispiel das Löschen eines Kalendereintrags. Hierbei dient der Action Sheet als Dialog und ersetzt den von Desktopsystemen bekannten Hinweis bzw. die Frage: „Wollen Sie wirklich löschen?“. Hierfür stellt Apple sogar eine eigene Schaltfläche für den Action Sheet bereit, diese befindet sich an höchster Stelle und ist rot hinterlegt.

Picker

Abbildung 2.59: Date & Time Picker und Custom Picker [22]

Zur Auswahl zwischen verschiedenen Elementen bietet Apple die sogenannten Picker. Diese erinnern an die Anzeige von Glücksspielautomaten. Hier kann der Nutzer durch vertikale Wischgesten ein Rad mit den verschiedenen Werten drehen bis er die gewünschte Auswahl getroffen hat. Ein Picker kann bis zu vier Räder besitzen welche sich unabhängig voneinander steuern lassen. Der Vorteil von Pickern ist die einfachere Auswahl von Werten, da der Nutzer diese nicht eintippen muss sondern bequem auswählen kann.
Für häufige Zwecke stellt Apple bereits vorgefertigte Picker zur Verfügung, die Date- und Time-Picker.
Diese liegen in vier Varianten vor:

- **Date and Time**: Auswahl des Datums und einer Uhrzeit (je nach Zeitformat optional mit AM / PM)
- **Time**: Zeitauswahl mit Stunden und Minuten, optional mit AM /PM
- **Date**: Datumsauswahl mit Monat, Tag und Jahr
- **Countdown Timer**: Auswahl von Stunden und Minuten bis maximal 23 Stunden und 59 Minuten

Diese Picker haben Standardmäßig die gleiche Größe wie die virtuelle Tastatur.
Neben diesen vorgefertigten Varianten, kann der Picker auch mit frei wählbaren Werten genutzt werden. Hierbei wählt der Entwickler die Anzahl und den Inhalt der jeweiligen

Räder. Apple gibt an, dass die Picker nur für eine überschaubare Anzahl von Auswahlmöglichkeiten verwendet werden sollten bzw. wenn der Nutzer die Auswahlmöglichkeiten bereits kennt. Für die Darstellung und Auswahl aus einer großen Anzahl von Werten, kann eine Table View verwendet werden.

Text Field

Abbildung 2.60: Tastaturarten: Standard, Nummer & eMail [22]

Um dem Nutzer die Eingabe eines beliebigen Textes zu ermöglichen, gibt es das Text Field. Die Verwendung dieses Standardelements bringt sowohl Vorteile für den Entwickler, als auch für den Nutzer. So besitzt das Text Field schon fertige Animationen und Funktionalitäten. Beispielsweise fährt beim Antippen des Text Field automatisch die Tastatur von unten in das Bild und der Platzhaltertext verschwindet. Des Weiteren wird bereits die gesamte Editierfunktionalität, wie Markieren, Ausschneiden und Kopieren bereitgestellt. Somit ist die Texteingabe anwendungsübergreifend konsistent gehalten.
An das Text Field ist auch das Erscheinungsbild der Tastatur gekoppelt. Man kann die Tastatur nicht vollständig frei modifizieren, sie ist immer abhängig von der gewählten Systemsprache, aber es besteht die Möglichkeit zwischen verschiedenen vordefinierten Arten zu wählen. Die Auswahlmöglichkeit ist sehr umfangreich, es existieren elf verschiedene Tastaturarten die alle Möglichkeiten abdecken, beispielsweise: Nummernblock, eMail, Dezimalzahl, Twitter, URL usw.

Sonstige UI Elemente

Abbildung 2.61: Switch, Segmented Control & Slider [22]

Neben den bereits vorgestellten UI Elementen, stellt Apple noch viele weitere zur Verfügung. Darunter sind auch Elemente die den von Desktopsystemen bekannten ähneln bzw. deren Funktion erfüllen, wie Check Boxes und Radio Buttons. Es fällt jedoch auf, dass diese nicht nur einfache Kopien ihrer Vorbilder sind, sondern dass auch bei diesen, auf den ersten Blick, simplen Elementen die Besonderheiten der Touchbedienung berücksichtigt wurden. Ebenso sind sehr schöne Metaphern für die einzelnen Elemente gewählt worden.

So ist die Check Box ein **Switch** (Schalter), welcher nicht nur durch Antippen umgeschaltet werden kann, sondern auch durch eine Wischgeste. Hierbei ändert der Schalter nicht nur einfach seine Position, sondern es geschieht in einer kleinen Animation. Dadurch erhält der Nutzer sofort positives Feedback und findet sich schneller bei der Bedienung zurecht.
Ebenso sind die Radio Buttons nun eine **Segmented Control**. Auch diese erfüllt den gleichen Zweck wie ihr Vorbild, jedoch werden die Auswahlmöglichkeiten nicht einfach durch Punkte repräsentiert, sondern durch zusammenhängende Schaltflächen, ähnlich wie bei alten Kassettenspielern.
Die Element besitzen standardmäßig die entsprechenden Animationen und Funktionalitäten und müssen nicht erst durch den Entwickler integriert werden. Es existiert ebenfalls ein Schieberegler (**Slider**) mit dessen Hilfe der Anwender stufenlose einen bestimmten Parameter, wie zum Beispiel die Lautstärke, wählen kann.

2.5.5 Zusammenfassung

Abschließend lässt sich sagen, dass Apple eine sehr ausführliche Guideline für die Programmierung von Anwendung bereitstellt. Sehr interessant ist auch, welche Gedanken sich Apple bei der Gestaltung der einzelnen Elemente gemacht hat und sogar psychische Aspekt berücksichtigt. Eine zentrale Vorgabe für die Entwicklung ist an dieser Stelle hervorzuheben; das Finden von Metaphern. Durch die Einhaltung dieser Forderung schafft man es, dass sich auch Menschen die wenig mit Computern und Smartphones zu tun haben, sehr schnell zurecht finden, da sie Erfahrungen aus der realen Welt einfach in die virtuelle Übertragen können (zum Beispiel bei Switch und Picker).
Apple macht generell starke Einschränkungen und Vorgaben wie die einzelnen Elemente zu verwenden sind und in welchem Maße diese modifiziert werden können. Dies äußert sich nicht nur in den Guidelines, sondern es wird ganz einfach dem Entwickler nicht ermöglicht über den Quellcode diverse Veränderungen durchzuführen.
Diese starken Einschränkungen haben jedoch einen ganz einfachen Grund: die Konsistenz. Dadurch dass sich die UI Elemente jederzeit gleich verhalten und auf Eingaben reagieren, wird dem Nutzer die Bedienung erleichtert. Sobald er einmal die Funktionsweise eines Steuerelements verstanden hat, kann er es jederzeit und in jeder anderen Anwendung wieder verwenden und hat keine Probleme bei der Nutzung.
Die Einschränkungen haben aber auch für den Entwickler eher Vor- als Nachteile. Zum einen bietet Apple für eigentlich jede Eventualität das entsprechende UI Element bzw. haben die einzelnen Elemente verschiedene Attribute über die man diese anpassen kann. Ebenso stellen die Standardelemente bereits viele Funktionen und Animationen bereit, wodurch der Entwickler mit geringem Aufwand ein sehr schönes und durchdachtes Ergebnis erhält.
Zum Schluss sei noch erwähnt, dass die Änderungen die durch das Update von iOS auf Version 5 entstanden sind noch nicht in die Guideline eingearbeitet wurden. Deshalb werden einige neue UI Elemente nicht in den Guidelines behandelt bzw. Änderungen wie man einige Elemente modifizieren darf, stehen nur im Developer Guide / Forum.

2.6 Vergleich

Ein zentraler Aspekt dieser Ausarbeitung ist der Vergleich zwischen den vier Geräten und deren Betriebssystem. In diesem Abschnitt werden zunächst Besonderheiten der einzelnen Komponenten noch einmal kurz aufgeführt, um zu verdeutlichen worin die expliziten Unterschiede liegen. Danach folgt eine tabellarische Darstellung der Gemeinsamkeiten.

2.6.1 Besonderheiten

BlackBerry

- **Hardwaretastatur**: Das gesamte Bedienkonzept und auch die Oberfläche sind auf die Steuerung durch eine Hardwaretastatur ausgelegt.

- **Abwärtskompatibilität**: Durch die Forderung der Kompatibilität zu älteren Geräten muss die Bedienung auch über ein Trackpad möglich sein (vier Wege Steuerkreuz bedienbar sein).

Android

- **Companion Widget**: Ein besonderes Merkmal der Android Platform sind die Companion Widgets, welche auf dem Home-Screen vom Nutzer platziert werden können und zur jeweiligen Anwendung aktuelle Informationen, grafisch aufbereitet darstellen.

- **Freiheitsgrad**: Bei Android kann jedes Element vollständig frei gestaltet und modifiziert werden. Google legt die Verantwortung für die Konsistenz in die Hände des Entwicklers. Als zentrale Aussage steht hier: „Guideline NOT Restriction“.

Windows Phone

- **Start**: Der zentrale Ausgangspunkt eines Windows Phone ist der Startbildschirm. Dieser ist in Kacheln gegliedert die frei vom Nutzer angeordnet werden können. Jede Kachel enthält Informationen zur jeweiligen Anwendung bzw. können auch einzelne Kontakte oder Mediendateien zum schnelleren Zugriff auf dem Startbildschirm platziert werden.

- **Panorama Control**: Das Strukturierungskonzept der Panorama Control findet sich in dieser Form nur auf dem Windows Phone. Hierbei werden einzelne Programmteile auf einer Art großen Wand (Panorama) platziert und der Nutzer kann fließend zwischen den einzelnen Abschnitten wechseln.

iPhone

- **Gestaltung der Elemente**: Bei Apple fällt auf, wieviel Ideen in zunächst simpel erscheinende Elemente eingeflossen sind. So wurden Komponenten die man von Desktopsystem kennt nicht einfach kopiert, sondern für die Bedienung über einen Touchscreen optimiert. Als zentrale Aussage steht das hier das Finden von Metaphern um die Bedienung in der virtuellen Welt, der Handhabung realer Objekte nachzuempfinden (z.B. Switch und Picker)

- **Konsistenz**: Beim iPhone spielt die Konsistenz eine wichtige Rolle. Um diese zu gewährleisten, schränkt Apple den Entwickler stark in der Modifizierung einzelner UI Elemente ein und fordert die Verwendung der Standardelemente, welche nur durch vorgegebene Attribute angepasst werden sollen und können.

2.6.2 Gemeinsamkeiten

Im Folgenden werden die einzelnen UI Elemente aufgeführt und gekennzeichnet welche Geräte diese besitzen bzw. ein nahezu identisches Variante. Für die Bezeichnung wird der Überbegriff genutzt, da die Namen der Elemente sich zwischen den Geräten geringfügig unterscheiden. Die Auflistung aller Elemente wäre zu umfangreich, deshalb werden nur die am häufigsten verwendeten UI Elemente untersucht.

	BlackBerry	Android	Windows Phone 7	iPhone
Tab Bar	x	x	x	x
Home-Screen	x	x	x	x
Listen und Tabellen	x	x	x	x
Toolbar	x	x	x	x
Mitteilungszentrale	x	x	—	x
Startansicht mit ausführlichen Informationen (Widgets)	—	x	x	—
Picker (Auswahlräder)	x	(x)	x	x
Textfelder mit anpassbarer Tatatur	x	x	x	x
Schieberegler (Slider)	—	—	x	x
Check Box	x	x	x	x
Radio Buttons	x	x	x	x
Vordefinierte Dialog Fenster	x	x	x	x
Drop-Down Listen	x	x	x	—
Kontextsensitive Menüs	x	x	x	x

Abbildung 2.62: Vergleich der am häufigsten genutzten UI Elemente

2.7 Fazit

Abschließend lässt sich sagen, dass die Hersteller bei der Entwicklung der Betriebssysteme für mobile Plattformen sehr interessante und kreative Lösungsmöglichkeiten gewählt haben. So hat sich weitestgehend die Bedienung über den Touchscreen durchgesetzt und es wurde auf Hardwaretasten verzichtet bzw. deren Anzahl auf ein Minimum reduziert. Es fällt auf, dass die Hersteller sehr ähnliche Wege bei der Umsetzung der GUI gehen. So werden, wie dies bereits bei Desktopsystemen der Fall war, sich immer die am besten zu nutzenden Ideen und Konzepte durchsetzen. Denn häufig war zu beobachten, dass Hersteller einfach die Ideen der Konkurrenz kopieren bzw. übernehmen und weiterentwickeln. Aufgrund der vielen Übereinstimmungen bei den einzelnen UI Komponenten, könnte es irgendwann durchaus denkbar sein, dass ein Entwickler nur noch eine Anwendung schreiben muss und diese dann ganz einfach für die einzelnen Systeme übersetzt bzw. umgewandelt wird. Mit dieser Thematik beschäftigt sich die Ausarbeit von Herrn Rolly näher.
Auch weil während der Erstellung dieser Arbeit Updates für Android und iOS erschienen sind, lässt sich sagen, dass die Entwicklung der GUI auf mobilen Geräten noch in den Anfangszügen liegt. So sind die Hersteller stets bemüht neue innovative Konzepte und UI Elemente zu entwickeln mit denen sie sich von der Konkurrenz abheben und als Vorreiter gelten.
Ebenso müssen sich die Entwickler von Anwendung an die neuen Steuerelemente und die neuen Designherausforderungen gewöhnen, da sich die bekannten Vorgehensweisen und Konzepte nicht einfach übertragen lassen. Des Weiteren wird der Entwickler je nach Plattform mehr oder weniger auf die vorhandenen UI Elemente beschränkt. Deshalb muss sich jeder potentielle Entwickler mit den GUI Guidelines der jeweiligen Hersteller beschäftigen um zu erfahren, was die Plattform an Möglichkeiten bietet bzw. was verboten ist. Dies wird besonders dann wichtig, wenn der Hersteller über einen eigenen Store verfügt und jede Anwendung zuvor überprüft. So kann es sein, dass eine Anwendung stabil läuft, jedoch nicht den Vorgaben der Hersteller genügt und somit nicht in den Store aufgenommen wird.
Alles in allem vermute ich, dass die Hersteller wie bereits schon geschehen, zum einen bei der Gestaltung der GUI und des allgemeinen Bedienkonzeptes immer weiter annähern werden, zum anderen aber auch noch viele Innovationen zu erwarten sind. Denn wie Steve Jobs bereits im Jahre 1994 sagte: „Good artists copy; great artists steal.“[29]
Dies war bei den Desktopsystemen zu beobachten, wo sich die Bedienung mit Maus, Tastatur und Fenstern als zentrale GUI Komponente durchsetzte und so wird es auch bei der Weiterentwicklung mobiler Betriebssysteme sein.

Quellen

[1] Schulz, Fabian. *Die Geschichte der grafischen Nutzeroberflächen*, http://www.imn.htwk-leipzig.de/~waldmann/edu/ss04/se/schulz/GUI.pdf, 08.11.2011.

[2] HTWK Leipzig, Swt Wiki. *Graphische Benutzeroberfläche*, http://www.imn.htwk-leipzig.de/~weicker/pmwiki/pmwiki.php/Main/GraphischeBenutzungsoberflaechen, 08.11.2011.

[3] Maul, Michael. *GUI - Graphical User Interface*, Arbeitspapiere WI Nr. 9/1998, http://geb.uni-giessen.de/geb/volltexte/2004/1682/pdf/Apap_WI_1998_09.pdf.

[4] apl. Prof. Dr. Schmitz, Lothar. *Skript Objektorientierte Programmierung*, 2009.

[5] BlackBerry Smartphones. *UI Guidelines, Version 7.0*, http://docs.blackberry.com/en/developers/deliverables/28627/, 08.11.2011.

[6] Samsung. *Galaxy SII*, http://samsung.de//webcontent/productimages/Mobiltelefone/i9100_m.jpg, 08.11.2011.

[7] NetbookNews.de. *Android Logo*, http://www.netbooknews.de/wp-content/uploads/android-logo-white.png, 08.11.2011.

[8] Computer-Bild.de. *LG P970 Optimus*, http://www.computerbild.de/fotos/Das-Smartphone-LG-P970-Optimus-Black-6084272.html, 08.11.2011.

[9] Fulcher, Richard & Nesladek, Chris & Palmer, Jim & Robertson, Christian. *Android UI Design Patterns*, http://dl.google.com/googleio/2010/android-android-ui-design-patterns.pdf, 19.05.2010.

[10] The Gadgeteer.com. *A Week with the Android G1 Smartphone*, http://the-gadgeteer.com/2009/01/29/a-week-with-the-android-g1-smartphone-day-3-4/, 08.11.2011.

[11] Phadke, Amar. *Amar Phadke's weblog*, http://amarphadke.com/category/mobile-computing/, 08.11.2011.

[12] Beckley, Adam. *android design guidelines, version 1*, http://www.mutualmobile.com/wp-content/uploads/2011/03/MM_Android_Design_Guidelines.pdf, 08.11.2011.

[13] Android Developers. *Dev Guide*, http://developer.android.com/guide/index.html, 19.10.2011.

[14] Android Developers. *User Interface Design Tips*, http://www.slideshare.net/AndroidDev/android-ui-design-tips, 08.11.2011.

[15] Microsoft. *Windows Phone*, http://www.microsoft.com/windowsphone/imageengine?img=/windowsphone/en-us/assets/devices/images/1699-8-PrimaryFront.png&scale=84, 08.11.2011.

[16] News4Mobiles. *Windows Phone 7 Logo*, http://news4mobiles.de/wp-content/uploads/2011/05/windows-phone-7-logo.jpg, 08.11.2011.

[17] Microsoft Developer Network. *User Experience Design Guidelines for Windows Phone*, http://msdn.microsoft.com/en-us/library/hh202915%28v=VS.92%29.aspx, 08.11.2011.

[18] Programming4Us. *Windows Phone 7 : Customizing the Start Screen*, http://mscerts.programming4.us/windows_phone/windows%20phone%207%20%20%20%20customizing%20the%20start%20screen.aspx, 08.11.2011.

[19] Microsoft. *Windows Phone*, http://www.microsoft.com/windowsphone/de-DE/howto/wp7/basics/what-do-the-icons-on-my-phone-mean.aspx, 08.11.2011.

[20] LuKe-B. *Windows & Apple Logo Ursprung*, http://luke-b.com/blog/2009/07/03/windows-apple-logo-ursprung/, 08.11.2011.

[21] Teebik Ltd.. *3Gstore.de*, http://www.iphoneohnevertrag.de/iphone-4-bilder-unter-cc-lizenz-frei-verwendbar/, 08.11.2011.

[22] Apple Inc. *iOS Human Interface Guidelines*, http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, 08.11.2011.

[23] Bono01-Live. *Das iPhone 4 - eine Review von euch*, http://bono01-live.blogspot.com/2010/06/das-iphone-4-eine-review-von-euch.html, 08.11.2011.

[24] Magnus.de. *AppStore und Flatrate*, http://www.connect.de/ratgeber/appstore-und-flatrate-376465.html, 08.11.2011.

[25] Hackphone. *Beautiful Notification Center Redesigns for iOS 5*, http://www.hackphone.co.il/2011/06/images/Notification_Center4.png, 08.11.2011.

[26] Apple Inc. *iOS Developer Library: UINavigationController*, http://developer.apple.com/library/ios/#documentation/uikit/reference/UINavigationController_Class/Reference/Reference.html, 08.11.2011.

[27] Apple Inc. *iOS Developer Library: Table View Programming Guide for iOS*, http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/TableView_iPhone/AboutTableViewsiPhone/AboutTableViewsiPhone.html#//apple_ref/doc/uid/TP40007451, 08.11.2011.

[28] Lesage, Stefaan. *How to use a custom background image in your grouped UITableView*, `http://cocoaheads.be/wordpress/wp-content/uploads/2010/11/Default_Grouped_TableView.png`, 08.11.2011.

[29] Jobs, Steve. *Triumph of the Nerds*, PBS, 1996.

Kapitel 3

Anwendungsentwicklung für Smartphones - Sicherheitskonzepte

Robert Hebestreit

Dieser Abschnitt beschäftigt sich mit der Sicherheit der vier Betriebssysteme Android, iOS, Windows Phone und BlackBerry OS. Dabei werden zunächst aktuelle Verschlüsselungsverfahren sowie Kommunikationsschnittstellen untersucht und anhand ihrer Sicherheit bewertet. Der Hauptteil beschäftigt sich mit der Analyse der Datensicherheit, Anwendungssicherheit und Kommunikationssicherheit der einzelnen Plattformen.

Inhaltsverzeichnis

3.1 Einführung und Problemstellung 91
3.2 Kryptografische Verfahren 91
3.2.1 Eingesetzte kryptografische Verfahren 92
3.2.2 Hashfunktionen 94
3.2.3 Sicherheit der Verfahren 95
3.3 Datenübertagung bei Smartphones 96
3.3.1 GSM 96
3.3.2 UMTS 98
3.3.3 Lokale Datenübertragung 99
3.4 Sicherheitsaspekte der Betriebssysteme 100
3.4.1 Android 101
3.4.2 iOS 103
3.4.3 Windows Phone 105
3.4.4 BlackBerry OS 108
3.5 Zusammenfassung 111

3.1 Einführung und Problemstellung

In der heutigen Zeit ist die Datensicherheit immer wichtiger geworden. Mit der zunehmenden Mobilität und dem technischen Fortschritt verlagert sich die Arbeit weg von dem klassischen Heim-PC, hin zu mobilen Geräten, wie z. B. Smartphones oder Tablet-PCs. Deshalb wird auch die Integration von Sicherheitsmechanismen in diesem Bereich zunehmend an Bedeutung gewinnen. Programme zur Verschlüsselung von Daten und Nachrichten sind nur einige Beispiele in diesem Zusammenhang. Aber auch die Sicherheit der Übertragungskanäle gerät zunehmend unter Beobachtung bzw. kann durch einfache Mittel bereits kompromittiert werden, wie das Beispiel GSM zeigt.
Ziel dieses Kapitels ist es die verschiedenen Sicherheitskonzepten der Smartphones zu beschreiben. Speziell werden im ersten Abschnitt verschiedene Verfahren vorgestellt Daten auf den Mobiltelefonen zu verschlüsseln. Dabei werden die einzelnen Verfahren zur Datenverschlüsselung kurz beschrieben und hinsichtlich ihrer Sicherheit bewertet. Weiterhin werden die verschiedenen Schnittstellen von Smartphones zur Datenübertragung auf ihre Sicherheitsaspekte untersucht. Der Hauptteil dieser Ausarbeitung beschäftigt sich mit der Analyse der vier mobilen Betriebssystemen Android, iOS, Windows Phone und BlackBerry OS im Hinblick auf deren Sicherheitsmechanismen. Untersucht wird dabei die Sicherheit der Daten auf dem Gerät, der darauf laufenden Anwendungen und die Kommunikationsbeziehungen. Dadurch soll ein umfassender Überblick über die verschiedene Sicherheitsmechanismen gegeben werden.

3.2 Kryptografische Verfahren

In diesem Abschnitt werden verschiedene kryptografische Verfahren erläutert, die zur Verschlüsselung von Daten auf Mobiltelefonen eingesetzt werden können. Des Weiteren werden diese Verfahren auch auf ihrer Sicherheit hin bewertet.
Die Verschlüsselungsverfahren kann man in symmetrische und asymmetrische Verfahren unterteilen. Der Hauptunterschied besteht dabei in der Anzahl und der Länge der Schlüssel. Symmetrische Verfahren benutzen für die Ver- und Entschlüsselung nur einen Schlüssel, der auch als gemeinsames Geheimnis bezeichnet wird. Dabei spielt der Austausch dieses geheimen Schlüssels eine zentrale Rolle. Meist wird der Schlüssel bereits vor der eigentlichen Kommunikation ausgetauscht, kann jedoch auch über einen sicheren Kanal oder mittels eines anderen Verfahrens, wie z. B. Diffie-Hellman ausgehandelt werden [1]. Nach der Wahl eines geeigneten Verfahrens und dem Austausch des Schlüssels kann der Sender eine Nachricht verschlüsseln und über eine Kanal übertragen. Der Empfänger kann nun mit dem geheimen Schlüssel und dem Entschlüsselungsverfahren den originalen Text wieder herstellen. Die Abbildung 3.1 soll diesen Vorgang noch verdeutlichen [1]. Die Verfahren Advanced Encryption Standard (kurz AES), Blowfish und Twofish sind Beispiele für symmetrische Verfahren.
Asymmetrische Verfahren zeichnen sich dadurch aus, das es ein Schlüsselpaar gibt, welches aus einem öffentlichen und einem geheimen Schlüssel besteht. Dabei dient der öffentliche Schlüssel zur Verschlüsselung von Nachrichten und kann öffentlich zugänglich sein. Daraus ergibt sich auch der Begriff Public-Key Verfahren. Der geheime Schlüssel hingehen, dient

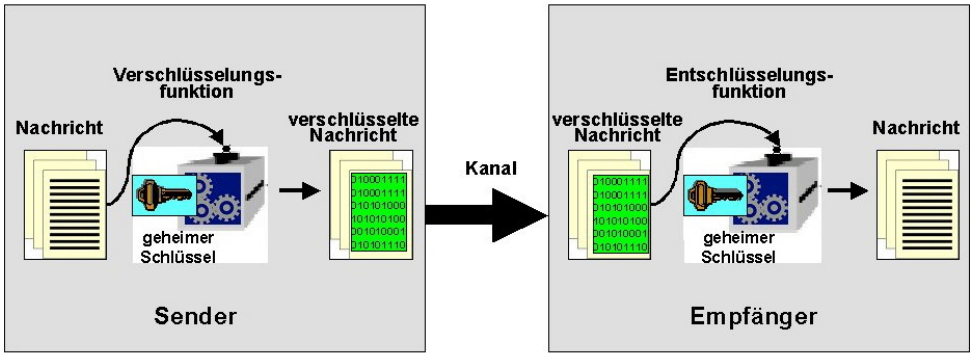

Abbildung 3.1: Symmetrisches Verschlüsselungverfahren

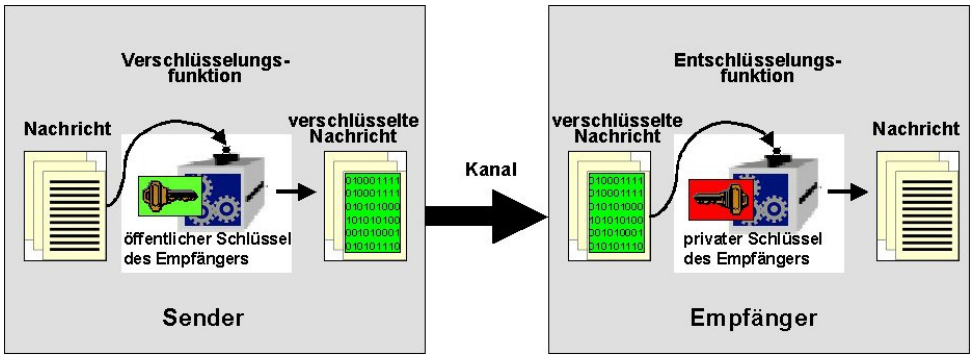

Abbildung 3.2: Asymmetrisches Verschlüsselungverfahren

zur Entschlüsselung von Nachrichten und wird geheim gehalten. Auch die Verschlüsselungsfunktion muss zugänglich sein, damit eine Nachricht von einem Sender verschlüsselt werden kann. Der Empfänger kann nun mit Hilfe der Entschlüsselungsfunktion und seinem privaten Schlüssel die Nachricht entschlüsseln. Die Abbildung 3.2 soll diesen Vorgang anschaulich darstellen [1]. Durch die mathematische Abhängigkeit zwischen dem öffentlichen und geheimen Schlüssel, kann die Nachricht nur von dem Empfänger mit dem passenden geheimen Schlüssel entschlüsselt werden. Ein Beispiel ist das Verschlüsselungsverfahren RSA, welches in diesem Kapitel noch genauer betrachtet wird. Für die Verschlüsselung von persönlichen Daten auf dem Smartphone eignen sich asymmetrische Verschlüsselungsverfahren nur bedingt, da zumeist die Daten nur von einer Person auf dem eigenen Rechner oder einem zentralen Server ver- und entschlüsselt werden.

3.2.1 Eingesetzte kryptografische Verfahren

Advanced Encryption Standard - AES

Der Advanced Encryption Standard (AES) ist ein symmetrisches Verschlüsselungsverfahren, das 2000 von dem National Institute of Standards and Technology (NIST) eingeführt wurde. Der Algorithmus beruht auf den Kryptografen Daemen und Rijmen und gehört zu den Blockchiffreverfahren, d. h. das der Klartext in Blöcke unterteilt ist und verarbeitet

wird. Das Verfahren bietet Schlüssellängen von 128, 192 und 256 Bit an und sollte den Data Encryption Standard (DES) ablösen. Dabei sollte das Verfahren gewisse Anforderungen erfüllen, wie z. B.

- Mindestens so sicher wie Triple-DES sein,
- Einfach zu implementierendes und der Öffentlichkeit frei zugängliches Design haben,
- Kostenlos zur Verfügung gestellt werden,
- Geringer Ressourcenbedarf haben,
- Sicherheit gegenüber allen bekannten Attacken sein und
- eine Blocklänge von 128 Bit haben.

Der Algorithmus basiert auf einem Rundenverfahren, wobei die Anzahl der Runden von der gewählten Schlüssellänge abhängig sind. Bei der Verwendung von einem 128 Bit Schlüssel werden 10 Runden durchlaufen, 12 Runden bei 192 Bit und 14 Runden bei 256 Bit. Für eine genauere Betrachtung der einzelnen Schritte und den mathematischen Grundlagen kann in der Quellen [5] nachgelesen werden. Die Quelle [6] bietet darüber hinaus noch anschauliche Beispiele.

Blowfish

Der Verschlüsselungsalgorithmus Blowfish ist eine Blockchiffre und wurde 1994 zum ersten Mal publiziert [3]. Da dieser Algorithmus auch als Pendant zu dem Data Encryption Standard (DES) steht, sollte er unteranderem auch den in dieser Zeit bekannten Verfahren zum Brechen der Verschlüsselung standhalten. Ein weiterer Aspekt ist, dass Alternativen zu dieser Zeit fast alle patentrechtlich geschützt und somit nicht frei zugänglich waren. Daraus ergab sich die Notwendigkeit, einen Verschlüsselungsalgorithmus zu finden. Der Algorithmus basiert auf 64 Bit großen Blöcken und kann Schlüssellängen bis zu 448 Bit verwalten. Die Verschlüsselung von Klartexten teilt sich bei Blowfish in 2 Teilbereiche auf. Der erste Bereich beschäftigt sich ausschließlich mit der Erstellung von 18 Teilschlüsseln aus dem anfangs definierten Schlüssel und der Berechnung der Einträge in den S-Boxen (engl. substitution box). Dieser Schritt wird auch als key-expansion bezeichnet. Im Anschluss erfolgt dann die eigentliche Verschlüsselung der Daten. Da Blowfish bereits 1995 erschienen ist, ist dieses Verschlüsselungsverfahren weit verbreitet und z. B. ab der Linux Kernel Version v2.5.47 ein fester Bestandteil des Betriebssystems.

Twofish

Der Verschlüsselungsalgorithmus Twofish [4] ist der Nachfolger von Blowfish und stellte sich auch wie der Algorithmus AES dem Auswahlverfahren 1997. Daher sind die Anforderungen bzw. die Eigenschaften dieses Algorithmus vergleichbar mit denen von AES.

Twofish verwendet ebenfalls Blöcke von 128 Bit und ist auch in der Lage die gleichen Schlüssellängen wie AES bereitzustellen, d. h. 128, 192 und 256 Bit. Eine Übersicht der detaillierten Anforderungen findet man in [4]. Der Algorithmus zur Verschlüsselung von Klartexten ist Rundenbasiert und wird durch das Whitening-Verfahren ergänzt. Dieses Verfahren soll die Sicherheit des Algorithmus erhöhen, da die Ein- und Ausgabe einer zusätzlichen XOR-Verknüpfung mit einem Schlüssel unterliegt [2].
Die gerade betrachteten Algorithmen sind Beispiele für symmetrische Verschlüsselungen. Das Problem bei dieser Art der Verschlüsselung, ist jedoch der vorherige Austausch der Schlüssel unter den Kommunikationspartnern. Hier besteht eine große Gefahr darin, dass falls der Schlüssel in die falschen Hände gerät, die Nachrichten entschlüsselt werden könnten. Erschwert wird der Tausch der Schlüssel noch durch die zunehmende Kommunikation im Internet. Daher entwickelte man die Idee, zwei Schlüssel einen öffentlichen und einen geheimen zu verwenden. Dieses Verfahren wird auch als asymmetrische Verschlüsselungsverfahren bezeichnet. Im Folgenden werden nun zwei Vertreter kurz vorgestellt.

RSA

Das asymmetrische Verschlüsselungsverfahren RSA [7] ist nach seinen Erfindern Ronald Rivest, Adir Shamir und Leonard Adleman benannt. Es wurde 1978 entwickelt und kann nicht nur zum Verschlüsseln, sondern auch zum Signieren verwendet werden. Die Sicherheit bei asymmetrischen Verfahren hängt davon ab, dass man mit Hilfe des öffentlichen Schlüssels nicht in der Lage sein darf, den privaten Schlüssel zu berechnen. Die Sicherheit bei RSA liegt vor allem darin, dass das RSA-Modul n nicht mit vertretbarem Rechenaufwand in seine Primfaktoren zerlegt werden kann. Damit dies jedoch gesichert ist, müssen die Primzahlen p und q, die für die Berechnung von n benutzt werden, hinreichen groß sein. Für eine sichere Verschlüsselung oder Signierung, sollte das RSA-Modul nicht kleiner als 1976 Bit sein. Eine genauere Betrachtung wird im Abschnitt 3.2.3 durchgeführt.

Diffie-Hellman für den Schlüsselaustausch

Ausgenommen von RSA galt bei allen symmetrischen Verschlüsselungen die Voraussetzung, dass die Kommunikationspartner die geheimen Schlüssel vorher austauschen mussten. Das Diffie-Hellmann-Schlüsselaustauschverfahren [8] ist ein Protokoll bei den zwei Kommunikationspartnern über einen unsicheren Kanal einen geheimen Schlüssel austauschen können. Die Sicherheit des Verfahrens beruht auf dem diskreten Logarithmusproblem. Der gemeinsame Schlüssel kann dann zur sicheren Kommunikation verwendet werden oder er dient als Sicherheit, um sich auf ein symmetrisches Verschlüsselungsverfahren wie z. B. AES oder Towfish zu einigen.

3.2.2 Hashfunktionen

Ein weiterer wichtiger Aspekt ist die Erkennung von Veränderungen bei sicherheitsrelevanten Daten. Mit Hilfe von Hashfunktionen kann die Integrität von Dateien überprüft

werden. Das Grundprinzip einer Hashfunktion besteht darin, das ein Autor einer Datei mit einer Hashfunktion den Hashwert seiner Datei ermittelt und öffentlich zugänglich macht. Falls nun eine zweite Person prüfen will, ob die Datei dem Werk des Autors entspricht, berechnet dieser einfach den Hashwert der Datei. Stimmt nun der berechnete mit dem veröffentlichten Wert überein, ist die Datei noch integer. Dabei gelten verschiedene Anforderungen für Hashfunktionen [11]:

1. Der Hashwert einer Nachricht muss einfach zu berechnen sein.
2. Die Berechnung der Nachricht aus einem gegebenen Hashwert darf nicht effizient möglich sein.
3. Die Hashwerte müssen kollisionsresistent sein.

Der Secure Hash Algorithm (SHA) [9] oder der Message-Digest Algorithm 5 (MD5) [10] sind 2 Beispiele für Hashfunktionen.

3.2.3 Sicherheit der Verfahren

Die Sicherheit der vorgestellten Verfahren kann zum einem auf die Länge der verwendeten Schlüssel zurückgeführt werden. Die Tabelle 3.1 zeigt eine Übersicht mit den empfohlenen Schlüssellängen des Bundesamt für Sicherheit in der Informationstechnik (BSI) [12] bzw. der Bundesnetzagentur [13]. Der Algorithmus Blowfish wird nur bedingt empfohlen, da

Verschlüsselungsverfahren	Schlüssellänge	Bewertung der Sicherheit
AES	128, 192, 256 Bit	Bis Ende 2012
Twofish	128, 192, 256 Bit	Bis Ende 2012
RSA	Mindestens 1976 Bit Empfohlen 2048 Bit	Bis Ende 2017
SHA	224 Bit 256, 384, 512 Bit	Bis Ende 2015 Bis Ende 2017

Tabelle 3.1: Bewertung der Schlüssellänge

die Blockgröße nur 64 Bit beträgt. Des Weiteren gibt es bei diesem Algorithmus eine reduzierte Version mit weniger Runden bei denen schwache Schlüssel gefunden wurden [14]. Jedoch beeinflusst dies nicht den originalen Algorithmus und zurzeit sind auch keine Sicherheitsrelevanten Erkenntnisse vorhanden, die Verschlüsselung von Blowfish zu brechen. Der Autor von Blowfish rät jedoch zur Verwendung des Nachfolgers Twofish. Ebenfalls als unsicher einzustufen ist die Hashfunktion MD5, da hier bereits mit Hilfe von einfachen Rechnern Kollisionen festgestellt werden können [14]. Auf der anderen Seite werden die Schranken für Twofish und AES aus den Quellen des BSI und der Bundesnetzagentur vermutlich weiter erhöht werden, da keine bekannten effizienten Angriffsverfahren für beide Verfahren bekannt sind und die Quelle [15] von einer Lebensspanne von 30 Jahren spricht.

3.3 Datenübertagung bei Smartphones

Dieser Abschnitt beschäftigt sich mit den häufigsten Kommunikationsnetzen, die heutige Smartphones nutzen. Dabei wird vor allem auf größere Netze wie GSM oder UMTS eingegangen. Die Hinsichtlich ihrer Sicherheitsmechanismen untersucht werden. Weiterhin wird kurz auf das Personal Area Netzwerk Bluetooth und das WLAN Netz eingegangen.

3.3.1 GSM

Das Global System for Mobile Communications (GSM) integriert drei verschiedene Sicherheitsmaßnahmen. Diese gliedern sich in die Bereiche

- Authentifizierung,
- Verschlüsselung,
- Schutz der Privatsphäre auf.

Bei der Authentifizierung, gilt es zwei Bereiche zu betrachten. Der erste Bereich ist die nötige Authentifizierung des Endgerätes mit dem Mobilfunkbetreiber. Dies geschieht über ein Challenge Response Verfahren. Wichtig für dieses Verfahren ist ein gemeinsames Geheimnis, das bei dem Kauf der SIM-Karte in der Datenbank des Betreibers und auf der SIM-Karte gespeichert wurde. Damit der Betreiber die gespeicherten Geheimise zu den Kunden zuordnen kann, verfügt jede SIM-Karte über eine weltweit eindeutige Kennnummer, die International Mobile Subscriber Identity (IMSI). Durch dieser eindeutigen Nummer kann der Betreiber das hinterlegte gemeinsame Geheimnis verwenden, um die Authentifizierung durchzuführen. Mit Hilfe einer 128 Bit großen Zufallszahl und dem Geheimnis kann nun das Ergebnis des sogenannten A3 Algorithmus verglichen werden und falls dies übereinstimmt, ist der Teilnehmer mit seinem Gerät identifiziert. Neben der Authentifizierung zu dem Betreiber, muss sich der Benutzer eines Gerätes erst mit der zugehörigen Personal Identification Number (PIN) bei der SIM-Karte authentifizieren. Die Eingabe des PINs ist dabei auf drei Versuche begrenzt und wird dann automatisch gesperrt. Danach kann die Karte nur mit dem Personal Unblocking Key (PUK) entsperrt werden. Nach zehn Fehleingaben dieses Codes, werden alle Daten auf der SIM-Karte gelöscht. Beide Nummern werden vorher durch den Betreiber generiert, jedoch kann nur die PIN durch den Benutzer verändert werden.
Die Verschlüsselung in GSM Netzen erfolgt nur über die Luftschnittstelle des mobilen Endgerätes zu einer Base Transceiver Station (BTS). Jedoch werden nicht alle Transmissionen verschlüsselt, da z. B. Broadcast-Nachrichten an alle Teilnehmer unverschlüsselt bleiben, da diese sonst nicht gelesen werden können. Für die Verschlüsselung wird zunächst ein Schlüssel benötigt, der aus dem gemeinsamen Geheimnis und einer Zufallszahl über den A8 Algorithmus berechnet wird. Dieser Schlüssel hat eine Größe von 64 Bit, wobei 8 Bit fest sind. Darauf aufbauend können die Daten mit dem A5 Algorithmus verschlüsselt und dann übertragen werden. Der Algorithmus teilt sich in vier Typen A5-0 bis A5-3 auf. Die Version A5-0 biete keine Verschlüsselung an, die Versionen A5-1 und

A5-2 sind Stromchiffre, wobei A5-2 lediglich eine abgeschwächte Form der Version 1 ist. Der letzte Typ A5-3 beschreibt eine Blockchiffre, die identisch zu der Verschlüsselung in UMTS-Netzen ist.
Der letzte Punkt ist der Schutz der Privatsphäre des Kunden. Dabei wird nur für die erste Verbindungsaufnahme und die Authentifizierung die IMSI der SIM-Karte verwendet. Weiterführende Verbindungen werden durch eine Temporary Mobile Subscriber Identity (TMSI) beschrieben. Dabei wird die TMSI in regelmäßigen Zeitabständen oder bei einem Wechsel der Funkzelle geändert. Dies soll verhindern, dass ein potentieller Angreifer mit dem Wissen einer bestimmten IMSI ein Bewegungs- oder Kommunikationsprofil erstellt [23].

Sicherheitsbewertung

Die Sicherheitsmechanismen von GSM, bieten jedoch zahlreiche Angriffspunkte. Bereits im Bereich der Authentifizierung der Geräte im GSM Netz können Angreifer die Transmissionen der Geräte abhören, da sich nur das mobile Endgerät nicht aber die Base Transceiver Station authentifizieren muss. Dieses Verfahren wird auch als IMSI-Catcher bezeichnet und kann z. B. dazu führen, dass die mobilen Endgeräte dazu gebracht werden, keine Verschlüsselung mehr einzusetzen. Aber selbst mit eingeschaltet Verschlüsselung können die Kommunikationsdaten ausgelesen werden. Auf der einen Seite gibt es Angriffe, die mit wenig Vorlaufzeit die Verschlüsselung in Echtzeit entschlüsseln können. Vor allem die Algorithmen A5-1 und A5-2 sind nicht mehr für den Gebrauch geeignet, da zahlreiche Angriffsmöglichkeiten existieren. Selbst die GSM Association drängt seit 2006 darauf, die Version A5-2 nicht mehr in den Geräten zu verwenden. Weiterhin gibt es Sicherheitsbedenken bei den Algorithmen A3 und A8, da diese keinen Standard folgen und vom Mobilfunkbetreiber selbst implementiert werden und damit eine korrekte Implementierung nicht nachgewiesen werden kann.
Auf der anderen Seite ist nur die Kommunikation zwischen den mobilen Geräten und der BTS verschlüsselt. Die Richtfunkverbindungen innerhalb eines Base Station Subsystem (BSS), d. h. beispielsweise die Verbindungen zwischen den BTS und den Base Station Controlern (BSC), wird unverschlüsselt übertragen und ist daher anfällig für Angriffe. Ein weiterer Angriffspunkt geht von Innentätern aus. Durch die Manipulation der Switching-Komponenten können z. B. Verbindungen an andere Endgeräte weitergeleitet werden. Weiterhin können durch die gespeicherten Daten Kommunikations- oder Benutzerprofile erstellt werden oder auf die Vorratsdatenspeicherung zurückgegriffen werden. Die Netzbetreiber müssen dabei Telekommunikationsdaten, wie z. B. die Anschlussnummern der Kommunikationspartner, SMS, MMS oder ähnliche Nachrichten oder Daten einer Internetverbindung speichern. Auch die Privatsphäre der Benutzer ist betroffen, da man mit der Rufnummer in der Lage ist, dass Gerät durch die Funkzellen zu orten und so dem Sicherheitsmechanismus der TMSI umgehen kann. Neben diesen Angriffsmöglichkeiten auf die Daten der Verbindungen können auch Denail of Service Angriffe durch Störer erreicht werden, so dass die Verbindung zwischen den mobilen Endgeräten und den Base Transceiver Station gestört wird [23].

3.3.2 UMTS

Im Gegensatz zu GSM wurden die Sicherheitsmaßnahmen bei dem Universal Mobile Telecommunications System (UMTS) verbessert. In allen drei Bereichen, der Authentifizierung, der Verschlüsselung und der Privatsphäre, gibt es Neuerungen, die zu der Erhöhung der Sicherheit beitragen sollen. Die Authentifizierung erfolgt nun in zwei Richtungen, d. h. zunächst muss sich das Mobilfunknetz bei dem mobilen Gerät authentifizieren. Dies geschieht über das Authentication Token (AUTN) und erst nach erfolgreicher Authentifizierung des Netzes, muss sich das mobile Gerät identifizieren. Durch die gehobene Sicherheit werden jedoch zwei weitere Werte benötigt, die in dem sogenannten Authentication Vector (AV) gespeichert werden. Dieser beinhaltet

- eine 128 Bit Zufallszahl,
- das erwartete Ergebnis (XRES) des Response Verfahrens bei der Authentifizierung,
- einen 128 Bit Chipher Key (CK) für die Verschlüsselung,
- einen 128 Bit Integrity Key (IK) für die Datenintegrität und
- ein 128 Bit Authentication Token (AUTN) für die Netzauthentifizierung.

Die Verschlüsselung der Daten ist bei UMTS Pflicht und wird nicht nur zwischen dem mobilen Gerät und der Empfängerstation einer Zelle verschlüsselt, sondern innerhalb eines kompletten Radio Network Controllers (RNC), der vergleichbar mit einem Base Station Subsystem (BSS) ist. Die Verschlüsselung erfolgt durch den f8-Algorithmus, der ähnlich dem A5-3 Algorithmus bei GSM ist. Die 64-Bit Blockchiffrierung greift auf einen 128 Bit Schlüssel zurück, der im Authentication Vector und auf dem Gerät gespeichert ist. Dazu kommt die Überprüfung der Datenintegrität durch den f9-Algorithmus auf der Luftschnittstelle zwischen dem mobilen Endgerät und dem Knoten in einem UMTS Netz. Für den Schutz der Privatsphäre wird, wie bei GSM, die IMSI nur für die Authentifizierung benötigt und danach durch die Extended Encrypted Mobile Subscriber Identity (XEMSI) ersetzt. Diese besteht aus der Adresse des Netzknotens und der Encrypted Mobile Subscriber Identity (EMSI). Für den Sitzungsschlüssel wird eine aus der XEMSI generierte Temporary EMSI (TEMSI) berechnet. Die Privatsphäre des Teilnehmers wird dadurch sichergestellt, dass die XEMSI periodisch gewechselt wird und die TEMSI nach jeder Transaktion neu berechnet wird [23].

Sicherheitsbewertung

Die zusätzlichen Sicherheitsmaßnahmen unterbinden vor allem die Angriffe auf die verschlüsselten Daten. Jedoch können weiterhin Daten durch Innentäter oder durch den Zugriff auf die Vorratsdatenspeicherung von Angreifern genutzt werden. Darunter zählt auch das Anfertigen von Bewegungs- und Kommunikationsprofilen. Weiterhin können die Geräte über ihrer Rufnummer geortet werden. Durch die Funktechnik können auch UMTS Verbindungen mit Stören unterbrochen werden, was einem Denail of Service Angriff ähnlich zu GSM gleich kommt. Ein letzter Sicherheitsmangel ist die Notwenigkeit

von UMTS-Geräten auch auf GSM zurückgreifen zu können, falls z. B. kein UMTS Empfang verfügbar ist. Damit werden wiederrum auch alle Sicherheitsgefährdungen von GSM relevant, wie z. B. der IMSI-Catcher für einen Man-in-the-Middle Angriff [23].

3.3.3 Lokale Datenübertragung

Bluetooth

Neben den größeren Netzen GSM und UMTS ist der Funkstandard Bluetooth auf Reichweiten bis zu 100 m ausgelegt und beherrscht drei verschiedene Modi. Der Funkstandard kann ausgeschaltet, eingeschaltet unsichtbar oder sichtbar betrieben werden und findet vor allem bei kleineren Geräten wie z. B. Headsets Verwendung. Im Bluetooth Standard sind Sicherheitsmechanismen verbaut, die jedoch auch Angriffspunkte liefern. Bluetooth setzt bis Version 2.0 auf eine einseitige Authentifizierung, was Man-in-the-Middle Angriffe ermöglicht. Erst mit der Version 2.1 wird das Konzept über die Abfrage von PINs durch ein Public Key Verfahren ersetzt, das auf beiden Geräten einen Schlüssel berechnet. Ist der Schlüssel identisch, kann davon ausgegangen werden, dass die Verbindung nicht kompromittiert ist. Neben der Authentifizierung können die gesendeten Daten auch Verschlüsselt werden. Jedoch ist die Verschlüsselung standardmäßig ausgeschaltet und muss erst von einem Teilnehmer gefordert werden. Die Daten werden dann entweder mit einem Geräteschlüssel oder einem Kombinationsschlüssel verschlüsselt. Das Problem hierbei ist, dass der Geräteschlüssel statisch ist und daher nicht als sicher eingestuft werden kann. Bei der Verwendung des Kombinationsschlüssels sollten die Empfohlenen Längen des BSI berücksichtigt werden. Das empfiehlt bei der Verwendung von Zahlen eine Mindestgröße von 12 Stellen oder bei der Verwendung des ASCII Alphabetes mindestens sechs Stellen, da der Schlüssel durch Abfangen von Nachrichten geknackt werden könnte. Ein weiteres Sicherheitsproblem bei Bluetooth ist das sogenannte Bluetooth-Tracking. Mit der Entsprechenden Hardware, d. h. genügend Bluetooth Empfänger und Verstärker, kann ein Gerät beispielsweise auch aus 2 km geortet werden. Auch wird bei dieser Methode nicht zwischen dem sichtbaren und unsichtbaren Modus unterschieden, hier zählt nur, ob Bluetooth eingeschaltet ist. Weiterhin gibt es für Bluetooth bekannte Hacking Software, mit der z. B. ein Gerät lahm gelegt oder die Konfiguration des Gerätes geändert werden kann. Mit dieser Software ist es auch möglich Daten auf dem Gerät, also beispielsweise das Telefonbuch zu ändern [23].

WLAN

Der WLAN Standard IEEE 802.11 ist eine weitere Schnittstelle auf die moderne Smartphones zurückgreifen können. Ein WLAN zeichnet sich zu aller erst durch einen bestimmten Namen, den Service Set Identifier (SSID), aus, welcher durch Broadcast Nachrichten alle Teilnehmer im Empfangsbereich Informationen zukommen lässt. Darunter auch die Sicherheitsmechanismen, die das WLAN Netz fordert. Für die Sicherheit sind die Abschnitte 802.11i und 802.1X des IEEE Standards verantwortlich. Neben diesen Mechanismen gib es noch die Wired Equivalent Privacy (WEP), die jedoch im Hinblick auf Sicherheit als

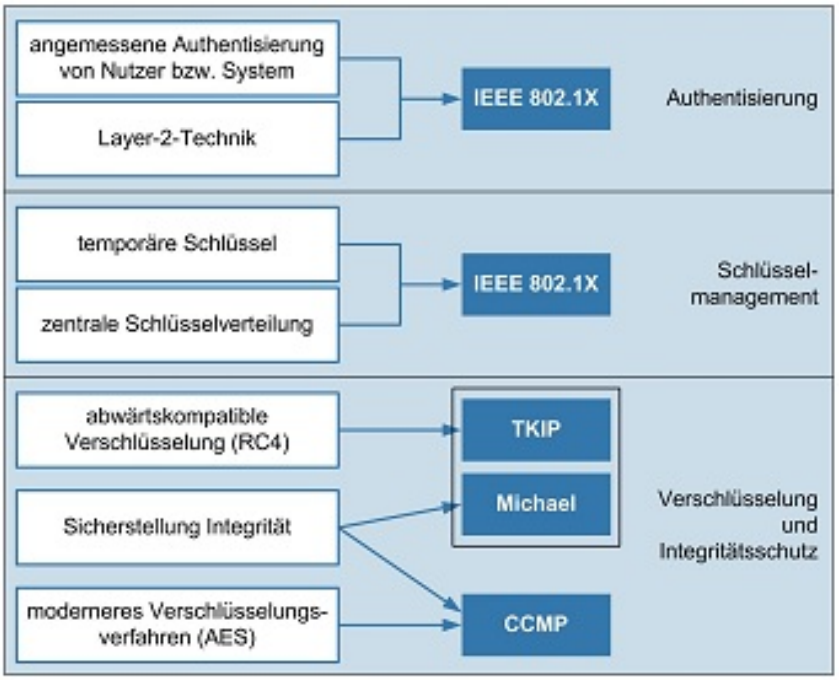

Abbildung 3.3: Überblick über die Sicherheitsmechanismen in 802.11i [24]

ungenügend sicher eingestuft werden muss. Der Standard 802.11i beschreibt Methoden für die Verschlüsselung, Authentisierung und das Schlüsselmanagement. Die wichtigsten Komponenten können der Abbildung 3.3 entnommen werden. Neben diesen Mechanismen kann der Zugang zu einem WLAN auch durch den Standard 802.1X gesichert werden. Dieser beruht auf der Authentifizierung mit Hilfe von RADIUS-Servern und nutzt das Extensible Authentication Protocol (EAP). Für die einzelne Darstellung der zur Verfügung stehenden Mittel verweise ich hier auf die Broschüre des Bundesamtes für Sicherheit in der Informationstechnik [24].
Abschließend ist zu sagen, das die Sicherheit in WLAN Netzen mit dem Standard 802.11i bzw. auch in Verbindung mit 802.1X als gute Absicherung der Luftschnittstelle aufgefasst werden kann. Angriffsmöglichkeiten bieten sich vor allem in falsch Konfigurierten Zugriffspunkten, die das komplette Netz dadurch gefährden. Bei der Verwendung mit WLAN sollte weiterhin auch das lokale Infrastrukturnetz über zuverlässige Sicherheitsmechanismen verfügen, da sich sonst Angreifer über das feste Netz den Zugriff verschaffen könnten. Auf der anderen Seite zählt die Störung des Funksignals für alle Funknetzwerke als Bedrohung, da dadurch das WLAN Netz lahm gelegt werden kann [24].

3.4 Sicherheitsaspekte der Betriebssysteme

Dieses Kapitel beschäftigt sich mit der Sicherheitsarchitektur der verschiedenen Betriebssysteme für Smartphones. Speziell wird auf die Sicherheitsfähigkeiten von Android, iOS, BlackBerry und Windows Phone eingegangen.

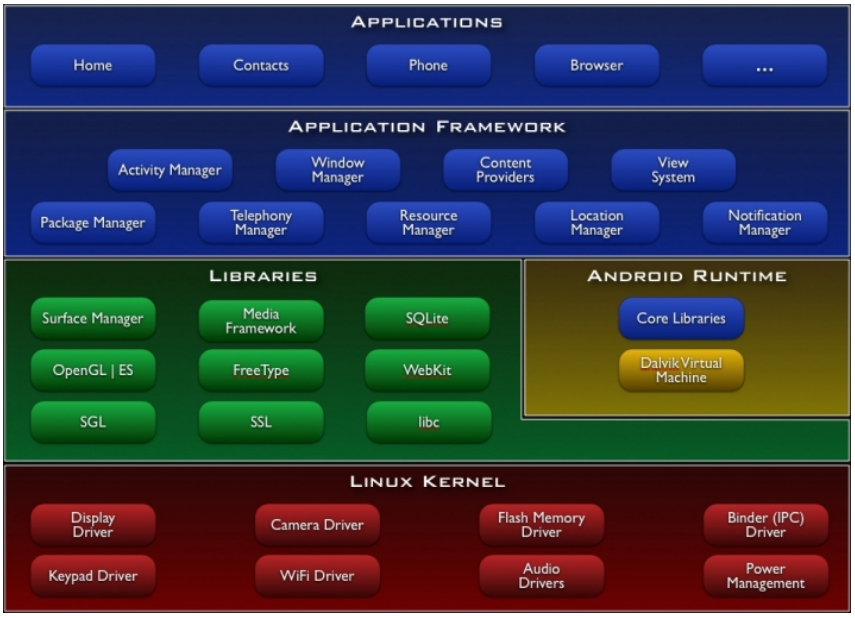

Abbildung 3.4: Die Android Architektur [16]

3.4.1 Android

Das Betriebssystem Android basiert auf dem Linux Kernel 2.6 und ist modular aufgebaut. Die Abbildung 3.4 zeigt den Aufbau von Android. Neben dem zugrunde liegenden Linux System ist der Hauptbestandteil von Android die Dalvik Virtual Machine, die für jedes Programm eine eigene virtuelle Umgebung (Sandbox) schafft. Die in Java entwickelten Programme werden dann in ein entsprechendes Format für die virtuelle Maschine von Dalvik umgewandelt.

Datensicherheit

Für die Sicherheit von Android bedeutet das jede Anwendung in einer virtuellen Umgebung arbeitet und durch den Linux Kernel mit Zugriffsrechten versehen wird. Durch das Linux System haben alle Anwendungen eine zugehörige Benutzer- bzw. Gruppen ID, die den Zugriff auf die Dateien regelt. Jede Anwendung wird dabei durch eine eindeutige Benutzer-ID identifiziert, um so die Zugriffsrechte der einzelnen Anwendungen auf die Daten festzulegen. Im Vergleich zu Linux, bei denen verschiedene Programme mit der gleichen Benutzer-ID laufen können, wird in Android strickt zwischen den einzelnen Anwendungen getrennt. Daraus folgt, dass jede Anwendung zunächst einmal keinen Zugriff auf andere Anwendungen hat und daher auch keine andere Anwendung, das Betriebssystem oder den Nutzer beeinflussen kann. Dies beinhaltet z. B. das Schreiben oder Lesen von Benutzerdaten oder den Netzwerkzugriff. Für die Speicherung von Daten bekommt jede Anwendung einen eigenen Bereich im Dateisystem zugewiesen, in denen Daten oder komplette Datenbanken abgespeichert werden können. Schließlich können durch diese Sicherheitsmechanismen auch kompromittierte oder gefährliche Anwendungen nur auf ihre eigenen Daten zugreifen, nicht jedoch andere Anwendungen oder Daten beeinflussen.Zur weiteren Absicherung des Betriebssystems, gibt es eine eigene Partition auf der sich der

Android Kernel, wichtige Bibliotheken und Anwendungen befinden. Auf diese Partition kann jedoch nur lesend zugegriffen werden.
Ein weiterer Speicherort bei einem Smartphone ist die SIM-Karte. Der Zugriff auf diese Karten ist nur dem Betriebssystem selbst vorbehalten und auch keine Anwendung von Dritten kann auf die Daten zugreifen, da keine Schnittstelle von Android bereitgestellt wird, die nicht über das Betriebssystem läuft.
Eine weitere mögliche Absicherung der Daten auf einem Android Smartphone ist die Dateisystemverschlüsselung. Dabei werden alle Daten auf dem Smartphone mit Hilfe von AES und einem 128 Bit Schlüssel verschlüsselt. Der Schlüssel setzt sich aus einem Benutzerpasswort, einem zufälligen Salt und der mehrfachen Ausführung der Hashfunktion SHA-1 zusammen. Dies sichert die Vertraulichkeit der Daten, solange ein Angreifer nicht das Passwort des Benutzers kennt.
Neben den Benutzer-IDs der Anwendungen gibt es auch den Root-Benutzer in Android, der jedoch nur für den Kernel und einige wichtige Kernanwendungen benutzt wird. Der Root-Benutzer hat Zugriff auf alle Daten der verschiedenen Anwendungen. Jedoch ist es im Sicherheitssystem von Android nicht vorgesehen, Benutzern diese Root-Privilegien zu geben. Falls sich Benutzer Zugriff auf das Gerät verschaffen wollen, um z. B. Root-Privilegien zu erhalten oder neue Betriebssysteme zu installieren, muss der Bootloader neu konfiguriert oder gelöscht werden. Android bietet einen Sicherheitsmechanismus an, der alle Daten auf dem Gerät löscht, falls der Bootloader verändert werden sollte. Diese Sicherung greift jedoch nicht, falls der ein Angreifer auf eine andere Art an die Root-Privilegien gekommen ist. Dieser kurze Abschnitt sollte die wichtigsten Sicherheitsmechanismen ansprechen, die Android bereits durch die Plattform bereitstellt.

Anwendungssicherheit

Der nächste Punkt beschäftigt sich mit der Sicherheit einzelner Anwendungen. Das wichtigste Merkmal einer Anwendung ist die *Manifest.xml* Datei. In dieser Datei müssen die benötigten Ressourcen der Anwendung vorher definiert werden. Diese Privilegien können nach der Ausführung der Programme nicht dynamisch zur Laufzeit verändert werden. Einige Ressourcen der Geräte können deshalb nur benutzt werden, wenn diese ausdrücklich durch die Anwendung gefordert werden. Beispiele für besonders geschützte Ressourcen sind:

- Kamerafunktion,
- GPS-Funktion,
- Bluetooth,
- Anruffunktionen sowie SMS/ MMS und
- Netzwerkverbindungen.

Weitere Beispiele sind z. B. Kontaktdaten, der Zugriff auf Kalenderdaten oder der Browserverlauf. Bei der Installation einer Anwendung wird der Benutzer über die benötigten

Ressourcen dieser Anwendung informiert und kann diesen entweder zustimmen und die Anwendung installieren oder ablehnen. Der Benutzer kann jedoch nicht einzelne Punkte erlauben, da er entweder allen zustimmen oder auf die Installation der Anwendung verzichtet muss.
Die Anwendungsentwicklung auf der anderen Seite erfordert eine Signierung der Anwendung durch ein Zertifikat. Das Zertifikat ist dabei der persönliche Schlüssel des Entwicklers und soll so den Autor und die zugehörigen Befugnisse der Anwendung identifizieren. Weiterhin können bei Android Anwendungen auch durch Dritte signiert werden. Jedoch gibt es keine spezielle Signierung durch den Market Places oder eine spezielle Untersuchung des Programmes. Die Sicherheit wird bei Android auf den Benutzter verschoben, da er bei jeder Installation eines Programmes den Bedingungen des Programmes zustimmen muss. Bei diesem Prozess entstehen auch die größten Sicherheitsbedenken, da die meisten Anwendungen auf kritische Funktionen zurückgreifen wollen, die sie unter Umständen nicht benötigen, aber von den Entwicklern in der Manifest-Datei gefordert werden. Beispiele dafür sind Spiele oder Kalenderanwendungen, die z. B. die GPS-Funktion des Handys nutzen wollen, diese aber überhaupt nicht benötigen [17].

Kommunikationssicherheit

Im Bereich der sicheren Kommunikation bietet Java verschiedene Schnittstellen und Implementierungen für Kommunikationsprotokolle an, die auch von Android Geräten genutzt werden können. Darunter sind z. B. das Transport Layer Security (TLS) Protokoll und dessen Vorgänger das Secure Sockets Layer (SSL) Protokoll, das Simple Authentification and Secure Layer Protokoll (SASL) oder das Secure Hypertext Transfer Protocol (HTTPS).
Den letzten Teil bildet die Public Key Infrastructure, die als Schnittstelle für die sichere Kommunikation auf Basis von asymmetrischen Verschlüsselungsalgorithmen arbeitet. Dabei steht vor allem die Verbindung zwischen Benutzern und die Verwaltung von Zertifikaten im Mittelpunkt. Ein weiterer Bestandteil ist die Langzeitverwaltung der verwendeten Schlüssel bzw. Zertifikate in sogenannten *KeyStores* oder *CertStores* bzw. die Erstellung von öffentlichen oder privaten Schlüsseln sowie das Erstellen von selbst-unterschriebenen Zertifikaten und deren Im- oder Export.

3.4.2 iOS

In diesem Abschnitt wird die Sicherheitsarchitektur von iOS untersucht. Das Sicherheitskonzept von iOS gliedert sich in die vier Bereiche

- Gerätesicherheit (device security),
- Datensicherheit (data security),
- Kommunikationssicherheit (network security) und
- Anwendungssicherheit (app security) auf.

Gerätesicherheit

Die Gerätesicherheit beschreibt dabei Methoden, die unautorisierte Zugriffe verhindern sollen. Die einfachste Möglichkeit ist die Aktivierung eines Zugangscodes für das komplette Gerät. Nur durch die Eingabe dieses Codes, kann danach das Gerät benutzt werden. Des Weiteren wird dieser Code auch zur Verschlüsselung der Daten auf dem Gerät verwendet, die im nächsten Abschnitt genauer beschrieben wird. Eine weiterer Eigenschaft, dass Gerät zu schützen, sind sogenannte Policies zu definieren. Dies ist z. B. für Unternehmen interessant, die bereits fest definierte Sicherheitsrichtlinien haben. Eine Policy beschreibt in diesem Kontext ein Regelwerk, das z. B. die Nutzung von Hardwareressourcen oder Anwendungen einschränkt. Damit lässt sich z. B. der Zugriff auf die Kamera oder auch die Verwendung von bestimmten Anwendungen, wie Safari einschränken. Diese Einstellungen können dann in Profilen gespeichert werden und auf dem Gerät etabliert werden. Profile können als Konfigurationsdateien auf dem Smartphone abgelegt und zusätzlich gesichert werden. Durch diese Absicherung können z. B. nur Benutzer mit einem entsprechenden Kennwort, in Unternehmen Administratoren, Änderungen an der Datei vornehmen. Bei vorsätzlichen Missbrauch dieser Datei können die Profile auch so angelegt werden, das alle Daten auf dem Gerät gelöscht werden und so nicht mehr zugänglich sind. Diese Profile können auch über das Mobile Device Management (MDM) oder über Microsofts ActiveSyn auf dem Smartphone gespeichert werden. Eine Liste mit Fähigkeiten, die eingeschränkt werden können, findet sich in der Quelle [18]. Ein weiterer Sicherheitsaspekt ist die Verschlüsselung oder Signierung der einzelnen Profile, um die Sicherheit weiter zu erhöhen.

Datensicherheit

Die Datensicherheit für iOS wird auf zwei verschiedenen Weisen sichergestellt. Zum einen bieten diese Geräte eine Hardwareverschlüsselung für alle gespeicherten Daten an. Diese Verschlüsselung greift auch auf den persönlichen Zugangscode zu. Ist das Smartphone mit Hilfe eines solchen Codes gesperrt, kann nicht auf die Daten zugegriffen werden, da sie verschlüsselt auf dem Gerät gespeichert sind. Die Verschlüsselung erfolgt mit dem AES Algorithmus und einem 256 Bit Schlüssel. Auf der anderen Seite bietet iOS eine Softwareverschlüsselung an, mit deren Hilfe E-Mails, Kalenderdaten oder andere Anwendungsdaten verschlüsselt werden können. Diese Verschlüsselung gilt auch für Daten, die über iCloud abgespeichert werden. Die Daten werden zunächst verschlüsselt übertragen und verschlüsselt auf dem Server abgespeichert. Falls das Gerät gestohlen oder verloren geht, kann man die Daten auch mittels einer Fernlöschung (remote wipe) vom Smartphone entfernen. Gleichzeitig wird das Gerät dabei auch deaktiviert. Diese Löschung kann auch lokal in den eben vorgestellten Profilen definiert werden, um z. B. BruteForce Angriffe zu verhindern. Standardmäßig wird hier seitens iOS bereits eine Grenze von zehn Fehleingaben festgelegt.

Kommunikationssicherheit

Der dritte Punkt ist die Kommunikationssicherheit. Neben der verschlüsselten Übertragung der Daten bei iCloud, können auch Browser, Mailprogramme oder der Kalender auf die Protokolle TLS und SSL in der Version 3 zurückgreifen. Neben der Verschlüsselung der Daten verfügt iOS auch über Mechanismen zur Authentifizierung der Kommunikationspartner. Darunter zählen digitale Zertifikate nach dem X.509 Standard, bei WLAN die Unterstützung des IEEE Standards 802.1X. Für Unternehmen, die mit *two-factor Token* arbeiten, können auch *RSA SecurID* und *CRYPTOCard* verwendetet werden, um Kommunikationspartner zu identifizieren. Des Weiteren können auch auf Proxy Konfigurationen oder *split IP Tunneling* zurückgegriffen werden und somit der Netzwerkverkehr gezielt gelenkt werden. Ein weiterer Aspekt ist die Möglichkeit, Virtual Private Networks (VPNs) zu verwenden. Dabei werden die Standards IPSec, L2TP und PPTP von Cisco unterstützt. Zusätzlich kann SSL VPN von den Anbietern Juniper, Cisco und F5 Networks genutzt werden [18].

Anwendungssicherheit

Der letzte Punkt ist die Sicherheit der Anwendungen auf dem Smartphone. Jede Anwendung wird dabei in einer Sandbox ausgeführt damit sich die einzelnen Anwendungen nicht gegenseitig beeinflussen können. Zusätzlich sind die Systemdateien sowie der Kernel vor dem Zugriff durch Benutzeranwendungen gesichert. Die Kommunikation zwischen den Anwendungen erfolgt nur auf Basis von Schnittstellen, die das Betriebssystem liefert. Für bestimmte Anwendungen, z. B. Bankanwendungen werden sogenannte Keychains verwendet. Diese kommen zum Einsatz, wenn Authentifizierungsinformationen auf dem Gerät abgespeichert werden sollen. Die Keychains sind verschlüsselt auf dem Gerät gespeichert und können z. B. Benutzernamen, Passwörter oder Zertifikate zur Identifizierung enthalten. Ein weiterer Sicherheitsmechanismus ist die Signierung aller Anwendungen. Dabei muss jede Anwendung entweder durch Apple selbst oder durch den Entwickler der Anwendung mit einem von Apple ausgestellten Zertifikat signiert werden. Dies soll verhindern, dass die Apps manipuliert werden können. Zusätzlich erfolgt eine Prüfung der Anwendungen, die in den Apple Store eingefügt werden sollen. Für selbsterstellte Anwendungen im Firmenbetrieb kann das *Provisioning Profile* benutzt werden und somit wieder spezielle Sicherheitsaspekte definiert werden.

3.4.3 Windows Phone

Auch bei dem Betriebssystem Windows Phone wurden Sicherheitsaspekte berücksichtigt, wie sie auch bei Android bzw. iOS betrachtet wurden. Die Sicherheitsarchitektur von Windows Phone orientiert sich an den folgenden drei Säulen:

- Quality of phone experience,
- Access to the users information,

- Billable events.

Der erste Punkt beschreibt die Erfahrung des Nutzers mit dem mobilen Gerät. Dazu zählt z. B., dass Anwendungen sich gegenseitig nicht behindern oder negativ beeinflussen sollen, damit der Benutzer in der Handhabung nicht behindert wird. Der nächste Punkt beschreibt die Sicherheit der persönlichen Daten auf dem Smartphone. Dabei sollen Anwendungen ohne die Genehmigung durch den Benutzer kein Zugriff auf vertrauliche Daten, wie z. B. Kontaktdaten oder Fotos, erhalten. Der letzte Punkt beschreibt spezielle Funktionen von Smartphones, die zusätzliche Kosten verursachen können. Darunter zählen z. B. Telefongespräche, SMS oder MMS, wobei Anwendungen nicht in der Lage sein sollen, unerlaubte Kosten zu verursachen. Dies ist das Basiskonzept der Sicherheitsmechanismen von Windows Phone, die in den folgenden Abschnitten genauer vorgestellt werden.

Datensicherheit

Jede Anwendung auf dem Windows Phone wird in einer Sandbox ausgeführt, welche die einzelnen Anwendungen voneinander isolieren soll. Sie beinhaltet weiterhin eine stricke Kontrolle über die verwendeten Telefonfunktionalitäten, wie z. B. das Senden von SMS. Zusätzlich wird auch die Kommunikation zwischen den einzelnen Anwendungen kontrolliert. Für Anwendungen, die auf sensible Telefonfunktionen zugreifen wollen, existiert eine Rechtevergabe ähnlich zu Android, die sich auch in einer Datei mit dem Namen *Manifest* finden. Diese Datei wird jedoch nicht direkt durch den Benutzer, sondern durch die jeweilige Entwicklungsplattform geändert, d. h. es werden nur Rechte vergeben, welche die Anwendung auch wirklich benötigt und keine die durch den Entwickler per Hand hinzugefügt wurden. Als weitere Sicherheitsmaßnahme sind alle benötigten Funktionen vor dem Kauf im Market Place durch den Benutzer einsehbar. Dies gilt auch für Standarddienste auf dem jeweiligen Gerät, die bei der ersten Benutzung die benötigten Funktionen anzeigen. Neben dem Sandbox-Prinzip für Anwendungen, speichern alle Programme ihre Daten in geschützten Ordnern ab. Diese sogenannten isolierten Ordner beinhalten nicht nur gespeicherte Daten, sondern auch Konfigurationsdateien für die jeweilige Anwendung. Die komplette Ein- und Ausgabe einer Anwendung läuft über diesen Ordner ab, der auch nur von dieser Anwendung benutzt werden kann. Als zusätzlichen Schutz können die Anwendungen nicht auf das Dateisystem darunter, also z. B. die Dateien des Betriebssystems zugreifen. Für die Speicherung von Objekten gibt es drei verschiedene Möglichkeiten, die auch in Abbildung 3.5 dargestellt sind. Die Speicherung von Schlüsseln oder besonderen Werten erfolgt in den *Setting-Objekten*. Dateien und Ordner werden durch das *Files and Folders-Objekt* gespeichert. Zum Schluss können auch komplette Datenbanken in diese isolierten Ordner gespeichert werden.
Ein weiterer Sicherheitsmechanismus ist der sogenannte Execution Manager, der auf der Basis von definierten Regeln Anwendungen überwacht. Falls eine Anwendung, die im Hintergrund läuft eine gerade aktuell laufende Anwendung beeinflusst, kann diese beendet werden, um z. B. die Reaktion der gerade bedienten Anwendung zu verbessern.
Die gerade analysierten Sicherheitsmechanismen schützen jedoch die gespeicherten Daten auf dem Smartphone nicht gegen Diebstahl oder Verlust. Dafür bietet Windows Phone

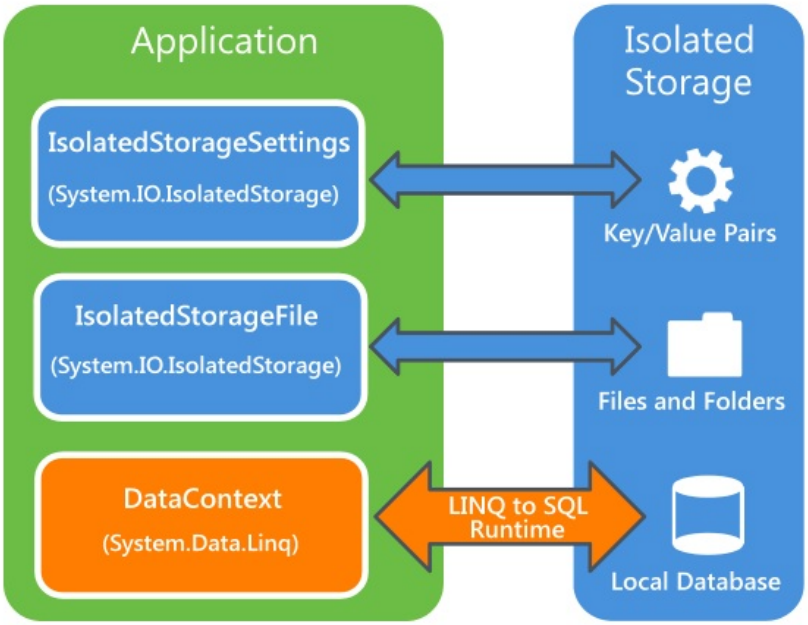

Abbildung 3.5: Überblick der isolierten Datenspeicherung in Windows Phone [19]

eine spezielle Schnittstelle an, bei der die Daten auf dem Gerät verschlüsselt werden können. Die *Data Protection API* (DPAPI) beinhaltet die *DataProtection* Klasse, die für die Ver- und Entschlüsselung der Daten genutzt werden kann. Die Verschlüsselung geschieht durch die Methode *Protect*, die neben der Eingabe des Klartextes auch einen Salt zur Verstärkung der Sicherheit gestattet. Die Methode *Unprotect* ist für die Entschlüsselung zuständig. Ein wichtiges Merkmal ist, dass jede Anwendungen über einen eigenen Schlüssel verfügt, der einmalig bei dem ersten Start berechnet wird. Windows Phone unterstützt dabei die folgenden Algorithmen:

- AES, RSA,
- SHA-1, SHA-256,
- HMACSHA-1, HMACSHA-256,
- den RFC 2895 DeriveByte.

Kommunikationssicherheit

Neben der Sicherheit auf dem eigentlichen mobilen Gerät, beinhaltet Windows Phone auch Mechanismen, die den Transport von Daten absichern sollen. In der Dokumentation wird dieser Bereich als Web Service Sicherheit bezeichnet und spaltet sich in die zwei Bereiche Authentifizierung und Verschlüsselung der Kommunikation auf. Die Authentifizierung erfolgt bei Windows Phone durch das sogenannte Basic Authentification. Hierbei

wird die Identitätsbeschreibung durch einen Benutzernamen und das zugehörige Passwort festgelegt. Das Problem hierbei ist jedoch, dass keine explizite Verschlüsselung der Informationen angewandt wird, falls sie übermittelt werden. Daraus ergibt sich die Notwendigkeit auch die Kommunikation zwischen zwei Partnern abzusichern. Dafür können die Protokolle SSL oder TLS genutzt werden, um z. B. mit Hilfe von HTTPS seine eigene Identität sicher zu bestätigen.

Anwendungssicherheit

Für die Entwicklung eigener Anwendungen gibt es noch weiterer Sicherheitsmechanismen. Jede Anwendung auf einem Windows Phone kann zunächst nur über den Hauseigenen Market Place von Windows bezogen werden. Weiterhin werden alle Anwendungen in diesem Market Place vorher durch zahlreiche Tests zertifiziert. Diese Tests sollen die Anwendungen untersuchen und ggf. Probleme erkennen, die dann zu einem Ausschluss führen würden. Ohne diese Zertifikate kann keine Anwendung auf einem Gerät mit Windows Phone ausgeführt werden. Neben der Zertifizierung soll auch die Verwendung von .NET Tools die Sicherheit der Anwendungen verbessern. Durch die strenge Typisierung und das Speichermanagement sollen programmierbedingte Fehler reduziert werden. Dies soll Hackern die Möglichkeit erschweren, Lücken oder Exploits auszunutzen [20].

3.4.4 BlackBerry OS

Die Smartphones der Firma Research In Motion (RIM) zeichnen sich vor allem durch ihre Sicherheitsmechanismen aus. Der Hauptunterschied zu den bereits vorgestellten Betriebssystemen und BlackBerry OS sind die Sicherheitszertifikate, die von unabhängigen Unternehmen ausgestellt wurden. Research in Motion ist in verschiedenen Programmen tätig, darunter z. B.:

- Cryptographic Module Validation Program in Nordamerika
- Internationales Common Criteria-Bewertungsprogramm
- Sicherheitsbewertung durch das Fraunhofer-Institut für Sichere Informationstechnologie in Deutschland

um die Sicherheit ihre Produkte bewerten zu lassen. In der Bewertungsskala der Common Criteria erhalten alle Geräte bzw. die Enterprise Server mit der Version 4 das Evaluation Assurance Level EAL2+. Dies bedeutet, dass diese Geräte strukturell getestet wurden und der Stufe Q1 der Deutschen IT Sicherheitskriterien entsprechen. Die Enterprise Server mit der Version 5 erhielten die Einstufung EAL4+, das nach den deutschen Sicherheitskriterien als Q3 und somit als Starker Schutz eingestuft wurde. Neben der Bewertung durch die Common Criteria, bestanden die Produkte von RIM auch den Test des Fraunhofer-Instituts für Sichere Informationstechnologie. Des Weiteren wurden die kryptographischen Module, die zur Berechnung der Verschlüsselung oder zur Entschlüsselung eingesetzt werden validiert und erhielten den FIPS 140-2 (Federal Information Processing Standard)

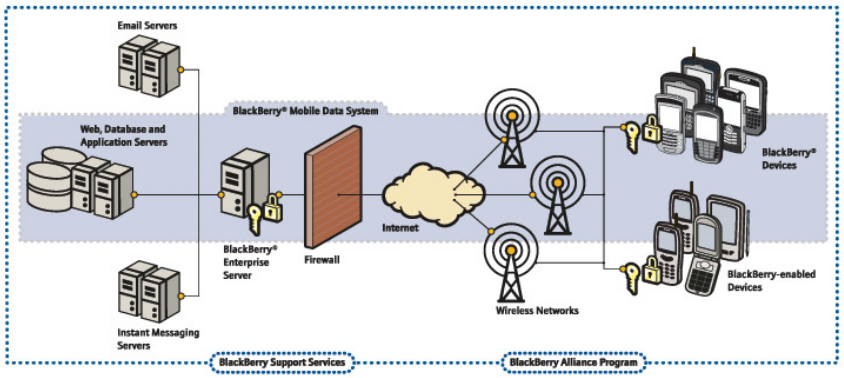

Abbildung 3.6: Überblick über die Architektur von BlackBerry Geräten [21]

Standard, der Sicherheitsanforderung für Kryptografische Module beschreibt [21]. Ein weiterer wichtiger Unterschied zu den bereits vorgestellten Betriebssystemen ist eine zusätzliche Komponente. Das Smartphone kommuniziert mit einem sogenannten Enterprise Server. Dieser Server ist im Firmennetzwerk integriert und soll die Kommunikation zwischen den Smartphones, dem Firmennetz und den Mobilfunkprovidern koordinieren. Ein Beispielaufbau wird in Abbildung 3.6 gezeigt. Neben diesen Zertifikaten und dem zusätzlichen Server gliedert sich die Sicherheitsarchitektur von BlackBerry in die Bereiche

- Mobile Datensicherheit und
- Datensicherheit auf den Geräten auf.

Mobile Datensicherheit

Die mobile Datensicherheit besteht zudem wieder aus mehreren Teilbereichen für Authentifizierung, Verschlüsselung und Signierung der Anwendungen. Der erste Sicherheitsaspekt ist, dass jede Übertragung von einem Enterprise Server zu einem Smartphone verschlüsselt gesendet wird. Die Verschlüsselung wird durch die Algorithmen AES oder TripleDES sichergestellt. Die passenden Schlüssel für diese Verschlüsselungen werden in einer sicheren Umgebung erstellt und dem Benutzer des Smartphones zugewiesen. Weiterhin wird der Schlüssel zu dem passenden Benutzer auf dem Server hinterlegt, damit dieser auch die Daten verschlüsseln kann. Dieser wird jedoch nur in der Unternehmens-Mailbox von den drei Anbietern Microsoft Exchange, Lotus Domino oder Novell GroupWise gespeichert. Der Benutzt kann jedoch jederzeit, auch mobil den Schlüssel neu generieren lassen, was für zusätzliche Sicherheit sorgen soll. Die komplette Verschlüsselung beruht also auf den oben beschriebenen symmetrischen Verfahren, bei dem die Nachricht mit dem gleichen Schlüssel bei dem Server verschlüsselt und auf dem Smartphone entschlüsselt wird. Für den sicheren Datenaustausch mit Anwendungsservern können die Protokolle TLS und SSL benutzt werden, um so HTTPS Verbindungen zu nutzen. Dabei werden zwei Modi unterstützt. Der Proxy-Modus beschreibt dabei erst die Kommunikation zwischen dem Anwendungsserver und dem Enterprise Server, die durch SSL oder TLS abgesichert ist. Der Anwendungsserver verschlüsselt danach erst die Daten mit AES oder TripleDES und

sendet diese an das Smartphone. Der zweite Modus beschreibt eine Ende-zu-Ende Verschlüsselung und wird vor allem genutzt, wenn nur den direkten Kommunikationspartner vertraut werden kann. Ein weiteres Merkmal der Sicherheitsmechanismen ist die Unterstützung der Lotus Notes E-Mail Verschlüsselung, die seit Version 4.1 besteht. Damit können auch per Lotus Notes verschlüsselte E-Mail auf dem Smartphone gelesen werden. Zur Authentifizierung benutzt BlackBerry das RSA SecurID Two-Factor Authentifizierungsverfahren. Einfach ausgedrückt benötigt der Benutzer eines Smartphones ein bestimmtes Wissen, in diesem Fall seinen Benutzernamen, und ein Token-Passwort für die Authentifizierung. Die Authentifizierung wird dann durch einen RSA Server sichergestellt, der die Kombination aus dem Benutzernamen und dem Token-Passwort prüft und dann den Zugriff auf das interne Netz oder einer Webseite frei gibt. Die Signierung von Anwendungen ist ein weiterer Sicherheitsaspekt. Dabei müssen alle Anwendungen, die auf potentiell sensible Daten zugreifen oder z. B. bei dem Start des Smartphones ausgeführt werden durch die Entwickler bei RIM signiert und registriert werden. Die Signierung erfolgt mit Hilfe von Zertifikaten nach dem X.509 Standard und kann auf Zertifikate von Zertifizierungsstellen (CAs) oder generierte Zertifikate zugreifen. Dadurch soll die Kontrolle und Vorhersagbarkeit des Smartphone Verhaltens verbessern. Zusätzlich erfolgt, wie bei Apple, eine Überprüfung der einzelnen Anwendungen, bevor sie indem Market Place angeboten werden [22].

Datensicherheit

Die Datensicherheit auf dem Gerät ist der zweite Punkt in der Sicherheitsanalyse. Wie bereits bei dem Betriebssystem iOS beschrieben, können auch bei BlackBerry IT Sicherheitsrichtlinien etabliert werden. Diese Regeln werden durch den Enterprise Server definiert und auf die Smartphones übertragen. Ein Beispiel wäre die Forderung auf allen Geräten Passwörter einzusetzten, um den direkten Zugriff durch Dritte zu verhindern. Verbunden mit der Einstellung eines Passwortes ist auch eine maximale Grenze von zehn Fehlversuchen nach der dann alle Daten auf dem Gerät gelöscht werden. Ein weiteres Beispiel ist die Vorgabe einer kompletten Verschlüsselung der Daten auf dem Gerät und der sicheren Ablage der Schlüssel in sogenannten Kennwort-Keepern. In diesen Kennwort-Keepern können auch die Anmeldedaten von Banking-Konten oder anderen Anwendungen gespeichert werden. Neben diesen Beispielen ist die Remote-Administrierung der Geräte durch den Enterprise Server ein weiterer Sicherheitsaspekt. So lassen sich z. B. Kontrollnachrichten erstellen und versenden, die das Passwort ändern, das Gerät sperren oder die Daten auf dem Smartphone löschen, falls dieses gestohlen wurde oder verloren ging.
Um die Sicherheit ihrer Produkte weiter zu fördern, können auch Sicherheitsschnittstellen von Drittanbietern genutzt werden. Beispiele für Erweiterungen sind die Unterstützung für die Pakete PGP und S/MIME, welche die Sicherheit des E-Mail Verkehrs durch eine weitere Verschlüsselung erhöhen sollen. Neben der Sicherung der Geräte und der Kommunikationswege wird auch der Enterprise Server abgesichert. Der Server dient dabei nur als Vermittler und leitet die E-Mail Nachrichten weiter. Er verfügt nicht über einen Speicher, in dem die Nachrichten abgelegt werden und der angegriffen und ausgelesen werden könnte. Weiterhin werde nur Verbindungen von den Smartphones zugelassen [21].
Die Sicherheitsaspekte beruhen jedoch fast ausschließlich auf der Enterprise Solution, d. h. der Nutzung der BlackBerry Geräte in einem Unternehmensumfeld. Für Privatanwen-

der steht ein solcher Server nicht zur Verfügung. Doch zumindest für den E-Mail Versand stehen dem Privatanwender sogenannte BlackBerry Internet Service (BIS) Server zur Verfügung. Diese Server werden von den Mobilfunkbetreibern verwaltet, so dass BlackBerry Nutzer ihre E-Mail Konten mit ihrem Gerät verwalten können.

3.5 Zusammenfassung

Abschließend ist zu sagen, dass alle vier Plattformen die Sicherheit ihrer Geräte durch zahlreiche Maßnahmen schützen wollen. Vor allem das Sandbox-Prinzip soll die Geräte vor schädlicher Software schützen, da sie keinen Zugriff auf andere Anwendungen oder deren Daten gestattet. Besonders starke Einschränkungen und Zugriffsbeschränkungen bietet die Plattform Android, da auch bei möglichen Einbrüchen durch Schadprogramme wenige Möglichkeiten für Datendiebstahl bestehen. Neben dieser Maßnahme sind vor allem auch die Tests der Anwendungen ein weiteres Sicherheitsmerkmal, dass es Angreifern erschweren soll, Malware auf den Geräten zu platzieren. Gute Beispiele dafür sind die Apps Stores von iOS und Windows Phone aber auch BackBerry unterzieht den Anwendungen einen Test.
Im Umkehrschluss fanden sich jedoch auf alle untersuchten Betriebssysteme Angriffsmöglichkeiten. Diese sind in vielen Fällen auch auf den Anwender zurückzuführen, der z. B. bei Android Geräte die geforderten Rechte der Anwendung einfach hinnimmt. Ein weiteres Beispiel sind Geräte mit iOS, die durch einen Jailbreak wichtige Sicherheitsmechanismen verlieren und so anfälliger für Malware werden. Da Smartphones in der heutigen Zeit immer stärker an Bedeutung gewinnen, werden sie auch stärker in den Fokus von Hackern geraten.
Daher sollte man auch mit seinem Smartphone verschiedene Regeln beachten. Falls der Nutzer des Gerätes einfache Sicherheitsmaßnahmen beachtet und die Anwendungen auch die zur Verfügung stehenden Maßnahmen benutzt sind die Smartphones z. B. im Gegensatz zu Desktop Rechnern noch sicher.

Literaturverzeichnis

[1] Projektgruppe E-Government. *Verschlüsselung und Signaturen*, Bundesamt für Sicherheit in der Informationstechnik, Bonn, 2005.

[2] J. Kilian and M. Robshaw. *How to Protect DES Against Exhaustive Key Search*, Advances in Cryptology - CRYPTO '96 Proceedings, Springer-Verlag, 1996.

[3] Bruce Schneier *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 191-204.

[4] Bruce Schneier, John Kelseyy, Doug Whitingz, David Wagnerx, Chris Hall, Niels Ferguson *Twofish: A 128-Bit Block Cipher*, Counterpane Internet Security Inc, 1998.

[5] Federal Information Processing Standards Publication 197. *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*, Federal Information Processing Standards Publications, 2001.

[6] StarShaper, Laurent Haan. *Advanced Encryption Standard (AES)*, `http://www.codeplanet.eu/tutorials/cpp/51-advanced-encryption-standard.html#aes_introduction`, Oktober, 2011.

[7] R.L. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM Volume 21, 1978, pp. 120-126.

[8] W. Diffie and M.E. Hellman. *New directions in cryptography*, IEEE Transactions on Information Theory 22, pp. 644-654, 1976.

[9] Federal Information Processing Standards Publication 180-2. *SECURE HASH STANDARD*, Federal Information Processing Standards Publications, 2002.

[10] R. Rivest. *RFC1321 - The MD5 Message-Digest Algorithm*, RFC Editor , United States, 1992.

[11] Udo Helmbrecht. *Skript Vorlesung Sicherheit in der Informationstechnik - Grundlagen der Kryptographie Verfahren, Protokolle*, 2011.

[12] BSI - Technische Richtlinie. *Kryptographische Verfahren: Empfehlungen und Schlüssellängen*, Bundesamt für Sicherheit in der Informationstechnik, 2008.

[13] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen. *Algorithmenkatalog 2011*, Veröffentlicht im Bundesanzeiger Nr. 85, p. 2034, 2011.

[14] ECRYPT II - European Network of Excellence in Cryptology II. *ECRYPT II Yearly Report on Algorithms and Keysizes (2010-2011)*, ICT-2007-216676, 2011.

[15] Hans Dobbertin, Vincent Rijmen, Aleksandra Sowa. *The Cryptanalysis of the AES - A Brief Survey*, Lecture Notes in Computer Science, SpringerLink, 2005.

[16] Android Open Source Project. *What is Android?*, `http://developer.android.com/guide/basics/what-is-android.html`, Oktober, 2011.

[17] Android Open Source Project. *Android Security Overview*, `http://source.android.com/tech/security/index.html`, November, 2011.

[18] Apple Inc. *Deploying iPhone and iPad - Security Overview*, 2011.

[19] Microsoft Corporation. *Isolated Storage Overview*, `http://msdn.microsoft.com/en-us/library/ff402541%28v=VS.92%29.aspx`, November, 2011.

[20] Microsoft Corporation. *Security for Windows Phone*, `http://msdn.microsoft.com/en-us/library/ff402533%28v=VS.92%29.aspx#BKMK_SecurityAppSafeguards`, November, 2011.

[21] Research In Motion Limited. *BlackBerry Sicherheit*, `http://de.blackberry.com/ataglance/security/features.jsp#tab_tab_wireless_data`, November, 2011.

[22] Dirk Fox. *BlackBerry Security - Das Sicherheitskonzept - Übersicht und Bewertung*, Datenschutz und Datensicherheit 29, 2005.

[23] Bundesamt für Sicherheit in der Informationstechnik. *Öffentliche Mobilfunknetze und ihre Sicherheitsaspekte*, Bonn, 2008.

[24] Bundesamt für Sicherheit in der Informationstechnik. *Drahtlose Kommunikationssysteme und ihre Sicherheitsaspekte*, Bonn, 2009.

[25] Heise Zeitschriftenverlag GmbH und Co KG. *ct kompakt - Security*, März, 2011.

Kapitel 4

Fernverwaltung und -administration

Stephan Mehner

Dieses Kapitel stellt einen Überblick über vorhandene Lösungen zur Fernverwaltung und -administration im Privat- und Unternehmenssektor dar. Dabei wird auf folgende Fragen näher eingegangen: Wie können Smartphones fernverwaltet und fernadministriert werden? Welche Software wird hierfür benötigt? Wie können Smartphones ferngesperrt und/oder ferngelöscht werden? Wie kann man Smartphones den gängigen Unternehmenssicherheitsrichtlinien entsprechend einschränken? Was genau kann eingeschränkt werden? Das hieraus entwickelte Anforderungsmodell wird mit der Lösungsauswahl verglichen und ein entsprechender Ausblick dargestellt.

Inhaltsverzeichnis

4.1 Einleitung 117
4.1.1 Begriffsklärung 117
4.1.2 Anforderungsmodell 119
4.2 Produkte für den privaten Sektor 120
4.2.1 Apple Mobile Me und iCloud 120
4.2.2 Samsung Dive 120
4.2.3 Microsoft My Phone 122
4.2.4 Trend Micro Mobile Security PE 122
4.3 Produkte im Enterprise-Bereich 123
4.3.1 Apple Mac OS X Lion Server 123
4.3.2 Google Apps for Business 124
4.3.3 BlackBerry Enterprise Server 125
4.3.4 BlackBerry Management Center 126
4.3.5 Microsoft Exchange Server 2010 127
4.3.6 Trend Micro Mobile Security 7.1 127
4.3.7 SAP/Sybase: Afaria 128
4.3.8 Odyssey Software: Athena 129
4.4 Bewertung der Lösungen 131
4.4.1 Ausblick 133

4.1 Einleitung

Jedes Unternehmen hat seine eigenen Richtlinien und Vorgaben zur Integration von Endgeräten in das eigene Unternehmensnetzwerk. Meist enthalten diese eine Forderung nach einem aktuellen Virenscanner, welcher auf dem Gerät installiert sein muss, sowie Forderungen bezüglich der Speicherung von unternehmenskritischen Daten auf den mobilen Endgeräten. Der Zugang über WLAN darf zum Beispiel nur über ein unternehmensspezifisches Zertifikat inklusive Intranetanbindung via Tunnel erfolgen und die Mitarbeiter dürfen in der Regel selbständig keine Software auf dem Gerät installieren.
Bezüglich der neuen Funktionsumfänge von Smartphones ergeben sich daraus neue Fragestellungen bezüglich der Verhaltensregeln für die Mitarbeiter. Dürfen diese die betriebssystemspezifischen App-Stores nutzen, welche Standarddienste gilt es zu sperren, nach welcher Zeitspanne der Inaktivität soll das Gerät automatisch gesperrt werden und wie sicher sollte das Passwort zur Entsperrung des Geräts sein. Entsprechend kann man die Einbindung eines mobilen Endgerätes in die Infrastruktur eines Firmennetzwerkes in drei Kategorien einordnen: Installation, Konfiguration und Service.
In die Kategorie Installation fallen alle notwendigen Maßnahmen zur Authentisierung und Autorisierung, die Installation von Unternehmenssoftware und Unternehmensdiensten. In den Bereich Konfiguration fallen die Anpassung an die vorhandenen Policies/Zertifikate, gegebenenfalls das Sperren von Diensten, die Generierung von Tokens und der Benutzerkennung. Der Servicebereich umfasst die Verwaltung der Benutzerrechte, das Monitoring, die Einbindung in den Betrieb und die Dokumentation (UDID, InventarNr.,...). Gerade für größere Firmen ist die Fernverwaltung der mobilen Geräte ihrer Mitarbeiter interessant. So sollten die Lösungen für diesen Markt auf die Bedürfnisse zugeschnitten sein [1].
Genau deswegen muss man aber eine Abgrenzung zwischen Produkten des Enterprise-Marktes und Lösungen für den Normalverbraucher treffen. Dies wird spätestens bei der Betrachtung des Einsatzszenarios klar. Der Verlust eines Gerätes und Nutzungsrestriktionen betreffen beide Gruppen, während Fernupdates, Profilmanagement und -verwaltung hauptsächlich aus unternehmerischer Sicht interessant sind. Da auf dem Markt nicht nur Unternehmenssoftware zu finden ist, werde ich auch Lösungen für den privaten Sektor vorstellen. Ein Anforderungsmodell soll bei der Unterscheidung und Klassifizierung helfen. Vorher werden noch einige Begriffe und Funktionen einführt.

4.1.1 Begriffsklärung

Unter Fernadministration versteht man allgemein den Fernzugriff auf Elemente der IT-Infrastruktur. Dieser Vorgang wird als ”remote administration” bezeichnet und umfasst alle Mittel, die der Administrator zur Administration auf Systemebene, via externer Server-Konsolen und des Command Line Interface einer Netzwerkkomponente, benötigt [2]. Dies entspricht also den technischen Gegebenheiten und entsprechend befasst sich die Fernverwaltung mit diesen Aufgaben. Im Fordergrund steht hierbei die ”Fehlersuche und Beseitigung von Problemen durch sichern Zugriff und Fernverwaltung von Computern, ohne die Leistung von Endbenutzern zu beeinträchtigen” [3]. Zusammengefasst beinhaltet dies alle Maßnahmen zur Aufrechterhaltung des Betriebs und der Nutzungsfähigkeit inklusive der drei Kategorien Installation, Konfiguration und Service. Die Erweiterung auf den

mobilen Sektor nennt sich "Mobile Device Managament" (MDM) und umfasst Funktionen zur Verwaltung von Smartphones (im Unternehmenseinsatz). Dies schließt die drahtlose Bereitstellung, "over the air", dieser Funktionen mit ein.
Grundlegend sei hierbei der Wunsch nach der "Kontrolle" des Gerätes aus der Ferne. Funktionen, welche am mobilen Endgerät nicht aus unmittelbarer Nähe ausgeführt werden können, sind von besonderem Interesse:

- remote alarm
  ... lässt das Gerät ferngesteuert klingeln und kann teilweise mit einer selbstdefinierten Nachricht versehen werden.

- remote detection
  ... lokalisiert die Position des Geräts mithilfe von GPS oder verbundener Netze. Die Genauigkeit ist von der Netzqualität, netzanbieterbezogenen Restriktionen und den verbundenen Netzen abhängig.

- remote lock
  ... Reduzierung der Gerätefunktionen auf ein bestimmtes Minimum, um den weiteren Gebrauch einzuschränken beziehungsweise zu verhindern.

- remote wipe
  ... Löschung aller, beziehungsweise vorher definierter, Daten auf dem Gerät inklusive der Einstellungen, Protokolle, Zertifikate und Sicherheitsschlüssel.

Beim letzten Punkt ist zu beachten, dass die Qualität der Löschung ausschlaggebend für den Erfolg und die Nutzerzufriedenheit ist. Zum Beispiel sollten private Daten von solchen Maßnahmen ausgeschlossen werden, da im Enterprise-Markt Mitarbeiter vermehrt von ihren privaten Geräten auf firmeninterne Daten zugreifen wollen. Dieses Verschwimmen der Grenze von Arbeitswelt und Privatleben und die daraus resultierenden Szenarien fässt man unter "Consumerisation" beziehungsweise "Bring your own Device" zusammen [4] [31].
Die Unternehmenssicherheitsrichtlinien müssen also im mobilen Bereich auch solche Fälle abdecken können. Es ist zu beachten, dass gerade hier die jeweilige Richtlinie abhängig vom jeweiligen Standort ist und nicht in jedem Unternehmen aufgrund der internen Struktur gleich umgesetzt werden kann. Anpassungsmöglichkeiten an verschiedene Profile sollten also vorhanden sein. Dies schließt auch eventuelle Nutzungsrestriktionen in Gerätefunktionen (Wi-Fi, Kamera, Bluetooth,...) und in Programmen (Browser, Facebook, App Stores,...), als auch Fernverwaltungsmöglichkeiten für E-Mail-Konten, Kalender und Adressbücher der Mitarbeiter mit ein. Diese Profile sollten natürlich aus Unternehmenssicht over-the-air-aktualisierbar sein, um eine schnelle Problemlösung garantieren zu können und den Vorteil der mobilen Effektivität der Mitarbeiter zu erhalten.

4.1.2 Anforderungsmodell

- Fähigkeit zu remote alarm, detection, lock und wipe
- Erstellung und Verwaltung von (Konfigurations-)Profilen für
  - E-Mail-Konten, Kalender, Adressbücher
  - Wi-Fi-Verbindungen, Zertifikate, VPNs
  - Apps, Gerätefunktionen
- Erstellung von Nutzungsrestriktionen für Programme und Funktionen
- Fähigkeit zu over-the-air-Updates und -Backups
- Verwaltung unterschiedlicher mobiler Betriebssysteme

Das von mir erstellte Anforderungmodell umfasst wichtige Funktionen für den Unternehmenssektor zur Fernverwaltung und -administaration von Smartphones. Sollten also alle Punkte erfüllt sein, dann ist aus meiner Sicht eine Software geeignet. Werden nicht alle Punkte erfüllt oder gar nur Funktionen aus dem ersten Stichpunkt, ist eine solche Software nur bedingt für den Enterprise-Markt geeignet. Dies gilt aber nicht zwangsläufig für den privaten Gebrauch, da hier im Allgemeinen die Anforderungen niedriger sind.
Nachdem also die theoretische Grundlage geschaffen ist, stelle ich im nächsten Abschnitt die am Markt befindlichen Lösungen vor.indlichen Lösungen vor.

4.2 Produkte für den privaten Sektor

Der Einblick in den privaten Sektor soll als Einstieg und gleichzeitig als Abgrenzung zu Enterprise-Lösungen dienen. Diese Lösungen sind meist betriebssystemspezifisch und je nach Anbieter auch mit Zusatzfunktionen ausgestattet, welche in den zugehörigen Quellen bei Bedarf nachgelesen werden können. Zu beachten sei außerdem, dass es von vielen Softwareherstellern (mindestens) eine Lösung für den privaten und den Enterprise-Markt gibt.

4.2.1 Apple Mobile Me und iCloud

Mit Mobile Me [11] hat Apple im Juni 2008 ein Webtool zu Verwaltung von iOS-Geräten als Nachfolger von .Mac vorgestellt. Nachdem man sein Gerät auf der Webseite registriert hat, kann man neben Funktionen wie Push-Diensten für E-Mail, iCal, Synchronisation, Webanwendungen und vielem mehr seit Juni 2009 die kostenpflichtigen Funktionen der Geräteortung, inklusive der Möglichkeit das Handy klingeln zu lassen oder eine Nachricht auszulösen und der Fernlöschung der kompletten Daten nutzen. Mit dem iOS 3.1 wurde auch die Möglichkeit eingebaut, mithilfe eines vierstelligen Zugangscodes das Gerät zu sperren. Leider wies der Dienst gerade in den Anfangszeit Probleme mit teilweise hohen Antwortzeiten auf. Seit November 2010 sind aber diese bisher kostenpflichtigen Funktionen für Geräte, welche das iOS 4.2.1 unterstützen, kostenlos.
Mobile Me nimmt momentan keine neuen Kunden mehr auf, Bestandskunden können aber bis Ende Juni 2012 den Dienst weiterhin nutzen [12]. Hintergrund ist die Migration in den neuen Dienst iCloud [14]. Dieser Nachfolger von Mobile me ist mit fast alle Funktionen des alten Dienstes ausgestattet [13]. Zumindest bleibt die iPhone-Suche erhalten und es wird wieder einen webbasierten Zugangspunkt geben. Mit einem Gerät, welches iOS 5 unterstützt, ist dieser Dienst auch kostenlos nutzbar. Interessant hierbei ist, dass die Server für diesen Dienst womöglich nicht von Apple selbst, sondern von Amazon und Microsoft bereitgestellt werden [19].
Zusätzlich ist es möglich, automatisch täglich ein Backup des Gerätes erstellen zu lassen [15]. Neben Fotos und Apps werden auch Geräteeinstellungen (unter anderem Accounts in E-Mail, Kontakte, Kalender), App-Daten und Nachrichten gesichert. Dies geschieht over-the-air via WLAN. Ein Backup kann auf dem gleichen Wege aufgespielt werden. Hierfür ist nur die eigene Apple-ID, das zugehörige Kennwort und eventuelle Kennwörter für Funktionen innerhalb des Backups notwendig.

4.2.2 Samsung Dive

Samsung bietet für seine Smartphones auf Android-Basis einen eigenen Dienst zur Fernverwaltung an, Samsung Dive [20]. Der Dienst an sich ist kostenlos, bis auf die Kosten für die Datenübermittlung, die je nach Netzbetreiber unterschiedlich sein können. Es wird aber zwingend ein Google Account zur Anmeldung benötigt. In über 130 Ländern ist Dive nutzbar, aber in China ist dies aufgrund interner Beschränkungen nicht möglich.

Zum Funktionsumfang gehören:

- Mobilgerät finden
- Ortung meines verlorenen Mobilgerätes
- Fernsperre des Mobilgerätes mit Meldung
- Anrufsperre
- Fernaktivierung des Tons
- Fernabfrage des Anrufprotokolls
- Fernlöschung des Mobilgerätes

Die zusätzliche Funktion "Mobile Tracker Service", welche nur von den Wave, Wave II und Galaxy S Modellen unterstützt wird, ist momentan in England, Deutschland und Thailand verfügbar und setzt die erstmalige Aktivierung im Gerät voraus. Nur wenn diese Funktion auf dem Gerät aktiv ist, funktioniert remote lock und wipe. Wird die Fernverwaltungsfunktion in den Einstellungen deaktiviert, bleibt nur die Anrufweiterleitung weiterhin funktionsfähig.
Dive funktioniert auch mit Roaming, solange eine Verbindung zwischen dem Gerät und dem Server zustande kommt. Ebenso ist es auch mit deaktiviertem GPS funktionsfähig, da nur die Funktion "Track my mobile" und "Find my mobile" dies benötigen. Es ist aber möglich, GPS per Fernzugriff über Dive zu aktivieren. Sollte dies nicht gelingen, kann eine ungefähre Position mithilfe der 3G-Netze und aktiver Wi-Fi-Verbindungen erfolgen. Außerdem aktualisiert "Track my mobile" den Aufenthaltsort bei Aktivierung alle 15 Minuten für insgesamt 12 Stunden. Die so gesammelten Aufenthaltsorte können innerhalb von Dive eingesehen werden.
mt 12 Stunden. Die so gesammelten Aufenthaltsorte können innerhalb von Dive eingesehen werden.
Wenn das Gerät gestohlen wurde und eine neue SIM-Karte eingelegt wird, meldet Dive dies dem Nutzer mittels einer PopUp-Nachricht. Je nach Mobilfunkanbieter wird dabei sogar die neue Nummer mit angezeigt. Der SIM-Kartenwechsel beeinträchtigt die Funktion von Samsung Dive nur im Bereich Anruf-/Nachrichtenweiterleitung, alle anderen Funktionen bleiben intakt. Bei Fernsperrung des Geräts sind nur noch Notrufe möglich, sowie in Dive vom Anwender definierte Nummern wählbar und Nachrichten, welche durch Dive gesendet werden, lesbar. Mit der Funktion "Wipe out my mobile" lässt sich entweder das Gerät in den Werkszustand zurücksetzen (alle Daten werden gelöscht) oder der angeschlossene externer Speicher und/oder die SIM wird gelöscht. Durch den Werkszustand entfallen aber auch alle Funktionen von Dive zu diesem Smartphone.
Der Zeitraum bis zur Erfolgsmeldung der ausgewählten Aktion kann bis zu über einer Stunde reichen, je nach Netzqualität am mobilen Endgerät [21].

4.2.3 Microsoft My Phone

Der Webdienst My Phone [32] von Microsoft für Windows Phone 7 Geräte ist der Knotenpunkt zwischen dem Smartphone und anderen Windows Diensten, wie iTunes, SkyDrive, xBox LIVE, Microsoft Office und seinen gekauften Apps. Interessant ist dabei der kostenlose "Find my Phone"-Dienst, welcher die Lokalisierung eines vorher mit einer Windows Live ID angemeldeten Smartphones erlaubt. Dies geschieht entweder via SMS, was auf 15 Einheiten pro drei Tage beschränkt ist, oder Push-Benachrichtigungen. Ebenso lässt sich auf der Web-Konsole, welche als Zugangspunkt dient, eine automatische Verfolgung einstellen. Weiterhin kann man über diese Konsole einen remote alarm, lock und wipe ausführen. Weitere Funktionen zur Fernsteuerung sind aber momentan nicht implementiert. Die "Backupfunktion" umfasst nur die bisher gekauften Anwendungen, gespeicherte Office Dokumente und Bilder und Videos via SkyDrive.

4.2.4 Trend Micro Mobile Security PE

Mit der Software Mobile Security Personal Edition von Trend Micro gibt es ein Produkt für den Normalverbraucher auf dem Markt, der neben Virenschutz und vordefiniertem Websiteblocker auch noch andere interessante Möglichkeiten anbietet [28]. Voraussetzung ist ein Smartphone, welches mindestens Android 2.2 unterstützt. Die Kindersicherung bietet drei verschiedene vordefinierte Richtlinien für die vorsorgliche Sperrung von Webseiten. Ebenso kann ein Anruf- und SMS-Filter eingesetzt werden. Dies alles kann aber nur manuell am Gerät konfiguriert werden.
Hingegen bietet Trend Micro einen eigenen Verlustschutz in Form einer Weboberfläche. Nach der Installation der Software auf dem Smartphone meldet man sein Gerät über die App auf jener Internetseite an. Dadurch werden folgenden Optionen freigeschalten:

- Gerät orten, sperren
- Alarm auslösen
- Gerätedaten löschen
- SIM-Kartensperre einrichten

Die SIM-Kartensperre erfolgt automatisch, sobald diese aus dem gestohlenen Gerät herausgenommen wird. Bei der Löschung der Gerätedaten wird sowohl der Geräte- als auch der Zusatzspeicher gelöscht. Das Produkt lässt sich 30 Tage kostenlos testen und verlangt nach Ablauf den Erwerb einer Lizenz.

4.3 Produkte im Enterprise-Bereich

Mit dem Einstieg durch die Produkte für den Normalverbraucher merkt man sehr schnell, dass all diese Lösungen nicht für den Unternehmensbereich ausgelegt sind. Hier will man die entsprechende Software auf dem eigenen Server haben, um eine entsprechende Verfügbarkeiten und Integritäten zu erhalten. Entsprechend werden in den folgenden Abschnitten Softwarelösungen von unterschiedlichen Anbietern präsentiert. Der gesteigerte Funktionsanspruch eines Unternehmens an eine Fernverwaltungssoftware im Gegensatz zum Normalverbraucher wird schon im ersten Produkt ersichtlich.

4.3.1 Apple Mac OS X Lion Server

Mac OS X Lion Server [17] ist ein Serverbetriebssystem für Mac Rechner, welches im Juni 2011 veröffentlicht wurde. Die Ausrichtung ist auf den Businessbereich zu sehen und entsprechende Funktionsumfänge sind vorhanden. Die Server App als zentrales Verwaltungstool dient zur Administration des Mac als Server. Damit lassen sich servertypisch Benutzer und Gruppen verwalten. Spezifische Sets von Einstellungen können jedem Benutzer und jedem Gerät, sowie den entsprechenden Gruppen, zugeordnet werden. Dabei kann zwischen Benutzer und zugeordnetem Gerät unterscheiden werden, was eine hohe Flexibilität mit sich bringt und so bestimmte Beschränkungen und Netzwerkeinstellungen je nach Gruppeneinteilung und Umstand konfiguriert und übernommen werden können [17].
sinessbereich zu sehen und entsprechende Funktionsumfänge sind vorhanden. Die Server App als zentrales Verwaltungstool dient zur Administration des Mac als Server. Damit lassen sich servertypisch Benutzer und Gruppen verwalten. Spezifische Sets von Einstellungen können jedem Benutzer und jedem Gerät, sowie den entsprechenden Gruppen, zugeordnet werden. Dabei kann zwischen Benutzer und zugeordnetem Gerät unterscheiden werden, was eine hohe Flexibilität mit sich bringt und so bestimmte Beschränkungen und Netzwerkeinstellungen je nach Gruppeneinteilung und Umstand konfiguriert und übernommen werden können [17].
Speziell genannt sei hier die MDM-Server Komponente "Server für Verwaltung mobiler Geräte" des Profil-Manager [16], welche es dem Administrator erlaubt, registrierte Geräte fernzuverwalten. Dadurch lässt sich zum Beispiel die Gerätekonfiguration anpassen oder es können auf dem Gerät Ereignisse ausgelöst werden, wie remote lock oder wipe. Interessant dabei ist, dass für die erfolgreiche Sperrung erst ein Passwort auf dem Gerät definiert sein muss, was aber über entsprechende Profile erzwungen werden kann. Ebenso werden erstellte Konfigurationsprofile so over-the-air manuell, automatisch oder durch Abruf des Benutzers auf dem jeweiligen Gerät übernommen. Dabei wird zum Beispiel der Benutzer gezwungen, die als Payload zum Profil befindliche VPN-Lösung mit zu installieren, wenn er das Profil verwenden will. Wird ein Profil vom Gerät entfernt, so werden auch alle zugehörigen Beschränkungen und Einstellungen mit aufgehoben.
Neben dem Profil-Manager ist auch, bei aktiviertem WebDAV-Protokoll auf dem Server, die Datenfreigabe zum Beispiel für das iPad möglich. Damit können Dokumente mit Apps wie Keynote, Numbers und Pages auf dem Server, je nach Benutzer- und Gruppeneinstellung, gesichtet, kopiert und verwendet werden. Weiterhin bietet der integrierte

Push-Benachrichtigungsdienst die Möglichkeit zur E-Mail-, Kalendereinladung- und Kontaktdatenweiterleitung auf das mobile Endgerät.
Durch den E-Mail Server kann der Benutzer auch auf die im Server hinterlegten E-Mails und Anhänge zugreifen und diese durchsuchen. Dabei werden die gängigsten Dateiformate unterstützt. Die iCal-Server App ist eine standardisierte Komplettlösung für Kalenderfunktionen, welche zum Beispiel einen Chef beim Terminabgleich der verschiedenen Benutzer und Gruppen unterstützt, indem Kalender und Terminpläne, welche regisitriert sind und verwaltet werden, ausgetauscht und Veranstaltungen festgelegt werden können.

iPhone Configuration Utility

Ich möchte noch einmal kurz auf die Geräteprofile zurückkommen, welche sich mit dem iPhone Configuration Utility erstellen lassen [18]. Diese als XML-Datein gespeicherten Konfigurationsprofile erlauben es, die Gerätefunktion je nach Einstellung durch Restriktionen einzuschränken oder durch festgelegte Einstellungen zu erweitern. Damit lassen sich unter anderem die gerätespezifischen Sicherheitsrichtlinien und Einschränkungen, VPN-Konfigurationsinformationen, Wi-Fi-Einstellungen, Accounts für E-Mail- und Kalenderfunktionen und Authentifizierungszertifikate festlegen. Ebenso können Einstellungen und Autorisierungsinformationen mit übertragen werden. Durch Bereitstellungsprofile können außerdem auch unternehmensspezifische Apps ausgeführt werden, welche nicht über iTunes vertrieben werden. Diese Profile erlauben die Verwendung des Gerätes für die iOS-Entwicklung.
Die Übertragung kann per angeschlossenem Mac, per E-Mail, über das Web mit Safari oder over-the-air mit dem Profil-Manager von Mac OS X Lion Server erfolgen.

4.3.2 Google Apps for Business

Google Apps for Business [22] bietet für den Enterprise-Markt eine Reihe von Anwendungen zur Kommunikation und Zusammenarbeit. Zusätzlich untersützt die Google Apps Suite mit Google Apps for Mobile for Businesses für Android, iPhone, BlackBerry und Windows Mobile folgende Funktionen:

- Google Mail, Calender, Contact Sync
- Push support und Google Docs
- 2-step Verifikation
- Enterprise admin controls

Interessant dabei ist der letzte Punkt. Dabei handelt es sich um MDM-Funktionen, die je nach mobilem Betriebssystem andere Funktionen bieten. Windows Mobile und iPhone Geräte lassen sich mit einem Set von Managementrichtlinien einrichten. Enthaltene Funktionen sind zum Beispiel remote wipe, Passwortzwang und Passwortstärkevorgaben. Für BlackBerry-Geräte gibt es Google Apps Conncetor for BlackBerry Enterprise Server,

welche ähnliche Funktionalitäten bereitstellt [23]. Eine Liste von Einstellungsmöglichkeiten, wie unter anderem Geräteverschlüsselung, Kamerarestriktionen und Synchronisationsmaßnahmen, kann auf der Support-Seite [24] eingesehen werden. Android-Geräte können darüber hinaus mit Google Apps Device Policy konfiguriert werden.

Google Apps Device Policy

Diese Software bietet MDM-Konfigurationsmöglichkeiten für Android-Geräte [25]. Dabei wird zwischen der Nutzer- und der Administratorsicht unterschieden, auf welche ich kurz eingehen möchte, da diese unterschiedliche Funktionalitäten bieten.
Aus der Sicht des Nutzers [26] betrachtet, muss dieser als Voraussetzung ein Smartphone mit mindestens Android 2.2 besitzen, sowie für seine Domain die Aktivierung der Anwendung für Android-Geräte durch seinen Google Apps-Administrator veranlasst haben. Damit ist gewährleistet, dass, konform mit den IT-Sicherheitsrichtlinien des jeweiligen Unternehmens, die installierte Anwendung funktionsfähig ist. Nur dann ist es zum Beispiel möglich, Daten wie E-Mails, Kontakte, Kalender,... mit der eigenen Google Apps-Domain zu synchronisieren. Mittels Fernzugriff ist es dem Nutzer aber auch selbst möglich per Konsole "Meine Geräte" die PIN des Geräts zurückzusetzen, es klingeln zu lassen, es zu sperren oder auch zu orten und die Sicherheitsrichtlinien der Anwendungen sind immer transparent einsehbar.
Mit Hilfe der Anwendung ist es nun für den zuständigen Administrator [27] möglich festzulegen, ob das Gerät eine PIN-/Passwortabfrage besitzt, wie komplex dieses sein muss und ob der Bildschirm bei Inaktivität gesperrt wird. Bei Geräten, die der Domain des Administrators beitreten wollen, ist es für ihn möglich, die entsprechenden Sicherheitsrichtlinien zu erzwingen und gleichzeitig bei Verlust oder Diebstahl des Geräts die Daten darauf per Fernzugriff zu löschen und das Gerät zurückzusetzen.
Die Löschung der Daten eines Geräts zieht die Löschung aller Konten auf dem Gerät nach sich, einschließlich der persönlichen. Die Nachrichten können aber weiterhin über das Internet abgerufen werden. Auch ist die Deinstallation des Dienstes durch den Nutzer möglich. Dadurch gehen aber alle Funktionen der Anwendung verloren.

4.3.3 BlackBerry Enterprise Server

Als Software und Service ist BlackBerry Enterprise Server (BES) im Enterprise-Markt für BlackBerry Smartphones die Lösung von BlackBerry zur Verbindung mit Nachrichtensoftware und Groupware, wie zum Beispiel Microsoft Exchange, leitet E-Mails weiter und stellt die Synchronisation zwischen Server, Desktop und mobilem Endgerät sicher [6] [5]. Vorhandene Datensätze von unterstützter Groupware lassen sich mithilfe von Enterprise Transporter in die aktuellste Enterprise Server Version migrieren.
Mit der webbasierten Konsole "Administration Service" lassen sich Benutzer, Gruppen, Richtlinien, Softwarekonfigurationen und Anwendungen als Profil erstellen und verwalten.

Dadurch ergeben sich unter anderem folgende Funktionen als MDM-Server [7]:

- Benutzer können E-Mail-Konten, Kalender,... zugeordnet werden
- Benutzer können verschiedenen Gruppen gleichzeitig zugeordnet werden
- Verwaltung von Benutzereinstellungen
- Steuerung von Gruppen und Benutzern mit über 500 vordefinierten IT-Richtlinien
- over-the-air Installation und Verwaltung von Softwareprofilen mit zugehörigen ausgewählten Anwendungen
- zeitgesteuerte Updates von Geräten
- drahtlose BlackBerry Geräte- und Softwareupdates
- Einrichtung und Verwaltung einer Hochverfügbarkeitsarchitektur

Dabei können für jeden Benutzer und jede Gruppe die momentan zugewiesenen Geräte verwaltet, die erteilten Aufträge aktualisiert und bearbeitet werden und die mobilen Aktivitäten überwacht werden. Mithilfe des kompletten Paketes ist es so möglich, BlackBerry-Geräte mit erweiterten Funktionen zur Verwaltung (inklusive der Remote-Festsetzung eines neuen Passworts) zu nutzen und durch proaktive Mechanismen (Berichterstattung, Warnmeldungen und Fehlerbehebungen) mithilfe einer grafischen Aufarbeitung den ordnungsgemäßen Zustand des Systems aufrecht zu erhalten. Durch die integrierte Hochverfügbarkeitsarchitektur können ausgefallene Server schnell wiederhergestellt werden. Vom Administrator gesetzte Grenzwerte erlauben außerdem einen automatischen Ausfallschutz durch Aktivierung von Standyby-Komponenten auf physischen und virtuellen Systemen.

4.3.4 BlackBerry Management Center

Ein kostenloses Cloud-basiertes Fernverwaltungstool für bis zu 100 Mitarbeiter bietet Blackberry für BlackBerry-Smartphones mit dem Management Center an [8]. Dieses Programm, mit einer nach eigenen Angaben intuitiven Benutzeroberfläche, erlaubt es einem Mitarbeiter mit geringen technischen Kenntnissen die Smartphones seiner Firma zu verwalten. Die Funktionalität ist gegenüber dem BES eingeschränkt und umfasst folgende Funktionen [9] [10]:

- Verwaltung von E-Mail-Konten
- lokalisieren, sperren und löschen von Geräten
- senden von Nachrichten und remote alarm
- drahtloses Backup und Wiederherstellung für Ersatzsmartphones
- unbeschränkter Zugriff auf das Tool (Netzverbindung vorausgesetzt)

Notwendig für die Nutzbarkeit des Dienstes ist ein BlackBerry-Smartphone mit BlackBerry Device Software v.4.6 oder höher und ein Internetbrowser mit Internetanschluss. Bei Problemen kann und soll man auf die kostenpflichtige Supportfunktion von BlackBerry zurückgreifen. Das Programm soll außerdem auch Consumerization unterstützen, da es sowohl firmen- als auch mitarbeitereigene Geräte zentral verwaltet [9].

4.3.5 Microsoft Exchange Server 2010

Exchange Server 2010 [33] ist eine Groupware und ein Nachrichtendienst von Microsoft und kam im November 2009 auf den Markt. In einer Client-Server Umgebung wird mit dieser Software die Serverseite gestellt. Dabei lassen sich die Benutzer in verschiedene Gruppen mit jeweiligen Restriktionen einteilen, um auf Funktionen und Daten wie E-Mail, Kalender, Dokumente, Adressbuch und erstellte Intranets zugreifen zu können. Der Datenaustausch erfolgt über ein propriäteres Protokol MAPI beziehungsweise für MAC OS X über WebDAV.
Mithilfe des auf HTTP und XML-basierten Exchange ActiveSync-Protokolls können mobile Endgeräte (Windows Mobile/Phone und iOS-Geräte [35]) eben jene Daten mit dem Nachrichtenserver synchronisieren und Geräterichtlinien und -eigenschaften erhalten.
Durch die Unterstützung von Information Rights Management ist es möglich, auch geschützte E-Mails anzeigen zu lassen. Eine komplette Auflistung aller Funktionen von Exchange ActiveSync kann hier [34] nachgelesen werde. Genannt seien zur Richtlinienverwaltung verschiedene Passwort- und Verschlüsselungsvorgaben sowie die Möglichkeit zum remote wipe. Windows Phone 7 unterstützt momentan noch nicht problemlos alle Richtlinien.
Wie schon genannt, werden auch iOS-Geräte unterstützt. Unter anderem lässt sich die Funktion des Webbrowser und der Kamera einschränken, sowie die Geräteverschlüsselung fordern und das Gerät fernlöschen. Eine Liste der unterstützen Funktionen kann hier [35, Unterpunkt: Policies] nachgelesen werden.

4.3.6 Trend Micro Mobile Security 7.1

Mobile Security 7.1 [29] ist eine plattformunabhängige Software, welche dem Administrator eine Vielzahl von Möglichkeiten zur Verwaltung bietet. Trend Micro will dabei nach eigenen Angaben mit seiner Softwarelösung auf den Faktor Consumerization eingehen [31]. Unterstützt wird Android 2.1+, iOS 3+, Windows Mobile 5/6 und Symbian 9.1. iOS-Geräte müssen dabei den Apple Push Notification Service aktiviert haben und für BlackBerry-Geräte muss ein BES-Server installiert sein. Ein Entsprechendes Serverbetriebssystem von Microsoft zuzüglich einer unterstützen Version von Trend Micro Office Scan Corporate Edition wird auf der Administratorseite vorausgesetzt.

Das Programm bietet unter anderem folgende Funktionen [30]:

- zentrale Management-Konsole
  - ein verlässliches, skalierbares und zentrales Management-Framework
  - Sicherheits- und Konfigurationsmanagement für mobile Geräte
  - geräteübergreifende Anwendungsrichtlinien
  - erlaubt die Sperrung der Kamera, von Bluetooth und SD Kartenlesern
- Mobile Device Management
  - Verschlüsselungs- und Passwortkonfiguration für iOS-Geräte
  - remote lock, remote wipe und Passwortzwang
  - Geräteinventarisierung und Fernortung
  - Netzwerk- und Anwendungszugangskontrolle für iOS- und Android Geräte inklusive VPN und Wi-Fi-Konfiguration, E-Mail und Home Settings

Durch diese Funktionen und der zusätzlichen Sicherheitsfunktionen (Malware Schutz, Firewall sowie Anruf- und Nachrichtenfilter) sollen die operationalen Kosten gesenkt, der Datenverlust limitiert, das Sicherheitsrisiko reduziert und die Übersicht in kritischen Situationen verbessert werden. Dabei kann die Software noch mit anderen Programmen von Trend Micro kombiniert werden, um zum Beispiel eine Endpunkt-Verschlüsselung zu gewährleisten (Endpoint Encryption) oder dem Datenverlust durch Backupsoftware entgegen zu wirken (Data Loss Prevention).

4.3.7 SAP/Sybase: Afaria

Afaria ist eine MDM- und Sicherheitssoftware von Sybase [36] für mobile Endgeräte. Unterstützt werden dabei nach eigenen Angaben alle mobilen Betriebssysteme (Windows Phone ist nicht mit aufgeführt) und alle kabellosen Übertragungswege. Neben der Funktion zur Erstellung eines eigenen Intranets über iAnywhere Relay Server können auch Profile an Benutzergruppen und einzelne Nutzer mit jeweils eigenen Richtlinien gebunden werden. Diese werden automatisch überprüft und geupdatet, wenn eine aktuellere Version auf dem MDM-Server erstellt wurde. Dadurch haben die jeweiligen Benutzer mit ihren Endgeräten Zugriff auf das eigene Intranet und die darin vorhandenen Datenbanken sowie durch Nutzung der veröffentlichten API auch die Möglichkeit Dritthersteller- und eigene Software auf dem Gerät aktuell zu halten.
Die wichtigsten Afaria-Komponenten im Überblick [37]:

- Backup Manager
  automatisches Archivieren von Daten und Dokumenten von mobilen Geräten
- Configuration Manager
  Gerätekonfigurationsüberwachung inklusive der Verbindungseinstellungen

- Document Manager
  automatische Dokumentenlieferung und -updates auf das Endgerät

- Remote Control
  Möglichkeit zur Fernsteuerung von mobilen Geräten

- Software Manager
  zentralisierte Softwaredistribution, -installation und -support für Endgeräte

Durch diese Cross-Plattfrom Unterstützung können somit zentral eine Vielzahl von Software und Hardwareeinstellungen gemanaged werden, wie Geräteidentifikation, Netzwerkeinstellungen, Verbindungsprofile, regionale Einstellungen und Alarme [38].

4.3.8 Odyssey Software: Athena

Athena als MDM-Software von Odyssey ist kein eigenständiges Programm, sondern sieht sich als eine Vervollständigung beziehungsweise Erweiterung bestehender Produkte um weitere Funktionsumfänge [39]. Neben einer Vielzahl anderer Partnerlösungen [42] werden nachfolgend kurz zwei Partnerlösungen und die resultierenden erweiterten Umfänge vorgestellt.

Microsoft Configuration Manager

In Verbindung mit Microsoft Configuration Manager 2007, welcher unter anderem die Möglichkeit zur Softwaredistribution, -verwaltung und -aktualisierung auf einem mobilen Endgeräten bietet, wird der Geräteumfang mit Athena 5.0 auf iOS, Android, BalckBerry, Windows Phone 7 und weiteren mobilen Betriebssystemen innerhalb einer Managementkonsole erweitert [40]. Dadurch stehen auch folgende Funktionen zur Verfügung:

- Echtzeitfernsteuerung

- Bereitstellung und Installation von Gerätesoftware, Anwendungen und Updates

- umfassende Informationen zum Software- und Hardwarebestand

- Fernortung via GPS, remote lock und wipe

- detaillierte Geräte- und Nachrichteninformationen

Microsoft Configuration Manager 2012 soll mit nativer Erstellung und Verbreitung von Exchange ActiveSync-Richtlinien ebenso durch Premiumeigenschaften von Athena im Bereich over-the-air Management profitieren.

BlackBerry Enterprise Server

Als Addon für BlackBerry Enterprise Server (Express) 5.0 bietet Athena eine umfassende Funktionserweiterung für BlackBerry Smartphones der Version 4.3+ [41]. Die Erweiterung bietet unter anderem das Athena App Inventory, interaktiven Support, Athena System Informationen, remote control, Module Explorer, Netzwerkstatus und die Athena Tunnel Funktion. Die Athena Smartphone App kann nur durch den Athena Tunnel Service Instruktionen erhalten.

4.4 Bewertung der Lösungen

Nachdem nun alle Softwarelösungen präsentiert sind, bleibt die Bewertung. Diese kann dank des Anforderungsmodells übersichtlich erfolgen. Die Vergleichbarkeit ist mit Vorsicht zu genießen, da nicht jede Software die gleiche Zielgruppe bedient und entsprechend andere Funktionen aufweisen kann. Zur Bewertung ist hier noch einmal das Anforderungsmodell dargestellt:

- Fähigkeit zu remote alarm, detection, lock und wipe
- Erstellung und Verwaltung von (Konfigurations-)Profilen für
  - E-Mail-Konten, Kalender, Adressbücher
  - Wi-Fi-Verbindungen, Zertifikate, VPNs
  - Apps, Gerätefunktionen
- Erstellung von Nutzungsrestriktionen für Programme und Funktionen
- Fähigkeit zu over-the-air-Updates und -Backups
- Verwaltung unterschiedlicher mobiler Betriebssysteme

Wenn eine Anforderung nicht erfüllt wird, ist das Kästchen am entsprechenden Feld rot. Wird die Anforderung erfüllt, ist es grün. Ist der Grad der Erfüllung nur ungenügend beziehungsweise nicht vollständig oder benötigt Drittherstellersoftware, ist das Feld gelb. In Abbildung 4.1 wird der Vergleich des Anforderungsmodells mit den Funktionen und Fähigkeiten der Produkte für den privaten Sektor und in Abbildung 4.2 und 4.3 für die Enterprise-Software dargestellt. Es ist dabei zu beachten, dass ich keiner Software ”grün” für eine vollständige Unterstützung der vier mobilen Hauptbetriebssysteme gegeben habe, da kein Hersteller den vollen Funktionsumfang laut entsprechenden Angaben ohne Einschränkungen gewährt.

Produkt	Mobile Me und iCloud	Samsung Dive	My Phone	Mobile Security PE
Fähigkeit zu remote alarm, detection, lock und wipe				
Erstellung und Verwaltung von (Konfigurations-) Profilen für				
E-Mail-Konten, Kalender, Adressbücher				
Wi-Fi-Verbindungen, Zertifikaten, VPNs				
Apps, Gerätefunktionen				
Erstellung von Nutzungsrestriktionen für Programme und Funktionen				
Fähigkeit zu OTA-Updates und -Backups				
Verwaltung unterschiedlicher mobiler Betriebssysteme				

Abbildung 4.1: Anforderungsmodellauswertung für Mobile Me und iCloud, Samsung Dive, My Phone und Mobile Security PE.

Produkt	Lion Server	Google Apps	BES	Management Center
Fähigkeit zu remote alarm, detection, lock und wipe				
Erstellung und Verwaltung von (Konfigurations-) Profilen für				
E-Mail-Konten, Kalender, Adressbücher				
Wi-Fi-Verbindungen, Zertifikaten, VPNs				
Apps, Gerätefunktionen				
Erstellung von Nutzungsrestriktionen für Programme und Funktionen				
Fähigkeit zu OTA-Updates und -Backups				
Verwaltung unterschiedlicher mobiler Betriebssysteme				

Abbildung 4.2: Anforderungsmodellauswertung für Lion Server, Google Apps, BES und Management Center.

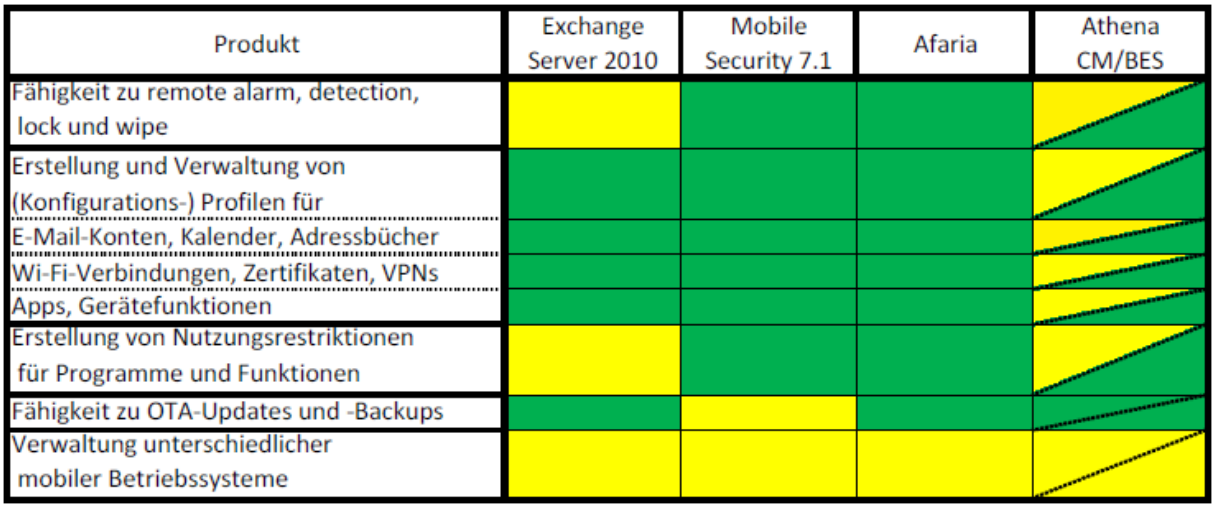

Abbildung 4.3: Anforderungsmodellauswertung für Exchange Server 2010, Mobile Security 7.1, Afaria und Athena CM/BES

4.4.1 Ausblick

Zusammenfassend möchte ich erwähnen, dass diese Informationen zum größten Teil aus den Dokumentationen und Präsentationen der Hersteller gewonnen wurden. Entsprechend schwer ist es festzustellen, in wie weit sich die genannten Funktionen und Fähigkeiten auch unter realen Bedingungen an ihre Vorgaben halten und was negative Punkte sind. Denn diese werden natürlich nicht offen von den Herstellern genannt, sondern müssen erst in der Praxis gefunden werden. Dass die Smartphones auch im Geschäftsbereich im kommen sind, sieht man unter anderem an den Veränderungen der Internetpräsentationen der Hersteller, welche neue Funktionen per Updates nachreichen und entsprechende Lösungen präsentieren, da sich manche Quellen im Verlauf der Ausarbeitung durchaus geändert haben, beziehungsweise auf einen aktuelleren Stand gebracht wurden. Da die benötigten Funktionen fest sind, besteht ein wirkliches Alleinstellungsmerkmal im Funktionalitätsbereich nur kurz, da anderen Hersteller schnell nachziehen.
Gerade für den Enterprise-Markt ist es aber sehr wichtig, in wie weit sich die Software wirklich den gegebenen Umständen anpassen lässt, wie benutzerfreundlich die Menüs und Funktionen sind (siehe Usability beziehungsweise Software-Ergonomie [44]) und in wie weit die ganzen Programme ausfallsicher sind. Wenn zum Beispiel ein Smartphone keinen Empfang mehr hat, da dieser gestört ist oder absichtlich unterbunden wird [43], können die meisten Funktionen nicht mehr genutzt werden. Ein Ansatz für dieses Problem wäre eine Geräteverschlüsselung, welche nach Ablauf einer gewissen Zeit ohne Netzverbindung automatisch die darauf befindlichen Daten unbrauchbar macht. Und gerade der Sicherheitsaspekt ist auch von Bedeutung, wenn es um Störanfälligkeit, Robustheit und sichere Datenübertragung geht. Auch die Umsetzung von Consumerization in der Software sollte unter realen Bedingungen untersucht werden, da wohl kaum ein Mitarbeiter den Wunsch hegt, dass sein IT-Administrator Zugriff auf seine persönlichen Daten hat.
Es bleibt zu sagen, dass Fernverwaltung und- administration von Smartphones heutzutage möglich ist und in seiner Funktionalität immer weiter ausgebaut wird, da dieser Sektor immer mehr an Bedeutung, im privaten wie im unternehmerischen Umfeld, gewinnt.

Literaturverzeichnis

[1] Oelmaier, F., Hörtreiter, J. und Seitz, A.. *Einbindung des iPad in die IT-Infrastruktur*, aus Apple's iPad im Enterprise-Einsatz, Springer Berlin Heidelberg, 2011, s.47-54

[2] IT Wissen. *Fernadministrierung*, Definition, `http://www.itwissen.info/definition/lexikon/remote-administration-Fernadministrierung.html`, zuletzt abgerufen am 08.12.2011

[3] Symantec. *ITProjekte: Fernverwaltung*, Definition, `http://www.symantec.com/de/de/business/solutions/projects/projectdetail.jsp?solid=sol_infrastruct_op&solfid=sol_endpoint_management&projectid=remote_management`, zuletzt abgerufen am 08.12.2011

[4] Byaer, Martin. *ByoD- Albtraum oder Segen für CIOs?*, Computerwoche, 23.08.2011, `http://www.computerwoche.de/management/it-strategie/2370513/`, zuletzt abgerufen am 08.12.2011

[5] Blackberry. *Enterprise Server*, deutscher Webauftritt, `http://de.blackberry.com/services/business/server/full/`, zuletzt abgerufen am 08.12.2011

[6] Blackberry. *Enterprise Server*, US Webauftritt, `http://us.blackberry.com/business/software/bes/`, zuletzt abgerufen am 08.12.2011

[7] Blackberry. *Enterprise Server Demo 5.0*, Video-Bröschure, `http://www.blackberry.com/select/server/autodemo/1551_RIM_BES_5.html`, zuletzt abgerufen am 08.12.2011

[8] Blackberry Ger. *Management Center*, deutscher Webauftritt, `http://de.blackberry.com/services/business/managementcenter.jsp`, zuletzt abgerufen am 08.12.2011

[9] Blackberry. *Management Center*, Broschüre, RIM, 2011

[10] Blackberry. *Management Center*, Präsentation, RIM, 2011

[11] Apple. *Mobile Me*, Webauftritt, `https://www.me.com`, zuletzt abgerufen am 08.12.2011

[12] Apple. *Mobile Me*, Apple Mobile Me Webauftritt, `https://www.apple.com/mobileme/`, zuletzt abgerufen am 08.12.2011

[13] Apple. *iCloud*, Übergangsbestimmungen, `https://www.apple.com/de/mobileme/transition.html`, zuletzt abgerufen am 08.12.2011

[14] Apple. *iCloud*, Webauftritt, `https://www.apple.com/de/icloud/`, zuletzt abgerufen am 08.12.2011

[15] Apple. *iOS: So erstellen Sie ein Backup*, Leitfaden, `https://support.apple.com/kb/ht1766?viewlocale=de_DE`, zuletzt abgerufen am 08.12.2011

[16] Apple. *Profil-Manager-Hilfe*, elektronisches Handbuch, `http://help.apple.com/profilemanager/mac/10.7/`, zuletzt abgerufen am 08.12.2011

[17] Apple. *Mac OS X Lion Server*, Webauftritt, `https://www.apple.com/de/macosx/server/`, zuletzt abgerufen am 08.12.2011

[18] Apple. *iPhone-Konfigurationsprogramm*, elektronisches Handbuch, `http://help.apple.com/iosdeployment-ipcu/`, zuletzt abgerufen am 08.12.2011

[19] Clarke, G.. *Apple's iCloud runs on Microsoft and Amazon services*, The Register, 02.09.2011, `http://www.theregister.co.uk/2011/09/02/icloud_runs_on_microsoft_azure_and_amazon/`, zuletzt abgerufen am 08.12.2011

[20] Samsung. *Dive*, Webauftritt, `http://www.samsungdive.com`, zuletzt abgerufen am 08.12.2011

[21] Samsung. *Dive*, FAQ, `http://v2.samsungdive.com/help/faq.do`, zuletzt abgerufen am 08.12.2011

[22] Google. *Apps for Business*, Webauftritt, `http://www.google.com/apps/intl/de/business/index.html`, zuletzt abgerufen am 08.12.2011

[23] Google. *Apps for Mobile*, Webauftritt, `http://www.google.com/apps/intl/en/business/mobile.html`, zuletzt abgerufen am 08.12.2011

[24] Google. *Sicherheitseinstellungen für Mobilgeräte*, Apps-Hilfe, `http://support.google.com/a/bin/answer.py?answer=173393`, zuletzt abgerufen am 08.12.2011

[25] Google. *Device Policy for Android*, Übersicht allgemein, `http://support.google.com/mobile/bin/answer.py?hl=en&answer=190930`, zuletzt abgerufen am 08.12.2011

[26] Google. *Device Policy for Android*, Übersicht Nutzer, http://www.google.com/support/mobile/bin/answer.py?answer=190930, zuletzt abgerufen am 08.12.2011

[27] Google. *Device Policy for Android*, Übersicht Administrator, http://www.google.com/support/a/bin/answer.py?answer=1056433, zuletzt abgerufen am 08.12.2011

[28] Trend Micro. *Mobile Security*, Personal Edition, http://us.trendmicro.com/us/products/personal/mobile-security-for-android/index.html, zuletzt abgerufen am 08.12.2011

[29] Trend Micro. *Mobile Securtity 7.1*, Enterprise Produkt, http://us.trendmicro.com/us/products/enterprise/mobile-security/, zuletzt abgerufen am 08.12.2011

[30] Trend Micro. *Mobile Securtity for Enterprises 7.1*, Datenblatt, 2011

[31] Trend Micro. *Enterprise Security*, "'Bring 'em on!"' - The Consumerization of Enterprise Mobility, White Paper, July 2011

[32] Microsoft. *My Windows Phone*, Webauftritt, https://www.windowsphone.com/en-US/my, zuletzt abgerufen am 08.12.2011

[33] Microsoft. *Exchange 2010*, Webauftritt, https://www.microsoft.com/exchange/2010/de/de/default.aspx, zuletzt abgerufen am 08.12.2011

[34] Microsoft. *Understanding Exchange ActiveSync*, TechNet Library, http://technet.microsoft.com/en-us/library/aa998357.aspx, zuletzt abgerufen am 08.12.2011

[35] Apple. *Exchange ActiveSync und iOS-Geräte*, elektronisches Handbuch, http://help.apple.com/iosdeployment-exchange/?lang=de, zuletzt abgerufen am 08.12.2011

[36] SAP/Sybase. *Mobile Lösungen für ihr Unternehmen: Afaria*, Webauftritt, http://www.sap.com/germany/solutions/mobility/afaria/index.epx, zuletzt abgerufen am 08.12.2011

[37] Sybase. *Afaria - Device Management and Security From a Single Console*, Datenblatt, 2010

[38] Sybase. *Sybase Afaria - Comprehensive Management and Security for the Mobile Enterprise*, Broschüre, 2010

[39] Odyssey Software. *ATHENA: Enterprise & Embedded Mobile Device Management*, Webauftritt, http://www.odysseysoftware.com/MDMPRODUCTS/AthenaforMobileDeviceManagement.aspx, zuletzt abgerufen am 08.12.2011

[40] Odyssey Software. *Enterprise Mobile & Embedded Devices: Managed, Supported & Secured*, Webauftritt, http://www.odysseysoftware.com/MDMPRODUCTS/Athenaformobiledevicemanagement/AthenaConfigurationManager.aspx, zuletzt abgerufen am 08.12.2011

[41] Odyssey Software. *Enhance Smartphone Management via BlackBerry Enterprise Server*, Webauftritt, http://www.odysseysoftware.com/MDMPRODUCTS/Athenaformobiledevicemanagement/AthenaBlackBerryEnterpriseServer.aspx, zuletzt abgerufen am 08.12.2011

[42] Odyssey Software. *Athena + Partner Solutions*, Webauftritt, http://www.odysseysoftware.com/mdmPRODUCTS/Athenaformobiledevicemanagement/AthenaPartnerSolutions.aspx, zuletzt abgerufen am 08.12.2011

[43] Disklabs. *The Mobile Phone Shield Faraday Bag*, Webauftritt, http://www.faradaybag.com/the-mobile-phone-shield.html, zuletzt abgerufen am 08.12.2011.

[44] Michael Herczeg. *Software-Ergonomie: Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme.*, 3. Auflage, Oldenbourg Wissenschaftsverlag, 2009, ISBN 978-3-486-58725-8.

Kapitel 5

App-Stores

Martin Hille

In diesem Teil werden die verschiedenen App-Stores vorgestellt, dies geschieht aus ökonomischer Perspektive. In dieser Seminararbeit werden die Fakten zusammengestellt, wie durch ein Geschäftsmodell Geld verdient wird. Für Entwickler gibt es unterschiedlichste Gründe Applikationen zu entwickeln, ob es als Hobbie oder Beruf verstanden wird, entscheidet sich letztlich über die Ansicht zu monetären Größen, speziell hier wird dies über den Gewinn für die Einzelperson wahr genommen. Dazu sind Einstiegsinvestitionen zu betrachten und daraus ergeben sich die Entscheidungen, bei welchem Hersteller der Entwickler seine Ideen einer Applikation umsetzen möchte. Die Marktperspektive ist das Interesse von Entwickler und Kunde, Entscheidungen zu einzelnen Herstellern und somit auch zu den Betriebssystemen der Smartphones werden über die Größe und das Wachstum des Marktes getroffen. Will ein Entwickler die breite Masse erreichen, so entscheidet er sich möglicherweise für den Hersteller mit dem größten Markt. Will ein Entwickler kreativ sein und Erfolg in einem noch kleinen aber dennoch wachsendem Markt haben, so entscheidet er sich für den Faktor Risiko mit hohen Gewinnchancen, weil er dadurch in der Masse der Produkte nicht untergeht oder erdrückt wird. Will der Entwickler sich dem geschäftlichen Bereich zuwenden und professionelle Business-Applikationen entwerfen, entscheidet dieser sich womöglich für die Marktnische. Aus diesen vielzähligen Möglichkeiten setzt sich letztlich die Marktperspektive des Entwicklers zusammen. Dem Interesse für diese existenziellen Fragen soll in dieser Seminararbeit nachgekommen werden, sodass das Wissen um die verschiedenen ökonomische Möglichkeiten in den nächsten Abschnitten zu erfahren ist.

Inhaltsverzeichnis

5.1 Einleitung 141
5.2 Grundlagen 141
5.2.1 Geschäftsmodell 141
5.2.2 BCG-Matrix 142
5.3 Hauptteil 144
5.3.1 Android Market 144
5.3.2 Blackberry App World 148
5.3.3 Apple App Store 150
5.3.4 Windows Phone 7 Marketplace 152
5.3.5 Vergleich 155
5.4 Fazit 156

5.1 Einleitung

Die Smartphones sind aus unserer Gesellschaft nicht mehr wegzudenken, seit der Markteinführung des ersten iPhones gelten diese Produkte als ein Zeichen von Erfolg und Ubiquität des Internets. Die Vision von Steve Jobs war es das Internet allgegenwärtig zu machen, heute können wir sagen, dass es ihm gelungen ist. Smartphones sind kleine Computer mit einem Mobiltelefon gepaart, dies ermöglicht erst die Funktionalität der Allgegenwärtigkeit. Computer funktionieren mit einem Betriebssystem und Software, dabei kann Software von einer kleinen Applikation bis hin zur kompletten Unternehmenssoftware unterteilt werden. Smartphones sind somit physikalisch betrachtet kleine Computer, sodass Applikationen auf deren Betriebssystemen arbeiten können. Dadurch werden diese Applikationen als Erweiterungen der Betriebssysteme der Smartphones angesehen. Um diese Weiterentwicklung der Betriebssysteme durch Erweiterungen zu beschleunigen, entwickeln heute nicht nur die Softwarehersteller der Betriebssysteme die Applikationen, sondern jede Person oder jedes Unternehmen darf dabei helfen.

Eine neue Unternehmensbranche ist durch dieses Geschäftsmodell der Applikationen entstanden, Applikationen sind zu einem handelbaren Gut geworden. Für die Applikationen gibt es genügend Kunden, diese suchen praktische und werterzeugende Erweiterungen oder Unterhaltungssoftware, bspw. vor einem Jahrzehnt wurde Software in einem Geschäft gekauft, dieses Geschäft ist auch als eine Art Marktplatz anzusehen. Auf einem Marktplatz verkaufen Verkäufer ihre Produkte an Kunden, diese Kunden kaufen somit ein Gut und bezahlen die Verkäufer. Übertragen auf die Applikationen musste hier auch ein Marktplatz geschaffen werden. Diese Marktplätze bezeichnen sich heute als App-Stores.

Diese Seminararbeit soll an diesen App-Stores ansetzen und die Funktionsweise klären. Die Möglichkeiten welche sich für die Betreiber und Verkäufer der App-Stores ergeben.

5.2 Grundlagen

Als Grundlagen für die gesamte Seminararbeit ist zum einen das Geschäftsmodell nach Stähler in seinen drei Hauptkomponenten beschrieben und zum anderen wird die BCG-Matrix beschrieben. Diese beiden Modelle sind die Grundlage des Vergleiches der einzelnen App-Stores und somit die Grundlage dieser Seminararbeit.

5.2.1 Geschäftsmodell

Zur Untersuchung der App-Stores ist es wichtig zu verstehen, wie damit Gewinn erwirtschaftet wird. Kein Service wird ohne Bezahlung generiert, das heißt wenn ein Service bereitgestellt wird, soll heute daran verdient werden. Ein Hersteller stellt somit den App-Store bereit und möchte damit Geld verdienen, doch wie kann dies veranschaulicht beschrieben werden, sodass die Betriebswirtschaft befragt werden muss.

Ein Geschäftsmodell kommt aus dieser wissenschaftlichen Sparte, sodass dieses den Sachverhalt beschreiben soll, wie sich ein Hersteller die Erwirtschaftung des Gewinns vorstellt.

Ein Geschäftsmodell besteht grundsätzlich aus drei Teilen, dies sind das Nutzenversprechen, die Architektur der Wertschöpfung und das Ertragsmodell. Sodass dadurch ziemlich genau aufgeschlüsselt wird, wie der Gewinn erwirtschaftet wird.

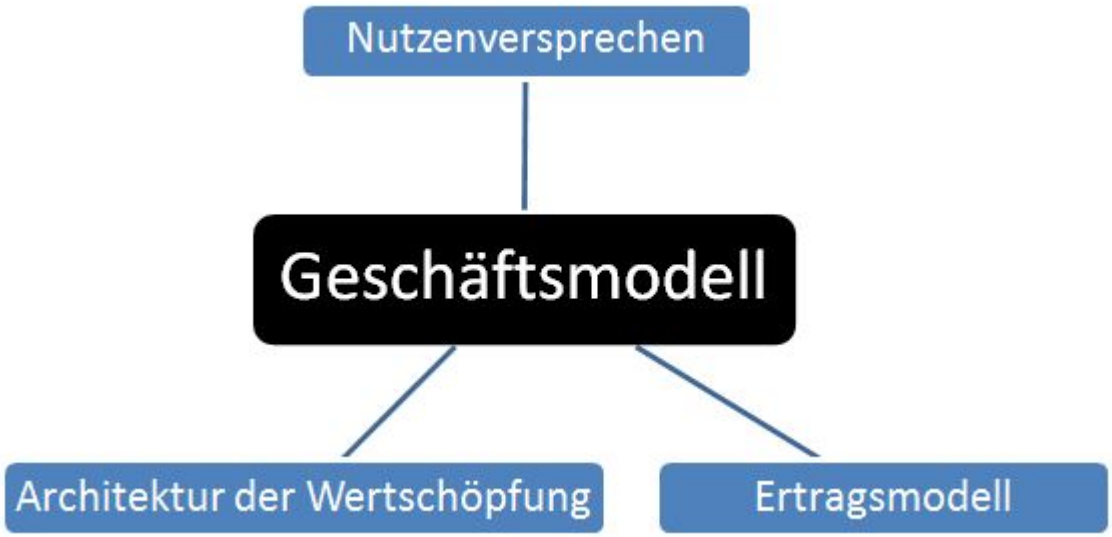

Abbildung 5.1: Das Geschäftsmodell nach Stähler
Quelle: Patrick Stähler (2001)

Das Nutzenversprechen soll den Nutzen beschreiben, dieser Nutzen ist sowohl für den Kunden als auch für einen Partner oder das Unternehmen selber zu differenzieren. Dazu soll genau beschrieben werden, welcher Nutzen für das Unternehmen gestiftet wird.

Die Architektur der Wertschöpfung ist gleichzusetzen mit dem Geschäftsmodell an sich, das heißt wie sich der Nutzen für den Kunden generiert. Dabei sind verschiedene Stufen zu berücksichtigen, sodass die Leistungserstellung im Vordergrund steht, darauf baut sich das Angebot in den jeweiligen Märkten auf. Bei diesen Märkten ist es ebenso erforderlich zu wissen, welche Produkt-Markt-Strategie verfolgt wird, dadurch kann ein Qualitätsprodukt mit hohem Preis nur auf einem Markt erfolgreich sein, wenn die Kunden dieses Marktes auch bereit sind, den Preis dafür zu bezahlen.

Das Ertragsmodell beschreibt das Geschäftsmodell, indem es sich mit den Einnahmen des Unternehmens beschäftigt. Dabei beantwortet es die Frage nach den Quellen, aus welchen die Einnahmen kommen, ebenso dem Vorgang an sich. Zudem soll beschrieben werden, wie nachhaltig Einnahmen generiert werden und somit die Zukunftsvorstellungen des Geschäftsmodelles. Insgesamt kann beim Ertragsmodell gesagt werden, dass das Wodurch Geld verdient wird, beschrieben wird.

5.2.2 BCG-Matrix

Die Matrix der Boston Consulting Group stellt in ihrem Sinne einen Zusammenhang zwischen Produkt und Kosten dar. Dabei dient sie hauptsächlich zur Veranschaulichung des aktuellen Standes, des Produktes im gegenwärtigen Zustand, sodass sich daraus eine

mögliche strategische Ausrichtung für das Unternehmen ergeben kann. Dies ist aus den vier Quadranten zu entnehmen, welche die Fragezeichen, die Stars, die "armen" Hunde und die "Cash" Kühe sind.

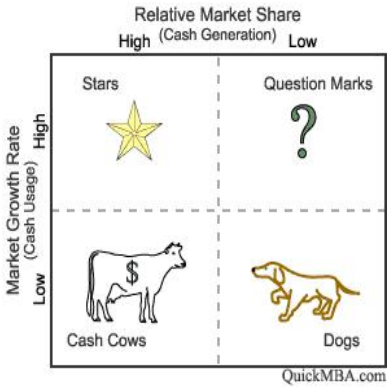

Abbildung 5.2: Die BCG-Matrix
Quelle: Boston Consulting Group (1980)

Die Boston Consulting Group gibt dazu Normstrategien heraus, wie mit den einzelnen Produkten in den jeweiligen Sektoren umzugehen ist. Dabei wird Wert auf die Möglichkeiten gelegt und auf die mögliche Laufzeit von den einzelnen Produkten.

Die Fragezeichen verdeutlichen bei dieser Matrix die "Newcomer", dabei haben diese einen relativ hohen Marktanteil oder Marktwachstum, sodass sich erst noch herausstellen wird, wie rentabel sie letztlich sind. Die Boston Consulting Group empfiehlt dabei eine Selektion der Produkte, sodass sich ein Fragezeichen zu einem Star entwickeln kann.

Die Stars sind Produkte, welche rentabel sind und einen hohen Marktanteil oder Marktwachstum besitzen. Für ein Unternehmen empfiehlt die Boston Consulting Group hier weiter zu investieren, sodass die Stars Stars bleiben und der Markt somit weiterhin rentabel genutzt wird.

Die "Cash" Kühe sind Produkte, welche "gemolken" werden wollen, dass soll heißen, dass die Rentabilität sehr wohl gegeben ist und die Strategie einer Abschöpfung angebracht ist. Aus dem hohen Marktanteil aber keinem Marktwachstum ergibt sich, dass diese Produkte im Markt in hoher Anzahl vertreten sind, jedoch der Markt stagniert. Dadurch soll nicht mehr investiert werden, sondern lediglich die maximale Rentabilität der Produkte abgeschöpft werden.

Die "armen" Hunde sind Produkte, welche im aktuellen Stand kein Marktwachstum mehr verspüren und auch in ihrer Rentabilität keinen Erfolg generieren. Daraus ergibt sich, dass diese Produkte schnellstmöglich durch andere ersetzt werden sollten, somit eine Desinvestitionsstrategie.

5.3 Hauptteil

In dem Hauptteil werden die einzelnen App-Stores auf ihr Geschäftsmodell untersucht, dabei werden auf die benötigten Zertifikate und Anforderungen für einen Entwickler jeweils eingegangen.

5.3.1 Android Market

Das Android Betriebssystem, von dem Hersteller Google, bietet seine Applikationen über den Android Market an. Der Android Market ist dadurch der Markplatz, sodass dieser trotz seines vielfältigen Angebotes von bspw. Klingeltönen oder Darstellungsschemata die Kernleistung der Vermarktung von Applikationen erbringt. Dadurch wird in dieser Seminararbeit, im weiteren Verlauf der Begriff App-Store synonym für den Android Market verwendet.

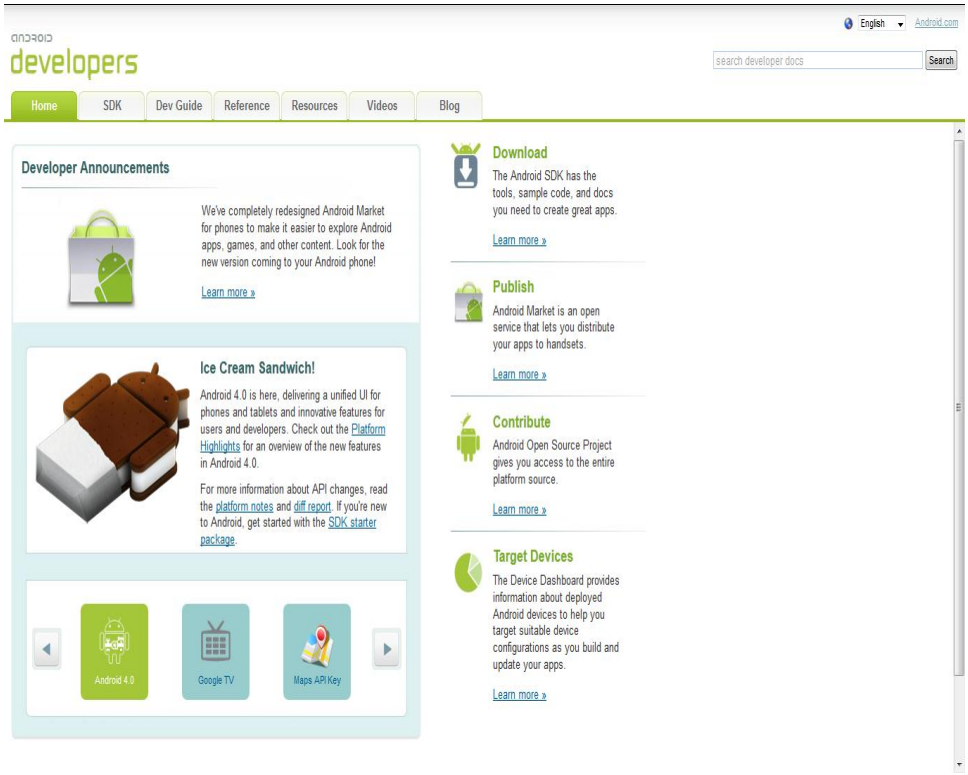

Abbildung 5.3: Die Developer Seite von Android
Quelle: Die Developer Seite von Android, `http://developer.android.com/index.html`

Hier in Abbildung 5.3 zu sehen, ist die für den Entwickler durch den Hersteller gebotene Seite, wobei die Offenheit im Vergleich zu den anderen App-Stores hervorzuheben ist. Daraus ergibt sich auch die Philosophie des Geschäftsmodelles.

Geschäftsmodell

Bei dem Geschäftsmodell des App-Stores gilt die Philosophie: "Android Market ist eine öffentlich zugängliche Website, über die Android-Entwickler Produkte für Geräte vertreiben können."

Dadurch kann die Frage nach dem Nutzenversprechen, also welchen Nutzen dieser App-Store stiftet, beantwortet werden. Indem die Entwickler ihre Produkte einfach vertreiben können und dabei der Einstieg einfach ermöglicht wird, verdient der Hersteller proportional mit. Je mehr Applikationen der Entwickler über diesen App-Store verkauft, desto mehr Umsatz macht auch der App-Store. Da aus den gesamten Einnahmen ein Teil pro Applikation an den Entwickler fließt und ein Teil an den Hersteller, wobei der Hersteller kaum Mehraufwand hat und trotzdem verdient. Dem Entwickler wird der freie Zugang zugesichert und das ist zum einen sein Nutzen, zum anderen hat der Entwickler den Nutzen des Verdienstes über den App-Store. Die Kunden haben den Nutzen, Zugriff auf die Applikationen des App-Stores zu erhalten und sich ihr Android Betriebssystem nach ihren Wünschen mit der jeweiligen Applikation anzupassen. Schließlich erbringt der Hersteller nur die Bereitstellung des Marktplatzes, sodass dort verschiedene Produkte gehandelt werden können und hat dabei sinnbildlich "Steuern", welche der Nutzen des Unternehmens sind. Also die Bereitstellung der Infrastruktur bringt einem Hersteller, einem Entwickler und einem Kunden den Nutzen.

Die Architektur der Wertschöpfung stellt eine Beschreibung für die gebotene Leistung am Kunden, in ihren einzelnen Stufen differenziert, dar. Dabei gilt es zum einen Herauszufinden, wer die einzelnen Rollen einnimmt und zum anderen welches Produkt auf welchem Markt angeboten wird. Sodass der Kunde, welcher die Applikation aus dem App-Store erwirbt, die Rolle des Käufers einnimmt. Mit der Rolle des Käufers, der bei seinem Kauf des Produktes einen Preis dafür aufbringt, ergibt sich die Grundlage eines Geschäftsmodells. Die Grundlage ist nichts Geringeres als der Kunde, welcher den Markt einseitig als Nachfrage darstellt. Die zweite Seite des Marktes ist das Angebot, sodass zu klären ist, wer dieses Angebot erbringt, dabei ist auf der einen Seite der Hersteller zu nennen und auf der anderen Seite ist der Entwickler zu nennen. Das Angebot an sich wird auf dem Marktplatz angeboten, dieser wird von dem Hersteller angeboten, jedoch muss das Produkt, welches das Angebot darstellt erst produziert werden. Bei der Produktion kommt der Entwickler zu seiner Arbeit, er produziert das Produkt und trägt somit die Grundlage für das Angebot durch sein Produkt bei. Zudem stellt sich für die Angebotsseite die Frage, wie Kunden durch eine Art "Marketing" mehr einkaufen und somit der App-Store mehr Gewinn erwirtschaften kann, jedoch ist dies im nächsten Abschnitt des Ertragsmodells beschrieben.

Die Stufen der Wertschöpfung (Abbildung 5.4) sollen hier genauer beschrieben werden, das heißt einen Benennung dieser muss stattfinden. Dabei haben wir Stufe 1, welche die Erstellung des Produktes betrifft, sodass eine Applikation fertig gestellt werden muss. Sobald die Applikation fertig gestellt wurde, kann mit Stufe 2 fortgefahren werden. In Stufe 2 soll die Bereitstellung der Applikation abgeschlossen werden, dass soll durch zur Verfügung Stellung auf einem Marktplatz sein. Konkret ist Stufe 2 abgeschlossen wenn die Applikation zum Download bereit steht. Bei der Stufe 3 ist der Kauf des Kunden

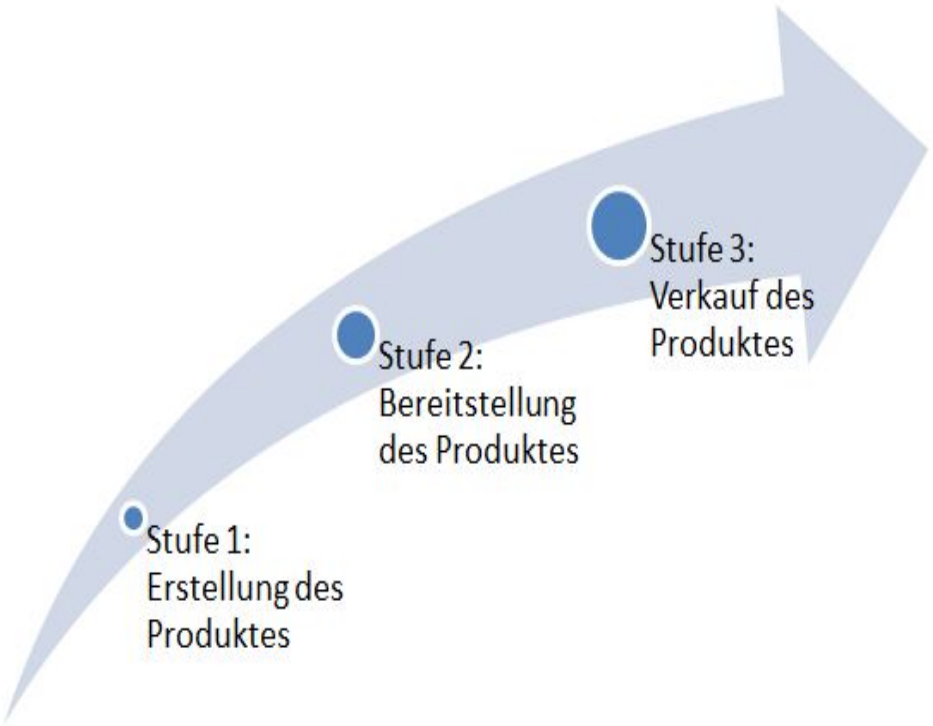

Abbildung 5.4: Die Stufen der Wertschöpfung

zu verzeichnen, sodass der Kunde den Betrag für die Applikation bezahlt hat und die Applikation nutzen kann.

Das Ertragsmodell beschreibt die Einnahmen, welche sich aus dem Geschäftsmodell erwirtschaften lassen. Geld will verdient werden und somit braucht Google Kunden, welche Produkte kaufen. Sodass der Kauf einer Applikation im Android Market für Google den Ertrag bringt. Dem Entwickler ergeht es ähnlich, sowohl bei ihm als auch bei Google macht die Anzahl der verkauften Applikationen den Faktor aus, wie erfolgreich das Produkt an sich ist. Google, im Vergleich zu dem Entwickler, macht über die Menge der Entwickler seinen Ertrag, dadurch spielt es keine Rolle mehr ob nun ein kostenpflichtiges Produkt Erfolg hat oder ein anderes. Google braucht die Entwickler, weil selbst nur der Marktplatz keinen Ertrag erbringt. Somit ist es für einen Hersteller von großer Bedeutung, dass dieser Entwickler wirbt und ein Entwicklerwachstum stattfindet. Je mehr Entwickler vorhanden sind, desto mehr Applikationen können produziert werden. Dem Entwickler selber wird um die Popularität seiner Applikation von Google aus zu steigern das Prinzip des indirekten Kaufes an die Hand gelegt. Dieses Prinzip ist mittels zweier Schritte zu verstehen, dabei wird eine erste kostenfreie Applikation zum Download gestellt und eine kostenpflichtige Applikation zusätzlich. Der Kunde soll dabei die Applikation für sich entdecken, sodass die kostenfreie Applikation, bspw. Werbung oder nicht, alle Funktionen enthält und durch den Kauf der kostenpflichtigen Applikation diese Nachteile für den Kunden nicht mehr vorhanden sind. Dadurch kann er die Applikation testen und bei Bedarf sich die Vollversion kaufen. Der direkte Kauf des Kunden ist für die Anbieter jeweils mit direktem Ertrag verbunden, sodass Applikationen für wenig Geld ein Modell an sich darstellen. Der

Gesamtertrag wird durch die Masse generiert, bspw. in einem 1-Euro-Laden, welcher viele Produkte für einen kleinen Preis anbietet, kaufen Kunden unter Umständen mehr ein, weil der Gedanke dahinter den Wert des einen Euro nicht mehr als Geld wahrnimmt. Auf einen App-Store projiziert, ergibt sich daraus die Möglichkeit, das gleiche Modell für diese Kunden anzuwenden.

Schließlich ist eine Applikation ein Produkt, welches preislich sehr unterschiedlich gestaltet werden kann. Dieses Produkt kann wie andere auch vermarktet werden, um bspw. den Preis zu rechtfertigen. Es können verschiedenste Produkt-Markt-Strategien angewandt werden, einzig zu beachten ist die Kaufkraft der Kunden und deren Anzahl. Diese Anzahl kann von Google beeinflusst werden, indem sie Smartphones, bspw. mit dem Betriebssystem Android, in den Markt drängen, sodass die Anzahl der Benutzer wächst und demzufolge auch die Anzahl der möglichen Kunden im Android Market.

Fakten und Voraussetzungen

In diesem Abschnitt gilt es für den Android Market die vorangegangenen Strukturen quantitativ aufzufüllen. Beide, also Hersteller und Entwickler verdienen Geld mit dem Verkauf einer Applikation, dabei verdient Google 30 Prozent und der Entwickler 70 Prozent an dem Verkaufspreis der Applikation. Der Entwickler jedoch muss sich zusätzlich, einmalig in diesem App-Store registrieren, was für ihn Kosten in Höhe von 25 US-Dollar hervorruft. Sozusagen eine Art Bearbeitungsgebühr, mit dem Hintergedanken, dass sich nur Entwickler registrieren und somit kein Unfug damit getrieben wird.

Die Frage, welche sich für einen Entwickler stellte, sind die Voraussetzungen die benötigt werden um überhaupt eine Applikation erstellen zu können. Zum einen verlangt Google, dass der Entwickler einen Email-Account bei ihnen hat und das Entwicklerprofil für 25 US-Dollar, zum anderen müssen dazu Kreditkarteninformationen angegeben werden. Die Kreditkarte ist bei Google das gängige Konto und wird somit für die Überweisung der Erlöse aus dem Verkauf der Applikationen verwendet. Zudem bietet Google ein Android Software Development Kit, mit welchem diese Applikationen erstellt werden. Alle hier nicht explizit mit Kosten benannten Voraussetzungen bietet Google dem Entwickler kostenfrei zu seiner Verfügung an.

Es sind Regeln bei der Veröffentlichung von Applikationen zu beachten, diese sind dabei direkt bei den Allgemeinen Geschäftsbedingungen angegeben. Dabei ist zu beachten, dass sich an die geltende Gesetzgebung gehalten werden soll, Datenschutzrechte sind einzuhalten, sodass eine Applikation vertraulich mit den Daten des Kunden umzugehen hat. Der Entwickler soll mit seiner Applikation keine Verbotenen Handlungen hervorrufen, hier ist ein Wettbewerbsverbot ebenfalls mit innbegriffen. Bspw. soll das GPS des Smartphones nicht den Standort des Kunden auswerten, um anhand dieser Daten den Preis für andere Produkte höher oder niedriger anzusetzen, als in ”reicheren” oder ”ärmeren” Städten. Der Entwickler hat sich außerdem an Best Practices der Programmierung zu halten, sodass schließlich die Nutzer eines Android Betriebssystems mit ergonomischen Applikationen arbeiten können. Eine weitere Einschränkung bei der Veröffentlichung ist die Größe der Applikation, sodass diese auf 50 Megabyte beschränkt ist und es eine APK-Datei sein muss.

5.3.2 Blackberry App World

Das Blackberry Betriebssystem, von dem Hersteller RIM, bietet seine Applikationen über die Blackberry App World an. Dabei ist die Kernleistung die Vermarktung von Applikationen zur Erweiterung des Smartphone Betriebssystems. Im weiteren Verlauf dieser Seminararbeit wird der Begriff App-Store synonym für die Blackberry App World verwendet.

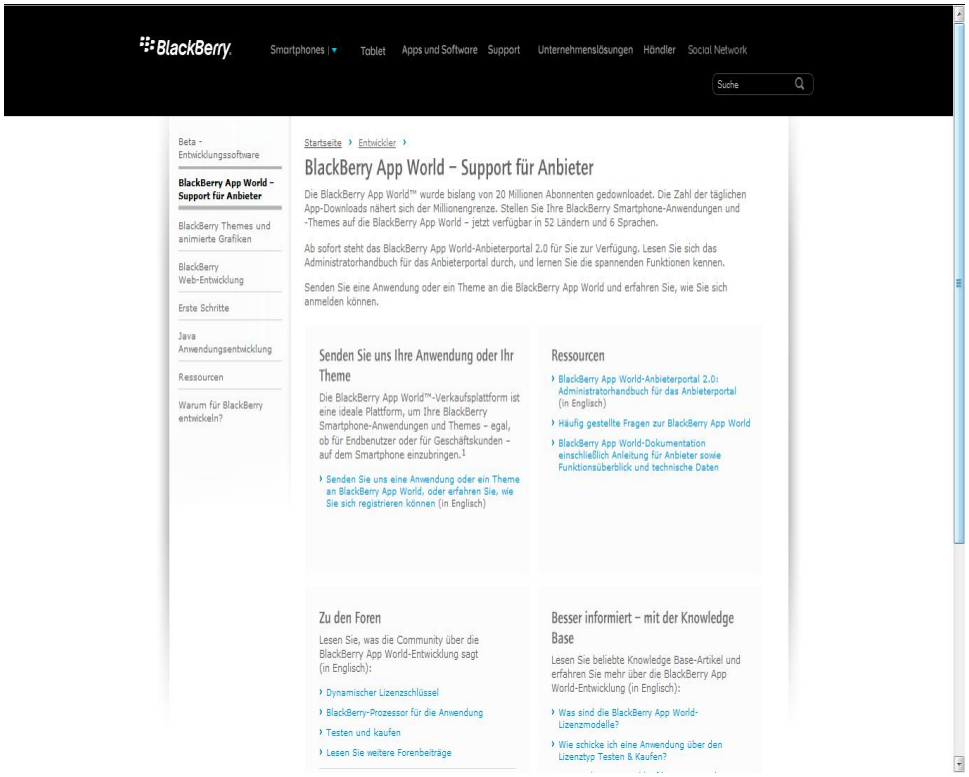

Abbildung 5.5: Die Developer Seite von Blackberry
Quelle: Die Developer Seite von Blackberry, `http://de.blackberry.com/developers/appworld/`

Hier in Abbildung 5.5 zu sehen, ist die für den Entwickler durch den Hersteller gebotene Seite, wobei Blackberry sein System sehr abschottet und im Vergleich zum Android Market Informationen zu finden, mit einem hohen Zeitaufwand verbunden sind. Im Hinblick auf das Geschäftsmodell ist es beinahe verständlich, weil Blackberry für den Einsatz bei Geschäftsleuten bekannt ist.

Geschäftsmodell

Bei dem Geschäftsmodell des App-Stores möchte Blackberry seine Unternehmensausrichtung auf die geschäftliche Nutzung seiner Smartphones erhalten, zugleich jedoch für Anwendungen unterschiedlicher Art zugänglich machen.

Dadurch kann die Frage nach dem Nutzenversprechen, also welchen Nutzen dieser App-Store stiftet, beantwortet werden. Indem die Entwickler ihre Produkte professionell entwickeln sollen und bereits vorab eine gute Kompetenz mit dem gesamten Applikationsmodell

haben sollen. Bei der Blackberry App World verdient RIM ebenso wie der Entwickler, ähnlich dem Prinzip des Android Market. Sodass hier auf die genau Beschreibung des Nutzenversprechens auf den Abschnitt im Geschäftsmodell des Android Market verwiesen wird.

Die Architektur der Wertschöpfung stellt eine Beschreibung für die gebotene Leistung am Kunden, in ihren einzelnen Stufen differenziert, dar. Diese Architektur ist bei der gesamten Vielzahl der Geschäftsmodelle der App-Stores gleich, sodass auch hier die drei Stufen der Wertschöpfung zu nennen sind und der Verweis auf den Abschnitt im Geschäftsmodell des Android Marktet verwiesen wird. Der Kunde schließt auch hier die Architektur der Wertschöpfung mit seinem Kauf ab.

Das Ertragsmodell beschreibt die Einnahmen, welche sich aus dem Geschäftsmodell erwirtschaften lassen. Analog zu dem Ertragsmodell von Google beschäftigt sich auch RIM mit allgemeinen Marktstrategien, um möglichst viele Applikationen verkaufen zu können. Speziell Blackberry gibt dazu seinen Entwickler die Auflagen, Applikationen immer mit 99-Cent-Beträgen darzustellen, dies hat in der Betriebswirtschafts bereits Erfolg ausweisen können, weil Kunden dazu neigen den Betrag geringer wahrzunehmen. Wichtiger ist jedoch, dass je mehr Entwickler vorhanden sind, desto mehr Applikationen können produziert werden. Dies ist bereits bei dem Ertragsmodell von Google hervorgegangen und stellt die Grundlage des Ertrag Modelles dar.

Der Gesamtertrag soll hier anders als bei Google durch qualitative Applikationen zu höherem Preise verwirklicht werden. Dadurch wird eine Qualitätsführerschaft von RIM angestrebt, welche durch Konkurrenzprodukte von Apple, Google und Windows durch Kostenführerschaft angegriffen wird.

Schließlich ist eine Applikation ein Produkt, welches preislich sehr unterschiedlich gestaltet werden kann. Dieses Produkt kann wie andere auch vermarktet werden, um bspw. den Preis zu rechtfertigen. Es können verschiedenste Produkt-Markt-Strategien angewandt werden, einzig zu beachten ist die Kaufkraft der Kunden und deren Anzahl. Diese Anzahl kann von RIM beeinflusst werden, indem sie ihre Smartphones durch Marketing an den Kunden bringen, sodass die Anzahl der Benutzer wächst und demzufolge auch die Anzahl der möglichen Kunden der Blackberry App World.

Fakten und Voraussetzungen

In diesem Abschnitt gilt es für die Blackberry App World, die vorangegangenen Strukturen quantitativ aufzufüllen. Beide, also Hersteller und Entwickler verdienen Geld mit dem Verkauf einer Applikation, dabei verdient RIM 30 Prozent und der Entwickler 70 Prozent an dem Verkaufspreis der Applikation. Der Entwickler jedoch muss sich zusätzlich, einmalig in diesem App-Store registrieren, was für ihn keine Kosten hervorruft.

Die Frage, welche sich für einen Entwickler stellte, sind die Voraussetzungen die benötigt werden, um überhaupt eine Applikation erstellen zu können. Zum einen verlangt RIM, dass ein Developer Account von jedem Entwickler angelegt wird. Zum anderen ist ein PayPal-Konto für die Abwicklung des Zahlungsverkehrs verantwortlich, welches der Entwickler

haben muss. Das PayPal-Konto ist bei RIM das gängige Konto und wird somit für die Überweisung der Erlöse aus dem Verkauf der Applikationen verwendet. Zudem bietet RIM ein Blackberry Software Development Kit, mit welchem diese Applikationen erstellt werden sollen. Alle hier nicht explizit mit Kosten benannten Voraussetzungen bietet RIM dem Entwickler kostenfrei zu seiner Verfügung an.

Es sind Regeln bei der Veröffentlichung von Applikationen zu beachten, diese sind analog wie bei Google beschrieben, deshalb wird hier auf den dazugehörigen Abschnitt des Android Market verwiesen. Im Gegensatz zu Google gibt es für die Applikations-Datei die Beschränkung auf das Format einer COD-Datei.

5.3.3 Apple App Store

Das Betriebssystem iOS, von dem Hersteller Apple, bietet seine Applikationen über den Apple App Store an. Dabei ist die Kernleistung, die Vermarktung von Applikationen zur Erweiterung des Smartphone Betriebssystems. Im weiteren Verlauf dieser Seminararbeit wird der Begriff App-Store synonym für den Apple App Store verwendet.

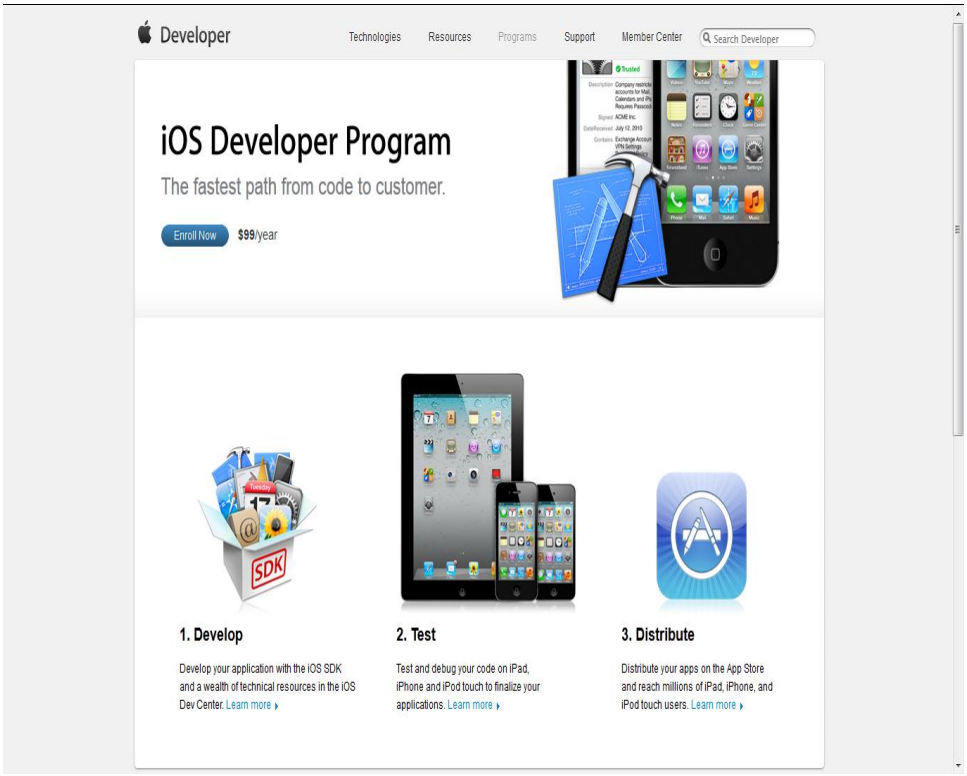

Abbildung 5.6: Die Developer Seite von Apple
Quelle: Die Developer Seite von Apple, `http://developer.apple.com/programs/ios/`

Hier in Abbildung 5.6 zu sehen, ist die für den Entwickler durch den Hersteller gebotene Seite, wobei Apple sein System als den schnellsten Weg zum Kunden anpreist. Im Hinblick auf das Geschäftsmodell will Apple durch gezieltes Marketing seine Marktführerschaft erhalten und zeigen, dass Entwickler sehr gut unterstützt werden sollen.

Geschäftsmodell

Bei dem Geschäftsmodell des App-Stores möchte Apple seinen Kunden ein besonderes Angebot unterbreiten, sodass nur geprüfte, Viren freie Applikation mit beinahe Garantie auf Funktionsfähigkeit angeboten werden. Dem Entwickler soll auf der anderen Seite zugesichert werden, dass nur dieser App-Store Applikationen für iPhones mit dem originalen Betriebssystem iOS zulässt.

Dadurch kann die Frage nach dem Nutzenversprechen, also welchen Nutzen dieser App-Store stiftet, beantwortet werden. Indem die Entwickler ihre Produkte professionell entwickeln sollen und bereits vorab eine gute Kompetenz mit dem gesamten Applikationsmodell haben sollen. Bei dem Apple App Store verdient Apple ebenso wie der Entwickler, ähnlich dem Prinzip des Android Market. Sodass hier auf die genau Beschreibung des Nutzenversprechens auf den Abschnitt im Geschäftsmodell des Android Market verwiesen wird. Dies mag dem Leser bizarr vorkommen, weil der App-Store von Apple in historischer Reihenfolge als erster "online gegangen" ist, aber für diese Seminararbeit ist es ohne Wertung zu verstehen, sodass eine beliebige Reihenfolge zulässig ist.

Die Architektur der Wertschöpfung stellt eine Beschreibung für die gebotene Leistung am Kunden, in ihren einzelnen Stufen differenziert, dar. Diese Architektur ist bei der gesamten Vielzahl der Geschäftsmodelle der App-Stores gleich, sodass auch hier die drei Stufen der Wertschöpfung zu nennen sind und der Verweis auf den Abschnitt im Geschäftsmodell des Android Markte verwiesen wird. Der Kunde schließt hier die Architektur der Wertschöpfung mit seinem Kauf ab.

Das Ertragsmodell beschreibt die Einnahmen, welche sich aus dem Geschäftsmodell erwirtschaften lassen. Analog zu dem Ertragsmodell von Google beschäftigt sich auch Apple mit allgemeinen Marktstrategien um möglichst viele Applikationen verkaufen zu können. Apple vermittelt seinen Entwickler die Einstellung den Preis eher niedrig zu halten, sodass die Kunden eher auf das Angebot eingehen und dadurch eine Mehrzahl von Kunden die Applikation kauft. Wichtiger ist jedoch, dass je mehr Entwickler vorhanden sind, desto mehr Applikationen können produziert werden. Dies ist bereits bei dem Ertragsmodell von Google hervorgegangen und stellt die Grundlage des Ertragsmodelles dar.

Der Gesamtertrag soll hier, anders als bei RIM, durch Applikationen zu niedrigem Preise verwirklicht werden. Dadurch wird eine Kostenführerschaft von Apple mit getesteter Qualität angestrebt, welche durch Konkurrenzprodukte zwar angegriffen werden kann, jedoch nicht in der vorhandenen Varietät und Vielzahl wie im Apple App Store zur Verfügung steht.

Schließlich ist eine Applikation ein Produkt, welches preislich sehr unterschiedlich gestaltet werden kann. Dieses Produkt kann wie andere auch vermarktet werden, um bspw. den Preis zu rechtfertigen. Es können verschiedenste Produkt-Markt-Strategien angewandt werden, einzig zu beachten ist die Kaufkraft der Kunden und deren Anzahl. Diese Anzahl kann von Apple beeinflusst werden, indem sie ihre Smartphones durch Marketing an den Kunden bringen, sodass die Anzahl der Benutzer wächst und demzufolge auch die Anzahl der möglichen Kunden des Apple App Store. Apple hat sich dazu bereits in der Vergangenheit von der Anbieterbeschränkung auf die Telekom getrennt, sodass jeder Anbieter iPhones verkaufen kann und der Kunde dieser Einschränkung nicht mehr unterliegt.

Fakten und Voraussetzungen

In diesem Abschnitt gilt es für den Apple App Store die vorangegangenen Strukturen quantitativ aufzufüllen. Beide, also Hersteller und Entwickler verdienen Geld mit dem Verkauf einer Applikation, dabei verdient Apple 30 Prozent und der Entwickler 70 Prozent an dem Verkaufspreis der Applikation. Der Entwickler muss sich zusätzlich, einmalig in diesem App-Store registrieren, was für ihn Kosten in Höhe von 99 US-Dollar per Anno hervorruft.

Die Frage, welche sich für einen Entwickler stellte, sind die Voraussetzungen die benötigt werden um überhaupt eine Applikation erstellen zu können. Zum einen verlangt Apple, dass eine Apple-ID von jedem Entwickler erstellt wird. Zum anderen ist eine Kreditkarte für die Abwicklung des Zahlungsverkehrs verantwortlich, welches der Entwickler haben muss. Die Kreditkarte ist bei Apple das gängige Konto und wird somit für die Überweisung der Erlöse aus dem Verkauf der Applikationen verwendet. Zudem bietet Apple ein iOS Software Development Kit, mit welchem diese Applikationen erstellt werden können, alternativ dafür kann ebenso Xcode verwendet werden. Zudem ist für die Programmierung von iOS-Applikationen die Verwendung eines Macs vorgesehen, sodass dies zusätzliche Kosten hervorruft. Alle hier nicht explizit mit Kosten benannten Voraussetzungen bietet Apple dem Entwickler kostenfrei zu seiner Verfügung an.

Es sind Regeln bei der Veröffentlichung von Applikationen zu beachten, diese sind analog wie bei Google beschrieben, deshalb wird hier auf den zugehörigen Abschnitt des Android Market verwiesen. Im Gegensatz zu Google gibt es für die Applikations-Datei die Beschränkung auf das Format einer IPA-Datei.

5.3.4 Windows Phone 7 Marketplace

Das Betriebssystem Windows Phone 7, von dem Hersteller Microsoft, bietet seine Applikationen über den Windows Phone 7 Marketplace an. Dabei ist die Kernleistung die Vermarktung von Applikationen zur Erweiterung des Smartphone Betriebssystems. Im weiteren Verlauf dieser Seminararbeit wird der Begriff App-Store synonym für die Windows Phone 7 Marketplace verwendet.

Hier in Abbildung 5.7 zu sehen, ist die für den Entwickler durch den Hersteller gebotene Seite, wobei Microsoft sein System als ganz einfach anpreist. Im Hinblick auf das Geschäftsmodell will Microsoft dadurch gezielt Entwickler anwerben, um mit diesem jungen Betriebssystem am Markt erfolgreich werden zu können.

Geschäftsmodell

Bei dem Geschäftsmodell des App-Stores möchte Microsoft seinen Entwicklern eine Möglichkeit von hohem Wachstum bieten. Der App-Store ist klein, also kann ein Entwickler hier besonders kreativ zur Arbeit gehen. Dabei ist generell die Programmierung von Applikation durch Vorbilder von vorhandenen Applikationen anderer App-Stores erwünscht.

Abbildung 5.7: Die Developer Seite von Windows
Quelle: Die Developer Seite von Windows,
`http://msdn.microsoft.com/de-de/windowsphone/ff191182`

Dies würde nicht nur den App-Store zu vergrößern helfen, sondern auch dem Ertrag aus den Applikationen.

Dadurch kann die Frage nach dem Nutzenversprechen, also welchen Nutzen dieser App-Store stiftet, beantwortet werden. Indem die Entwickler ihre Produkte professionell entwickeln sollen und bereits vorhandene Produkte in den App-Store integrieren. Bei dem Windows Phone 7 Marketplace verdient Windows ebenso wie der Entwickler, ähnlich dem Prinzip des Android Market. Sodass hier auf die genau Beschreibung des Nutzenversprechens auf den Abschnitt im Geschäftsmodell des Android Market verwiesen wird.

Die Architektur der Wertschöpfung stellt eine Beschreibung für die gebotene Leistung am Kunden in ihren einzelnen Stufen differenziert dar. Diese Architektur ist bei der gesamten Vielzahl der Geschäftsmodelle der App-Stores gleich, sodass auch hier die drei Stufen der Wertschöpfung zu nennen sind und der Verweis auf den Abschnitt im Geschäftsmodell des Android Markte verwiesen wird. Der Kunde schließt auch hier die Architektur der Wertschöpfung mit seinem Kauf ab.

Das Ertragsmodell beschreibt die Einnahmen, welche sich aus dem Geschäftsmodell erwirtschaften lassen. Analog zu dem Ertragsmodell von Google beschäftigt sich auch Windows mit allgemeinen Marktstrategien, um möglichst viele Applikationen verkaufen zu können. Windows gibt seinen Entwicklern hier nur die Vorbilder an die Hand, sodass bereits funktionsgetestete Modelle sinnvoll umzusetzen sind. Wichtiger ist jedoch, dass je mehr Entwickler vorhanden sind, desto mehr Applikationen können produziert wer-

den. Dies ist bereits bei dem Ertragsmodell von Google hervorgegangen und stellt die Grundlage des Ertragsmodelles dar.

Der Gesamtertrag soll hier, anders als bei RIM, durch Applikationen zu niedrigem Preise verwirklicht werden. Dadurch wird eine Marktdurchdringung erst ermöglicht, weil Microsoft mit seinem App-Store erst ganz am Anfang seiner Möglichkeiten steht.

Schließlich ist eine Applikation ein Produkt, welches preislich sehr unterschiedlich gestaltet werden kann. Dieses Produkt kann wie andere auch vermarktet werden, um bspw. den Preis zu rechtfertigen. Es können verschiedenste Produkt-Markt-Strategien angewandt werden, einzig zu beachten ist die Kaufkraft der Kunden und deren Anzahl. Diese Anzahl kann von Windows beeinflusst werden, indem sie ihr Betriebssystem durch Zusammenschließung mit einem Smartphone-Hersteller an den Kunden bringen, sodass die Anzahl der Benutzer wächst und demzufolge auch die Anzahl der möglichen Kunden des Windows Phone 7 Marketplace. Microsoft hat sich dazu bereits in Vergangenheit mit Nokia zusammengeschlossen, sodass dies einen Einstieg für Microsoft in den Smartphone-Betriebssystem-Markt erst ermöglicht.

Fakten und Voraussetzungen

In diesem Abschnitt gilt es für den Windows Phone 7 Marketplace die vorangegangenen Strukturen quantitativ aufzufüllen. Beide, also Hersteller und Entwickler verdienen Geld mit dem Verkauf einer Applikation, dabei verdient Microsoft 30 Prozent und der Entwickler 70 Prozent an dem Verkaufspreis der Applikation. Der Entwickler muss zusätzlich für die Einstellung in den App-Store bezahlen, dass bedeutet bspw. um 100 Applikationen in den App-Store einstellen zu dürfen, zahlt der Entwickler einen Betrag von 99 US-Dollar.

Die Frage, welche sich für einen Entwickler stellte, sind die Voraussetzungen die benötigt werden, um überhaupt eine Applikation erstellen zu können. Zum einen verlangt Microsoft, dass eine Windows Live-ID von jedem Entwickler erstellt wird. Zum anderen ist eine Kreditkarte für die Abwicklung des Zahlungsverkehrs verantwortlich, welches der Entwickler haben muss. Die Kreditkarte ist bei Microsoft das gängige Konto und wird somit für die Überweisung der Erlöse aus dem Verkauf der Applikationen verwendet. Zudem bietet Microsoft die Windows Phone Developer Tools mit einer angepassten Version von Microsoft Visual Studio und Expression an, mit welchen diese Applikationen erstellt werden. Alle hier nicht explizit mit Kosten benannten Voraussetzungen bietet Microsoft dem Entwickler kostenfrei zu seiner Verfügung an.

Es sind Regeln bei der Veröffentlichung von Applikationen zu beachten, diese sind analog wie bei Google beschrieben, deshalb wird hier auf den zugehörigen Abschnitt des Android Market verwiesen. Microsoft setzt zudem User Experience Guidelines ein, welche von dem Entwickler zu befolgen sind. Im Gegensatz zu Google gibt es für die Applikations-Datei die Beschränkung auf das Format einer XAP-Datei.

5.3.5 Vergleich

Für den Vergleich der vorgestellten App-Stores soll sich anhand von Zahlen zur quantitativen Weiterentwicklung bedient werden und anhand von aktuellen strategischen Entscheidungen der Hersteller sollen diese qualitative Zusatzinformationen bieten.

	Android	Blackberry	iOS	Windows Phone 7
Zusatzgebühr	25 USD einmalig	Nichts	99 USD/Jahr	99 USD/100 Apps
Verdienst:				
Entwickler	70%	70%	70%	70%
Hersteller	30%	30%	30%	30%
Verfügbare Apps	352.800 (Stand: Okt. 2011)	43.000 (Stand: Nov. 2011)	512.749 (Stand: Okt. 2011)	40.000 (Stand: Nov. 2011)
App Download pro Tag	16.660.000	3.000.000	33.000.000	ca. 900.000

Abbildung 5.8: Der Vergleich der einzelnen App-Stores

Diese Abbildung 5.8 vergleicht die einzelnen App-Stores anhand der quantitativen Größe miteinander. Auffällig ist, dass aus den Geschäftsmodellen immer eine 70-30-Erlös Struktur gegeben ist. Nach der Anzahl der Verfügbaren Applikationen steht Apple an höchster Stelle. Bei der Downloadrate pro Tag ist Apple ebenfalls führend.

Zusätzlich macht RIM auf den ersten Blick relativ zu den anderen weniger, aber durch die Information, dass RIM im Vergleich zu Apple pro Applikation im Durchschnitt einen höheren Erlös pro Applikation erzielt, wiegt diese falsche Betrachtungsweise wieder auf.

Der aktuelle Stand dieser Abbildung 5.8 ist für die Zukunft nicht mehr zu verwenden, da durch Wachstum oder Abbau diese Zahlen nur jeweils zum aktuellen Stand aussagefähig sein können.

Eine weitere Möglichkeit alle App-Stores miteinander zu vergleichen ist eine Einordnung in das Vorbild der BCG-Matrix. Diese wurde für diese Seminararbeit zur Vereinfachung angepasst, da die Entwicklung und Sicht in die Zukunft nicht bestimmt werden kann.

Dazu wurden die App-Stores von Apple (schwarzer Kreis) und Google (grüner Kreis) wegen ihres hohen Umsatzes und weiten Verbreitung zu den Stars zugeordnet. Rim (blauer Kreis) wurde wegen der geringen Anzahl der verfügbaren Applikationen zu den ”Cash” Kühen zugeordnet. Windows (roter Kreis) wurde wegen des geringen Umsatzes und des kleinen Marktes zu den ”armen Hunden” zugeordnet. Die einzelnen Strategien sind den Grundlagen dieser Seminararbeit zu entnehmen, wobei deutlich gesagt werden muss, dass durch die Anpassung der BCG-Matrix die Normstrategie nicht anzuwenden ist. Sondern, Windows hat wie bereits erwähnt durch die Kooperation mit Nokia neue Zukunftsperspektiven und dadurch sollte dies eher als Chance betrachtet werden.

Abbildung 5.9: Die BCG-Matrix befüllt mit den App-Stores

5.4 Fazit

Dem Entwickler wurden in dieser Seminararbeit die einzelnen Geschäftsmodelle dargestellt, dadurch hat er einen Einblick in die ökonomische Perspektive der App-Entwicklung gewonnen, welche ihm für seine Entscheidung der Wahl des Herstellers die einzelnen Möglichkeiten aufschlüsseln soll.

Genauer wurde beschrieben wie der aktuelle Stand der einzelnen App-Stores ist und mit quantitativen Zahlen belegt. Ergänzend wurden strategische Entscheidungen der Hersteller in Vergangenheit erläutert und somit ein breites Spektrum für den Entwickler geboten.

Der Vergleich der einzelnen App-Stores hat eine Einordnung in einem betriebswirtschaftlichen Modell gezeigt, dadurch wäre eine Entscheidung bspw. für den Hersteller Apple zu rechtfertigen, indem der Entwickler bei einem gut situierten Hersteller einsteigen kann und damit für die nahe Zukunft eine Sicherheit hat. Diese Sicherheit setzt sich dadurch zusammen, dass dieser Hersteller weniger schnell als andere möglicherweise, vom Markt verschwindet. Ebenso steht die Möglichkeit offen, an einem neuen Markt einzusteigen und dadurch mit hohem Risiko aber großen Gewinnchancen und Entwicklungspotential diesen Markt prägen kann. Dies wäre bei einem Einstieg in Windows möglich.

Erfahren wurde in dieser Seminararbeit, dass schließlich das 70-30-Erlös Modell überall gleich ist, dies ist wichtig für den Entwickler, sodass er seine Entscheidung von anderen Zielen abhängig machen kann.

Der Entwickler kann jetzt, nach seiner Entscheidung, über den Hersteller seine Idee verwirklichen. Dabei weiß er jetzt um seine Risiken, aber auch Möglichkeiten und hat zusammengefasst ein geschliffeneres Gesamtbild über seine persönliche oder berufliche Zukunft.

Literaturverzeichnis

[1] Patrick Stähler. *Geschäftsmodelle in der digitalen Ökonomie* Merkmale, Strategien und Auswirkungen, Josef Eul Verlag, Köln-Lohmar, S. 38-52.

[2] Hannes Weber. *App Stores im Vergleich* http://www.computerworld.ch/news/software/artikel/app-stores-im-vergleich-57152/6/.

[3] Xcubelabs. *Die App Store Mitbewerber im Vergleich* http://www.iphone-ticker.de/infografik-die-app-store-mitbewerber-im-vergleich-22350/.

[4] Wikipedia. *List of digital distribution platforms for mobile devices* http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices#cite_note-cumulative-0.

[5] McKinsey. *BCG Growth-Share Matrix* http://www.quickmba.com/strategy/matrix/bcg/.

[6] Google. *Developer Seite AGB* http://www.android.com/de/developer-distribution-agreement.html.

[7] RIM. *Developer Seite* http://de.blackberry.com/developers/appworld/.

[8] Apple. *Developer Seite* http://developer.apple.com/devcenter/ios/index.action.

[9] Windows. *Developer Seite* http://msdn.microsoft.com/de-de/windowsphone/ff191182.

Kapitel 6

IDEs, Simulatoren, Zertifizierung

Ronny Vogel

Dieses Kapitel gibt einen Überblick über die Konzepte zur Entwicklung von Anwendungen für Smartphones, die auf den vier bedeutendsten Plattformen Android (Google), iOS (Apple), Windows Phone 7 (Microsoft) und Blackberry (RIM) betrieben werden.
Dabei werden zu jeder der vier Architekturen die folgenden Fragestellungen bearbeitet:

- *In welchen IDEs werden Smartphones programmiert?*
- *Welche Programmiersprachen kommen zum Einsatz?*
- *Gibt es Simulatoren oder kann nur auf dem Endgerät getestet werden?*
- *Welche Einschränkungen hinsichtlich der Funktionalität haben die Simulatoren?*
- *Gibt es Software zur Unterstützung der Testfälle?*
- *Welche Kosten entstehen für IDEs, Entwicklerzertifikate usw.?*

Inhaltsverzeichnis

6.1 Einleitung 161
6.2 Google: Android 162
6.2.1 Aufbau Android 162
6.2.2 Android SDK 163
6.2.3 Hallo Android! 163
6.2.4 Ausführen und Testen von erstellten Anwendungen 164
6.2.5 Debuggen und Testen von Android-Apps 165
6.2.6 Zertifikate 166
6.3 Apple: iOS 167
6.3.1 Allgemeiner Überblick 167
6.3.2 Hallo iOS! 168
6.3.3 Debuggen und Testen von iOS-Apps 169
6.3.4 Zertifikate 170
6.4 Microsoft: Windows Phone 7 171
6.4.1 Allgemeiner Überblick der WP7-Anwendungsplattform 171
6.4.2 Entwickeln für WP7 172
6.4.3 Hallo Windows Phone 7! 173
6.4.4 Debuggen und Testen von WP7-Anwendungen 173
6.4.5 Zertifikate 174
6.5 Research in Motion: BlackBerry 175
6.5.1 Allgemeiner Überblick BlackBerry OS 175
6.5.2 Entwickeln für BlackBerry OS 176
6.5.3 Hallo BlackBerry! 177
6.5.4 BlackBerry-Simulatoren 178
6.5.5 Debuggen und Testen von BlackBerry-Apps 178
6.5.6 Zertifikate 179
6.6 Zusammenfassung 180

6.1 Einleitung

Egal, was auch immer man für ein Problem hat: heutzutage gibt es für alles eine App. Mit dieser Botschaft jedenfalls hat Apple für sein Smartphone, das iPhone geworben. Und auch die drei bedeutendsten Konkurrenten werben in erster Linie mit dem immer größer werdenden Angebot in ihren App-Marktplätzen. Nachdem die Email zu einer "Killeranwendung" des 20. Jahrhundert geworden ist, so scheint diese Entwicklung gerade auch bei den Apps für Smartphones einzusetzen. Jedenfalls stellt die Verfügbarkeit von vielen Apps ein wichtiges Kaufargument bei der Entscheidung für bzw. gegen eine der vier den Markt dominierenden Smartphone-Technologien iOS, Android, BlackBerry und Windows Phone 7 dar (Abbildung 6.1 zeigt ein Gerät jeder Plattform).

Abbildung 6.1: Aktuelles Smartphone mit dem Betriebssystem Android, iOS, Windows Phone 7 und BlackBerry (von links nach rechts)

Da es bei jeder dieser Plattform inzwischen mehr Apps gibt, als ein Mensch jemals ausprobieren könnte, steht besonders die Qualität der vertriebenen Software im Vordergrund. Besonders wichtig wird das, wenn man bedenkt, das alle Smartphone-Besitzer fast ständig erreichbar sein wollen und damit gleichbedeutend auch ständig online sind, was sie als stets verfügbare Zielscheibe für jedwede Malware und sonstige Sicherheitsrisiken machen, die eine Verbindung zum Internet mit sich bringen.
Aus diesen Gründen muss den Vertreibern dieser Plattformen - Apple, Google, Microsoft und Research in Motion - nicht nur daran liegen, das möglichst viele Entwickler in kurzer Zeit das App-Angebot erweitern und dafür sorgen, dass es für jedes Problem eine App gibt. Sie müssen ihnen im Besonderen Tools an die Hand geben, mit denen diese Apps auch qualitativ hochwertig sind. In den folgenden Abschnitten wird daher untersucht, welche Konzepte die vier genannten Hersteller für ihr System bei der Entwicklung neuer Anwendungen verfolgen. Es wird gezeigt, welche IDEs unterstützt werden und inwiefern es möglich ist, die eigenen Anwendungen auch auszuführen und zu testen, sei es in einem Simulator oder tatsächlich auf einem mobilen Endgerät.

6.2 Google: Android

Bei Android handelt es sich um ein Betriebssystem für mobile Endgeräte wie Smartphones und Tablets. Die Entwicklung dieser Softwareplattform begann im Jahr 2005 mit der Übernahme der Firma Android Inc. durch Google. Ziel war und ist es, eine im Sinne von Open Source offene Plattform zu schaffen, die gleichzeitig innovativ und kostengünstig sein sollte. Mit diesen Zielsetzungen wurde schließlich die Open Handset Alliance gegründet, der sich mittlerweile ca. 80 Unternehmen - im Wesentlichen Hersteller mobiler Endgeräte, Netzbetreiber und Softwarefirmen - angeschlossen haben. Weitere Details zur Geschichte von Android finden sich in [2] bzw. auf der Seite der Open Handset Alliance http://www.openhandsetalliance.com.
Es soll nun ein allgemeiner Überblick von Android gegeben werden, um danach zu zeigen, wie Software für Android entwickelt wird.

6.2.1 Aufbau Android

Wie man in Abbildung 6.2 erkennen kann, bildet ein Linux-Kernel die Basis von Android, der sich um die grundlegenden Dinge wie Speicherverwaltung, Prozessmanagement und die Energieverwaltung kümmert. Weiterhin ist zu sehen, das es daneben nur noch einige C/C++-Bibliotheken wie Webkit, OpenGL und SQLite gibt, also für Aufgaben, die besonders performant ablaufen müssen. Alles andere bei Android sind Java-Anwendungen, die auf einer speziellen virtuellen Maschine (VM) laufen, der Dalvik Virtual Machine.
Applikationen werden zwar zunächst wie ganz normale, in einer JVM lauffähige Java-

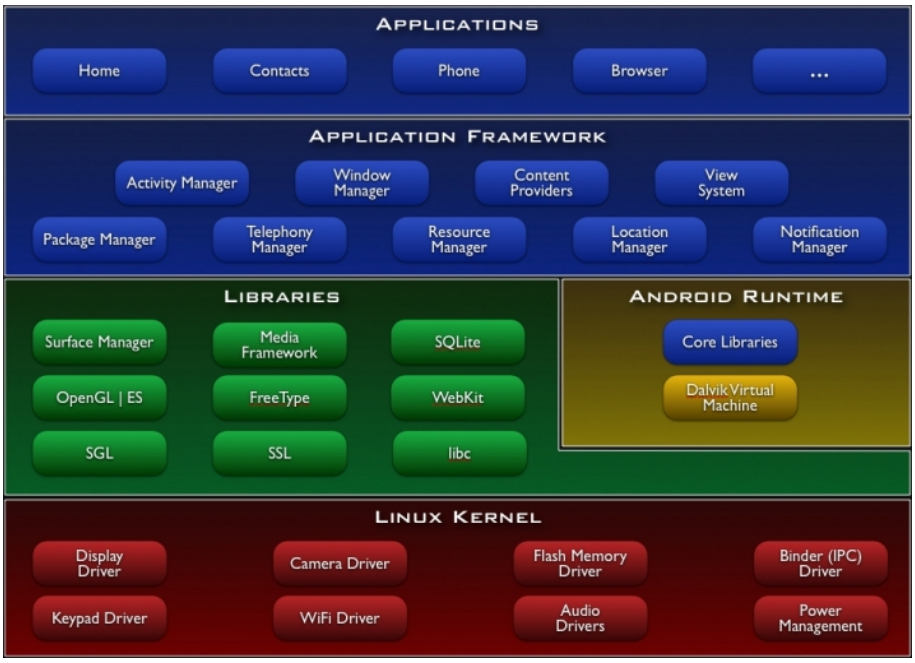

Abbildung 6.2: Komponenten der Android-Architektur aus [1]

Anwendungen erzeugt, aber danach für die Dalvik VM kompiliert. Hintergrund ist, das Anwendungen für mobile Endgeräte möglichst wenig Speicher verbrauchen dürfen und

idealerweise nur einen geringen Energieverbrauch verursachen. In [2] (Seiten 5ff.) wird das sehr anschaulich anhand der Browser-App vermittelt: während die Jar-Variante für die JVM 500 kBytes umfasst, sind es bei der Dalvik VM-Version nur noch 200 kBytes.
Zusätzlich wurde der Garbage Collector für die Anforderungen mobiler Endgeräte optimiert. Schließlich ist noch ein wesentlicher Unterschied zur üblichen JVM auf Assembler-Ebene zu nennen: Während üblicherweise Stacks zum Einsatz kommen, nutzt die Dalvik VM Register. Damit soll die Anzahl der Anweisungen erheblich verringert werden ([2] beziffert die Verbesserung auf 30 Prozent).
Offensichtlich ist also Java-Bytecode auf Android-Systemen nicht ausführbar, sondern hierfür müssen die Java-Class Dateien in für die Dalvik VM geeignete *.dex-Dateien umgewandelt werden. Ähnlich wird bei der Benutzeroberfläche vorgegangen: Die entsprechenden Dateien sind zwar im XML-Format, werden aber für die Verwendung auf Android noch in ein spezielles, binäres Format konvertiert.

6.2.2 Android SDK

Alle zur Erstellung von Android-Anwendungen erforderlichen Tools werden im Android SDK zur Verfügung gestellt. Dieses kann kostenlos vom Android Developer Guide [1] für Windows, Mac OS X und Linux bezogen werden. In Verbindung mit einem Java Development Kit könnte man nun bereits mit einem Texteditor und den Android SDK Tools entwickeln, viel komfortabler ist aber natürlich die Verwendung einer vollwertigen Entwicklungsumgebung. Das Android SDK kann problemlos in die von Java-Entwicklern häufig verwendete und ebenfalls kostenfrei nutzbare IDE Eclipse integriert werden. Generell ist die Nutzung jeder Java-Entwicklungsumgebung möglich, aufgrund der Bedeutung von Eclipse und seiner kostenlosen Verfügbarkeit soll nur dieses hier betrachtet werden.
Im SDK sind im Wesentlichen die folgenden Komponenten enthalten:

SDK Tools Entwicklungswerkzeuge v. a. zum Testen und Debuggen

SDK Platform-Tools Plattformabhängige Tools zur Entwicklung und Debugging

Samples Eine umfangreiche Sammlung an Codebeispielen und Beispiel-Apps, geordnet nach Android-Plattformen

Documentation Offline-Kopie des Android Developer Guides [1].

6.2.3 Hallo Android!

Wenn man schon einmal mit Eclipse gearbeitet hat, wird man kaum Probleme haben damit Anwendungen für Android zu entwickeln. Als Beispielanwendung soll hier das klassische HelloWorld zum Einsatz kommen, denn im Mittelpunkt dieser Abhandlung steht die Entwicklung. Hierfür ist neben Eclipse und dem Android SDK noch die Installation der Android Development Tools (ADT) erforderlich, die die Entwicklungstools aus dem SDK in Eclipse integrieren.
Nach Erstellung des Projektes wird im Dateisystem eine wie in Abbildung 6.3 ersichtliche

Ordnerstruktur angelegt. In *src* wird der Quelltext gespeichert, der Ordner *gen* enthält die von den Android SDK Tools automatisch generierten Dateien, *Google-APIs* enthält die Bibliotheken für die dem Projekt zugeordnete Android-Version und im Ordner *res* mit seinen Unterordnern werden Ressourcen abgespeichert wie Bilder, Icons, lokalisierte Strings. Zusätzlich sind in res auch noch die XML-Dateien enthalten, mit denen die Benutzeroberfläche, also die einzelnen Views, definiert werden.
Im folgenden Codeausschnitt sind die einzigen drei einzufügenden Codezeilen ersichtlich,

Abbildung 6.3: Struktur eines neu erstellten Android-Projektes

die für das gewünschte ”Hallo Welt” sorgen werden:

```
public void onCreate(Bundle savedInstanceState) {
    ...\\gekürzte Fassung
    TextView tv = new TextView(this);
    tv.setText("Hallo Welt!");
    setContentView(tv);
}
```

Es wird mit TextView ein Label erzeugt, also ein Steuerelement, das Text anzeigt. Diesem wird dann mit setText() der auszugebende Text zugewiesen und dann wird dieses Label zur aktuellen View hinzugefügt. Die View ist das, was der Benutzer auf dem Handy zu sehen bekommt. In dieser einfachen Anwendung ist diese natürlich äußerst primitiv, doch um auch anspruchsvollere Views zu erstellen, bietet ADT einen entsprechenden WYSIWYG-Editor, mit dem die Oberfläche entsprechend zusammengestellt werden kann.

6.2.4 Ausführen und Testen von erstellten Anwendungen

Um nun die erstellte Anwendung auszuführen und zu testen, gibt es zwei Möglichkeiten: entweder man nutzt ein Android-Gerät oder man verwendet den im Android SDK beinhalteten Emulator. Es soll nun die Verwendung eines Emulators betrachtet werden. Abbildung 6.4 zeigt die entsprechende Ausgabe des Hallo-Welt-Programmes in einem Emulator. Diesen muss man zunächst erzeugen mit Hilfe des AVD Managers. AVD steht dabei für Android Virtual Device. Dieses virtuelle Gerät läßt sich dann auswählen, um das erstellte Programm auszuführen und dann erhält man eine Ausgabe wie in Abbildung 6.4 dargestellt.
Es wird eine Android-Umgebung emuliert, wobei folgende Funktionalitäten im virtuellen Androiden in der aktuellen Version des Android SDK nicht zur Verfügung stehen bzw. diese können noch nicht nachgebildet werden (nach [1]):

Abbildung 6.4: Ausgabe von "'Hallo Welt!"' im Emulator

- USB-Verbindungen,
- Camera/Video Capture,
- Kopfhörer, die am Gerät angeschlossen sind,
- Bestimmung des Verbindungsstatus',
- Bestimmung des Ladezustandes und der Art der Stromversorgung,
- Erkennung des Einlegens/Entnahme einer SD-Karte und
- Bluetooth.

Neben dem Emulator, der einem Entwickler die Möglichkeit bietet, seine Anwendung unter den verschiedenen Versionen von Android mit zudem unterschiedlichen Gerätekonfigurationen zu testen, kann man die entwickelte Anwendung auch auf einem tatsächlichen Gerät ausführen.

6.2.5 Debuggen und Testen von Android-Apps

Zu Eclipse gehört bereits ein JDWP-Debugger (Java Debug Wire Protocol), sodass man ohne zusätzliche Konfigurationen im Quellcode Breakpoints setzen, Variablenwerte überwachen, Codeausführung anhalten kann und so weiter. Zum Debugvorgang von Android-Anwendungen gehören im Wesentlichen drei Hauptkomponenten:

adb Das Tool adb stellt eine Verbindung zwischen Entwicklungssystem und dem Android-Gerät her.

Dalvik Debug Monitor Server DDMS ist gewissermaßen die graphische Oberfläche zum Debuggen von Android-Systemen über adb. Es erlaubt Screenshots, das Auslesen von Log-Dateien, das Simulieren eingehender Anrufe/SMS, um die wichtigsten Funktionalitäten zu nennen.

Android-System Natürlich muss eine Anwendung auf einem Android-System laufen, das entweder ein echtes Gerät ist oder eben ein AVD. In beiden Fällen läuft auf dem Gerät bzw. Emulator ein adb-Dämon, sodass man von seiner Entwicklungsumgebung aus mit adb auf das System zugreifen kann als Voraussetzung für die Nutzung von DDMS bzw. den in Eclipse integrierten Debug-Tools.

Zum Testen gibt es ein umfangreiches Android Testing Framework. Für Programmteile, die keinen Gebrauch von der Android API machen, kann man JUnit nutzen. Für Klassen, die die API verwenden, gibt es Android JUnit Erweiterungen, die entsprechende Hilfsklassen enthalten, um spezielle Hilfsobjekte zur Nachahmung von Android-Funktionalität zu erstellen. So werden entsprechende TestCases bereitgestellt, um die verschiedenen Android-Komponententypen wie Activities, Services und Content Provider zu testen.
Daneben bietet das SDK noch zwei weitere Tools zum Testen. Das ist zum Einen der UI/Application Exerciser Monkey, kurz monkey. Monkey ist ein Kommandozeilentool, mit dem zufällig Tastenfolgen, Touchgesten und sonstige Benutzereingaben an ein Gerät gesendet werden - also ein Stress-Test-Tool. Zum Anderen gibt es das monkeyrunner-Tool. Das ist eine API, mit deren Hilfe man sehr umfangreiche Testprogramme schreiben kann.

6.2.6 Zertifikate

Alle auf einem Android-Sytem installierten Anwendungen müssen mit einem Zertifikat signiert sein, dessen privater Schlüssel dem jeweiligen Anwendungsentwickler gehört. Diese Anfordung trifft auch in der Entwicklungsphase zu, egal ob auf einem physischen Gerät oder dem Emulator entwickelt wird. Sinn dieser Richtlinie ist es, das einerseits der Autor einer Anwendung identifiziert werden kann und andererseits dient es dazu, Vertrauensbeziehungen zwischen Anwendungen zu erstellen. Das Zertifikat muss allerdings nicht von einer Zertifizierungsstelle authorisiert werden, sondern kann selbst signiert sein.
In der Entwicklungsphase wird mittels der Debug bzw. Buildtools automatisch ein Key und ein Keystore mit einem bekannten Standard-Passwort erzeugt. Somit kann automatisch die *.apk-Datei von den SDK-Tools signiert werden, ohne das dafür eine Benutzereingabe erforderlich wäre. Hierbei ist es interessant zu wissen, dass das Debug-Zertifikat nach 365 Tagen abläuft. Will man nach Ablauf dieser Zeit die Anwendung kompilieren, erhält man eine Fehlermeldung. Um weiterhin entwickeln zu können, muss man dann einfach nur die debug.keystore-Datei löschen (Details zum Vorgehen siehe [1]).
Um die Anwendung schließlich zu veröffentlichen, ist es notwendig, sie mit einem eigenen privaten Schlüssel zu signieren, wofür das SDK entsprechende Tools bereit hält (Keytool, Jarsigner, ADT Export Wizard). Man kann das Debug-Zertifikat zum Veröffentlichen nicht verwenden. Das genaue Vorgehen zum Veröffentlichen von Android-Anwendungen ist nicht Thema dieses Kapitels und daher wird hier auf den Android Developer Guide [1] verwiesen.

6.3 Apple: iOS

Wenn von Smartphones die Rede ist, denkt man wohl zuerst an das iPhone. Mit dem iPhone hat Apple 2007 das erste Smartphone auf den Markt gebracht, das sich erfolgreich verkaufte und somit diese im Endkonsumentbereich zu diesem Zeitpunkt kaum verbreitete Smartphone-Technologie als ein neues, nachgefragtes Produkt etablieren konnte. Ähnliche Produkte waren schon vorher da, nur hatte keines diese intuitive Touchoberfläche in Verbindung mit herausragenden Multimediaeigenschaften und der mittlerweile riesigen App-Welt: Für jedes Problem gibt es bekanntlich eine App.
In diesem Abschnitt geht es um die Entwicklung von Apps für das iPhone und damit für das Betriebssystem iOS. Es handelt sich dabei jedoch nicht um das Router-Betriebssystem von Cisco, nur der Name ist derselbe. Als Quelle zur Erstellung des iOS-Abschnittes diente vor allem die Apple eigene Online-Dokumentation für iOS-Entwickler unter [16].

6.3.1 Allgemeiner Überblick

Der Kernel von iOS basiert auf demselben MACH-Kernel wie auch Mac OS X. Auf diesem Kernel bauen die in Abbildung 6.5 dargestellten Schichten auf, die es ermöglichen, Anwendungen für iOS zu implementieren, indem sie entsprechende Frameworks bereit halten. Dabei sind in der obersten Schicht, dem Cocoa Touch Layer, die Schlüsseltechnologien enthalten wie Dateimanagement, Netzwerkoperationen u. ä.

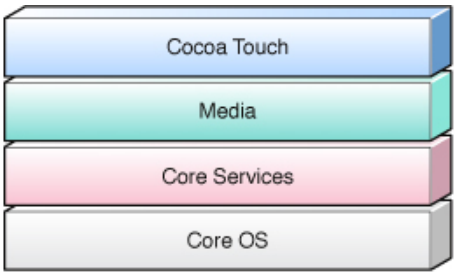

Abbildung 6.5: Schichtenmodell von iOS [16]

Als Programmiersprache kommt Objective-C zum Einsatz, bei Verwendung von Frameworks der unteren Schichten wird auch reines C benötigt. Objective-C basiert auf C mit objektorientierten Erweiterungen von SmallTalk, einer der ersten objekt-orientierten Sprachen. Weitergehende Informationen zu dieser Sprache finden sich in [16], [17] und [18].
Um Anwendungen für iOS zu erstellen, benötigt man das iOS SDK und die IDE Xcode, die Entwicklung ist damit ausschließlich unter Mac OS X möglich. Mit Xcode kann man nicht nur C bzw. Objective-C programmieren, sondern ähnlich wie Eclipse viele andere Programmiersprachen. Es sei darauf hingewiesen, das auch die gesamte Werkzeugsammlung für die iOS-Entwicklung mit Xcode bezeichnet wird. Zum Erstellen der grafischen Benutzeroberflächen findet der Interface Builder Anwendung, von dem ein Screenshot in Abbildung 6.6 zu sehen ist. Das iOS SDK beinhaltet wiederum das iOS Simulator SDK zur Entwicklung von Apps für den Simulator und das iOS Device SDK für die Entwicklung auf iPhones.

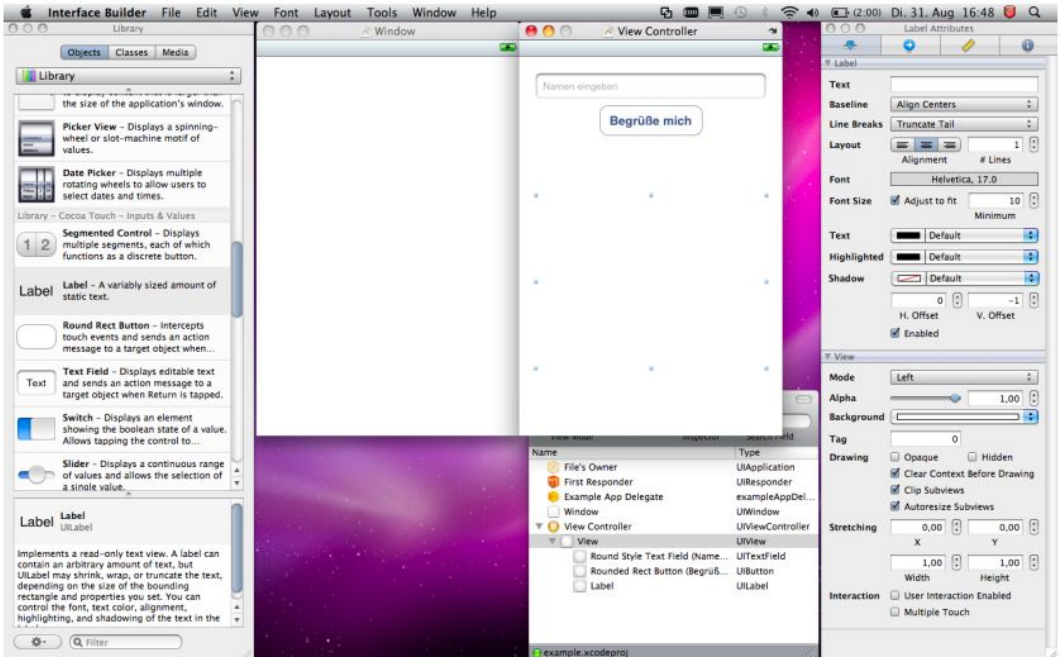

Abbildung 6.6: Screenshot vom Interfacebuilder aus [18]

6.3.2 Hallo iOS!

Wie man es von C kennt, so wird auch bei Objective-C die Deklaration und Definition getrennt. In Header Dateien mit der Endung *.h erfolgen die Definitionen und die Deklaration findet dann in einer gleichnamigen Datei mit der Endung *.m statt. Daneben gibt es noch *.plist-Dateien zum Speichern von Benutzer-Konfigurationen und die *.xib-Dateien, in denen die Views im XML-Format abgespeichert werden. Um einen Eindruck zu erhalten, wie Objective-C aussieht, sei hier ein Ausschnitt aus dem HelloWorld-Codebeispiel aus [16] aufgeführt:

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    // Set up the view controller
    MyViewController *aViewController = [[MyViewController alloc]
       initWithNibName:@"HelloWorld" bundle:[NSBundle mainBundle]];
    self.myViewController = aViewController;
    [aViewController release];
    [[UIApplication sharedApplication]
       setStatusBarStyle:UIStatusBarStyleBlackOpaque];

    // Add the view controller's view as a subview of the window
    UIView *controllersView = [myViewController view];
    [window addSubview:controllersView];
    [window makeKeyAndVisible];
}
```

6.3.3 Debuggen und Testen von iOS-Apps

Die eigens erstellte Anwendung kann man nun im Simulator oder auf einem iPhone testen. Da es einen Unterschied macht, ob man im Simulator oder auf einem echten iPhone testet, gibt es die Prä-Prozessormakros
TARGET_IPHONE_SIMULATOR und TARGET_OS_IPHONE, mit denen man festlegt, welcher Code in welchem Einsatzfall (also iPhone oder Simulator) kompiliert werden soll.

Abbildung 6.7: Xcode-Umgebung und Simulator

Um auf einem iPhone zu testen, muss man Mitglied im iOS Development Team sein, was - mit Ausnahme für Universitäten - kostenpflichtig ist (Stand November 2011: 99 USD Jahresbeitrag für Individual- & Company-Account, 299 USD/Jahr im Enterprise Programm). Folgende Funktionalitäten kann man nur auf einem Gerät testen bzw. es bestehen sehr starke Unterschiede in der Simulatorausgabe und dem Verhalten auf einem iPhone:

- Kamera,
- Beschleunigungssensor,
- Ereignisse, für die mehr als 2 Finger notwendig sind,
- GPS-Sensor,
- Rendering mit OpenGL ES.

Xcode bietet alle notwendigen Tools, um eine Anwendung zu Debuggen. Beispielsweise werden Anwendungsmeldungen angezeigt, mit Hilfe von NSLog kann man auch selbst solche Meldungen in einer eigenen Anwendung implementieren. Zudem gibt es ein ähnliches Testpaket wie JUnit in Java, mit dem man Testfälle definieren kann.

Zum Optimieren der Anwendung steht mit Instruments eine umfangreiche Toolsammlung bereit, mit Hilfe derer man zum Einen den Ressourcenverbrauch zur Laufzeit messen und zum Anderen die Anwendung entsprechend optimieren kann.

6.3.4 Zertifikate

Für das Entwickeln auf einem iPhone wird ein Entwicklungs-Zertifikat benötigt, das man nur im Rahmen einer Mitgliedschaft im iOS Development Team erhält. In Abbildung 6.8 wird dieser Zusammenhang verdeutlicht. Wenn man sein Gerät registriert hat, wird der entsprechende Schlüssel am persönlichen "Schlüsselanhänger" befestigt, daher der Begriff *keychain*. Bei der Registrierung eines iPhones gibt es zudem zwei Stufen. Die erste ist sozusagen die Basisstufe, damit man überhaupt auf dem Gerät entwickeln kann. Wenn man in seiner Anwendung Funktionen wie iCloud Storage, Push-Dienste, In-App-Käufe und/oder Game Center nutzt, bedarf es der Einrichtung für "Specialized Development", was wiederum die Basisstufe voraussetzt.

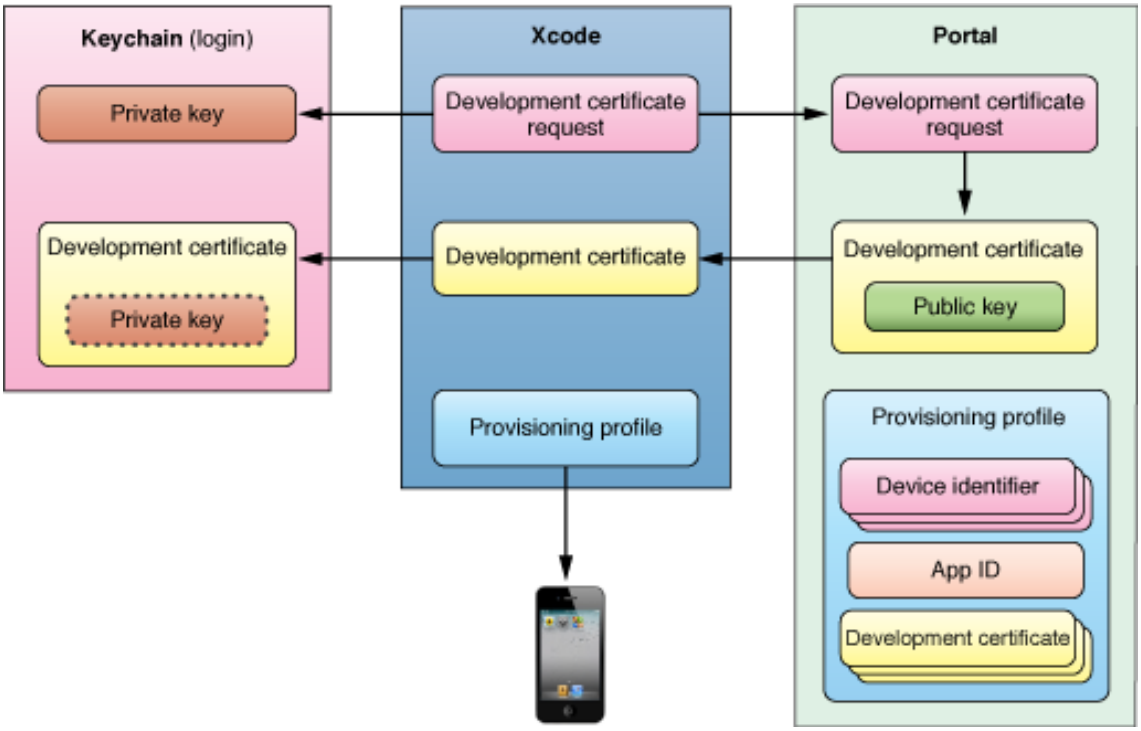

Abbildung 6.8: Voraussetzungen zum Entwickeln für iOS auf einem iPhone

6.4 Microsoft: Windows Phone 7

Windows Phone 7 (im Folgenden mit WP7 abgekürzt) ist das neueste Betriebssystem für Smartphones aus dem Hause Microsoft. Im Gegensatz zu den Vorgängerbetriebssystemen für mobile Endgeräte, die noch die Bezeichnung Windows Mobile trugen, hat man hier bewusst einen neuen Namen gewählt. Es wurde von Microsoft im Februar 2010 auf dem Mobile World Congress der Öffentlichkeit vorgestellt und unterscheidet sich im Wesentlichen durch die neue, komplett Multitouch-fähige Benutzeroberfläche mit Codenamen Metro. Eine weitere wichtige Änderung im Vergleich zu den Mobile-Versionen sind die restriktiveren Hardwareanforderungen, die Microsoft für den Einsatz von WP7 voraussetzt. Damit wird den Endgeräteherstellern zwar wenig Spielraum zur Herstellung von Geräten, die sich von Konkurrenzprodukten mit WP7 unterscheiden, gelassen, aber dafür ist es einfacher bzw. überhaupt erst möglich, eine Anwendung einmal zu erstellen, sodass diese dann auf allen Windows Phone-Geräten lauffähig ist. Denn nur wenn die von Microsoft vorgegebenen Hardwarevoraussetzungen (vgl. [13], S. 2ff) erfüllt sind, darf ein Smartphone als Windows Phone bezeichnet werden. Der folgende Abschnitt soll einen Überblick zu WP7 geben.

6.4.1 Allgemeiner Überblick der WP7-Anwendungsplattform

Die WP7-Anwendungs-Plattform wird in Abbildung 6.9 dargestellt. Bei den "Windows Phone Frameworks" kann man die von einem WP7 angebotenen Dienste erkennen, die "Common Base Class Library" zeigt die verfügbaren .NET-Bibliotheken auf.
Besonders interessant ist der Bereich "Application Objekt", denn hier ist zu sehen, das

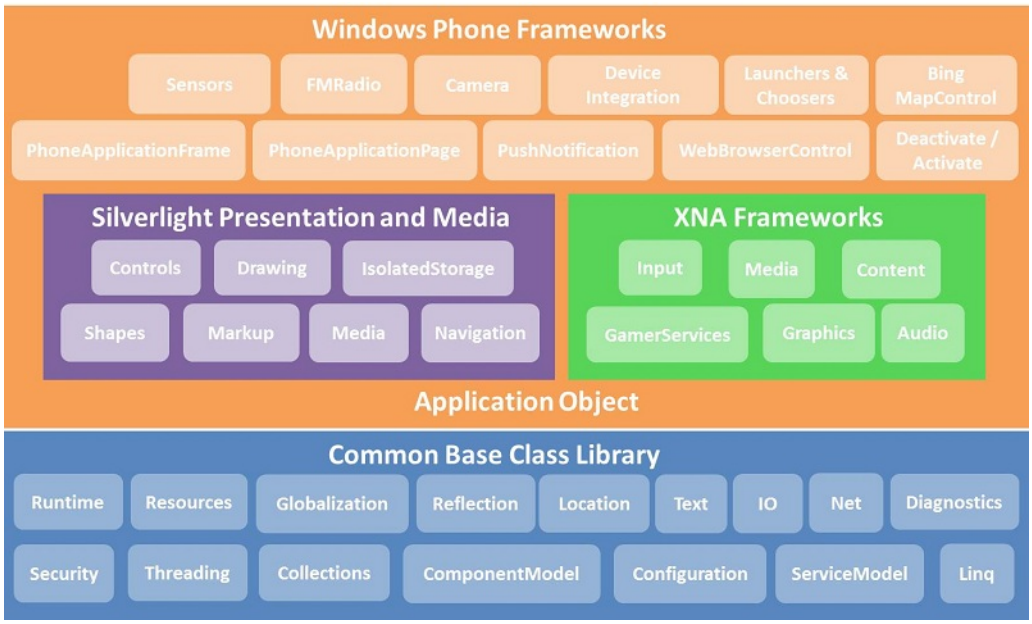

Abbildung 6.9: WP7-Anwendungsplattform aus [12]

sowohl Silverlight als auch XNA bei der Erstellung von WP7-Anwendungen Verwendung findet:

Silverlight ist ein Framework zur Erstellung von umfangreichen Web-Anwendungen, vergleichbar mit Adobe‘s Flash. Es ist aus der Windows Presentation Foundation (WPF) abgeleitet und bietet die Möglichkeit, die Entwicklung der Oberfläche von der Programmierung zu trennen.

XNA ist ein Framework, welches Software, Dienste und Ressourcen zur Erstellung von Spielen für Microsoft‘s Spiele-Plattformen - Windows, Xbox 360, Zune und eben auch Windows Phone - beinhaltet. Es ist besonders dafür geeignet, Spiele mit aufwändigen 2D/3D-Grafiken zu erstellen.

Das heißt also im Wesentlichen, das man zur Erstellung von Anwendungen das Silverlight-Framework benutzt und wenn man ein Spiel entwickeln möchte, ist XNA die geeignete Wahl. Seit Version Windows Phone 7.5 ist es sogar möglich, beide Frameworks bei der Entwicklung miteinander zu kombinieren, näheres dazu erfährt man in der Hauptquelle für dieses WP7-Kapitel, dem Microsoft Developer Network (MSDN) [15].

6.4.2 Entwickeln für WP7

Abhängig von der Wahl eines der beiden Frameworks, wird auch das dazu notwendige Toolset bestimmt. Microsoft stellt mit WP7 auch die dazugehörigen Entwicklungswerkzeuge kostenlos zur Verfügung. Diese Windows Phone 7 Developer Tools enthalten die folgenden Programme:

Visual Studio Express ist die Entwicklungsumgebung,

Expression Blend for Windows Phone zur Erstellung professioneller Oberflächen,

XNA Game Studio für die Entwicklung von Spielen,

Windows Phone Emulator zur Emulation eines WP7-Smartphones,

Silverlight zur Erstellung von Anwendungen und

.NET-Framework mit den benötigten Klassenbibliotheken und der Laufzeitumgebung.

Als Programmiersprachen kommen also die .NET-Sprachen Visual Basic oder C# zum Einsatz. Um eine einfache Anwendung zu erstellen reicht das Visual Studio vollkommen aus, denn hier ist auch ein vollwertiger GUI-Editor integriert. Abbildung 6.10 zeigt, wie diese IDE aussieht.
Im Folgenden wird nur auf die App-Entwicklung mit Silverlight unter Verwendung von C# eingegangen, da XNA tatsächlich recht speziell ist - beispielsweise erfordert allein schon das Verwenden von Text bereits einigen Mehraufwand (vgl. [14] S. 30ff). da es in XNA keine TrueType-Schriftarten gibt, sondern alles als Bitmap behandelt wird.

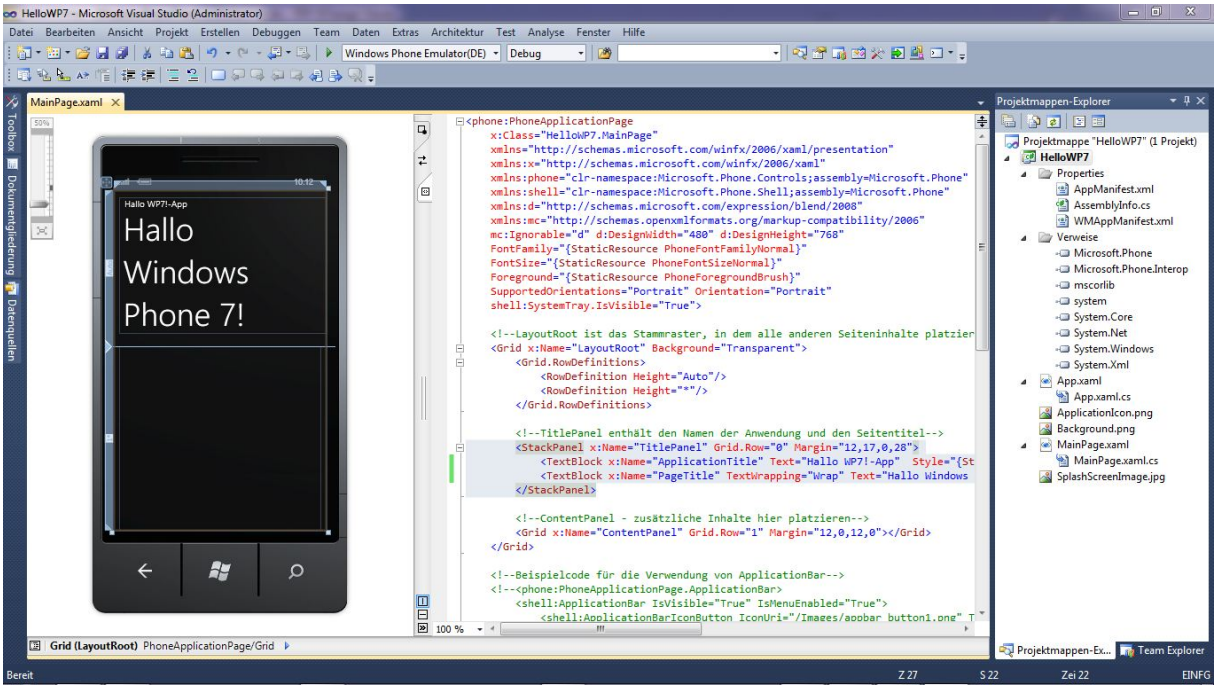

Abbildung 6.10: Entwicklung einer WP7-Anwendung in Visual Studio

6.4.3 Hallo Windows Phone 7!

Wenn man in Visual Studio ein neues WP7-Projekt angelegt hat, wird man folgendes Konzept erkennen: Es gibt zum Einen XAML-Dateien und zum Anderen zu jeder XAML-Datei eine gleichnamige Datei mit der zusätzlichen Endung **cs**, also beispielsweise App.xaml und App.xaml.**cs**. In der XAML-Datei steht in XML-Format die Beschreibung einer Ressource, beispielsweise eines Programm-"Fensters" und in der zugehörigen CS-Datei der dazugehörige Quellcode, daher heißt diese auch Code-Behind-Datei. Sinn dieses Konzeptes ist die Trennung der Programmierung vom Oberflächendesign. Der Entwickler erstellt eine XAML-Datei, in der die benötigten Steuerelement mit entsprechende Bezeichnung enthalten sind und gibt diese einem Grafik-Designer, der die Oberfläche dann entsprechend mit dem Profi-Tool Expression Blend erstellt, während der Entwickler programmieren kann. Diese Trennung ist allerdings nicht unbedingt erforderlich, man kann alles, was mit XAML definiert wird, auch allein mit Quellcode beschreiben. Wie man an Abbildung 6.10 erkennen kann, reicht eine einfache Änderung des automatisch generierten Codes aus, um die "Hallo WP7"-Anwendung zu erstellen. Zudem sieht man, das bereits während der Entwicklung im WYSIWYG-GUI-Editor alle Änderungen im XAML-Quellcode übernommen werden und man sofort die Auswirkung auf das Layout sieht.

6.4.4 Debuggen und Testen von WP7-Anwendungen

Standardmässig wird unter Visual Studio mit dem Windows Phone Emulator getestet. Dieser emuliert ein Windows Phone, lediglich der Beschleunigungssensor und die Ortsbestimmung müssen simuliert werden, was entweder durch Wiedergabe abgespeicherter Daten oder entsprechende Eingabe konkreter Werte erfolgt. Um auf einem physischen

Gerät zu testen, muss dieses Gerät hierfür registriert sein. Dazu ist eine Windows Live ID erforderlich und eine kostenpflichtige App Hub-Mitgliedschaft. Es wird hierfür ein Jahresbeitrag von derzeit (November 2011) 99 USD fällig. Für Studenten gibt es das sogenannte Dream Spark Programm, womit die Mitgliedschaft kostenlos ist. Mit dieser Mitgliedschaft erwirbt man auch das Recht, eine bestimmte Anzahl Apps über den WP7-Marketplace zu veröffentlichen.
Um die Performance der Anwendung zu optimieren gibt es das Windows Phone Performance Analysis Tool, welches in Abbildung 6.11 dargestellt wird. Damit können Leistungsdaten während des Testens aufgezeichnet werden.

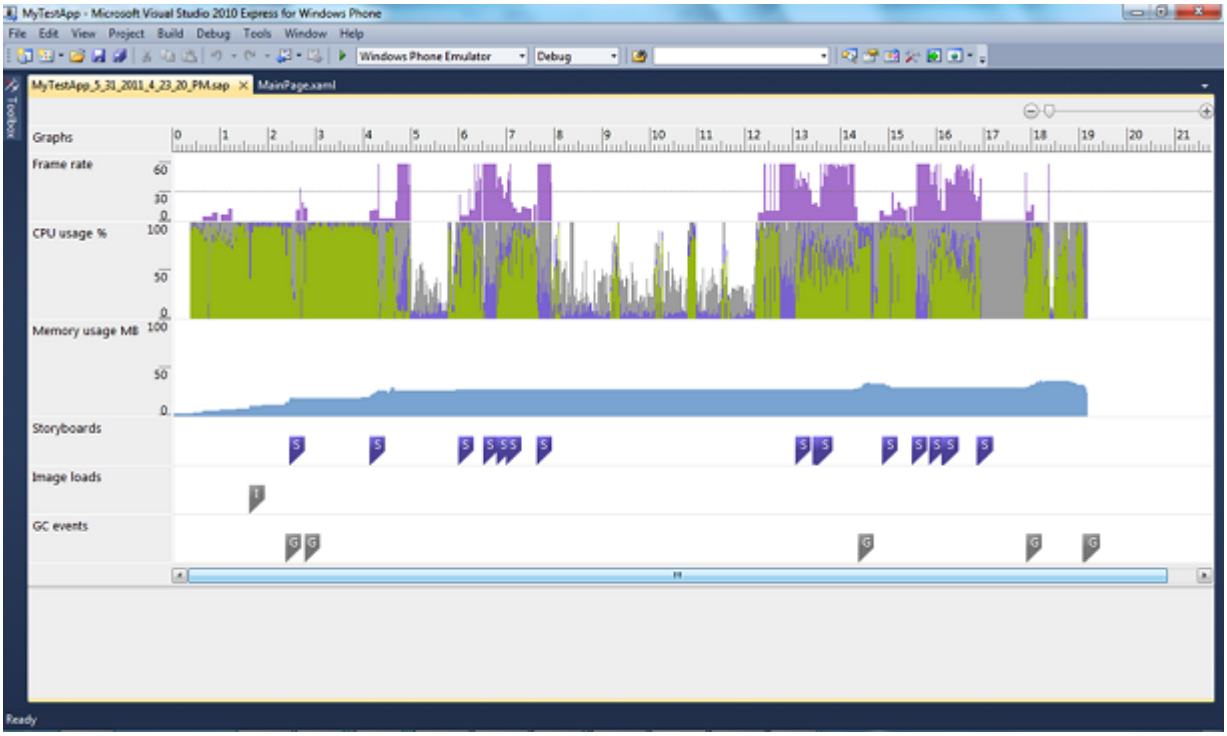

Abbildung 6.11: Windows Phone Performance Analysis Tool [15]

6.4.5 Zertifikate

Bevor eine WP7-App veröffentlicht werden kann, muss sie von Microsoft zertifiziert werden. Dabei wird von Microsoft innerhalb von 5 Werktagen geprüft, ob die Anwendung zuverlässig arbeitet, ressourcenschonend, frei von Malware ist und die Telefonfunktionalität nicht beeinträchtigt. Im Rahmen dieses Zertifizierungsprozesses wird die Anwendung schließlich auch signiert. Nur wenn die Zertifizierung nicht erfolgreich ist, erhält man in der Regel einen Testbericht. Genauere Angaben und vor allem akuelle Anforderungen sind der MSDN zu entnehmen.

6.5 Research in Motion: BlackBerry

BlackBerry ist die Bezeichnung für Smartphones, welche vom kanadischen Unternehmen Research in Motion (RIM) entwickelt wurden. RIM zielt vor allem auf Unternehmensnetzwerke ab und entsprechend sind BlackBerries für diesen Einsatzzweck spezialisierte, mobile Endgeräte. Hierzu gehören aber nicht nur die Endgeräte, sondern auch Serversoftware. Mit Hilfe dieser Serverdienste werden zu versendende Daten, im Wesentlichen natürlich Emails, so aufbereitet, das die Informationen so schnell wie möglich beim Benutzer ankommen. Das heißt im Falle von Emails also, das der Benutzer bei Mails mit vielen Anhängen nicht erst warten muss, bis sein Endgerät alle Daten vom Server erhalten hat, sondern er sofort die Informationen lesen kann. Das geschieht zum Einen dadurch, das die Server die Emails aufbereiten, sodass sie möglichst wenig Speicher verbrauchen und somit schnell übertragen werden können, was wiederum in kleinen Einheiten erfolgt. Auf einem BlackBerry werden Emails zum Anderen so behandelt, als wären es SMS: sobald eine neue Email im Postfach landet, erhält sie der Empfänger sofort. Dieses Push-Feature ist aber schon lange kein Alleinstellungsmerkmal von BlackBerries mehr, sondern es gehört mittlerweile zur Grundfunktionalität eines Smartphones egal welchen Herstellers, dass regelmäßig auf neue Nachrichten geprüft wird und so der Benutzer zeitnah seine Emails auf seinem Handy hat . Aus diesem Grunde verliert BlackBerry auch mehr und mehr an Bedeutung. In den folgenden Abschnitten soll dargestellt werden, wie die Entwicklung von Anwendungen für die BlackBerry-Architektur von statten geht.

6.5.1 Allgemeiner Überblick BlackBerry OS

Wie bereits erwähnt, sind die Blackberries in eine Client-Server-Architektur eingebettet. Die Server sorgen dabei dafür, dass das zu übertragende Datenaufkommen, so klein wie möglich ist. Die Daten werden stark komprimiert und "häppchenweise" an das Endgerät verschickt. In Zeiten von UMTS scheint das überflüssig, aber angesichts des hohen Energiebedarfs für das Senden und Empfangen von Daten ist diese Vorgehensweise von Vorteil, da die Akkus der Geräte länger halten. Zudem kann der gesamte Datenverkehr und das BlackBerry selbst verschlüsselt werden. BlackBerry richtet sich an Unternehmen und ist optimiert für die Verwendung im Geschäftsbereich. Den daraus resultierenden Mangel an Multimedia-Fähigkeiten im Vergleich zu den anderen Smartphone-Herstellern versucht RIM mit jeder neuen Betriebssystemversion dennoch zu beseitigen, da man BlackBerry auch für den Privatkundenbereich attraktiv machen will. Die aktuelle Version ist BlackBerry OS 7, welches sich u. a. durch einen schnellen Browser, optimierte Touch-Bedienung (RIM bezeichnet das als Liquid Graphics Technology), HD-Videoaufnahmefunktion und eine Sprachsuche auszeichnet (vgl. [4]). Am 18. Oktober 2011 hat RIM auf der hauseigenen Entwicklerkonferenz DevCon bereits den Nachfolger BlackBerry BBX vorgestellt, der Cloud-Dienste integriert und sowohl für Smartphones als auch Tablets gedacht ist [5]. Für Tablets gibt es bisher eine eigene Betriebssystem-Version, das BlackBerry Tablet OS.

6.5.2 Entwickeln für BlackBerry OS

BlackBerry stellt Anwendungen eine MIDP Java-Laufzeitumgebung bereit (s. [7]). MIDP steht für Mobile Information Device Profile und bietet eine Java-Laufzeitumgebung für mobile Endgeräte. BlackBerry nutzt MIDP 2.0 in Verbindung mit CLDC 1.1 (Connected Limited Device Configuration), womit die minimalen Hardware-Voraussetzungen spezifiziert werden (s. [8], [9]). Dementsprechend werden BlackBerry-Applikationen mit Java programmiert, somit ist auf dem Entwicklungsrechner zumindest eine JRE erforderlich. Als Entwicklungsumgebung stellt RIM eine Eigenentwicklung zur Verfügung, die sogenannte BlackBerry JDE (JDE steht für Java Development Environment). Die JDE (Screenshot s. Abbildung 6.12) ist derzeit nur für das Windows-Betriebssystem verfügbar und enthält alle Tools, um unter BlackBerry OS lauffähige Anwendungen zu erstellen.
Ebenso bietet RIM Unterstützung zur Entwicklung von BlackBerry-Anwendungen mit

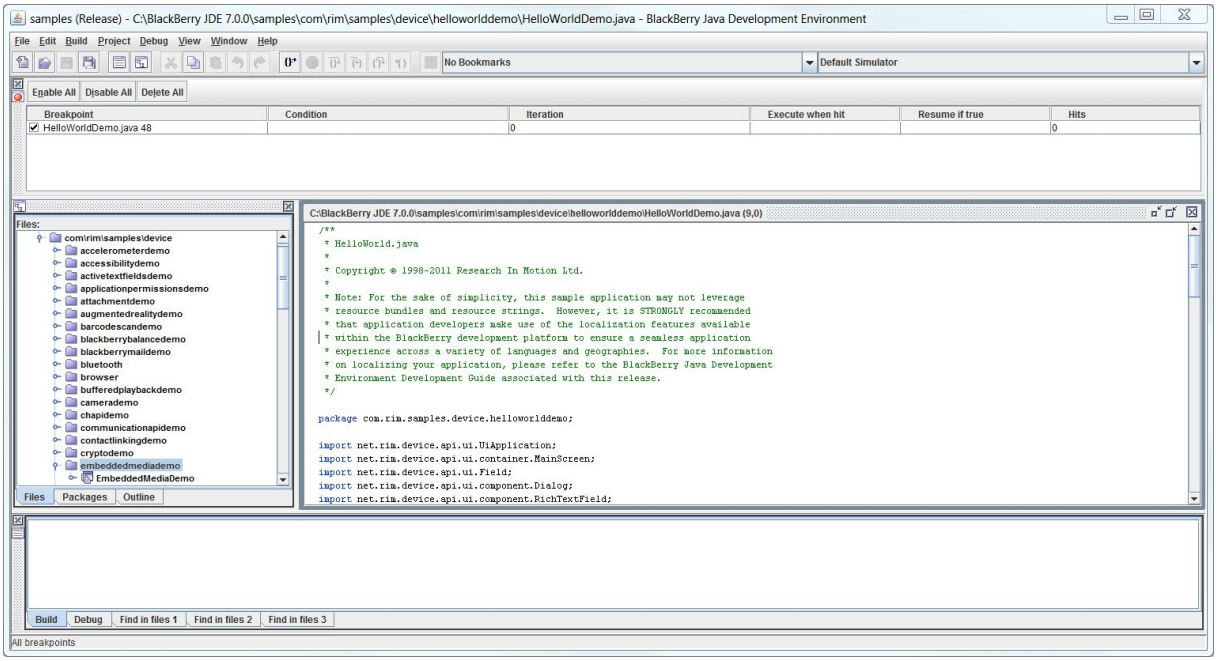

Abbildung 6.12: Screenshot der BlackBerry JDE

Hilfe von Eclipse und stellt dafür ein Plugin zum Download bereit. Damit hat man ebenfalls die volle Funktionalität zur Anwendungsentwicklung wie mit der JDE. Daneben wird auch die Entwicklung mit Netbeans unterstützt.
Allerdings gilt als Einschränkung für die Nutzung des JDEs bzw. der JDE-Plugins für die anderen Entwicklungsumgebungen, dass nur 32 Bit Java genutzt werden kann und auch nur Windows Betriebssysteme. Im Weiteren soll die Entwicklung von Anwendungen unter Eclipse dargestellt werden. Die Entwicklung von Webandwendungen für BlackBerry, die mit Tools wie Eclipse aber auch dem Visual Studio möglich ist, wird hier nicht behandelt, sondern nur die Erstellung klassischer Java-Apps.

6.5.3 Hallo BlackBerry!

Zur Entwicklung unter Eclipse werden also eine Java-Laufzeitumgebung, Eclipse und die JDE Plugins für Eclipse benötigt. Um die Plugins herunterladen zu können, muss man sich bei BlackBerry registrieren, wodurch zumindest keine Kosten entstehen. Bei der Installation des Plugins muss man auch mindestens eine BlackBerry Version auswählen, für die dann das entsprechende BlackBerry SDK mitgeliefert wird, in dem die zur Entwicklung benötigten Java-Klassen enthalten sind, um beispielsweise Zugriff auf den BlackBerry Messenger oder BlackBerry Mail zu erhalten. Weiterhin wird in [6] vorgeschlagen, die BlackBerry Desktop Software und den BlackBerry Device Manger zu installieren, um damit verschiedene Firmware-Versionen auf den Entwicklungsgeräten zu installieren oder Dateien zu transferieren.
Es gibt bei BlackBerry drei Arten von Anwendungen [6]: Midlets, CLDC Anwendungen und Libraries. Sowohl Midlets als auch CLDC-Anwendungen sind MIDP konform und ein Midlet ist technisch gesehen auch eine CLDC-Anwendung. Der Unterschied ist, das Midlets auf jedem Java-ME-Gerät (MicroEdition) laufen, aber CLDC-Anwendungen eine von RIM für das BlackBerry optimierte Midlet-Form sind und daher auch nur für BlackBerries gedacht sind. Bis auf die UI-Klassen für die Darstellung der Benutzeroberflächen können bei beiden Anwendungsarten dieselben Klassen genutzt werden. Abbildung 6.13 zeigt den Lebenszyklus und damit die drei möglichen Zustände an, in denen sich ein Midlet während seiner Laufzeit befinden kann. Die 3. Anwendungsart, die Libraries, sind keine

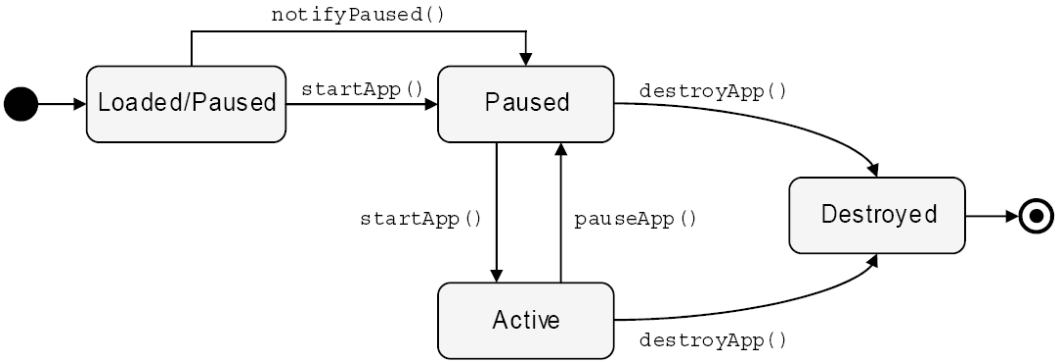

Abbildung 6.13: Midlet-Lebensyklus

Anwendungen im eigentlichen Sinne, sondern es sind Bibliotheken, in denen von mehreren Anwendungen verwendete Funktionen gebündelt werden können.

Eine Hallo-Welt-CLDC-Anwendung sieht wie folgt aus:

```
public class HalloBlackBerry extends UiApplication
{
    public void start()
    {
        MainScreen main = new MainScreen();
        LabelField label = new LabelField("Hallo BlackBerry!");
        main.add(label);
        UiApplication app = UiApplication.getUiApplication();
        app.pushScreen(main);
        app.enterEventDispatcher();
    }

    public static void main(String[] args)
    {
        (new HalloBlackBerry()).start();
    }
}
```

Im Unterschied zu einer Midlet-Anwendung muss man sich hier nicht selbst um die Lebenszyklen kümmern und zudem hat man wie in Java SE üblich eine main-Methode als Programmeinstiegspunkt.

6.5.4 BlackBerry-Simulatoren

Um die gerade geschriebene Hallo-Welt-Anwendung auszuführen, gibt es Simulatoren. Man kann zu jedem BlackBerry-Gerät einen eigenen Simulator verwenden, da es zu jeder BlackBerry-Version auch abhängig vom Netzbetreiber für ein Gerät mehrere angepasste Unterversionen gibt. Die verfügbaren Simulatoren können jederzeit in der Developer Zone [7] eingesehen und heruntergeladen werden. Danach stehen Sie in Eclipse bzw. der JDE zur Verfügung, um selbst erstellte Anwendungen zu testen. Zusätzlich wird der BlackBerry Email Simulator benötigt, wenn Email-Funktionalität getestet werden soll. Um Internetzugang, GPS oder Push-Dienste zu simulieren, ist der BlackBerry MDS Simulator erforderlich. Die Ausgabe der Programmausführung der ”Hallo BlackBerry!”-Anwendung wird in Abbildung 6.14 auf Seite 179 visualisiert.

6.5.5 Debuggen und Testen von BlackBerry-Apps

Sowohl beim Eclipse Plugin als auch bei der BlackBerry JDE ist ein JDWP konformer Debugger enthalten, sodass die bekannten Debugfunktionalitäten beim Ausführen einer Anwendung im Simulator problemlos möglich ist. Ein anderer wichtiger Aspekt bei der Fehlersuche sind Logfiles. Die einfachste Methode unter BlackBerry hierfür ist das Schreiben auf System.out bzw. System.err an den entsprechenden Programmstellen. Diese Informationen werden vom BlackBerry Logger festgehalten und können ausgelesen werden,

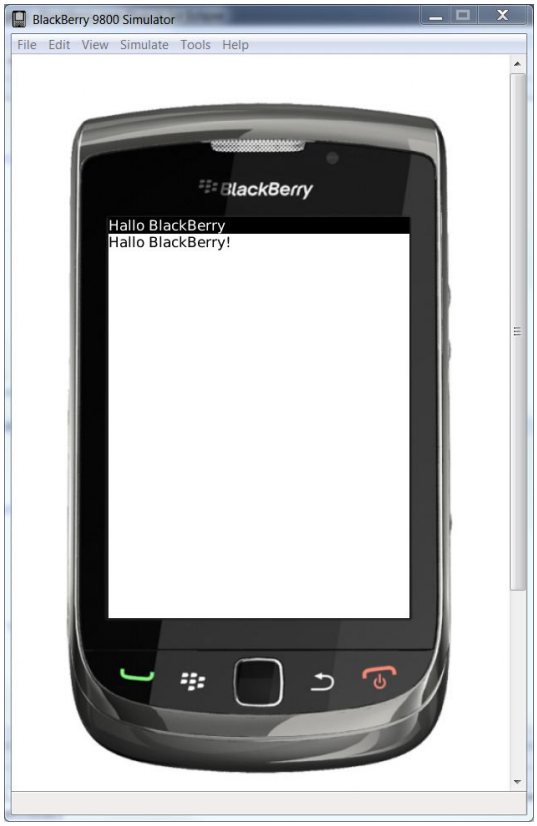

Abbildung 6.14: Ausgabe im Simulator

indem das Gerät per USB-Kabel und der Desktop-Software mit einem Computer verbunden wird. Weitere Ansätze zum Testen/Debuggen von BlackBerry Anwendungen sind in [6] zu finden.

6.5.6 Zertifikate

Um ein Programm schließlich auf einem physischen BlackBerry auszuführen, ist es unter Umständen erforderlich, sogenannte Code signing keys zu erwerben. Mittlerweile erhält man diese Schlüssel kostenlos, muss sich aber registrieren. Code signing ist erforderlich, wenn man in seiner Anwendung auf geschützte Klassen zugreift. Die Klassen, für die diese Einschränkung gilt, sind in der API-Dokumentation mit einem Schlosssymbol markiert bzw. die Methoden und Eigenschaften dieser Klassen. Es handelt sich dabei um solche Teile der API, die RIM für besonders schützenswert hält, wie kryptographische Funktionen, Nachrichten- sowie Telefonfunktionalität. Ziel ist es zum Einen, den Autor einer Anwendung identifizieren zu können und zum Anderen aber auch, um die Nutzung von entsprechend sensiblen Teilen der API zu überwachen, um bei Missbrauch (z. Bsp. Virenprogramme) den Autor ermitteln zu können. Die Tools, um seinen Quellcode zu signieren, sind in die Entwicklungsumgebung integriert.

6.6 Zusammenfassung

Die Aufgabe dieses Kapitels war es, zu zeigen, welche Konzepte zur Entwicklung von Smartphone-Anwendungen der vier bedeutendsten Smartphone-Plattformen existieren. Allen gleich war die Tatsache, das eine für die speziellen Anforderungen von mobilen Endgeräten optimierte Laufzeitumgebung - eine jeweils angepasste Java-VM im Falle von Android und BlackBerry, Cocoa Touch bei iOS und das .NET-Framework bei Windows Phone 7 - bereitgestellt wird. Zur Entwicklung von Anwendungen für die Plattform kann in allen dargestellten Technologien eine bereits existierende Entwicklungsumgebung verwendet werden, sodass diese nicht "von der grünen Wiese" her neu erstellt werden mussten. Das erleichtert den Einstieg in die Smartphone-Anwendungsentwicklung ungemein.
Ausgehend von einem Java-Entwickler dürfte die Entwicklung für Android, BlackBerry und auch WP7 einfach fallen, da C# sehr Java-ähnlich ist, schließlich diente Java als Vorbild bei der Entwicklung von C#. Bei BlackBerry allerdings sollte sich ein Entwickler zunächst umfassend mit der im Vergleich zu den anderen Architekturen komplexeren BlackBerry-Architektur beschäftigen, damit die erstellten Anwendungen von den Vorteilen von BlackBerry profitieren. Die Einarbeitung in die Entwicklung für das IPhone wiederum gestaltet sich für einen Java-Entwickler in der Hinsicht etwas aufwändiger als bei den anderen Smartphone-Technologien, weil mit Objective-C eine neue Sprache erlernt werden muss. Zudem ist man hier auf Mac angewiesen, andererseits ist die Entwicklung für WP7 und BlackBerry auch ausschließlich unter Windows möglich (von der Verwendung virtueller Maschinen abgesehen).
Festzuhalten bleibt, das alle vier Hersteller, also Apple, Research in Motion, Google (bzw. die Open-Handset-Alliance) und Microsoft versuchen, es Entwicklern so einfach wie möglich zu machen, für die jeweilige Plattform Anwendungen zu erstellen. Das macht sich zum Einen dadurch bemerkbar, dass man alle Tools kostenlos erhält, aber auch damit, das umfangreiche Dokumentation zur Verfügung steht. Das Geschäftsmodell besteht darin, das man am Verkauf der Anwendungen mit verdient, wofür man die entsprechenden Marktplätze zur Verfügung stellt und auch bereits für das Verwenden von physischer Hardware anstelle von Simulatoren/Emulatoren Geld verlangt. Als Maßnahme des Qualitätsmanagements wird zudem besonders im Falle von Apple und Microsoft, die Anwendung vor der Veröffentlichung geprüft, um Schadsoftware, aber auch "ungewünschte" Inhalte vom Markt fernzuhalten, wofür oft der Vorwurf der Zensur gemacht wurde. Genau hierin liegt somit auch eine Schwierigkeit für die vier Unternehmen: das Abwägen zwischen Offenheit und Sicherheit für den Smartphone-Nutzer.

Literaturverzeichnis

[1] Android Developer Guide:
http://developer.android.com

[2] Sayed Hashimi, Satya Komatineni, Dave MacLean:
Pro Android 2, Apress, 2010

[3] W. Frank Ableson, Robi Sen, Chris King:
Android in Action, Manning Publications, Stamford, 2011

[4] Offizielle Website von RIM zu Blackberry OS 7:
http://us.blackberry.com/apps-software/blackberry7/

[5] Bekanntgabe von BBX:
http://us.blackberry.com/newsroom/news/press/release.jsp?id=5230

[6] Chris King:
Advanced BlackBerry 6 Development, Apress, 2011

[7] BlackBerry Developer Zone:
http://www.blackberry.com/developers/

[8] CLDC 1.1 Spezifikation:
http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html

[9] MIDP 2.0 Spezifikation:
http://www.oracle.com/technetwork/java/index-jsp-138820.html

[10] Sarah Allen, Vidal Graupera, Lee Lundrigan:
Pro Smartphone Cross-Platform Development, Apress, 2010

[11] Paul Thurrot:
Windows Phone 7 Secrets, Wiley Publishing Inc., 2011

[12] Windows Phone Developer Center:
http://msdn.microsoft.com/de-de/windowsphone

[13] Nick Randolph, Christopher Fairbairn:
Professional Windows Phone 7 Application Development, Wiley Publishing Inc., 2011

[14] Charles Petzold:
Programming Windows Phone 7, Microsoft Press, 2010

[15] Microsoft Developer Network (MSDN):
http://msdn.microsoft.com

[16] Apple Developer Library:
https://developer.apple.com/library/ios/navigation/

[17] Matt Neuburg:
Programming iOS 4, O'Reilly, 2011

[18] Alexander Reeb, Christian Rolly:
Objective-C-Quickstart-Guide für Programmierer mit Java-Kentnissen, Studienarbeit an der UniBw München, HT2010

Kapitel 7

Universelle Entwicklung

Christian Rolly

Der Smartphone-Markt ist mit fünf großen Betriebssystemen - Apple iOS, Google Android, RIM Blackberry, Nokia Symbian und Microsoft Windows Phone 7 - wesentlich fragmentierter als der PC-Markt. Daher besteht großes Interesse an der universellen Entwicklung. Dies bedeutet, dass eine App nur einmal programmiert werden muss und für alle Plattformen kompiliert werden kann. Dies wird in diesem Kapitel untersucht.
Dazu wird im ersten Abschnitt zunächst zur Thematik hingeführt und die Motivation für die universelle Entwicklung erläutert. Anschließend werden die verschiedenen Ansätze der universellen Entwicklung genau beleuchtet, eh im dritten Abschnitt eine Übersicht über alle Frameworks der drei gängigsten Ansätze - Cross-Compiler, Pure-Web Apps und Hybrid Apps (Wrapper Frameworks) - gegeben wird. Im vierten Kapitel werden dann die jeweils erfolgreichsten Vertreter dieser Frameworks im Detail betrachtet und Vor- und Nachteile ausgearbeitet, eh in einem abschließenden Fazit der aktuelle Stand der universellen Entwicklung zusammengefasst wird.

Inhaltsverzeichnis

7.1 Einleitung 185
7.2 Ansätze 187
7.2.1 Native Entwicklung 187
7.2.2 Cross-Compilierung 187
7.2.3 Virtuelle Maschine 188
7.2.4 Web Apps 188
7.3 Übersicht 191
7.3.1 Cross-Compiler Frameworks 192
7.3.2 Pure Web App Frameworks 195
7.3.3 Wrapper Frameworks 199
7.4 Bewertung 202
7.4.1 Appcelerator Titanium 202
7.4.2 Rhomobile Rhodes 203
7.4.3 Sencha Touch 204
7.4.4 jQuery Mobile 205
7.4.5 Phone Gap 205
7.5 Fazit 207

7.1 Einleitung

Die Veröffentlichung des Apple iPhones im Jahr 2007, des Apple AppStore und Android Betriebssystems 2008 haben den Smartphone-Markt revolutioniert. Laut Gartner sind allein im zweiten Quartal 2011 die Verkaufszahlen von Smartphones um 74 Prozent gestiegen. Wurden 2010 insgesamt 296 Millionen Geräte verkauft, sind es 2011 bereits 467 Millionen. Gartner prognostiziert über eine Milliarde verkaufte Geräte im Jahr 2015. [1] Besondere Beachtung sollten hier jedoch nicht nur die Geräte selbst, sondern vielmehr die Distributionsplattformen erhalten. So ermöglicht Apples AppStore beispielsweise Apps mit nur einem Klick auf dem Gerät zu installieren. Somit sind auch wenig versierte Anwender nicht von der Softwarevielfalt exkludiert. Allein in Apples AppStore wurden in den letzten drei Jahren 18 Milliarden Apps gedownloadet. Eine Milliarde monatlich im letzten Jahr mit einer finanziellen Ausschüttung von bis dato $3 Milliarden an App-Entwickler.[2] Es ist also unverkennbar, dass Smartphones ein großartiges Potential für Softwareentwickler darstellen. Jedoch birgt der große Vorteil einer zentralen Distribution mit vielen potentiellen Kunden auch einen großen Nachteil. Die Apps müssen zunächst von den jeweiligen Plattformbetreibern kontrolliert werden, eh sie zum Beispiel für den Apple AppStore zugelassen werden. Im schlimmsten Fall kann das bedeuten, dass monatelange Entwicklungsarbeit umsonst war, da die App abgelehnt wird. Selbst bei erfolgreicher Annahme erhält man nur 70% des Umsatzes. Darüber hinaus ist entgegen des PC-Markts mit drei großen Betriebssystem - Windows, Mac OS, Linux - der Smartphone-Markt wesentlich fragmentierter. Aktuell gibt es fünf große mobile Betriebssysteme, die unter einander einen Marktanteil von knapp 94% ausmachen - Nokias Symbian, Googles Android, Research in Motion (RIM) Blackberry, Apple iOS und Microsoft Windows Phone 7 (Abbildung 1.1). Dabei besitzt jede Plattform eine eigene Entwicklungsumgebung (IDE), Programmiersprache (Tabelle 1.1), spezielle Schnittstellen, Limitierungen durch die Geräte selbst (Bildschirmgröße, Eingabemethoden, Richtlinien) und jeweils zentrale Distributionsplattformen. Daher müssen Apps momentan für jede Plattform separat entwickelt werden und da keine einheitliche API existiert, ist Code-Wiederverwendung ebenfalls sehr eingeschränkt. Somit ist die Entwicklung einer universellen App für alle Plattformen besonders zeit- und kostenintensiv. Eine Entwicklung für nur eines der Betriebssysteme hingegen scheint abwägig. Laut Gartner wird das führende Betriebssystem 2014 nur einen Marktanteil von 30% halten. Eine Forschung im Bereich der universellen Entwicklung sollte daher besonders interessant für Softwareentwickler sein, möchte man sicher den gesamten Markt bedienen. Neue Standards, wie HTML 5, CSS 3 und JavaScript scheinen hierbei mögliche Lösungsansätze zu bieten. Auf den folgenden Seiten werde ich zunächst die verschiedenen Ansätze der universellen Entwicklung vorstellen, eine Übersicht über alle bis dato verfügbaren Lösungen geben, besonders erfolgreiche im Detail vorstellen und abschließend alle gesammelten Informationen in einem Fazit zusammenfassend bewerten.

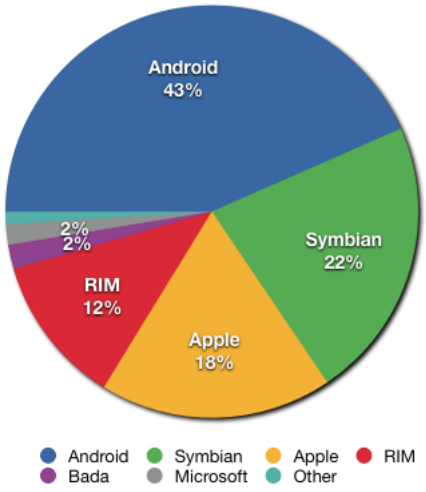

Abbildung 7.1: Plattformverteilung der Smartphones [1]

Plattform	Programmiersprache	IDE	Kosten
iOS	Objective-C	Xcode	$99 / Jahr
Android	Java	Eclipse (ADT)	einmalig $25
RIM	Java	Eclipse	$20 / App
Symbian	C++	Eclipse	$50
Windows Phone	.Net, C, C++, C#	Visual Studio	$99 / Jahr

Tabelle 7.1: Vergleich der Entwicklungsplattformen

7.2 Ansätze

7.2.1 Native Entwicklung

Eine native App ist eine auf die jeweilige Plattform angepasste Anwendung, die im Rahmen der verfügbaren APIs und Bibliotheken vollen Zugriff auf die Hardware des jeweiligen Geräts hat. Sie ist in der jeweiligen Programmierumgebung mit der dafür vorgesehen Sprache entwickelt worden und wird über die für die Plattform spezifische zentrale Stelle (Apple AppStore, Android Market, BlackBerry App World, Nokia Ovi Store, Windows Marketplace) vertrieben.

Vorteile:

- Hohe Performance
- Hardware/Software-APIs (z.B. Kamera, Adressbuch, Kompass etc.)
- Vertrieb über App Stores (Abrechnung)

Nachteile:

- Plattformabhängigkeit
- Hohe Entwicklungskosten bei universellen Apps
- Kosten für Lizenzen (SDK und Vertriebsplattform (Tabelle **??**)

Man sieht schnell, dass die native Entwicklung nicht wirklich ein Ansatz für die universelle Entwicklung ist. Eigentlich werden alle Apps separat für die eigene Plattform entwickelt und einfach in Werbekampagnen zusammengeführt für alle Plattformen vorgestellt. Dadurch entstehen sehr hohe Kosten - zeitlich und finanziell. In den letzten Jahren ist jedoch eine enorme Entwicklung auf diesem Bereich zu vernehmen. Die gängigsten Ansätze werden im Folgenden vorgestellt: [3]

7.2.2 Cross-Compilierung

Ein Cross-Compiler separiert die Entwicklungsumgebung von der Umgebung der Zielplattform. Dazu wird ein Framework bereitgestellt, welches plattformunabhängige APIs zur Verfügung stellt. Als Programmiersprache dienen Mainstream-Sprachen wie JavaScript, Ruby oder Lua. Die Entwickler können die gegebenen APIs verwenden, um eine vollständige App inklusive GUI, Datenpersistenz und Anwendungslogik zu implementieren. Der Cross-Compiler transformiert diesen Code dann in eine plattformspezifische, native App.

Vorteile:

- Hohe Performanz
- Sehr gute Usability (von nativen Apps gewohntes Bedienkonzept)
- Voller Zugriff auf Hardware (z.B. Kamera, Sensoren etc.)

Nachteile:

- Komplexität (Implementierung und Konsistenz, insb. bei Weiterentwicklung der Plattformen)

7.2.3 Virtuelle Maschine

Ein weiterer Ansatz sind Virtuelle Maschinen, welche den Programmcode für die jeweilige Plattform abstrahieren. So werden in einem Framework nicht nur API sondern auch die Laufzeitumgebung zur Verfügung gestellt. Diesen Ansatz verfolgt Java bereits seit geraumer Zeit für den PC-Markt. Vorteile, die sich hieraus ergeben, sind ganz klar die Portabilität und die Flexibilität. Insbesondere bei Erweiterungen der Geräte kann man schnell Schnittstelen schaffen. Wie sich jedoch auch auf dem PC-Markt zeigt, haben Virtuelle Maschinen meist Probleme mit der Performanz. Sehr ressourcenintensive Anwendungen sind oftmals nicht möglich. Dies ist ein umso größeres Problem für mobile Geräte, da diese meist nur sehr begrenzte Ressourcen haben. Insbesondere Prozessorleistung, Arbeitsspeicher und viel wichtiger der Akku sind in ihrer Kapazität begrenzt. Ich möchte daher im Weiteren nicht gesondert auf diesen Ansatz eingehen, sondern ihn an dieser Stelle nur der Vollständigkeit halber aufführen.

7.2.4 Web Apps

Ein zunehmend an Popularität gewinnender Ansatz ist der der Web Apps. Eine Web App ist hierbei nichts anderes als eine Webseite, die mit Hilfe von Webtechnologien, wie HTML, CSS und Javascript die GUI und Bedienung nativer Apps nachempfindet. Besonders interessant bei diesem Ansatz, ist die Tatsache, das es bei mobilen Geräte keine große Vielfalt von Browserengines gibt, wie man es von PCs gewohnt ist (Trident [Internet Explorer], Gecko [Mozilla Firefox], WebKit[Google Chrome, Apple Safari], Presto [Opera] etc.). 64% des mobilen Surfens wird mit Apples iOS getätigt, 27% mit Googles Android. Da Androids Browser jedoch auf Apples Web-Engine WebKit basiert, kann man feststellen, dass 91% aller mobilen Browser WebKit nutzen. [4] Diese Einheitlichkeit macht Web Apps also besonders interessant. Hierbei zeichnen sich jedoch wiederum zwei unterschiedliche Ansätze bei der Verwendung von Web Apps ab.

Pure Web

Bei diesem Ansatz werden die Apps im mobilen Browser geöffnet. Diese können auch offlinefähig sein und bedürfen somit nicht immer eine Internetverbindung. War früher der Zugriff auf die Hardware des Geräts nicht möglich, eröffnet HTML 5 inzwischen verschiedenste Möglichkeiten zumindest auf Teile der Gerätesensorik zuzugreifen. Die Leistungsfähigkeit verbleibt hierbei jedoch ähnlich wie bei den virtuellen Maschinen zu erörtern.

Allerdings bieten zusätzliche Frameworks, wie z.B. Sencha Touch oder jQuery Mobile eine Vielzahl von Bibliotheken und Hilfsklassen die native UI-Komponenten, Animationen und das generelle Look & Feel nachempfinden. Dadurch sind Web Apps kaum noch von nativen Apps zu unterscheiden. Vor allem die Entwicklung von Web App Stores ermöglicht wiederum den Vorteil der zentralen Distribution.

Vorteile:

- Geringe Entwicklungskosten
- Plattformunabhängigkeit
- Einfache Distribution (Veröffentlichung der Webseite)

Nachteile:

- Eingeschränkter Zugriff auf Plattformfunktionen
- Kein nativer App Store
- Teilweise schlechtere Performance

Hybrid Apps

Hybrid Apps sind eine Mischung aus Web und nativen Apps. Ziel ist es hierbei die Nachteile des beschränkten Hardwarezugriffs zu maskieren. So werden die Webseiten in nativen Apps über eine sogenannte WebView aufgerufen. Die nativen Apps bieten hierbei eine Schnittstelle zu den wichtigsten Plattformfunktionen mit Hilfe von JavaScript APIs (Wrapper Frameworks). [5] Dieser Ansatz bietet hierbei gegenüber den Web Apps den weiteren Vorteil des Zugriffs auf Plattformfunktionalitäten.

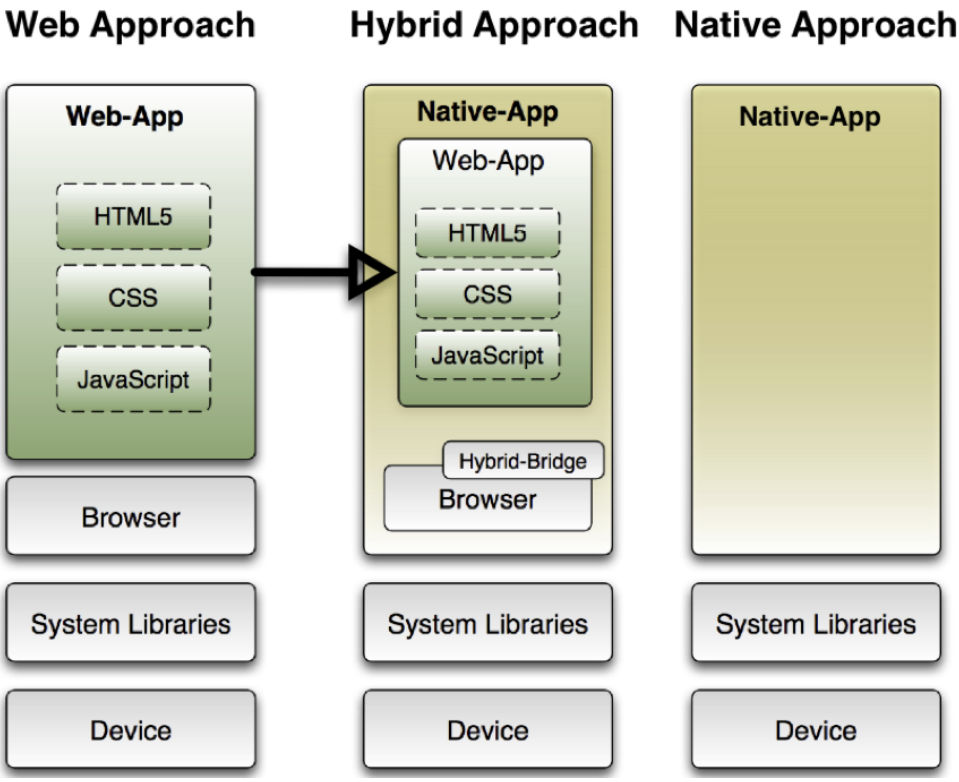

Abbildung 7.2: Architektur von Hybrid Apps [5]

7.3 Übersicht

Bis dato existieren gut zwei Dutzend Frameworks, welche versuchen universelle Entwicklung zu ermöglichen. In diesem Kapitel möchte ich eine Übersicht über diese Frameworks geben und mit Hilfe von Tabellen Unterschiede verdeutlichen. Dabei werde ich darstellen, für welche Plattformen die Frameworks überhaupt verfügbar sind, welche Sprachen verwendet bzw. benötigt werden und auf welche Gerätefunktionen zugegriffen werden kann. Dabei findet folgende Legende Anwendung:

An dieser Stelle sei ein Dank an Markus Falk gerichtet, welcher die nachfolgend aufgeführten Daten eigenhändig und in Kooperation mit den Publishern der Frameworks als Unterstützung für seine Masterarbeit zusammengetragen hat. So nutzte er diese Daten, um festzustellen, welches Framework für sein Projekt - Multiplattform-Fahrplananwendung für den Verkehrsbund Oberelbe GmbH (Nahverkehr Dresden) - am geeignetsten war. [6]

7.3.1 Cross-Compiler Frameworks

Unterstützte Plattformen

Framework	Platform (Rendering Engine)				
	iOS (Webkit)	Android (Webkit)	Windows Phone (Trident)	Blackberry OS (Webkit)	Symbian (Webkit/ Gecko)
Appcelerator Titanium Mobile					
Rhodes					
Corona					
Gideros Studio					
Moscrif					

Die Tabelle verdeutlicht den größten Nachteil der Cross-Compiler. Aufgrund der hohen Kompliziertheit, alle Fähigkeiten der einzelnen Plattformen zu bedienen, ist es besonders schwierig, einen Compiler zu entwickeln, der native Apps für die gängigsten Plattformen erzeugt. Lediglich dem Rhodes Framework der Firma Rhomobile gelingt dieser Spagat. Die anderen Frameworks setzen hauptsächlich auf den lukrativsten Absatzmarkt und unterstützen fast ausschließlich nur iOS und Android.

Programmiersprachen

Framework	Development Languages					
	PHP	Java	Ruby	Action Script	C#	Lua
Appcelerator Titanium Mobile						
Rhodes						
Corona						
Gideros Studio						
Moscrif						

Framework	Development Languages		
	HTML	CSS	Java Script
Appcelerator Titanium Mobile			
Rhodes			
Corona			
Gideros Studio			
Moscrif			

Wie bereits angesprochen, setzt der Großteil der Cross-Compiler auf alternative Programmiersprachen im Vergleich zu den gängigsten - C, C++, C#, Objective-C, Java. Insbesondere Ruby und Lua scheinen hierbei präferiert. Diese beiden interpretierten Skriptsprachen bieten besonders für den Zweck des Cross-Compiling einige Vorteile - u.a. die dynamische Typisierung, die Möglichkeit der Metaprogrammierung, die Vielzahl der unterstützten Programmierparadigmen, Plattformunabhängigkeit, geringe Größe nach der Übersetzung in Bytecode, die Erweiterbarkeit und hohe Geschwindigkeit. Zwar unterstützen die beiden bekanntesten Cross-Compiler-Vertreter auch die üblichen Web-Development-Sprachen (HTML, CSS, JavaScript), jedoch hauptsächlich für die Entwicklung der GUI, statt der tatsächlichen Programmlogik.

Hardwarezugriffe

Framework	Hardware						
	Beschleunigungssensor	Kamera	Aufnahme	Kompass	GSM / WiFi Verbindung	Adressbuch	Dateisystem
Appcelerator Titanium Mobile							
Rhodes							
Corona							
Gideros Studio							
Moscrif							

Framework	Hardware					
	GPS	Benachrichtigungen	Gesten / Multitouch	Nachrichten / Telefon	Bluetooth	Vibration
Appcelerator Titanium Mobile						
Rhodes						
Corona						
Gideros Studio						
Moscrif						

Erwartungsgemäß ist an dieser Stelle viel “Grün” zu finden. Ich habe bereits im vorhergehenden Kapitel erwähnt, dass aufgrund des Kompilierens in eine native App, auf einen Großteil der Sensorik der Smartphones zugegriffen werden kann. Es ist daher nicht überraschend, dass z.B. Rhodes alle wichtigen Gerätefunktionen nutzen kann.

7.3.2 Pure Web App Frameworks

Unterstützte Plattformen

Framework	Plattform (Web-Engine)				
	iOS (Webkit)	Android (Webkit)	Windows Phone (Trident)	Blackberry OS (Webkit)	Symbian (Webkit/ Gecko)
Sencha Touch					
jQuery Mobile					
Adobe Flex					
Zepto.js					
iUI					
CNET iPhone UI					
JQ Touch					
GWT Mobile					
mobione					
XUI					
Jo HTML5					
NS Basic / App Studio					
iWebKit					
qooxdoo					
iPhone Universial					
WebApp.net					

Web App Frameworks bieten die beste Multiplattformunterstützung. Grund hierfür ist vor allem die gleiche Web-Engine der unterschiedlichen Plattformen (WebKit). Auffällig ist an dieser Stelle vor allem die vgl. geringe Unterstützung für Windows Phone und Symbian. Dies ist wohl hauptsächlich der Tatsache geschuldet, dass Windows Phone das jüngste der mobilen Plattformen ist und die Entwickler der Frameworks hier noch etwas Zeit brauchen, um eine Unterstützung zu gewährleisten. Bei Symbian hingegen zeichnet sich ab, dass die Tage dieses Betriebssystem gezählt sind. Zwar gibt Hersteller Nokia an, nach wie vor an dem Betriebssystem festhalten zu wollen, jedoch deutet die geschlossene Kooperation mit Microsoft darauf hin, dass man in Zukunft vermehrt Smartphones mit Windows Phone 7 herstellen wird. [7]

Programmiersprachen

Framework	Development Laguages					
	PHP	Java	Ruby	Action Script	C#	Lua
Sencha Touch						
jQuery Mobile						
Adobe Flex						
Zepto.js						
iUI						
CNET iPhone UI						
JQ Touch						
GWT Mobile						
mobione						
XUI						
Jo HTML5						
NS Basic / App Studio						
iWebKit						
qooxdoo						
iPhone Universial						
WebApp.net						

Framework	Development Languages		
	HTML	CSS	Java Script
Sencha Touch			
jQuery Mobile			
Adobe Flex			
Zepto.js			
iUI			
CNET iPhone UI			
JQ Touch			
GWT Mobile			
mobione			
XUI			
Jo HTML5			
NS Basic / App Studio			
iWebKit			
qooxdoo			
iPhone Universial			
WebApp.net			

Im letzten Kapitel wurde bereits erläutert, dass die Web App Frameworks mit Hilfe von HTML5 und CSS die grafische Benutzeroberfläche der Plattformen nachahmen und der Großteil der eigentlichen Programmlogik in JavaScript geschrieben wird. Daraus ergibt sich trivialerweise die obige Verteilung zugunsten der klassischen Web-Development-Sprachen.

Hardwarezugriff

Framework	Hardware						
	Beschleunigungssensor	Kamera	Aufnahme	Kompass	GSM / WiFi Verbindung	Adressbuch	Dateisystem
Sencha Touch							
jQuery Mobile							
Adobe Flex							
Zepto.js							
iUI							
CNET iPhone UI							
JQ Touch							
GWT Mobile							
mobione							
XUI							
Jo HTML5							
NS Basic / App Studio							
iWebKit							
qooxdoo							
iPhone Universial							
WebApp.net							

Framework	Hardware					
	GPS	Benachrichtigungen	Gesten / Multitouch	Nachrichten / Telefon	Bluetooth	Vibration
Sencha Touch						
jQuery Mobile						
Adobe Flex						
Zepto.js						
iUI						
CNET iPhone UI						
JQ Touch						
GWT Mobile						
mobione						
XUI						
Jo HTML5						
NS Basic / App Studio						
iWebKit						
qooxdoo						
iPhone Universial						
WebApp.net						

Erwartungsgemäß sind Web App Frameworks nicht die richtige Wahl, wenn man auf die Sensorik der Smartphones zugreifen möchte. Zwar werden einzelne Funktionen (GPS, Multitouch) mittlerweile unterstützt, eine hundertprozentige Umsetzung ist jedoch nicht abzusehen. Hauptgrund hierfür ist mit Sicherheit auch der Sicherheitsaspekt. So ist es nicht zielführend, über eine Web-Schnittstelle vollen Zugriff auf sein Gerät zu erlauben und diese Daten durch das Internet schicken zu lassen.

Eine weitere Funktion, die hier nur begrenzt betrachtet wird, aber eine sehr große Einschränkung darstellt, ist darüber hinaus die Möglichkeit, Daten auf dem Gerät zu speichern, sollten sie beispielsweise des öfteren benötigt werden. So laufen der Großteil der Webanwendungen in Sandbox-Umgebungen, die einen begrenzten Speicher (meist 5MB) zur Verfügung stellen, mehr jedoch nicht. Hierbei ist wieder einmal die Sicherheit der bestimmende Aspekt. Apps, die unlimitiertes Nachladen von Daten erlauben, sind ein viel zu großes Sicherheitsrisiko im Hinblick auf Schadsoftware.

7.3.3 Wrapper Frameworks

Unterstützte Plattformen

Framework	Plattform (Web-Engine)				
	iOS (Webkit)	Android (Webkit)	Windows Phone (Trident)	Blackberry OS (Webkit)	Symbian (Webkit/ Gecko)
NS Basic / App Studio					
Phone Gap					
Quick Connect					
TapLynx					
OpenPlug Studio					

Ähnlich wie die Cross-Compiler Frameworks werden hauptsächlich die absatzstärksten "App-Märkte" bedient. Jedoch ist zu erkennen, dass eine größere Abdeckung in Planung ist. Vor allem Phone Gap wird schon bald (Stand: 26. Oktober 2011) auch für Windows Phone 7 volle Unterstützung bieten.

Programmiersprachen

Framework	Development Laguages					
	PHP	Java	Ruby	Action Script	C#	Lua
NS Basic / App Studio						
Phone Gap						
Quick Connect						
TapLynx						
OpenPlug Studio						

Framework	Development Laguages		
	HTML	CSS	Java Script
NS Basic / App Studio			
Phone Gap			
Quick Connect			
TapLynx			
OpenPlug Studio			

Da die Wrapper Frameworks nur eine Schnittstelle in Form einer nativem App, welche eine WebView enthält darstellen, ergibt sich, dass fast ausschließlich in Web-Development-Sprachen entwickelt wird.

Hardwarezugriff

Framework	Hardware						
	Beschleunigungssensor	Kamera	Aufnahme	Kompass	GSM / WIFI Verbindung	Adressbuch	Dateisystem
NS Basic / App Studio							
Phone Gap							
Quick Connect							
TapLynx							
OpenPlug Studio							

Framework	Hardware					
	GPS	Benachrichtigungen	Gesten / Multitouch	Nachrichten / Telefon	Bluetooth	Vibration
NS Basic / App Studio						
Phone Gap						
Quick Connect						
TapLynx						
OpenPlug Studio						

Auch an dieser Stelle bestätigen sich die im vorherigen Kapitel angesprochenen Charakteristika. So werden wesentlich mehr Gerätefunktionen unterstützt, jedoch nicht annähernd so viele, wie die Cross-Compiler Lösungen anbieten. Wrapper Frameworks scheinen also genau dann angebracht, wenn nur bestimmte Hardwarezugriffe benötigt werden, man aber trotzdem mit einer einzelnen Implementierung möglichst viele Plattformen abdecken möchte.

7.4 Bewertung

Auf den vorherigen Seiten wurden allgemein die Charakteristika der verschiedenen Ansätze im Kontext der universellen Entwicklung angesprochen. An dieser Stelle möchte ich nun noch einmal im Detail auf die aktuell jeweils erfolgreichsten Frameworks eingehen. Diese sind Appcelerator Titanium und Rhomobiles Rhodes für Cross-Compiler, Sencha Touch und jQuery Mobile für Web App Frameworks und Phone Gap für Wrapper Frameworks.

Als Grundlage dienen dazu die Ausarbeitungen von Gustavo Hartmann, Geoff Stead, Asi DeGani und Michael Jaser. [3, 5]

7.4.1 Appcelerator Titanium

Abbildung 7.3: Appcelerator Titanium Logo [8]

Seit Dezember 2008 stellt Appcelerator Titanium eine Vielzahl von Tools für die plattformübergreifende Softwareentwicklung zur Verfügung. Dabei beschränkt man sich nicht nur auf das Bedienen der mobilen Systeme, sondern stellt auch Desktoplösungen bereit. Dabei kann als Entwickler auf eine sehr gute Dokumentation zurückgegriffen werden, welche einen sehr schnellen Einstieg gewährleistet.

Das Framework basiert auf eine unabhängige JavaScript API, welche auf die einzelnen Plattformen in drei Schritten kompiliert werden kann. Erster Schritt ist die Pre-Compilation, der zweite die Front-End-Compilation und der letzte Schritt die Platform- und Package-Compilation.

- Pre-Compilation
  - Optimierung des JavaScript Codes
  - Erstellen einer Hierarchie von Abhängigkeiten aller Titanium APIs
- Front-End Compilation
  - Generierung des plattformspezifischen, nativen Codes
  - Erstellen eines evtl. benötigten nativen Projekts
  - Erstellen für Platform Compilation benötigten spezifischen Code

- Platform & Package Compilation
  - Kompilieren des Codes in native, ausführbare Dateien
  - Bündeln der Dateien für Ausführung in Simulator, auf Gerät oder für Distribution

Titanium verfolgt hierbei einen anderen Ansatz im Vergleich zu den übrigen Frameworks. Zwar ist JavaScript die Hauptsprache der Entwicklung, sowohl für den Programmcode als auch für die grafische Oberfläche, jedoch nutzt das Framework hierfür - zumindest für mobile Geräte - keine Browser Engine um die GUI zu rendern. Die API stellt eine Abstraktion aller gängigen UI-Elemente zur Verfügung, was für eine wesentlich flüssigere Bedienung sorgt.

Vorteile

- Natives Programm
- Leichter Einstieg (Entwicklungsumgebung und Framework in einfachem Installationspaket)
- Sehr gute Dokumentation
- Aktive Community zur Unterstützung des Projekts

Nachteile

- Aktuell nur wenige Plattformen unterstützt
- Zum Teil keine vollständigen Lösungen ("viele Baustellen", insb. Desktop- und Mobilbereich)

7.4.2 Rhomobile Rhodes

Abbildung 7.4: Logo der Firma Rhomobile [9]

De facto ist Rhodes eine Mischung zwischen einem Cross-Compiler und einer virtuellen Maschine. So wird der Code, welcher komplett in Ruby erstellt wird, für die verschiedenen

Plattformen separat kompiliert, ist jedoch nur in der bereitgestellten Laufzeitumgebung lauffähig. Diese wird in eine native App integriert und übernimmt die Kommunikation zwischen der App und dem Gerät. Bei der App handelt es sich jedoch nicht um eine Web App, welche mit Hilfe einer WebView dargestellt wird. Daher wird das Framework eher den Cross-Compilern zugeordnet.

Ein interessanter Ansatz des Frameworks ist die Bereitstellung objektrelationaler Abbildung, mit der man Objekte in relationalen Datenbanken ablegen kann. Dies ermöglicht eine sehr gute Persistenz und Skalierbarkeit.

Die grafische Oberfläche wird mit Hilfe von HTML, JavaScript und CSS erstellt. Dabei ist es besonders erwähnenswert, dass auch andere Frameworks wie bspw. iWebKit, XUI und Sencha Touch angebunden werden können, um ein noch besseres Bediengefühl zu erzeugen.

Darüber hinaus stellt Rhomobile auch weitere Tools zur Verfügung. Mit Hilfe von RhoSync sind geräteübergreifende Synchronisationslösungen möglich, mit RhoHub können Betatests aber auch die gesamte Distribution durchgeführt werden und mit RhoGallery gibt es Anbindungen für Cloud Management.

Vorteile

- Unterstützung aller Plattformen
- Vollständige Unterstützung des Model-View-Controller Paradigmas
- Objektrelationale Abbildung
- Kompilierung in Java Code für Java-basierte Plattformen (Android, Blackberry)

Nachteile

- Keine echtes Cross-Compiling
- Kein nativer Code → plattformspezifische Verbesserungen nicht möglich
- Basiert auf weniger populäre Programmiersprache

7.4.3 Sencha Touch

Abbildung 7.5: Logo der Firma Sencha [10]

Sencha Touch basiert auf dem JavaScript Framework ExtJS. Die Entwicklung erfolgt programmatisch in JavaScript. Das heißt, dass sämtliche Elemente der grafischen Oberfläche und die passende Logik mit JavaScript entwickelt werden und kein HTML oder CSS geschrieben werden muss. Aus den JavaScript-Klassen lässt sich schließlich HTML generieren, welches standardkonform nach HTML5 ist. Weiteren Anpassungen der Oberflächen werden durch CSS möglich. Dabei können auch eigene CSS-Klassen eigefügt werden.

Die Basis ExtJS bietet bereits eine umfangreiche Sammlung an Funktionen und erleichtert die Anwendungsentwicklung, beispielsweise durch die Unterstützung von AJAX und JSON für die Datenbeschaffung.

Wie bereits bei Rhodes angesprochen lässt sich Sencha Touch auch mit anderen Frameworks kombinieren.

7.4.4 jQuery Mobile

Abbildung 7.6: Logo der Firma Rhomobile [11]

jQuery Mobile setzt auf dem JavaScript-Framework jQuery auf und hat erst vor kurzem das Betastadium verlassen. Dabei wird es von einer sehr großen Community und insbesondere von zahlreichen, großnamigen Sponsoren (u.a. Nokia, BlackBerry, Palm) unterstützt. jQuery Mobile setzt dabei komplett auf Web-Standards. Dazu werden zahlreiche UI-Komponenten, Event-Handling für die Touch-Bedienung so wie auch Effekte und Hilfsfunktionen für AJAX-Requests bereitgestellt. Im Gegensatz zu Sencha Touch wird dabei ein deklarativer Ansatz verfolgt. Das heißt, die Programmierung erfolgt innerhalb des HTML-Markup, wodurch entsprechende JavaScript-Methoden angesteuert werden können.

7.4.5 Phone Gap

Abbildung 7.7: Logo der Firma Rhomobile [12]

Phone Gap ist ein Open Source Framework, welches Anfang 2008 veröffentlicht wurde. Es stellt einen Werkzeugkasten für das Erstellen mobiler Anwendung rein auf Basis von HTML, JavaScript und CSS dar. Basis ist hierbei wiederum eine geräteunabhängige JavaScript API, die Zugriff auf Hardware der Geräte erlaubt. Die API stellt jedoch nur sehr begrenzte Möglichkeiten zur Gestaltung der GUI bereit. Hier wurde wesentlich mehr auf die Abdeckung aller Funktionen wert gelegt, welche auf allen Geräten verfügbar sein sollten. Vor allem durch Einbinden modularer Bibliotheken gelingt dies sehr gut. Die Apps jedoch wie native Apps aussehen zu lassen, verbleibt Aufgabe des Entwicklers. Hierfür können jedoch wiederum andere Frameworks, wie XUI und Sencha Touch angebunden werden.

Erwähnenswert sind die fast nicht existierenden Strukturen oder Richtlinien, die Vorschreiben, wie die App am besten designed werden sollte. Hier werden Entwicklern alle Freiheiten gegeben, was bekanntermaßen Fluch und Segen sein kann. Erfahrene Entwickler können hier besonders leistungsstarke Apps entwickeln, Einsteiger hingegen werden jedoch auf viele Fehlerquellen stoßen. Dieser minimalistische Ansatz ist jedoch wahrscheinlich der Hauptgrund, warum so ziemlich alle existierenden Plattformen bedient werden können.

Vorteile

- Quellcode des Wrappers ist nativ für jeweilige Plattform → spezifische Änderungen möglich
- Modulare Bibliotheken
- Unterstützung aller gängigen Plattformen
- Leichter Einstieg für Webentwickler, da nur HTML, JavaScript und CSS

Nachteile

- Einbindung von WebApp durch WebView → Netzabhängigkeit
- Mangelnde Bereitstellung von GUI-Elementen

7.5 Fazit

Abschließend bleibt festzustellen, dass es zum jetzigen Zeitpunkt vielfältige Möglichkeiten für die universelle Entwicklung gibt. Vor allem die drei Ansätze des Cross-Compilings, der Pure Web App Entwicklung und der hybride Ansatz können sich zufriedener Entwickler erfreuen. Diese müssen jedoch ihre Kunden genau analysieren. So können nicht alle Funktionen mit einem Ansatz abgedeckt werden.

Hat man simple Usability Apps, welche nur schlichte Informationen darstellen, bieten sich Web Apps an. Diese können sehr schnell und vor allem plattformunabhängig entwickelt werden. Die gemeinsame Engine der meistgenutzten mobilen Browser sprechen hier einen sehr großen Absatzmarkt an.

Hat man bereits eine funktionierende Web App und möchte diese als eine Art native App veröffentlichen, um evtl. noch gewisse Sensorikzugriffe zu erhalten oder schlicht weg in einer der Distributionsplattformen präsent zu sein, so bietet sich der hybride Ansatz an. Es bedarf so gut wie keinen Programmieraufwand und kann wohlmöglich zusätzliche Kunden erschließen.

Der beste Ansatz um möglichst performante Apps plattformunabhängig zu entwickeln ist das Cross-Compiling. Leider ist dieser Ansatz auch der komplizierteste, was leider bedeutet, dass nur zwei oder drei Plattformen voll unterstützt werden. Darüber hinaus sind die cross-compilierten Apps meist nicht so performant, wie man es sich wünschen würde. Es müssen also wiederum plattformspezifische Modifikationen getätigt werden. Dies setzt wieder die entsprechende IDE so wie die entsprechenden Programmierkenntnisse voraus. Somit ist zu befürchten, dass man zwar mit einer universellen Entwicklung beginnt, jedoch schlussendlich wieder für jede Plattform separat Verbesserungen vornehmen muss.

Alles in allem steht die universelle Entwicklung also noch am Anfang ihres Weges. Diese jedoch noch sehr junge Technik hat sehr viel Potential und wird vielleicht in wenigen Jahren schon die standardisierte Vorgangsweise für die Entwicklung von Apps sein.

Literaturverzeichnis

[1] Gartner says Android to Command nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012, *http://www.gartner.com/it/page.jsp?id=1622614*, 7. April 2011

[2] Apple Special Event, *http://www.apple.com/apple-events/october-2011*, 4. Oktober 2011

[3] Cross-Platform Mobile Development, *Gustavo Hartmann, Geoff Stead, Asi DeGani*, März 2011

[4] Apple WWDC 2011 Keynote Address, *http://www.apple.com/apple-events/wwdc-2011/*, 6. Juni 2011

[5] Evaluation, Bewertung und Implementierung verschiedener Cross-Platform Development Ansätze für Mobile Internet Devices auf Basis von Web-Technologien, Bachelorarbeit, *Michael Jaser*, 21. März 2011

[6] Mobile Frameworks Comparison Chart, *Markus Falk, http://www.markus-falk.com/mobile-frameworks-comparison-chart/,* 26. Oktober 2011

[7] Nokia tut sich mit Microsoft zusammen, *heise Developer, http://www.heise.de/developer/meldung/Nokia-tut-sich-mit-Microsoft-zusammen-1187890.html,* 11. Februar 2011

[8] Appcelerator Titanium, *http://www.appcelerator.com,* 15. November 2011

[9] Rhomobile, *http://www.rhomobile.com,* 15. November 2011

[10] Sencha, *http://www.sencha.com, 15. November 2011*

[11] jQuery, *http://www.jquery.com,* 15. November 2011

[12] Phone Gap, *http://www.phonegap.com,* 15. November 2011

Kapitel 8

Smartphone-Apps auf Basis von HTML5

Christopher Rohde

Mit der Einführung von Apples iPhone wurde der Markt für Smartphones 2007 vollkommen revolutioniert. Die neuen Möglichkeiten der Nutzung mobiler Geräte waren vielfältig und so wurde vor allem der einfachere und unkompliziertere Zugriff vom Smartphone auf das Internet ermöglicht. Die Konkurrenz ließ aufgrund der großen Nachfrage nicht lange auf sich warten und so finden sich heute eine Vielzahl an unterschiedlichen Smartphones unterschiedlichster Hersteller und unterschiedlichster Betriebssysteme auf dem Markt wieder. Neben dem vom Marktführer Apple verwendeten iOS hier allen voran als härtester Apple-Konkurrent Google mit seinem Betriebssystem Android. Desweiteren folgten Hersteller wie RIM mit Blackberry oder auch der Softwaregigant Microsoft mit Windows Phone 7, aber auch kleinere wenn gleich nicht uninteressantere Systeme wie bspw. webOS von HP.

Aufgrund dieser hohen Anzahl an verfügbaren Plattformen, alle mit ihren eigenen App-Stores und eigenen Entwicklungsumgebungen für die entsprechenden Apps, gestaltet es sich als schwierig bzw. oft einfach zu teuer, bei der Entwicklung eines Programms den kompletten Markt an Betriebssystemen und somit Nutzern abzudecken bzw. zu erreichen. Mit dem kommenden Standard HTML5 stellt die Entwicklung von sogenannten Web-Apps einen möglicherweise günstigeren Weg dar, ein Programm auf den unterschiedlichen Systemen für eine Zielgruppe zur Verfügung zu stellen.

In den folgenden Abschnitten soll daher zuerst auf HTML5 genauer eingegangen werden und im Anschluss die Möglichkeiten erörtert werden, inwiefern die Entwicklung solcher Web-Apps auf Basis von HTML5 im Gegensatz zu den nativen Applikationen von Vorteil bzw. auch von Nachteil sein kann.

Inhaltsverzeichnis

8.1 Einführung 211
8.2 HTML5 212
8.2.1 Was ist neu? 213
8.2.2 Offline Web Applications 214
8.2.3 Geolocation-API 215
8.2.4 Das Canvas-Element 217
8.2.5 Multimedia Objekte 218
8.3 Native App vs. Web-App 220
8.3.1 Erstellung einer nativen Applikation 220
8.3.2 Erstellung einer Web-Applikation 220
8.3.3 Von einer Web-App zur nativen Anwendung 221
8.4 Fazit 223

Abbildungen

8.1 Logo vom kommenden Standard HTML5.
Quelle: http://www.w3.org/html/logo/ 212
8.2 Beispiel für den Aufbau einer HTML-Datei. 213
8.3 Beispiel für den Aufbau einer Manifest-Datei. 215
8.4 Beispiel für eine einfache Positionsabfrage. 216
8.5 Positionsbestimmung und Ausgabe mit HTML5. 216
8.6 HTML-Code für ein Canvas-Elements. 217
8.7 Ein Spiel visualisiert mit dem Canvas-Element.
Quelle: http://www.felinesoft.com/blog/index.php/2010/09/ 218
8.8 HTML-Code für Video- und Audio-Elemente. 218
8.9 Beispiel für einen HTML5-Audioplayer.
Quelle: http://demos.w3avenue.com/ 219
8.10 Web-Apps mit dem jQTouch-Framework. 221
8.11 Umwandlung einer Web-App mit Titanium.
Quelle: http://www.appcelerator.com/ 222

8.1 Einführung

Das iPhone verfügte als erstes Telefon standardmäßig über einen integrierten Browser, der mit den Funktionen vergleichbar eines Browsers eines Desktop-PCs ausgestattet war [1]. Hierdurch stand und steht für Anbieter von Programmen und Diensten für mobile Geräte die Möglichkeit der Wahl zwischen einer für ein Smartphone optimierten Website oder eine speziell für ein mobiles Gerät bzw. Betriebssystem entworfene Applikation. Aufgrund der beschränkten Möglichkeiten in Bezug auf Design und vor allem Interaktivität mit dem Benutzer fiel die Wahl hierbei selten auf die Web-Applikation, sodass die Entwicklung mühsam über die betriebssystem-spezifische Entwicklungsumgebung der Zielplattform erfolgen musste. Auch die Option der Nutzung der interaktiviereren Flash-Technologie von Adobe konnte und kann auch heute aufgrund fehlender Unterstützung seitens des Marktführers Apple meist nicht in Betracht gezogen werden zumal Adobe zu Gunsten von HTML5 angekündigt hat, die Entwicklung von Flash für die mobile Plattform einzustellen.

Vor diesem Hintergrund ist der neue momentan noch in der Entwicklung steckende Internetstandard HTML5 also besonders interessant und daher genauer zu betrachten, haben doch alle verfügbaren mobilen Geräte trotz aller Unterschiede wenigstens eine entscheidende Gemeinsamkeit: einen Browser. HTML5 ist für die Anwendung in Web-Applikationen für Smartphones also besonders prädestiniert, wurde und wird doch bei der Festlegung der zukünftigen Spezifikationen von HTML5 viel Wert auf eine hohe Interaktivität mit dem Benutzer gelegt, sodass die Grenzen zwischen nativem Programm und Web-Applikation weiter verschwinden sollen.

Was aber genau ist HTML5 und was macht es so besonders im Gegensatz zum aktuell noch vorherrschenden Standard HTML4? Dies ist eine wichtige Frage und soll daher zum besseren Verständnis in Bezug auf die Nutzung von HTML5 zur Smartphone-App-Entwicklung im folgenden Abschnitt genauer betrachtet werden und als Grundlage dienen, um in den anschließenden Abschnitten dieses Wissen auf die Entwicklung flexibler und nativ wirkender Programme für mobile Geräte anzuwenden.

8.2 HTML5

HTML5 ist als neuer geplanter Standard der *Hypertext Markup Language*, kurz *HTML*, ebenso wie seine Vorgänger eine Auszeichnungssprache zur Strukturierung von Inhalten wie Bildern, Texten und Hyperlinks in Dokumenten.

Abbildung 8.1: Logo vom kommenden Standard HTML5.

Unter HTML5 versteht man jedoch nicht nur einen neuen Standard von HTML, vielmehr sieht man als HTML5 auch oft die Verbindung der Webtechnologien HTML, *CSS3* sowie *Javascript*. Mit der Entwicklung von HTML allgemein wurde 1989 zur besseren Kommunikation und zum schnelleren Datenaustausch zwischen den Forschungsmitarbeitern der Europäischen Organisation für Kernforschung (CERN) begonnen. Seit dem Beginn der Entwicklung gab es mehrere gültige Versionen, an denen sich Webprogrammierer zu orientieren hatten. Die letzte und aktuell gültige Version, HTML 4.01, wurde 1999 spezifiziert. Die Sprache bildet seitdem die Grundlage für die Visualisierung von Webseiten des World Wide Web und wird von Browsern unterschiedlichster Plattformen vom Desktop-PC bis zum Smartphone dargestellt, ist also somit plattformunabhängig. Daran wird sich auch mit dem kommenden Standard HTML5 nichts ändern, was es gerade bei der Entwicklung für den diversifizierten Smartphone-Markt besonders interessant macht.

Obwohl bereits einige Elemente von HTML5 fertig umgesetzt sind und in einigen Browsern auch bereits unterstützt werden, so befindet sich die neue Spezifikation dennoch immernoch in der Entwicklung, ist also keinesfalls fertig. Das *World Wide Web Concortium*, kurz *W3C*, begann mit der Entwicklung im Jahr 2008 und eine offiziellen Verabschiedung des neuen Standard durch das W3C, auch als *Recommendation* bezeichnet, wird nicht vor 2014 erwartet.

Durch die Einbindung von Webautoren sowie Entwicklern von Browsern bei der Festlegung der Ziele des neuen Standards, also den Gruppen, die vorrangig HTML verwenden, wurden folgende Gestaltungsrichtlinien für den neuen Standard als Zielsetzungen festgelegt:

- *Kompatibilität*:
  Bereits existierender Inhalt muss weiterhin nutzbar bleiben. Desweiteren dürfen neue Elemente bestehende Inhalte nicht beeinflussen.
- *Verwendbarkeit*: Neue Elemente müssen auch sinnvoll sein und sollen keine bereits vorhandenen Funktionen ersetzen.

- *Sicherheit*:
  Neue Elemente müssen entsprechend sicher sein, d. h. bestimmte Sicherheitsaspekte müssen bei der Entwicklung berücksichtigt werden.

- *Konsistenz*:
  Elemente aus *XML*[1], welche auch in *XHTML* verwendet werden, sollen auch in HTML erlaubt werden.

- *Vereinfachung*:
  Der Zugang zur Sprache muss durch geringe Komplexität und eine genaue Beschreibung des Verhaltens aller Elemente einfach gehalten werden.

- *Universalität*:
  HTML muss auf allen Plattformen und inhaltlich multilingual verwendbar sein.

- *Zugänglichkeit*:
  Der Zugang zur Spezifikation, also zu Funktion und Inhalt, muss gewährleistet sein.

Zur Umsetzung dieser Richtlinien werden bzw. wurden bereits vorhandene Elemente der Sprache teils erweitert, teils mit neuen ergänzt, auf was im Folgenden genauer eingegangen werden soll.

8.2.1 Was ist neu?

Am grundlegenden Aufbau einer HTML-Datei soll sich auch mit HTML5 nichts ändern.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Titel der Webseite</title>
    <!-- Evtl. weitere Header-Informationen -->
  </head>
  <body>
    <p>Inhalt der Webseite</p>
  </body>
</html>
```

Abbildung 8.2: Beispiel für den Aufbau einer HTML-Datei.

Mit der Einführung von HTML5 sollen jedoch zahlreiche neue Elemente in HTML Einzug halten und vieles vereinfachen bzw. ermöglichen, was in HTML4 so noch nicht möglich war. So sollen bspw. spezielle Elemente, wie *section*, *nav*, *article* oder auch *header* und *footer*, für eine bessere Seitenstrukturierung sorgen. Desweiteren kommen neue Elemente für die Formulareingabe hinzu, hier vor allem speziellere Eingabefelder für Telefonnummern oder auch Datums- und Zeitangaben, also Felder, welche eine bestimmte formatierte

[1] Extensible Markup Language

Eingabe erwarten. Dies ermöglicht in Hinblick auf die Programmentwicklung für mobile Plattformen bereits eine bessere Optimierung von Webseiten für die Anzeige und Nutzung auf mobilen Geräten, da durch die Wahl des richtigen Eingabefeldes der Browser bereits die für die erwartete Eingabe beste Eingabetastatur anzeigen kann. Im Falle eines Telefonnummerneingabefeldes also zum Beispiel eine reine Zahlentastatur des Telefons wie sie auch beim Wählen einer Rufnummer zur Eingabe verwendet wird. Darüber hinaus erfolgt auch eine im Hinblick auf die Smartphone-App-Entwicklung besonders interessante Erweiterung in Hinblick auf das Einbinden von Audio- und Videoinhalten, wodurch die Nutzung von Flash-Objekten hierzu entfallen soll. Diese Neuerung wird ebenfalls wie das neue *Canvas*-Element zur besseren Visualisierung und Berechnung von komplexen 2D- und auch 3D-Grafiken direkt im Browser des Benutzers in den folgenden Abschnitten noch eingehender vertieft werden.

Vorerst jedoch zwei weitere wichtige Neuerungen in der HTML5-Spezifikation, die an dieser Stelle gerade in Bezug auf die Applikationsentwicklung für mobile Geräte ebenfalls nicht unerwähnt bleiben sollen: *Offline Web Applications* sowie die *Geolocation-API*.

8.2.2 Offline Web Applications

Bei mobilen Geräten ist aufgrund schwankender Signalstärken des Handynetzes eine beständige Verbindung zum Internet nicht immer gewährleistet. Dies kann zu Problemen bei Webapplikationen führen, wenn diese bspw. noch nicht komplett fertig geladen wurden. Abhilfe soll hier eine weitere Neuerung in der HTML5-Spezifikation schaffen, die sogenannten Offline Web Applications. Hierunter versteht man das Zwischenspeichern einer beliebigen HTML5-Webseite mit allen Ressourcen die für eine korrekte Anzeige und Funktion der Webseite nötig sind, um diese auch offline verfügbar zu machen.

Das Prinzip dahinter funktioniert wie folgt: Beim ersten Aufrufen einer Website lädt der Browser im Hintergrund alle in einer sogenannten *Manifest*-Datei genannten Dateien herunter. Die komplette Seite wird hiermit also auf dem Gerät gespeichert. Wird diese Webseite erneut aufgerufen, wird zunächst überprüft, ob eine aktive Verbindung zum Internet besteht. Ist dies der Fall, so wird die Webseite ganz normal angezeigt, während im Hintergrund der Browser jedoch bereits erneut neue oder geänderte Dateien herunterläd. Besteht keine Verbindung zum Internet oder sind keine Änderungen an der Webseite erkannt, so wird die zwischengespeicherte Version der Webseite angezeigt. Die Manifest-Datei ist dafür nötig, dem Browser mitzuteilen, welche Dateien genau er zu cachen hat. Diese Datei muss auf dem Server auf dem die Webseite liegt zugänglich sein und von der Webseite selbst referenziert werden. Die Webseite selber muss nicht explizit in der Manifest-Datei angegeben werden.

Zusätzlich zu den zu cachenden Ressourcen können in der Manifest-Datei auch bestimmte Schlüsselwörter verwendet werden. So werden Dateien, welche unter dem Schlüsselwort *NETWORK* stehen, vom Caching ausgeschlossen, sodass diese Dateien immer neu geladen werden. Ein weiteres Schlüsselwort stellt *FALLBACK* dar. So ausgezeichnete Dateien werden verwendet, sobald eine Ressource die unter NETWORK genannt ist, somit also neu geladen wird bei jedem Aufruf, nicht verfügbar ist bzw. geladen werden kann. Hier

```
CACHE MANIFEST

/js/functions.js
/css/style.css
/images/logo.jpg
```

Abbildung 8.3: Beispiel für den Aufbau einer Manifest-Datei.

wird dann eine entsprechende unter FALLBACK genannte Ressource stattdessen angezeigt.

Der Zugriff auf den aktuellen Zustand des Browser, also ob aktuell eine Verbindung zum Internet besteht, kann über eine JavaScript-Abfrage einfach ermittelt werden. Ebenso kann der Zugriff auf den Offlinecache auch über eine JavaScript-API erfolgen, um bspw. den aktuellen Cache-Fortschritt abzufragen oder eventuell aufgetretene Fehler beim Zwischenspeichern abzufangen.

Zur Zeit wird dieses neue Feature von HTML5 noch nicht vollständig unterstützt, auf dem iPhone ist es bereits implementiert, funktioniert allerdings noch nicht komplett. Auf Android ist es derzeit ebenfalls noch nicht nutzbar, soll jedoch in naher Zukunft realisiert werden.

Weitere Funktionen für Offline Applikationen sind, auch wenn diese nicht direkt zur HTML5-Spezifikation gehören sondern eine gesonderte W3C-Spezifikation darstellen, sogenannte *WebStorages* und *SQL-Storages*, welche wichtige Daten einer Webapplikation auf dem Gerät des Benutzers abspeichern können, um diese später wiederverwenden zu können, selbst wenn keine Internetverbindung bestehen sollte. Diese Funktionen sollen an dieser Stelle nur kurz genannt, hier aber nicht weiter vertieft, werden.

8.2.3 Geolocation-API

Die Geolocation-API ist ein neues Interface in HTML5, welches den Zugriff auf die geographische Position eines mobilen Gerätes per JavaScript ermöglichen soll. Diese Funktion ist auch bereits umfassend implementiert und auf den wichtigsten Plattformen verfügbar, bspw. auf dem iPhone und auf Android-basierten Geräten. Durch den Zugriff auf die aktuelle Position eines Nutzers ergeben sich vielfältige Anwendungsmöglichkeiten, gerade im mobilen Segment, wo der Nutzer ein Gerät meist ständig dabei hat.

Die API selbst stellt verschiedene Funktionen für die Positionsabfrage bereit, als wichtigste zum einen eine *getCurrentPosition*-Funktion und zum anderen eine *watchPosition*-Funktion. Die Funktionen unterscheiden sich im Interval der erneuten Positionsabfrage. Während die watchPosition-Funktion für mehr für Positionsänderungen genutzt wird, so stellt die getCurrentPosition-Funktion eine einfache Positionsabfrage dar. Ein Beispiel für eine einfache Positionsabfrage zeigt Abbildung 8.4. Die Funktion getCurrentPosition bekommt also eine Callback-Funktion sowie eine Funktion für eventuell auftretende Fehler als Parameter übergeben. Als letzter Parameter kommt zudem ein Array mit bestimmten

```
//Callback-Funktion, wird aufgerufen, sobald Position vorhanden ist
function showMap(position) {
var lat = position.coords.latitude;
var lon = position.coords.longitude;
}

function handleError(error) {
...
}

// Positionsabfrage
var options = {timeout:3000, maximumAge:30000, enableHighAccuracy:false};
navigator.geolocation.getCurrentPosition(showMap, handleError, options);
```

Abbildung 8.4: Beispiel für eine einfache Positionsabfrage.

Eigenschaften hinzu. Im gezeigten Beispiel sind dies *timeout*, *maximumAge* sowie *enableHighAccuracy*. Während der Wert für timeout in Millisekunden angibt, wie lange der Prozess der Positionsbestimmung maximal dauern darf, so gibt die maximumAge-Eigenschaft an, ob bereits zuvor ermittelte Positionsbestimmungen verwendet werden dürfen und falls ja, wie lange solche Werte zurückliegen dürfen, ebenfalls in Millisekunden angegeben. Als letzte Eigenschaft ist noch enableHighAccuracy zu betrachten. Da es auf mobilen Geräten mehrere Möglichkeiten zur Ortsbestimmung gibt, kann man hier zwischen der zeitintensiveren, dafür aber auch genaueren, Variante über einen eventuell verfügbaren GPS-Chip und der schnelleren, aber auch ungenaueren, Variante der Bestimmung per Wi-Fi-Netzwerk wählen.

Die nach erfolgreicher Positionsermittlung erhaltenen Werte für Längen- sowie Breitengrad können im Anschluss bspw. für die Anzeige der Position auf einer Karte, wie sie zum Beispiel von GoogleMaps zur Verfügung gestellt wird, verwendet werden.

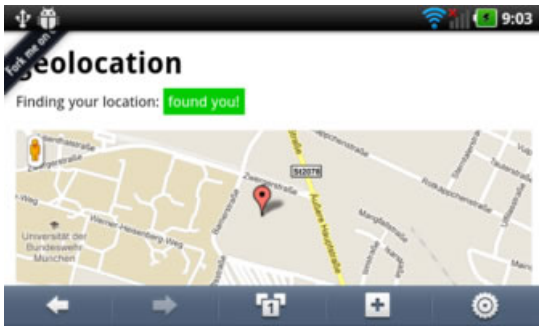

Abbildung 8.5: Positionsbestimmung und Ausgabe mit HTML5.

Ein oft genannter Kritikpunkt an ortsbasierten Diensten ist die ständige Preisgabe der eigenen Position, was auf der einen Seite natürlich einen großen Wert aus Sicht eines Entwicklers darstellt, allerdings diese Informationen auf der anderen Seite auch missbraucht werden können. Aus diesem Grund wurde bei der Implementierung dieser Funktionen sehr auf die Wahrung der Sicherheit und der Privatsphäre geachtet. Daraus folgend muss jeder

Positionsbestimmung im Browser durch den Benutzer ausdrücklich zugestimmt werden, was bei den Marktführern Apple und Google gleichermaßen durch eine einfache Dialogabfrage im Browser gelöst wird.

8.2.4 Das Canvas-Element

Eine weitere Neuerung in HTML5 ist das Canvas-Element. Es erlaubt das Erstellen von Grafiken auf einer Webseite und kann nachdem es auf einer Webseite eingefügt wurde mithilfe von Javascript angesprochen werden. Es ist also mit dem Canvas-Element möglich, während ein Nutzer eine Seite betrachtet, Änderungen an der Grafik vorzunehmen oder diese auch komplett neu berechnen zu lassen. Dies bietet eine ganze Reihe potentieller Anwendungsmöglichkeiten, z. B.:

- Finanzapplikationen, die in Echtzeit Börsencharts aktualisieren und clientseitig gerendert werden, was Serverkapazitäten für andere Rechenarbeiten freihält.
- Spiele, die den kompletten Spielinhalt direkt im Browser erzeugen.
- Zeichenprogramme ähnlich Microsoft Paint, bei denen ein erstelltes Bild dank SQL-Storage direkt auf dem Gerät abspeichert wird, wodurch die Applikation auch komplett offline genutzt werden könnte.

Das Erzeugen eines solchen Canvas-Elements erfolgt ganz einfach in gewohnter HTML-Schreibweise, wie auch in Abbildung 8.6 dargestellt.

```
<canvas>Ihr Browser kann kein HTML5.</canvas>
```

Abbildung 8.6: HTML-Code für ein Canvas-Elements.

Über die Attribute *width* und *height* kann wie in HTML üblich zusätzlich die Festlegung der gewünschten Anzeigegröße erfolgen. Weitere Anpassungen des Anzeigenlayouts erfolgen mit CSS-Attributen.

Für das Erzeugen einer Grafik in diesem Element stehen besondere Funktionen in Javascript bereit. Da das Canvas-Element nur primitive Formen, wie Rechtecke und Pfade, unterstützt, müssen alle komplexeren Visualisierungen mithilfe der entsprechenden Funktion realisiert werden. Zu den Funktionen zählen daher Methoden zum Zeichnen und Füllen von Rechtecken und Kreisen, sowie von Pfaden von einer Koordinate zur anderen. Ebenfalls sind Funktionen zur Anzeige von Text sowie ganzer Bilder implementiert. Ein gutes Beispiel, was mit dem Canvas-Element alles möglich ist, zeigt Abbildung 8.7. Dieses Beispiel ist auch unter der Internetadresse [2] zu finden. Hier findet sich zudem auch gleichzeitig ein passendes Tutorial zu diesem Thema.

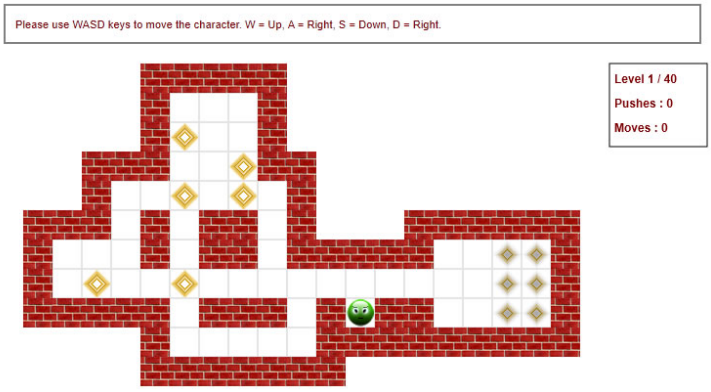

Abbildung 8.7: Ein Spiel visualisiert mit dem Canvas-Element.

8.2.5 Multimedia Objekte

Als letzte hier zu erwähnende interessante Neuerung in HTML5 sollen die neuen Möglichkeiten zur Einbindung von Audio- und Video-Dateien betrachtet werden. Diese neue Komponente soll vor allem die vorhandene Abhängigkeit von Flash-Komponenten reduzieren, weshalb gerade Apple diese Entwicklung besonders forciert, da Apple-Produkte keinen Flash-Support bieten und somit viele Anbieter, wie z. B. YouTube, ihr Angebot bereits auf die hier beschriebene HTML5-Spezifikation umgestellt haben.

An dieser Stelle sei angemerkt, dass Flash damit keinesfalls ausgedient hat, Windows Phone 7 soll laut Angaben Microsofts bspw. immernoch Flash unterstützen. Es bietet sich jedoch für die Nutzung auf mobilen Geräten einfach aufgrund der besseren Stabilität und Benutzerfreundlichkeit an, die Einbindung von Multimediainhalten nicht zuletzt auch wegen der Ankündigung Adobes, die weitere Entwicklung von Flash für die mobile Plattform einzustellen, mit den genannten HTML5-Elementen zu bevorzugen. Flash bietet allerdings auf der anderen Seite den Vorteil, das man sich bei der Verwendung von Audio- und Videoinhalten keine Gedanken um deren Format machen muss, was hingegen bei der HTML5-Variante zur Zeit noch ein Problem darstellt, da sich noch nicht auf ein einheitliches Format für die Inhalte geeinigt werden konnte und somit nicht jeder Browser jede Video- oder Audio-Datei abspielen kann. Aufgrunddessen müssen bei der Nutzung der HTML5-Elemente notfalls immer Versionen in unterschiedlichen Formaten zur Verfügung gestellt werden, was wiederum Speicherplatz kostet und somit als suboptimal bezeichnet werden kann.

Für das Einbinden der Inhalte selbst werden zwei neue Tags definiert, das *audio*- sowie das *video*-Tag. Über das src-Attribut kann dann eine Zieldatei angegeben werden, die im Player wiedergegeben werden soll.

```
<audio src="beispiel.oga" controls="controls"></audio>

<video src="film.ogv"></video>
```

Abbildung 8.8: HTML-Code für Video- und Audio-Elemente.

Mithilfe weiterer Attribute wie dem controls-Attribut lassen sich noch Festlegungen bezüglich möglicher Steuerungselemente treffen. Aber auch noch andere Attribute wie Anzeige- und Abspieloptionen existieren. Für den Zugriff auf ein audio- bzw. video-Element steht zudem wie bei canvas-Elementen eine Javascript-API zur Verfügung mit deren Hilfe das Element zusätzlich gesteuert werden kann.

Abbildung 8.9: Beispiel für einen HTML5-Audioplayer.

8.3 Native App vs. Web-App

Trotz aller genannten Neuerung und Verbesserungen in Zusammenhang mit HTML5 steht ein Entwickler von Applikationen für die mobilen Plattformen dennoch vor der Entscheidung, entweder eine native Applikation oder eine Web-Applikation zu programmieren. Für einen besseren Einblick soll daher im folgenden differenziert betrachtet werden, wie denn die eigentliche Entwicklung bei den beiden Möglichkeiten abläuft.

8.3.1 Erstellung einer nativen Applikation

Für die Erstellung einer nativen Applikation für ein Smartphone muss ein Entwickler sich zunächst für eine der vorhandenen Plattformen entscheiden, verfügen doch die verbreitesten Betriebssysteme alle über eigene *Software Development Kits*, kurz SDKs genannt. Zudem wird schon mit der Wahl einer Zielplattform bereits festgelegt, in welcher Sprache das Programm geschrieben werden muss und somit auch, welche Sprache der Entwickler beherrschen können sollte. Während zum Beispiel für iOS des Marktführers Apple Programme in *Objective-C* bzw. *C++* geschrieben werden müssen, so erfolgt das Programmieren beim direkten Konkurrent Google für die Android-Plattform in Java. Die Entwicklung selber kann dann im Anschluss nach bekannten Prinzipien des *Software Engineering* erfolgen und als Ergebnis erhält der Entwickler eine native Anwendung, die im günstigsten Fall auf genau einer Plattform optimal läuft. Eine Portierung auf eine andere Plattform erfordert meist nochmal ebenso lange wie die ursprüngliche Implementierung, zumal meist das Design auch noch an das Aussehen der neuen Zielplattform angepasst werden muss.

8.3.2 Erstellung einer Web-Applikation

Bei der Erstellung einer Web-Applikation muss der Entwickler sich auf keine bestimmte Plattform festlegen, da jedes verfügbare Smartphone über einen Browser verfügt. So kann die Entwicklung in einfachem HTML-Code erfolgen, wie bei normalen Webseiten auch. Durch die Möglichkeit den *User Agenten* des Benutzers abzufragen, können zudem unterschiedliche Stylesheets verwendet werden, um unterschiedliche Designlinien der verschiedenen Plattformen auch in der Web-App widerspiegeln zu können oder aber um verschiedene Anzeigemodi wie Portrait oder Landscape optimal für die Darstellung anzupassen. Hier existieren auch bereits einige Frameworks, welche Entwickler bei der Implementierung unterstützen und gleichzeitig CSS-Vorlagen und Javascript-Funktionen für die grafische Anpassung der Applikation an die Zielplattform zur Verfügung stellen. Dadurch wird das Erstellen der HTML-Dateien der Applikation enorm erleichtert, da nur noch die vom Framework zur Verfügung gestellten CSS-Klassen verwendet werden müssen, um das gewünschte Aussehen zu erreichen. Zwei bekannte Frameworks seien an dieser Stelle kurz erwähnt, zum einen iWebKit5 und zum anderen jQTouch. Mithilfe des Frameworks jQTouch lassen sich sogar vorgefertigte Animationen bei einem Seitenwechsel realisieren, wobei diese Funktion zur Zeit nur auf dem iPhone richtig funktioniert. Ein weiteres Framework, *Sencha Touch*, ermöglicht sogar die Definition von komlpexeren Touch-Events.

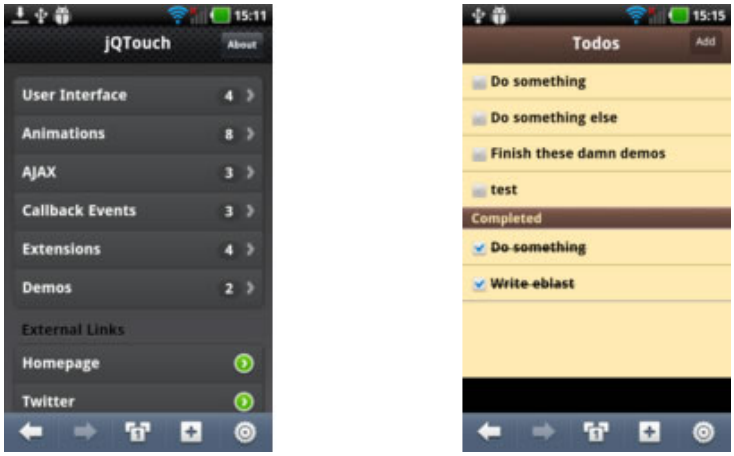

Abbildung 8.10: Web-Apps mit dem jQTouch-Framework.

Aus einer Web-Applikation lassen sich bestimmte Funktionen des Telefons direkt aufrufen. Hierzu zählen neben der Telefonfunktion auch das Versenden einer E-Mail bzw. Kurznachricht. Dafür ist die Benutzung des *tel:*-, *sms:*- und des *mailto:*-Protokolls in einem Verweis-Tag vorgesehen, um die entsprechende Anwendung aus der Webseite heraus aufzurufen. Das Aufrufen von GoogleMaps aus einer Web-Applikation heraus ist ebenfalls ohne Probleme möglich.

Speziell auf dem iPhone werden weitere Funktionen unterstützt, um eine Web-Applikation bereits wie eine native Anwendung wirken zu lassen. So kann für die Web-Applikation ein eigenes App-Icon definiert werden, welches zum Aufrufen der Web-App auf dem Home-Bildschirm hinzugefügt werden kann. Zusätzlich können Steuerungselemente des Safari-Browsers des iPhones ausgeblendet werden und sogar ein Startup-Bildschirm angegeben werden, um die Startzeit bzw. Ladezeit kürzer wirken zu lassen.

Der Vorteil einer solchen Web-Applikation liegt in dem nicht notwendigen Kompilieren der Applikation, sowie in der kürzeren Entwicklungsdauer. Mit den neuen Funktionen von HTML5 lassen sich eine große Zahl von Applikationen bereits als Web-Applikation realisieren und da Webentwickler im Allgemeinen kostengünstiger als Programmierer von Hochsprachen wie Java oder C sind, liegt in den günstigeren Entwicklungskosten ein weiterer Vorteil. Nachteilig wirkt sich zur Zeit noch die nicht durchgängige Unterstützung der verschiedenen neuen Elemente von HTML5 aus, was aber mit kommenden Browsergenerationen mehr in den Hintergrund treten wird. Features, welche zur Zeit nur von nativen Anwendungen aus nutzbar sind, sind die Bluetooth-Funktion sowie Bewegungs- und Neigungssensoren. Um diese Lücken zu schließen kann aber die im folgenden Abschnitt vorgestellte Konvertierung in eine native Anwendung helfen.

8.3.3 Von einer Web-App zur nativen Anwendung

Die Möglichkeit aus einer Web-Applikation eine native Anwendung zu generieren, unterstreicht den Nutzen von Web-Applikationen einmal mehr. Um aus einer Web-Applikation eine native Anwendung zu erzeugen, stehen verschiedene Tools im Internet zur Verfügung.

Zum einen *Titanium* vom Hersteller Appcelerator und zum anderen *PhoneGap* von Nitobi. Beide Programme liefern ein natives Programm, welches in einem AppStore vertrieben werden kann und zudem noch einen erweiterten Zugriff auf Gerätefunktionen bietet, als dies allein aus dem Browser heraus möglich wäre. Im Wesentlichen unterscheiden sich PhoneGap und Titanium in Hinblick auf die Art der Konvertierung. Während PhoneGap noch verhältnismäßig wenig nativen Code produziert, so ist eine Umwandlung mit Titanium sehr viel näher an einer richtigen nativen Applikation, produziert also mehr nativen Code. PhoneGap kann nicht nur für die Plattformen iOS und Android native Anwendungen aus Web-Applikationen erstellen, sondern ebenfalls für BlackBerry, Palm, Symbian oder Windows Mobile. Titanium auf der anderen Seite kann nicht nur für mobile Geräte native Anwendungen erzeugen, sondern ebenfalls für Windows-, Linux- oder Mac-basierte Systeme. Bei PhoneGap erleichtert die nähe zur Webtechnologie zusätzlich die Einarbeitung, da viele der gerätespezifischen APIs gleich oder zumindest ähnlich der Javascript-API sind, was bei Titanium in dieser Form nicht der Fall ist.

Ausgangspunkt für beide Programme stellt jedoch jeweils immer der HTML-Code der Web-Applikation dar.

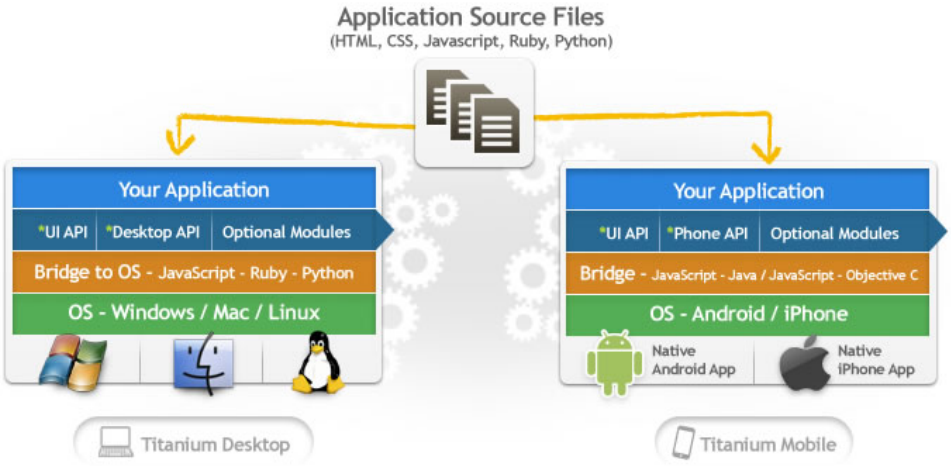

Abbildung 8.11: Umwandlung einer Web-App mit Titanium.

Als Nachteil bei beiden Tools ist zu erwähnen, dass immernoch die jeweiligen SDKs der Zielplattform benötigt werden, um letztendlich die native Anwendung aus der Web-Applikation erstellen zu können.

8.4 Fazit

In den vorangegangenen Abschnitten wurden zunächst die neuen Elemente und Vorzüge des kommenden Standards HTML5 behandelt. Die Enwicklung geht hier in die Richtung von eigenständigen Web-Applikationen, die auch ohne aktive Internetverbindung lauffähig bleiben. Die zukünftige Unabängigkeit von Flash erleichtert zunächst das benutzerfreundliche Einbinden von Multimediainhalten und erspart das clientseitige Installieren von Adobe Flash, jedoch gilt es in Bezug auf HTML5 noch abzuwarten, bis sich Standards für Dateiformate in diesem Bereich bei den Browseranbietern durchsetzen.

In den darauffolgenden Abschnitten wurden die native Applikation und die Web-App gegenübergestellt. Hierbei wurde deutlich, das in Zukunft und mit der zunehmenden Unterstützung von HTML5 viele native Applikationen bereits durch Web-Applikationen ersetzt werden können, was vor allem auch durch die damit verbundenen geringeren Kosten für die Entwicklerfirmen begünstigt wird.

Zusammenfassend bietet die Entwicklung von Web-Applikationen mit HTML5 gegenüber normalen nativen Anwendungen viele Vorteile, welche die Nutzung von Web-Applikationen gegenüber nativen Anwendungen in Zukunft begünstigen werden. Einzig die grafiklastige Spieleentwicklung für die mobilen Plattformen wird auch in naher Zukunft noch auf dem nativen Weg erfolgen, auch wenn mit dem Canvas-Element der HTML5-Spezifikation schon große Fortschritte auf diesem Gebiet gemacht wurden. Dies wird jedoch auch der einzige Bereich bleiben, in welchem native Anwendungen auf mobilen Geräten weiterhin die Mehrheit bilden.

Literaturverzeichnis

[1] Markus Spiering, Sven Haiges. *HTML5-Apps für iPhone und Android*, Franzis Verlag, Poing 2011.

[2] FelineSoftBlog. *Accelerated Game Programming with HTML5 and canvas*, `http://www.felinesoft.com/blog/index.php/2010/09/accelerated-game-programming-with-html5-and-canvas/`, Entnahmedatum: 07.09.2011.

[3] Johan Harjono, Gloria Ng, Ding Kong, Jimmy Lo. *Building Smarter Web Applications with HTML5*, CASCON '10: Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research.

[4] Appcelerator Inc. *Titanium makes cross-platform native application development easy*, `http://www.appcelerator.com/products/titanium-cross-platform-application-development/`, Entnahmedatum: 07.09.2011.

[5] Kathleen Buettner und Anna M. Simmons. *Mobile Web and Native Apps: How One Team Found a Happy Medium*, Lecture Notes in Computer Science, 2011, Volume 6769, Design, User Experience, and Usability. Theory, Methods, Tools and Practice, Seiten 549-554.

Kapitel 9

3D-Grafik

Christian Haupt

Ein zunehmend wichtiger und kommerziell sehr interessanter Bereich des Smartphone-Einsatzes ist das Darstellen dreidimensionaler Inhalte. Gerade im Unterhaltungsbereich sind durch die schnell wachsende Leistungsfähigkeit der mobilen Grafikprozessoren Anwendungsfelder entstanden, die so vor wenigen Jahren noch nicht abzusehen waren. Doch auch der professionelle Umgang mit den Möglichkeiten, die Smartphones hier zu bieten haben, hat stark zugenommen. So ist der Gebrauch im Bildungsbereich, zum Beispiel bei der Veranschaulichung komplexer Lehr-Inhalte, schon Realität. Auch der Einsatz im Militär wird besonders von der U.S. Army angestrebt und birgt große Potentiale, beispielsweise in der Visualisierung von Einsatzlagen. Dieses Kapitel soll sich mit den für die Darstellung von 3D-Szenen auf dem Smartphone notwendigen technischen Grundlagen auseinandersetzen. Es wird erläutert, welche 3D-APIs zum Einsatz kommen und wie sich diese entwickelt haben. Darüber hinaus liegt das Hauptaugenmerk auf den Unterschieden zu APIs auf dem PC, für deren Verständnis aber im Vorhinein die Funktionsweise und der Aufbau einer solchen Spezifikation veranschaulicht werden sollen. Abschließend werden Praxisbeispiele gegeben, wie diese Techniken für Applikationen im professionellen Bereich, z.B. im Militär, bereits umgesetzt wurden und in Zukunft genutzt werden können.

Inhaltsverzeichnis

9.1 Die Grundlagen 227
9.1.1 Das Rendering 228
9.2 Der Standard für mobile 3D-Grafik - Open Graphics Library for Embedded Systems 231
9.2.1 Die geschichtliche Entwicklung der mobilen Grafik 231
9.2.2 OpenGL und OpenGL ES - Die Unterschiede 233
9.2.3 OpenGL ES 1.0 und OpenGL 1.3 234
9.2.4 OpenGL ES 1.1 237
9.2.5 OpenGL ES 2.0 238
9.3 Die Konkurrenz zu OpenGL ES - DirectX 238
9.3.1 DirectX auf Windows Phone 7 und auf dem PC 239
9.3.2 OpenGL ES & Direct3D 9.0 - Das Fazit 240
9.4 Anwendungsbeispiele für die professionelle Nutzung von 3D-APIs auf Smartphones 241
9.4.1 ImageVis3D Mobile 242
9.4.2 Lageinformationssystem für den Kriseneinsatz 244
9.4.3 Ausblick 246

9.1 Die Grundlagen

Vor ein paar Jahren noch schien der Einsatz von in Echtzeit gerenderten dreidimensional (3D)-Grafiken auf den mobilen Geräten reine Zukunftsvision zu sein. Handys hatten zu dieser Zeit kleine monochrome Displays, Batterien, die im laufenden Betrieb nur wenige Stunden durchhielten, sowie – im Vergleich zu heute – langsame Prozessoren. Auch die Speicherkapazität sowohl des Hauptspeichers als auch der verbauten Massenspeicher war sehr gering.

Doch diese Situation änderte sich rasant. Innerhalb weniger Jahre entwickelten sich Handys von reinen Kommunikationsgeräten – ca. seit dem Jahr 2002 – zu mobilen Computern weiter, die heutzutage nicht umsonst den Namen *Smartphone* tragen. Smartphones sind vielseitig einsetzbar und vereinen neben Selbstorganisationswerkzeugen und Internet Browsern noch eine Vielzahl anderer Funktionen auf ein und derselben Plattform. Besonders hervorzuheben ist ihr stetig gewachsenes Leistungspotential im Bereich der Generierung und Ausgabe von Bildern virtueller Szenen, dem sogenannten *Rendering* [1].

Grundlage dieses Vorgangs ist auf heutigen Smartphones die Grafikkarte - im Folgenden auch Graphics Processing Unit (GPU) genannt - und ein 3D-Application Programming Interface, zu deutsch Programmierschnittstelle (API), welches die Kommunikation zwischen der Anwendung, zum Beispiel einer Spiele-App, und der GPU ermöglicht.

Eine API ist dabei erst einmal nur eine Vielzahl von Regeln und Funktionen. Diese müssen für ein bestimmtes System implementiert werden und können daraufhin vom Programmierer genutzt werden, um mit einer bestimmten anderen Software- oder Hardware-Komponente zu kommunizieren. Er muss sich nur noch mit der API auseinandersetzen, um z.B. auf vielen Systemen seine Anwendung zum Laufen zu bringen, und braucht nicht für jedes System einzeln die Kommunikationsschnittstellen zur Hardware herzustellen.

In Bezug auf die 3D-Grafik unterscheidet man zwei Arten von APIs, die sogenannten *'High-Level'* und die *'Low-Level'* APIs. Letztere unterscheiden sich vor allem dadurch von High-Level APIs, dass sie nur grundlegende geometrische Formen wie Punkte und Dreiecke verarbeiten können, nur eine relativ kleine Auswahl an Datentypen unterstützen und nur begrenzte Möglichkeiten zum Laden von zweidimensional (2D)-Bilddatein oder 3D-Modellen bieten. Sie sind aber durch ihr nahe Anbindung an die Hardware in heutigen Computersystemen und besonders beim Darstellen von 3D-Inhalten unersetzlich [25]. In diesem Kapitel soll es deswegen um die auf Smartphones eingesetzten Low-Level APIs gehen.

Aufgrund der immer noch eingeschränkten Hardwareausstattung eines Smartphones im Vergleich zu traditionellen Computern, was zum Beispiel die Leistungsfähigkeit der GPU betrifft, kann eine API nicht den gleichen Funktionsumfang zur Verfügung stellen wie eine entsprechende Spezifikation auf dem Computer.

Die Unterschiede, die hierbei gemacht werden, sollen im Folgenden für die beiden großen APIs im mobilen Bereich analysiert werden. Um sich aber mit dieser Materie beschäftigen zu können, müssen zuerst die technischen Grundlagen im Bereich des Renderings verstanden worden sein. Deswegen folgt zunächst eine kurze Einführung in den Prozess der Darstellung von virtuellen 3D-Szenen im Allgemeinen.

9.1.1 Das Rendering

Der Begriff des *Renderings* bezeichnet die Berechnung eines Bildes aus einer virtuellen 3D-Szene [1], welches danach auf dem Bildschirm des Gerätes ausgegeben werden kann. Hierfür werden die Daten von der 3D-Applikation an die API weitergegeben. Sie durchlaufen dann die sogenannte *Rendering Pipeline*, die die eigentliche Berechnung des Bildes unter Zuhilfenahme der Grafikkarte durchführt. Abbildung 9.1 zeigt die Rendering Pipeline der Open Graphics Library for Embedded System (OpenGL ES), einer weit verbreiteten 3D-API im mobilen Bereich. Da alle großen APIs, sei es für Smartphones oder für traditionelle Computer, eine solche oder eine ähnliche Pipeline implementieren, soll am Beispiel von OpenGL ES erläutert werden, wie der Rendering Prozess abläuft.

Wenn bis jetzt der Begriff *Daten* gefallen ist, die auf dem Bildschirm angezeigt werden sollen, so waren damit immer Geometrie-Daten z.B. folgender Form gemeint:

```
GLfloat vertexArray[] = { -0.5f, -0.5f, 0.0f,
                           0.5f, -0.5f, 0.0f,
                           0.0f,  0.5f, 0.0f };
glVertexPointer(3, GL_FLOAT, 0, vertexArray);
glDrawArrays(GL_ TRIANGLES, 0, 3);
```

Dieser OpenGL ES Befehl erstellt drei *Vertices* (Singular: *Vertex*). Dies sind Punkte im virtuellen dreidimensionalen Raum, die über je drei Koordinaten festgelegt werden. Danach werden sie mit Linien zu einem Dreieck verbunden, einem sogenannten *Polygon*. Diese drei geometrischen Formen – Punkt, Strecke und Dreieck – sind die Grundbausteine einer jeden virtuellen 3D-Szene. Aus ihnen werden alle Objekte modelliert, die der Nutzer später auf dem Bildschirm sehen kann. Hierbei werden alle Informationen über die Vertices eines Objektes in einer Matrix gespeichert. Jedes Vertex erhält neben seiner Position im Raum noch Farbwerte, einen *Normal Vektor*, der in die Richtung zeigt in der die Oberfläche des Vertex dargestellt werden soll, einen mit *Alpha* bezeichneten Transparenz-Wert und weitere Attribute [25].

Zum Anzeigen der so modellierten Objekte werden die Vertices dann an die API weitergegeben. Jetzt beginnt der Durchlauf der Vertices durch die Rendering Pipeline.

Im ersten Schritt der Pipeline – dem **'Primitive Processing'** werden die Vertex-Daten zu sogenannten **'Vertex Buffer Objects'** zusammengefasst. Diese machen einen bestimmten Teil eines Objektes aus – z.B. das Bein eines Läufers in der Szene – und werden im Arbeitsspeicher der Grafikkarte gehalten, was einen enormen Geschwindigkeitsvorteil gegenüber Objekten im Hauptspeicher des Systems bringt. Außerdem finden hier unterschiedliche Optimierungen der eingehenden Vertex-Daten statt, die eine schnelle Verarbeitung erleichtern.

An dieser Stelle kann es auf zwei Wegen weiter gehen, je nachdem, ob die Pipeline sogenannte *Shader* unterstützt und diese vom Programmierer genutzt werden oder nicht. Shader sind in diesem Fall Software-Module, die bestimmte benutzerdefinierte Rendering-Effekte bei der 3D-Computergrafik implementieren.

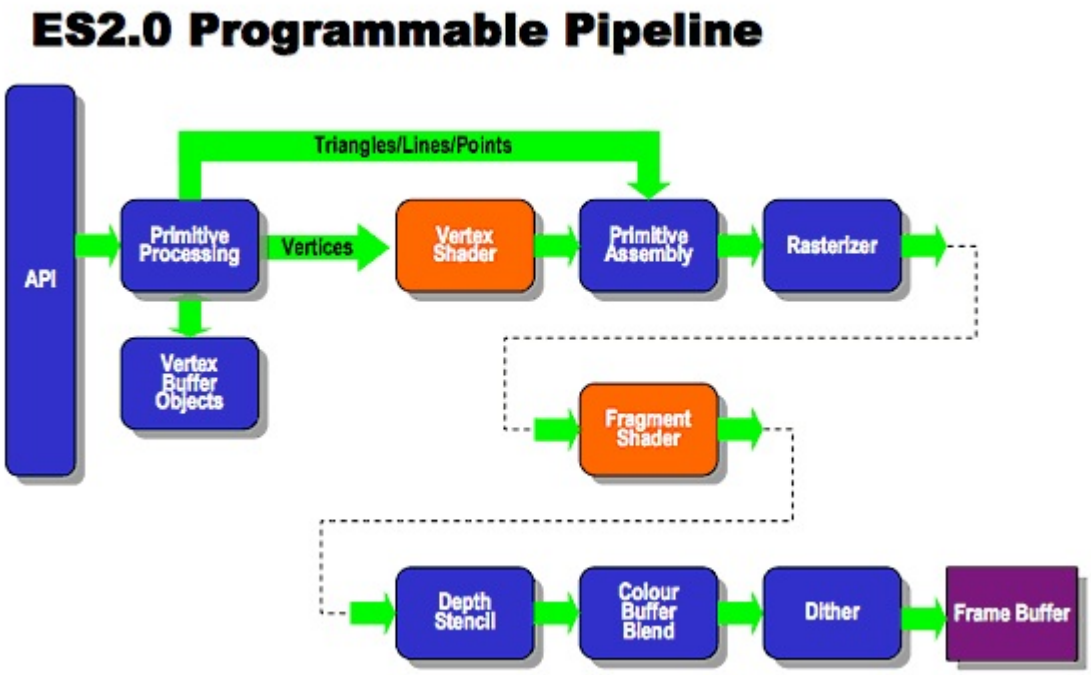

Abbildung 9.1: Die OpenGL ES 2. Rendering-Pipeline
Quelle: Khronos Group (2011), [19], `http://www.khronos.org/opengles/2_X/`

Die Abbildung 9.1 zeigt eine sogenannte „*programmierbare Shader Pipeline*“. In dieser werden zwei Arten von Shadern unterstützt, die *Vertex* und die *Fragment Shader*. Diese Art der Pipeline Architektur ist aber noch relativ neu und wird im mobilen Bereich erst seit ca. 2007 - eingeführt mit OpenGL ES 2.0 – unterstützt. Vorher war der Programmierer an die „*Fixed Function Pipeline*“ gebunden, die keine Shader unterstützte und jeden Rendering Schritt genau vorgab. Es konnte nur durch wenige einstellbare Parameter auf das Rendering Einfluss genommen werden. Somit waren auch nur wenige Effekte umsetzbar [25].

Nutzt der Programmierer an dieser Stelle den „**Vertex Shader**“ kann er die Attribute der eingehenden Vertices zur Laufzeit der Anwendung variieren. Er kann beispielsweise ihre Position im Raum in unterschiedlichen Situationen auf bestimmte Weise verändern. Somit hat er völlig freie Hand beim Entwurf von Effekten für seine Applikation, wie z.B. Gesichtsanimationen oder virtuelles Fell. Nutzt der Programmierer die Shader hingegen nicht werden die Vertices so verarbeitet wie in der Fixed Function Pipeline.

In diesem Schritt wird auch die Beleuchtung der einzelnen Vertices berechnet sowie deren Position mit mehreren Matrixmultiplikationen und Transformationsmatrizen in das *Welt-Koordinatensystem* überführt. Dies ist nötig, da Objekte in 3D-Applikationen im sogenannten *Objekt-Koordinatensystem* modelliert werden. Jedes in einer Szene auftauchende Objekt wird dann in das Koordinatensystem der 3D-Welt gebracht. So ist sichergestellt, dass ein Objekt beispielsweise mehrfach an unterschiedlichen Positionen der Welt verwendet werden kann.

Zusätzlich sind weitere Schritte nötig um den Ausschnitt der 3D-Welt, die der Nutzer auf seinem Bildschirm sehen soll, anzuzeigen.

Als nächstes müssen die Objekte einer Szene in das *Bild-Koordinatensystem* des virtuellen Betrachters der Szene überführt werden. Er ist das Auge des Nutzers einer 3D-Applikation in dieser Welt. Was er sieht bekommt, auch der Nutzer nach dem Rendering zu sehen. Da er aber nur einen rechteckigen Ausschnitt der 3D-Welt sieht, zu vergleichen mit dem menschlichen Blickfeld, findet hier das *Culling* und *Clipping* statt. Das *Clipping* sorgt dafür, dass keine Vertices mit in das Bild-Koordinatensystem übernommen werden, die außerhalb des Sichtbereiches des Betrachters liegen. Das *Culling* dagegen entfernt alle Vertices, die durch andere geometrische Formen im Raum verdeckt werden, und testet den *Normal Vektor* jedes Vertices. Weist dieser vom Betrachter weg, so kann in den meisten Fällen das Vertex entfernt werden, da es z.B. die Rückseite eines Objektes abbildet [4].

Zum Schluss sorgt eine letzte Transformation für den Übergang der Bild-Koordinaten-Matrix in das *Geräte-Koordinatensystem*. Dieses entspricht dem Bild, das der Nutzer auf dem Bildschirm sieht. An dieser Stelle ist aus den 3D-Objekt-Daten ein zweidimensionales Bild entstanden.

Die folgende Stufe der Pipeline ist das „**Primitive Assembly**“. An diese werden sowohl die transformierten Vertices als auch deren Verknüpfungsdaten übergeben. So können aus den einzelnen noch allein im Raum stehenden Vertices Polygone gemacht werden.

An dieser Stelle liegen nun die korrekten Attribute wie Farbe und Position für die Vertices vor. Das Bild ist bereits in einer zweidimensionalen Darstellung und die Polygone wurden eingezeichnet. Da Computer-Bildschirme – und Smartphones bilden dabei keine Ausnahme – aber jedes Bild in einzelnen Bildpunkten sogenannten Pixeln darstellen, muss für jedes dieser Pixel ein genauer Farbwert bestimmt werden. Diese Aufgabe übernimmt der „**Rasterizer**“. Er bringt jedoch noch keine Pixel hervor, sondern sogenannte *Fragmente*. Diese werden erst Pixel genannt sobald sie die restlichen Stufen der Pipeline überstanden haben und wirklich auf dem Bildschirm angezeigt werden [31].

Im sich daran anschließenden „**Fragment Shader**“ können vom Programmierer wieder eigene Effekte für die Fragmente eines Bildes definiert werden. Er übernimmt also die Funktion des Vertex Shaders für die Fragmente und ermöglicht so z.B. andere oft realistischere Beleuchtungsmodelle.

Die nächsten drei Schritte der Pipeline dienen der Bearbeitung des bis jetzt enstandenen Bildes und sorgen zum Beispiel mit dem „**Dither**“ für fließende Farbübergänge. Es soll hier aber nicht weiter auf sie eingegangen werden, da sie für die folgenden Betrachtungen nicht ausschlaggebend sind.

Im „**Frame Buffer**“ wird zum Schluss das fertig gerenderte Bild gespeichert und kann auf dem Bildschirm ausgegeben werden [31].

Nachdem nun das grundlegende Verständnis geschaffen wurde, was eine API ist und wie der Rendering Prozess mit ihrer Hilfe auf dem Computer abläuft, wird sich der nächste Abschnitt nun mit den beiden im mobilen Bereich großen APIs beschäftigen. Zuerst wird dabei auf OpenGL ES eingegangen. Für mobile Geräte ist diese API die am weitesten

verbreitete. Da sie sich von der im Festrechner-Bereich wichtigen Schnittstelle Open Graphics Library (OpenGL) ableitet, sollen vorher ein paar Sätze deren Historie erläutern. Hinterher folgt ein kurzer Überblick über die Entwicklungsgeschichte der 3D-Grafik auf Handys.

9.2 Der Standard für mobile 3D-Grafik - Open Graphics Library for Embedded Systems

OpenGL ist die API, die den Computer in Bezug auf die Darstellung von dreidimensionalen Bildern revolutioniert hat. Sie sorgte Anfang der 90er Jahre des letzten Jahrhunderts für eine einheitliche Programmierschnittstelle, die Anwendungsentwickler ansprechen konnten, um die Grafik-Hardware so gut wie jedes Systems zu nutzen. Diese vereinheitlichte Schnittstelle war der Grund für einen Entwicklungsboom auf Softwareseite und den darauf folgenden rasanten Hardwarefortschritt im Bereich der GPU aufgrund steigender Konsumenten-Nachfrage [21].

Die Khronos Group ist seit 2006 für die Weiterentwicklung von OpenGL verantwortlich. Sie ist ein non-Profit Industrie-Konsortium, das sich aus über 100 Unternehmen – wie *NVIDIA, Google, AMD* und vielen anderen – zusammensetzt. Ziel der Khronos Group ist die Erstellung offener APIs zur Bearbeitung und Wiedergabe von Medien aller Art auf unterschiedlichen Plattformen und Geräten [19]. Von ihr wurde OpenGL zu einer der beiden führenden Schnittstellen auf diesem Gebiet ausgebaut und ist heute für alle wichtigen Betriebssysteme implementiert.

Neben dem technischen Fortschritt im Personal Computer (PC)-Bereich entwickelte sich auch die Leistungsfähigkeit von mobilen Geräten wie Handys, tragbaren Spielekonsolen oder *Personal Digital Assistants (PDAs)* im letzten Jahrzehnt rasant weiter. Dies bewog die Khronos Group dazu, eine offene API zu entwickeln, die das Erstellen von 3D-Grafik Anwendungen auf – im Vergleich zu stationären Rechnern – leistungsschwachen, unzureichend mit Energie versorgten oder anderweitig eingeschränkten Geräten vereinfachen und standardisieren sollte. Die API wurde 2003 unter dem Namen *OpenGL ES* veröffentlicht [25].

9.2.1 Die geschichtliche Entwicklung der mobilen Grafik

Die Geschichte der 3D-Grafik auf Handys ist noch relativ jung, kann aber trotz ihres kurzen Bestehens schon auf beträchtliche Entwicklungserfolge zurück blicken. 2001 betrat im Hightech-Land Japan mit der *Mascot Capsule Engine Micro3D* der *HI Corporation* die erste kommerzielle 3D-API für Handys den Endkonsumentenmarkt. Der japanische Mobilfunkanbieter J-Phone setzte die API auf dem *Sharp SH-51* ein und bot gleichzeitig einen Downloadservice für 3D-Anwendungen an. Die Versionen 1 und 2 dieser API wiesen rudimentäre Fähigkeiten zum Rendern von einfachen 3D-Szene auf. Grundlegende Eigenschaften waren ein Tiefensortierungs-Algorithmus, der Polygone nach der Z-Koordinate

(Tiefe) sortiert auf dem Bildschirm ausgab, und eine orthografische Blickwinkelprojektion. Beide entsprachen nicht den aktuellen Industriestandards zur damaligen Zeit, waren aber Voraussetzung für das Rendering auf Handys [31].

Obwohl schon bald nach Erscheinen des *SH-51* alle anderen Mobilfunkanbieter Japans mit ähnlichen Angeboten nachzogen, dauerte es noch ein Jahr, bis auf dem europäischen Markt mit dem Nokia *3410* ein ähnliches Produkt aufkam. Das *3410* konnte aber wegen seines monochromen nur 96 zu 56 Pixel auflösenden Bildschirms in keiner Weise mit dem japanischen Angebot mithalten. Es brachte aber rudimentäre Rendering-Fähigkeiten für 3D-Szenen mit, welche u.a. zum Darstellen dreidimensionaler Texte als Bildschirmschoner benutzt wurden [31].

Während sich in Japan mit der Micro3D API ein de-facto Standard herausbildete, war der Rest der Welt noch weit davon entfernt. Zu dieser Zeit schuf fast jeder Mobilfunkanbieter, vor allem aus Differenzierungsgründen, eine eigene zu anderen Herstellern meist inkompatible Software-Umgebung zum Erstellen und Darstellen von 3D-Szenen auf Handys.

Da man jedoch aus der Geschichte der standardisierten PC APIs gelernt hatte, konnte ein Schritt in Richtung Vereinheitlichung der Herstellerbemühungen schneller und gezielter getan werden als noch ca. 10 Jahre zuvor auf dem Computer Markt.

Schon 2002 erkannte die Khronos Group, dass der Mobilfunkindustrie ein starkes Wachstum bevorstand und sich das Konsumenteninteresse dabei nicht auf die reine Telefonie beschränken würde. Mindestens genauso wichtig sollte die mobile Unterhaltung in Form von Musik, Filmen und natürlich auch Spielen werden, wie man es in etwa vom Nintendo *Gameboy* gewohnt war. Um den zersplitterten 3D-Markt im Mobilfunkbereich zusammenzuführen sowie eine schnellere und einfachere Entwicklung von Produkten zu fördern, setzte die Khronos Group im selben Jahr ein Komitee ein [2]. Dieses Komitee, bei seiner ersten Sitzung bestehend aus *"3d4W, 3Dlabs, ARM, ATI, Imagination Technologies, Motorola, Nokia, Seaweed, Silicon Graphics Incorporated (SGI), und Texas Instruments"* [31], war dazu aufgerufen, auf Basis der damals aktuellen OpenGL Version 1.3 ein eigenes Set an Funktionen zu entwickeln, das Flexibilität und Schnelligkeit bei der Darstellung von 3D-Inhalten auf Geräten mit geringer Rechenleistung und Speicherkapazität sicherstellen sollte.

Die API mit Namen *Open Graphics Library for Embedded Systems* 1.0 wurde noch vor Ende des Jahres 2003 in Kalifornien veröffentlicht und im darauf folgenden Jahr waren erste Geräte mit dem implementierten Standard auf dem Endkonsumentenmarkt verfügbar [24].

Die stark wachsende Nachfrage nach einer leistungsfähigeren 3D-API mit größerem Funktionsumfang für mobile Geräte gab den Anstoß zur schnell anschließenden Entwicklung von OpenGL ES 1.1, das 2004 veröffentlicht wurde. Diese Version nahm etwas Abstand von dem auf das absolut Notwendigste getrimmten Release 1.0 und verlangte vom Gerät mehr Leistung, unterstützte dafür aber auch auf die Grafikdarstellung spezialisierte Chips. Da sich jede Version von OpenGL ES an der Fähigkeiten-Liste von OpenGL orientierte, welche in ihrer Entwicklung einige Jahre Vorsprung hat, wurde OpenGL ES 1.1 an die Version 1.5 des OpenGL Standards angelehnt [30].

Seit März 2007 ist OpenGL ES 2.0 verfügbar. Dieses wurde gegen OpenGL 2.0 spezifiziert und ist eine komplette Überarbeitung von OpenGL ES, welche nicht zu Versionen 1.x. kompatibel ist.

Zur besseren Übersichtlichkeit ist in Abbildung 9.2 die Entwicklungsgeschichte von OpenGL und OpenGL ES noch einmal in einem Zeitstrahl zusammengefasst, in Blau dabei die zu OpenGL gehörenden Entwicklungsschritte und in Violett die zu OpenGL ES gehörigen. Mit den violetten Pfeilen wird angedeutet gegen welche Version von OpenGL die einzelnen OpenGL ES Versionen spezifiziert wurden [16], [18].

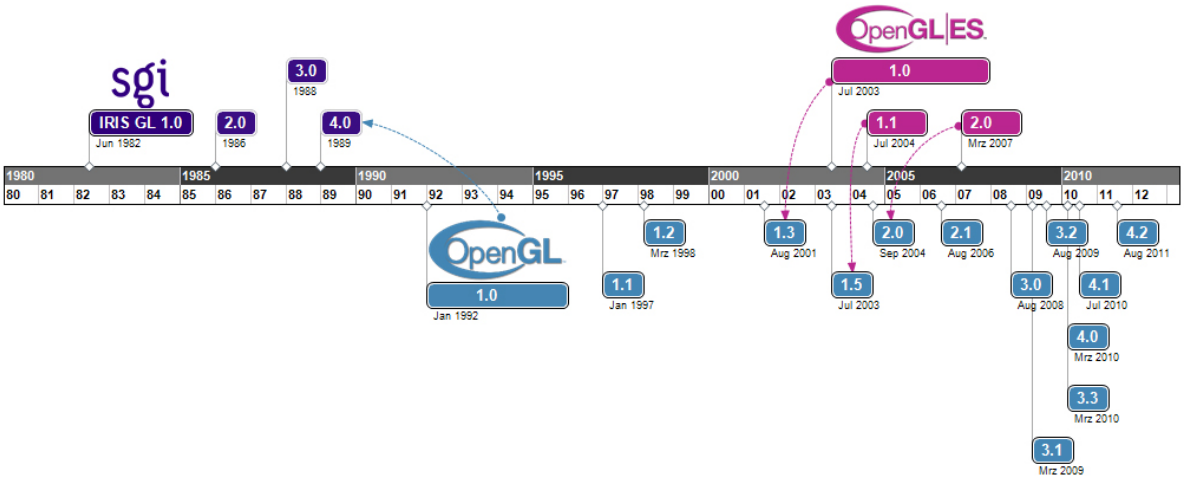

Abbildung 9.2: Die historische Entwicklung von OpenGL und OpenGL ES

OpenGL ES wird heute auf allen großen Handy Betriebssystemen – mit Ausnahme von Windows Phone 7 – eingesetzt und ist damit die am weitesten verbreitete 3D-Grafik API im mobilen Bereich [17].

9.2.2 OpenGL und OpenGL ES - Die Unterschiede

Als das von der Khronos Group eingesetzte Komitee mit dem Entwurf von OpenGL ES begann, musste es sich an ein paar grundlegende Designprinzipien halten, die ein Ausführen von 3D-Inhalten auf mobilen Geräten ermöglichen sollten. Unter anderem sollte eine Implementierung von OpenGL ES möglichst klein sein, was zum Wegfall einiger alter Funktionen aus OpenGL und einer Trennung in sogenannte *Versionsfamilien* führte. Diese Trennung stellt die Abwärtskompatibilität zwischen den Hauptversionssprüngen von OpenGL ES zu Gunsten einer einfacheren Umsetzbarkeit neuer Funktionen zurück. Aus diesem Grund ist OpenGL ES 2.0 Code nicht mehr auf Implementierungen der Versionen 1.0 oder 1.1 ausführbar [31]. Obwohl keine Kompatibilität zwischen diesen Versionen mehr gegeben ist, wurde die API jedoch so gestaltet, dass ein Hersteller Treiber für beide Versionen auf einem Gerät bereitstellen kann [30]. Dadurch kann sich die Anwendung die von ihr genutzte Version auswählen und ist so in jedem Fall lauffähig.

Im Folgenden soll nun zuerst detailliert auf die Unterschiede zwischen OpenGL 1.3 und OpenGL ES 1.0 eingegangen werden.

9.2.3 OpenGL ES 1.0 und OpenGL 1.3

Die Unterschiede zwischen den beiden APIs können in mehreren Gebieten zusammengefasst werden, die hier zu Gunsten der Übersichtlichkeit per Stichpunkt getrennt werden.

1. **Profile:** Da OpenGL ES auf einer Vielzahl teilweise grundsätzlich unterschiedlicher Geräte laufen soll, wurden sogenannte *Profile* (*Common* und *Common Lite*) eingeführt. Durch diese werden die grundlegenden, von einem bestimmten 3D-Programm benötigten, Rechenmodelle beschrieben. Das *Common Profil* rechnet beispielsweise sowohl mit Gleitkomma- (`FLOAT`) als auch mit Fixkommadatentypen (`FIXED_POINT`), während das Common Lite Profil nur letztere unterstützt. Außerdem ist das *Common Lite Profil* in der Genauigkeit, mit der es z.B. bei Vertex Daten rechnet, eingeschränkt. Es nutzt zwar auch eine Auflösung von 32 Bit pro Datentyp, kann aber nur im Bereich von $\pm 2^{-15}$ rechnen. Das *Common Profil* kann einen größeren Zahlenbereich von $\pm 2^{-32}$ benutzen [3], siehe dazu Stichpunkt 4 „*GLfixed und die Einschränkung der erlaubten Datentypen*“.

   Zusätzlich wird noch ein sogenanntes *Safety-Critical Profil* ([9], [20]) angeboten, welches spezielle auf Anwendungsfelder ausgelegt ist, in denen Fehlervermeidung zu den obersten Prioritäten zählt. Es richtet sich daher eher weniger an den Konsumentenmarkt als an den militärischen, medizinischen oder den Avionik-Markt. Im Unterschied zu den anderen beiden Profilen steht *OpenGL ES Safety Critical Profile (OpenGL SC)* nur in der Version 1.0.1 zur Verfügung, da es bisher nur gegen OpenGL 1.3 spezifiziert wurde. Außerdem wurden die Datentypenunterstützung für `BYTE` und `SHORT` entfernt.

2. **Entfernung redundanter Funktionen:** Um eine Implementierung von OpenGL ES auf möglichst kleinem Speicherplatz zu ermöglichen, wurden Funktionen weggelassen, die sich in ihrem Zweck nicht unterschieden. Zurück blieben nur die am meisten genutzten und universellsten Funktionen eines Typs. **Ein Beispiel** hierfür ist die Funktion glColor(), welche in OpenGL in ca. 30 verschiedenen Versionen existiert, die sich durch Parameter unterscheiden [2].

3. **Vertex Arrays und das glBegin()-glEnd()-Model:** Neben redundanten Funktionen umfasst OpenGL auch unterschiedliche Methoden, um ein und dasselbe Problem zu lösen. OpenGL ES wählt von diesen jeweils die effizienteste aus und lässt den Rest wegfallen.

   Ein Beispiel hierfür ist die Übergabe von Geometriedaten an die API. Dies kann über den sogenannten *Immediate Mode* geschehen, in dem die Daten mit Hilfe des *glBegin()-glEnd()-Models* übergeben werden. Diese Methode ist jedoch bei umfangreichen 3D-Szenen wenig ressourcenschonend. Deswegen nutzt OpenGL ES die sogenannten Vertex Arrays, die mit OpenGL 1.1 eingeführt wurden [2].

   Vertices können damit in Arrays abgelegt und direkt mit dem Funktionsaufruf an die API übergeben werden.

```
void glVertexPointer(GLint SIZE, GLenum TYPE,
                     GLsizei STRIDE, GLvoid * POINTER)
```

Dieser Funktionsaufruf kann Polygone unterschiedlicher Eckenanzahl oder auch Linien erzeugen. Der Parameter ***SIZE*** gibt dabei an, wie viele Vertices übergeben werden, daraus schließt OpenGL auf das zu erzeugende Primitiv. ***TYPE*** spezifiziert den Datentyp, in welchem die Vertex-Daten abgelegt sind, und ***POINTER*** zeigt schließlich auf das Array, in dem diese Daten liegen. Der ***STRIDE*** Parameter dient dem Übergeben der Distanz zwischen den auf einander folgenden Array Elementen in Bytes [3]. Im Einsatz könnte dies wie folgt aussehen:

```
GLfloat vertexArray[] = { 1.0, -1.5, 5.5, 3.0, 0.0, 1.0 };
glVertexPointer(2, GL_FLOAT, 0, vertexArray);
glDrawArrays(GL_LINES, 0, 2);
```

Dieses Code-Snippet erzeugt ein Array mit sechs `FLOAT` Werten. Danach wird dieses Array mit dem `glVertexPointer()`-Befehl in Vertices umgewandelt, wobei die Vertices mit Hilfe von je drei Koordinaten (die `FLOATs`) im Raum bestimmt werden. Zum Schluss werden die Vertices mit einer Linie verbunden und ausgegeben.

OpenGL ES nutzt nicht alle in OpenGL 1.3 spezifizierten Funktionen zum Umgang mit Vertex Arrays, bzw. wandelt diese leicht ab.

4. **GLfixed und die Einschränkung der erlaubten Datentypen:** Um Berechnungen im dreidimensionalen Raum durchzuführen, werden in allen großen APIs auf dem Computer – inklusive OpenGL – Gleitkommazahlen eingesetzt, die z.B. als Datentyp `FLOAT` (nach *Institute of Electrical and Electronics Engineers (IEEE)* Standard 754) abgebildet werden. Fixkomma Operationen gehören eher zur Ausnahme, da sie nicht genau genug sind. Leider besitzt ein Großteil der in mobilen Geräten arbeitenden Prozessoren keine eigene Gleitkomma-Einheit zum effizienten Rechnen mit diesem Datentyp, wie sie auf Computer-Prozessoren heutzutage zum Standard gehören. Ein in Software ausgelagertes Rechnen mit Pseudo-Gleitkommazahlen ist zwar möglich, würde aber bei Grafik-Anwendungen zu starken Performance-Einbußen führen und ist somit auf schwachen mobilen Prozessoren nicht durchführbar. Deswegen wurden unterschiedliche Wege beschritten, um eine Implementierung ohne Gleitkommazahlen zu ermöglichen. Zum einen wurden Funktionen, die nur mit dem 64-bit präzisions Datentyp `DOUBLE` arbeiteten, durch neue ersetzt, welche `FLOAT` Parameter besitzen und durch ein „`f`“ gekennzeichnet sind, bspw. `glFurstum()` und `glFrustumf()`.

   Zusätzlich wurde der 32-bit Datentyp `GLfixed` in der Erweiterung `OES_fixed_point` eingeführt. `OES` steht dabei für OpenGL ES und weist offizielle Erweiterungen des OpenGL ES Standards aus [3].

   Mit Hilfe von `GLfixed` können `FLOAT` Daten durch eine Multiplikation mit 2^{16} einfach in Fixkommazahlen umgewandelt werden. Dazu speichert GLfixed den ganzzahligen Teil der Zahl inklusive Vorzeichen in den ersten 16 Bit und den Nachkommateil in den letzten 16 Bit. Abschließend wurde zu jeder Funktion, die nur `FLOAT` Parameter besaß, eine Variante entworfen, die mit Fixkommazahlen arbeiten kann [3]. Diese neuen Funktionen werden mit einem „`x`“ hinter dem Namen gekennzeichnet, bspw. `glFrustumf()` und `glFrustumx()`. Sie werden vor allem vom *Common Lite* Profil genutzt.

Zusammengefasst arbeitet OpenGL ES nur mit den Zahlen-Datentypen `BYTE`, `SHORT`, `FLOAT` und `FIXED`. `INTEGER` an sich und `DOUBLE` werden nicht unterstützt [3].

5. **Transformationen:** In diesem entscheidenden Bereich der Darstellung von 3D-Inhalten geht es vor allem um die Transformation der für verschiedene Zwecke eingesetzten Matrizen. Es müssen zum Beispiel die Koordinaten für jedes in einer Szene auftauchende Vertex in drei Schritten so transformiert werden, dass sie auf dem Bildschirm ausgegeben werden können. Um diese Transformationen zu erleichtern, nutzt OpenGL sogenannte *Stacks*, also Stapel von Matrizen, welche dazu dienen, mehrere Matrizen auf einmal zu verändern und auch nach der Veränderung den Ursprungszustand wieder herzustellen. Diese Stacks haben in OpenGL ES eine kleinere maximale Größe als in der Desktop Version. Für den am häufigsten genutzten `GL_MODELVIEW`-Stack ist die Größe beispielsweise von 32 auf 16 Matrizen geschrumpft. Es gibt noch drei weitere Stacks (`GL_PROJECTION, GL_TEXTURE und GL_COLOR`), erstere beiden sind auf die Größe von 2 Matrizen geschrumpft und letzterer ist komplett weggefallen [3].

6. **Lighting:** Das Lighting ist für die Beleuchtung einer 3D-Szene zwingend notwendig, deswegen hat sich in diesem Bereich der API nicht viel geändert. OpenGL ES unterstützt genauso viele Lichtquellen (bis zu 8) pro Szene wie OpenGL. Es wurden nur einige Änderungen an den Materialeigenschaften von Polygonen unternommen. Diese können zum Beispiel nur noch ein Material haben, in dem sie dargestellt werden, und nicht mehr je eines für Vorder- und Rückseite. Eine weitere wichtige Änderung ist der Wegfall des lokalen Betrachter-Modells (`GL_LIGHT_MODEL_LOCAL_VIEWER`), was zwar die Lichteffekt teilweise weniger realistischer erscheinen lässt, dafür aber die Berechnung erheblich beschleunigt [3].

7. **Texture Mapping:** Texturen spielen in 3D-Anwendungen als Oberfläche von so gut wie allen Objekten eine bedeutende Rolle für das Aussehen, aber auch für den Rechenaufwand einer Szene. Daher werden nur noch 2D-Texturen unterstützt. 1D Texturen sind leicht durch 2D-Texturen zu emulieren und 3D-Texturen – eher im Bereich des *Volumen Renderings* eingesetzt, z.B. für die Simulation von Flüssigkeiten – stellen zu hohe Anforderungen an Hardware und Software [31]. Außerdem wurde mit der Erweiterung `OES_compressed_paletted_texture` eine effiziente Möglichkeit zum Komprimieren von Textur-Daten geschaffen.

8. **Verkleinerung des benötigten Datenvolumens:** Da auf mobilen Geräten sowohl der Speicherplatz als auch die Bandbreite zur Kommunikation mit dem Arbeitsspeicher begrenzt war wurden die Parameter einiger Funktionen so umgeändert, dass kleinere Datentypen akzeptiert werden können. Ein Beispiel hierfür sind Funktionen, die statt `INTEGER` nun `BYTE` akzeptieren [2].

9. **Bereiche, die wenig bis gar nicht verändert wurden:** Das *Culling* – der Prozess der Wegschneidens aller nicht sichtbaren Polygone innerhalb einer Szene – und das *Clipping* – der Prozess der Wegschneidens aller nicht sichtbaren Polygone außerhalb des Sichtbereichs des Betrachters einer Szene – in OpenGL ES wurden so gut wie gar nicht eingeschränkt. Genauso wurden fast alle Funktionen unverändert gelassen, die die Möglichkeiten betreffen, Nebel zu erschaffen. Alle *Framebuffer*-Funktionen – Operationen, die das komplette ausgegebene Bild betreffen – sind, so-

weit sie nicht zu rechenintensiv sind, unterstützt. Gleiches gilt für die *Per-Fragment* Operationen.

10. **Entfernung der Zustands-Abfrage:** In OpenGL 1.3 gibt es die Möglichkeit, zu jeder Zeit den sogenannten dynamischen Zustand abzufragen, in dem sich die API augenblicklich befindet. Dieser Zustand wird durch unterschiedliche Attribute beschrieben, die in der API zur Laufzeit an- oder abgeschaltet bzw. verändert werden können, wie z.B. die Beleuchtung der Szene [31]. Diese Abfrage wird dafür genutzt, von der API unabhängig laufende Systeme mit ihr zur synchronisieren, um Berechnungen auszulagern. Leider ist sie sehr Speicherplatz raubend und wurde, da sie zusätzlich schwer auf Systemen mit mehreren Software-Schichten wie Smartphones zu implementieren ist, weggelassen. Deswegen ist nur eine Abfrage des statischen Zustands (Version der API, Renderer, Erweiterungen) möglich [2].

11. **Kein Einsatz der OpenGL Utility Library:** Die *OpenGL Utility Library (GLU)* ist eine durch das Architecture Review Board (ARB) geprüfte Bibliothek, die für unterschiedliche Implementierungen von OpenGL zur Verfügung steht. Sie bietet Operationen an, die eigentlich in den Bereich einer High-Level API fallen [25]. Beispielsweise erlaubt sie das einfache Erstellen von multipolygonalen Objekten wie Zylindern und bietet Zugriff auf nützliche Matrix-Operationen. Obwohl die Bibliothek ein nützliches Feature-Set bereitstellt, gab es bei Erscheinen von OpenGL ES keine offiziellen Bemühungen zu einer Portierung [2]. Heute werden einige privat gepflegte Implementierungen von GLU für unterschiedliche Plattformen angeboten, siehe Stawarz (2008) [34]für iOS (z.B. eingesetzt im iPhone) und Gorchak (2009) [8]für QNX (z.B. eingesetzt im BlackBerry Playbook).

9.2.4 OpenGL ES 1.1

Während OpenGL ES 1.0 eine wirklich minimal gehaltenen API war, zielte die Version 1.1 auf Geräte mit etwas mehr Rechenleistung und allgemein einer besseren Hardwareausstattung, wie z.B. einem eigenen Grafik Chip, ab. Insgesamt wurden sieben Änderungen gemacht, von denen 5 zwingend erforderlich für eine Implementierung dieser Version sind. Auf die wichtigsten drei soll hier eingegangen werden [30].

1. **Verbesserung der Textur-Darstellung:** Jedes Gerät, das eine Implementation dieser Version ausführen soll, muss mindestens zwei Hardware-Einheiten zur Berechnung von Texturen vorweisen können. Durch diese Einschränkung kann das Limit von Texturen pro Objekt von einer auf zwei angehoben werden. Dies kommt dem Aussehen eines so gerenderten Objektes zugute und nähert die API der PC-Version an, in der teilweise vier oder mehr Texturen pro Objekt verwandt werden. Eine Einsatzmöglichkeit mehrerer Texturen für ein Objekt ist das sogenannte *Bump Mapping*. Dabei wird eine Textur über das Objekt gelegt, die die Obeflächenfarbe und das Material simuliert. Zusatzlich wird noch eine zweite Texture mit Namen *Normal Map* benutzt. Diese simuliert Unebenheiten in der Oberfläche des Objektes, indem sie die Lichtreflexion verändert. So können realistische rauhe Oberflächen dargestellt werden, ohne dass mehr Polygonen eingesetzt werden müssten. Dadurch kann im Vergleich viel Rechenaufwand gespart werden.

2. **Schnelle Partikel-Effekte:** Durch den Einsatz von sogenannten *Point Sprites* – einzelne Vertices, denen jetzt auch Texturen zugewiesen werden dürfen – können nun schnelle Partikel-Effekte wie Wasser, Rauch und Feuer umgesetzt werden.

3. **Dynamic State Abfragen:** Es ist nun erlaubt, den dynamischen Zustand abzufragen, in dem sich OpenGL ES gerade befindet. Anwendungen, die z.B. eigene Berechnungen zur Laufzeit einer OpenGL Anwendung durchführen, können so auch in vielschichtigen Betriebssystemen, wie man sie auf Smartphones findet, programmiert werden, siehe dazu auch Seite 10 „*Entfernung der Zustands-Abfrage*“.

9.2.5 OpenGL ES 2.0

Abschließend soll hier noch auf die in OpenGL 2.0 gemachten Änderungen gegenüber seinen Vorgängerversionen eingegangen werden. Diese API unterscheidet sich von ihrem Vorgänger ganz erheblich darin, dass sie die *OpenGL Shading Language* einführt [15]. Die Sprache erlaubt es, *Vertex* und *Fragment Shader* zu schreiben, die in bestimmter Weise auf das Rendering Einfluss nehmen und damit auch auf das Aussehen der auf dem Bildschirm ausgegebenen Objekte. Während die OpenGL ES Versionen 1.x noch auf der *Fixed Function Pipeline* beruhten, die nur bestimmte Effekte auf den einzelnen Stufen der Rendering Pipeline unterstützte (vgl. Abschnitt 9.1.1 (S. 228)), hat der Nutzer nun die Möglichkeit, sämtliche Effekte selbst zu erstellen. Die *Shading Language* wurde mit wenigen Änderungen gegenüber ihrem Desktop-Pendant übernommen. Wichtig sind vor allem der Wegfall aller *Fixed Function* Funktionalitäten und die Vereinfachung der Pipeline, indem logische Operationen ausgeschlossen wurden [15].

9.3 Die Konkurrenz zu OpenGL ES - DirectX

Nachdem im letzten Abschnitt detailliert auf OpenGL ES eingegangen und damit der größte Teil aller Smartphones heutzutage abgedeckt wurde, wird im Folgenden nun eine Plattform betrachtet, die bisher noch nicht durch diesen offenen Standard erobert werden konnte.

Microsoft stieg im Jahr 2000 mit Windows Mobile in den Markt für PDAs ein. Zum damaligen Zeitpunkt hatte keines der Zielgeräte Mobilfunkfunktionen, sie dienten eher als Taschencomputer, der die Windows-Funktionalität in einem Notizbuchformat bereitstellen sollte. Microsoft konnte sich damit erfolgreich – besonders auf dem Markt für den Einsatz in Unternehmen – mit Research in Motion (RIM) messen, dem Blackberry-Hersteller und damaligen Branchenprimus [37].

Die Situation änderte sich grundlegend als 2007 das iPhone erschien. Durch seinen Bedienkomfort und die allgemeine Benutzerfreundlichkeit setzte es völlig neue Maßstäbe. Trotzdem erkannte Microsoft den damit einhergehenden Wechsel im Anspruch der Nutzer an ein modernes Handy nicht und begann die Arbeit an Windows Mobile 7 ohne grundlegend z.B. von der Stifteingabe, die vielen Nutzern schon damals als antiquiert erschien, Abstand zu nehmen. Das spiegelte sich im Erfolg des Windows Mobile 6.5 Updates wider,

welches vor dem eigentlichen Erscheinen der 7. Version des Betriebssystems bereits auf dem Markt kam. Als dann auch noch Google 2008 den Markt mit Android betrat, sah Microsoft sich im Angesicht schwindender Marktanteile dazu gezwungen, seine Softwareplattform Windows Mobile von Grund auf zu überdenken. Das Ergebnis dieses Schrittes wurde im Oktober 2010 unter dem Namen *Windows Phone 7* veröffentlicht.

Windows Phone 7 unterscheidet sich nicht nur oberflächlich von seinem Vorgänger Windows Mobile, sondern zeigt auch intern große Unterschiede. Einer dieser Unterschiede ist die Abkehr von *OpenGL ES* als treibende Kraft für 3D-Anwendungen hin zu *Direct3D 9.0*. Das Direct3D, welches auf Windows Phones implementiert ist, entspricht dabei ungefähr dem Direct3D, das zu Microsofts PC-Schnittstellensammlung *DirectX* gehört [37].

Zwar kann über die Gründe, warum Microsoft OpenGL ES für sein neues Smartphone-Betriebssystem nicht mehr nutzt, nur gemutmaßt werden, doch erscheint die Portabilität von Code zwischen den verschiedenen Microsoft Plattformen wie Windows, der Xbox und dem Windows Phone als sehr plausibel. In den letzten Jahren führte Microsoft eine Entwicklungsumgebung unter dem Namen „*XNA*“ ein. Sie kapselt die Direct3D-Befehle, um sie in unterschiedlichen Programmiersprachen ansprechbar zu machen. Außerdem kann XNA sowohl PCs, als auch die Xbox und das Windows Phone mit Software beliefern. Um hier die Smartphone-Sparte besser zu integrieren, könnte versucht worden sein diese möglichst wenig von den anderen in Sachen Grafik-Bibliotheken zu differenzieren.

9.3.1 DirectX auf Windows Phone 7 und auf dem PC

XNA stellt seit Version 4.0 für das Windows Phone 7 die primäre Programmierumgebung dar. Unter der Oberfläche arbeitet ein von Microsoft vorgeschriebener mindestens Direct3D 9.0 fähiger Grafikchip [36]. Das Windows Phone 7 Betriebssystem unterstützt also offiziell keine andere Grafik API neben Direct3D.

Trotz der schon erwähnten begrenzten Leistungsfähigkeit von mobilen Geräten unterscheidet sich die Direct3D Variante, die hier zum Einsatz kommt, in vielen Bereichen nur gering von ihrem Desktop-Pendant. Es wurden zwar einige Funktionen, die untereinander ähnliche Aufgaben erledigten, gestrichen, diese Änderungen gingen aber in keiner Weise soweit wie bei OpenGL ES. Außerdem wurden die durch die nachfolgenden Versionen der PC APIs gemachten Erweiterungen aus Performanz-Gründen nicht durch die Smartphone-Schnittstelle übernommen. Deswegen ist sie noch auf dem Stand von Direct3D 9.0, während die Grafik-Bibliothek auf dem PC schon Versionsnummer 11.0 schreibt.

Die einzige große und wirklich erwähnenswerte Lücke im mobilen Bereich ist das Fehlen selbstprogrammierbarer Shader - sowohl die *Fixed Function* Einstellungsmöglichkeiten betreffend als auch *Vertex* und *Fragment Shader* –, da diese zu hohe Anforderungen an die von Microsoft vorgeschriebene Hardware stellen würden.

Um dies zu kompensieren, bietet XNA eine Reihe vordefinierter Effekte an, die vom Nutzer angepasst werden können [36].

1. **BasicEffect:** Dieser Effekt wird für Transformationen und für die Beleuchtung einer Szene genutzt. Außerdem können über ihn Nebel und verschiedene Arten des Texturings realisiert werden.

2. **SkinnedEffect:** Mit diesem Effekt können Charakteranimationen erstellt werden. Er stellt dafür sogenannte *Knochen* zur Verfügung, über die die Vertices eines Objekts bewegt werden können.

3. **EnvironmentMapEffect:** Über diesen Effekt können Cubemaps erzeugt werden, die – wie oben bereits erwähnt – Spiegelungen für unbewegte Objekte und im begrenzten Rahmen auch für bewegte Objekte realisierbar machen.

4. **DualTextureEffect:** Dieser Effekt ermöglicht das Belegen eines Vertex mit zwei Texturen, welche für anspruchsvolle Effekt genutzt werden.

5. **AlphaTestEffect:** Dieser Effekt verbessert die Performance einer Grafikanwendung, indem er dafür sorgt, dass nur Pixel des Bildschirms pro Render-Durchlauf aktualisiert werden, die auch ein Objekt aus der Szene darstellen müssen. Über diesen Test werden auch durchsichtige Objekte realisiert.

Das Windows Phone besitzt also eine sehr starke Grafik-Schnittstelle, die sich nur partielle von der auf Festrechnern unterscheidet. Jedoch stellen die Einschränkungen, die im Shader-Bereich gemacht werden mussten, massive Einschnitte für Anwendungsentwickler dar. Heutzutage wird ein Großteil der für Realismus in der Darstellung nötigen Effekte über Shader realisiert. Jedoch sind die von Microsoft vorgeschriebenen Effekte nur ein schwacher Ersatz für selbstprogrammierbare Shader. Besonders im professionellen Einsatz hat diese Situation enorme Auswirkungen. Shader werden dort vor allem eingesetzt, um bestimmte Visualisierungstechniken umzusetzen. Hier kann zum Beispiel das Echtzeit *Ray Tracing* genannte werden. Seine Rendering Abläufe versuchen die physikalische Realität von Lichtstrahlen, die ausgesendet, gespiegelt und vom Auge aufgenommen werden, nachzuahmen. Somit ist eine Verwendung des Windows Phones für bestimmte Einsatzzwecke gar nicht oder nur erschwert möglich. Trotzdem scheint es so, als ob Microsoft diesen Weg bewusst gewählt hätte. Ein Indiz hierfür ist zum Beispiel die vollständige Kapselung der Direct3D Schnittstelle in XNA, welches vorranging zur Spieleprogrammierung entwickelt wurde.

Somit stellt Direct3D auf dem Windows Phone für den professionellen Einsatz nicht ausreichend Flexibilität bereit. Trotzdem können auf Smartphones mit diesem Betriebssystem ästhetische Anwendungen realisiert werden. Die API zielt somit bisher eher auf die Unterhaltungsindustrie ab.

9.3.2 OpenGL ES & Direct3D 9.0 - Das Fazit

Durch das Beschreiben dieser beiden APIs konnte der komplette Markt für Smartphones bezüglich seiner eingesetzten Grafik-Bibliotheken abgedeckt werden. Es wurden die Entwicklungsgeschichten von *OpenGL*, seiner mobilen Abwandlung *OpenGL ES* und von

Direct3D beschrieben. Darüber hinaus wurden jeweils die Unterschiede zwischen der Desktop- und der mobilen Variante aufgezeigt.

Man kann sehen, dass die mobilen APIs sich in beiden Fällen aus ihren schon lange im Desktop-Bereich eingesetzten Vorfahren entwickelt haben. OpenGL ES weist eine ganze Reihe von – gut dokumentierten – Unterschieden zu OpenGL auf. Diese ziehen sich durch sämtliche Teile der Rendering Pipeline. Aufgrund seiner jetzt fast 10-jährigen Entwicklung konnte sich OpenGL ES aber bereits bedeutend an OpenGL annähern.

Das mobile Direct3D unterscheidet sich hingegen auf den ersten Blick nicht großartig von dem auf PCs eingesetzen. Doch eine etwas längere Auseinandersetzung mit diesem Thema fördert bedeutende Unterschiede ans Tageslicht. Besonders wichtig ist dabei, dass hinter die Fassade von XNA geschaut wird. Dieses erscheint zuerst so, als würde es allein sämtliche 3D-Funktionalität zur Verfügung stellen. XNA ist dabei aber eigentlich nur eine Abstarktionsschicht für die Komplexität von Direct3D und reicht eingehende Befehle nur an letzteres weiter. Direct3D selbst taucht in der Literatur zum Windows Phone nur am Rand auf und wird auch nur oberflächlich behandelt. Der Grund hierfür ist, dass Microsoft es erfolgreich vor den Augen von Programmierern und Nutzern durch XNA versteckt.

Trotzdem können bei genauem Hinschauen einige gewichtige Abwandlungen gefunden werden, die ein Umsetzen von Spielen auf dem Windows Phone in ästhetischer Hinsicht schwieriger erscheinen lassen als auf Smartphones, die OpenGL ES unterstützen. Ganz abgesehen von einem professionellen Einsatz des Windows Phones. Da Shader heute für alle wichtigen Effekte im Grafikbereich eingesetzt werden, die hier zum Teil schon zur Sprache kamen, besitzt OpenGL ES gerade in Version 2.0 einen wesentlichen größeren Spielraum für visuell anspruchsvolle und ansprechende Anwendungen. Microsoft hat in dieser Hinsicht noch Aufholbedarf.

Die Geschichte von OpenGL ES und die erste Version von Direct3D mit seinen zahlenmäßig kleinen Unterschieden zur Desktop-Version haben deutlich gemacht, dass auf langfristige Sicht mit einer Wiedervereinigung der mobilen und der PC APIs zu rechnen ist. Wann diese eintritt, hängt im ganz entscheidendem Maße von der Entwicklung der Energiespeicher für Smartphones ab. Diese sind heutzutage noch der limitierende Faktor, der z.B. auch den Geschwindigkeitszuwachs von mobilen Prozessoren in erster Linie mitbestimmt.

9.4 Anwendungsbeispiele für die professionelle Nutzung von 3D-APIs auf Smartphones

Im Folgenden sollen zwei Anwendungsbeipsiele gegeben werden, wie die oben beschriebenen Techniken auf Smartphones eingesetzt werden können. Bisher liegt das Haupteinsatzgebiet von 3D-APIs bei Videospielen. Trotzdem sind immer öfter Bestrebungen zu finden, das Potential zu nutzen, welches mobile Geräte im Bereich der Darstellung von komplexen photorealistischen dreidimensionalen Inhalten haben. Um dies zu veranschaulichen, soll zuerst ein Beispiel aus dem *Volumen Rendering* gegeben werden. Daran schließt ein

selbstentwickeltes Einsatzsezenario an. Es zeigt auf wie die Nutzung der Smartphones in diesem Bereich in Zukunft aussehen könnte.

9.4.1 ImageVis3D Mobile

ImageVis3D Mobile ist eine Open-Source Volume Rendering App des *Center for Integrative Biomedical Computing* der University of Utah in den USA [5]. Es ist bisher nur für das iOS Betriebssystem von Apple erschienen. Die Software wird eingesetzt, um große volumetrische Datensets zu visualisieren. Diese enstehen bei unterschiedlichen Verfahren der bildgebenden Diagnostik, wie z.B. bei der Magnetresonanztomographie oder Computertomographie. Darüber hinaus sind sie oft Ergebnis von Computer-Simulationen, wie Strömungsberechnungen oder ähnlichem.

Alle eben genannten Verfahren produzieren sogenannte *Point Clouds*. Ein CT nimmt z.B. den zu durchleuchtenden Körper in Schichten von heutzutage ca. 0,3 mm Dicke auf. Pro Aufnahme ensteht genau ein Bild mit einer bestimmten Auflösung. Diese Bilder werden anschließend in einem 3D-Modell zusammengefasst. Dabei werden die Bilder in ihrer Aufnahmereihenfolge übereinandergestapelt und jeder Bildpunkt zu einem Vertex umgesetzt. So entsteht der Eindruck einer Wolke, die aus Tausenden kleiner Punkten besteht [12].

Mit speziellen Visualisierungstechniken wie dem *Raycasting* werden die Point Clouds gerendert. Das Raycasting sendet dabei für jeden Bildpunkt auf dem Bildschirm des Nutzers einen Sehstrahl aus. Die Farb- und Transparenzwerte der Objekte in der 3D-Szenen, auf die der Strahl trifft, werden dabei festgehalten. Um am Schluss für den einzelnen Bildpunkt den Gesamtfarbwert zu berechnen, werden die vorher zwischengespeicherten Werte aufsummiert. So entsteht ein realistisches Bild der Point Cloud, das frei dreh- und zoombar ist und sich in unterschiedlichen Tiefeebenen betrachten lässt [22].

Da Raycasting auf mobilen Geräten noch nicht ausreichend schnell ausgeführt werden kann, werden auf dem Smartphone unterschiedliche Tricks angewandt, um volumetrische Inhalte zu rendern ohne auf 3D-Texturen zurück greifen zu müssen. Diese sind nämliche Grundlage des Raycastings. Einer dieser Tricks ist das sogenannte *2D Texture based Volume Rendering*, bei dem die einzelnen Ebenen der Point Cloud als 2D-Texturen angezeigt werden. ImageVis3D Mobile wandelt dessen ursprünglichen Prozess leicht ab, um ein besseres Bild zu erzielen (vgl. [23]). Abbildung 9.4 zeigt das Programm auf dem Apple iPad2. Man sieht den gerenderten Oberkörper eines Menschen.

Der Umgang mit der Software gestaltet sich sehr einfach. Durch Fingergesten kann das Modell gezoomt, verschoben und gedreht werden. Es ist auch erlaubt, mit der Kamera in das Modell hineinzufahren. Durch diese einfache Bedienung ist auch die Benutzung durch Computer-Laien möglich oder durch Menschen, die erst seit Kurzem mit der Software und dem Gerät, auf dem sie läuft, arbeiten.

Um ein flüssiges Arbeiten mit einem dargestellten Modell sicherzustellen, rendert ImageVis3D Mobile die Szenen erst hochauflösend neu, nachdem der Benutzer eine Veränderung der Darstellung abgeschlossen hat. Während eines Zoom-Vorgangs oder eines Schwenks

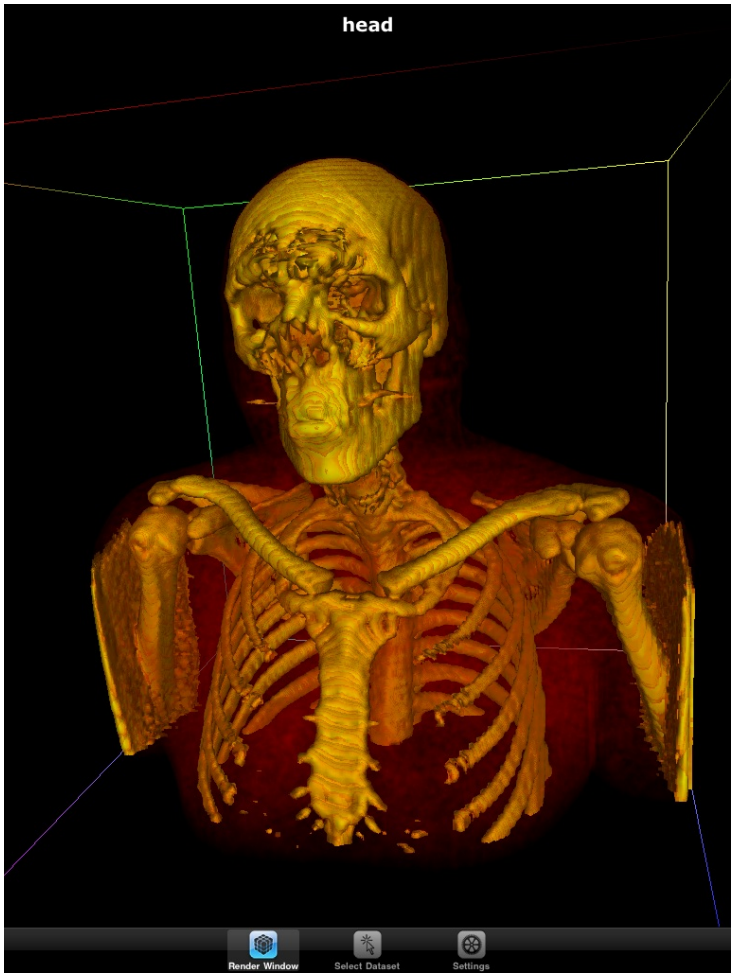

Abbildung 9.3: ImageVis3D Mobile auf dem iPad2 - Oberkörper eines Menschen

mit der Kamera um das Modell herum wird dem Nutzer ein Modell mit geringerer Auflösung und Details präsentiert. Diese Behelfslösung erlaubt eine ruckelfreie Ausführung auch auf Geräten mit weniger Leistung als bei traditionellen Computern.

Die Software könnte zum Beispiel in Krankenhäusern eingesetzt werden. Ärzte würden sich dann mit ihr direkt im Krankenzimmer einen Überblick über den Zustand des Patienten verschaffen können, z.B. bei inneren Verletzungen. So kann der Arzt direkt beim Kranken die benötigten Informationen einsehen und dieser muss nicht an einen Ort verbracht werden, an dem dies nur durch einen Festrechner möglich wäre.

Ein weiterer Einsatzbereich wäre die Qualitätskontrolle in Fabriken. Hier könnten direkt an der Produktionslinie die Durchleuchtungsergebnisse der letzten gefertigten Stücke mit den Computer Aided Design (CAD) Skizzen verglichen werden und ggf. die Maschinen neu justiert werden. Zeit und Kostenersparnisse wären die Folge.

ImageVis3D Mobile befindet sich noch nicht offiziell im professionellen Einsatz, veranschaulicht aber schon hervorragend, wie neue Rendering-Techniken, kombiniert mit moderner mobiler Hardware, erstaunliche Ergebnisse hervorbringen können. Durch die jetzt schon hohen Benutzerzahlen lässt sich abschätzen, welches Nutzerinteresse gerade in die-

sem Bereich besteht. Im Moment wird das Programm vor allem zum Visualisieren von Simulationsergebnissen benutzt, z.B. zum Erforschen des Verhaltens zweier Flüssigkeiten.

9.4.2 Lageinformationssystem für den Kriseneinsatz

Gerade der Einsatz von 3D-APIs im Volumen Rendering muss nicht auf die oben beschriebenen Anwendungsgebiete beschränkt bleiben. Eine weitere Möglichkeit, diese Technik nutzenbringend einzusetzen, wäre die Lagedarstellung für Militärs im Einsatzgebiet. Schon heutzutage gibt es mit Produkten wie der *Lidar Fusion Tool Suite* von *CG2* Applikationen, die aus Point Cloud Daten Obeflächenkarten – inklusive der Erhebungsparameter – generieren. Grundlage dieser Karten sind dabei meist Satelliten Scans der Gebiete von Interesse [6].

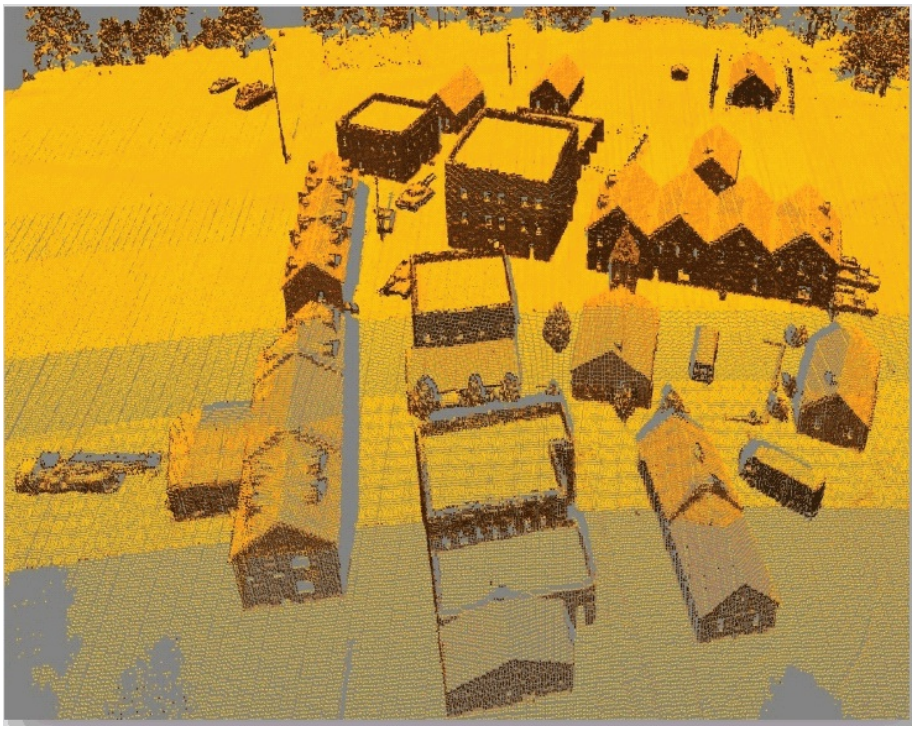

Abbildung 9.4: Lidar Fusion GEOINT - Digitales Erhebungsmodell eines Dorfes
Quelle: CG2 (2011), [6], `http://www.cg2.com/Products_LIDAR_GEOINT.html`

Normale Satelliten- oder allgemein Luftaufnahmen haben den Nachteil, dass sie zwar einen Überblick über das Gebiet geben aber dieser Überblick auf eine zweidimensionale meist direkt von oben betrachtete Karte beschränkt ist. Gerade für taktische Überlegungen sind aber die Höhenunterschiede in einem Gebiet von ausschlagebender Bedeutung. So kann eine Senke z.B. Deckung für die eigenen Truppen bedeuten oder eine Anhöhe leichter zu verteidigen sein.

Besonders in bergigem oder urbanem Gebiet in dem mit Hinterhalten gerechnet werden muss, spielt die genaue Planung der eigenen Truppenbewegungen eine große Rolle. Diese werden schon heute mit stationären Rechnern an Karten durchgeführt, die die Topografie

des Geländes abbilden. Doch sobald die Einheiten das Lager verlassen haben, ist die Übersicht nur mit normalen Karten möglich.

Eine Applikation, die auf Grundlage vorheriger Satelliten-Scans eine 3D-Karte des Geländes auf einem mobilen Gerät rendern könnte, wäre hier eine große Hilfe. Abbildung 9.4 zeigt einen Screenshot der *Fusion GEOINT* aus der oben schon erwähnten *Lidar Fusion Tool Suite*. Die Oberflächen könnten zusätzlich noch mit Texturen versehen werden, die für mehr Überblick sorgen würden. Die Software in diesem Beispiel würde als Grundfunktion das Anzeigen dieser Karte auf mobilen Geräten ermöglichen. Für die Einheitsführer im Gelände böte allein dies schon Vorteile. Es wäre möglich Gegenden noch im Gelände zu bestimmen, die sich für einen Hinterhalt anbieten könnten und deswegen möglichst gemieden werden sollten. Außerdem wäre es möglich Wege zu identifizieren, die mehr Deckung bieten als andere. Mit heutiger Software können zu diesem Zweck sogar schon die Unebenheiten des Bodens unter bewaldetem Gebiet bestimmt werden.

Erweitert man diese Softwareidee jetzt z.B. um die Funktion, entdeckte Feindeinheiten sowie Verbündeteteinheiten per GPS in der Karte zu verorten, würden sich wieder andere Nutzungsmöglichkeiten eröffnen. Anhand der Position von Freund und Feind wäre es möglich eine Sichtweitenbestimmung über das sie umgebende Gelände durchzuführen. Hier spielen auch die Gebiete mit hinein, in denen von der eigenen Artillerie gewirkt werden kann. Diese könnten durch eine Schattierung verdeutlicht werden.

Über weitere Funktionen könnte der Einheitsführer vor Ort seinen Untergebenen Befehle geben. Es wäre so möglich den Ausführungsort im dreidimensionalen Raum zu bestimmen. Dadurch könnten Wegpunkte exakter für einzelne Truppen vorgegeben werden.

Abbildung 9.5: DARPA Tigr Screenshot - Lageinformationen
Quelle: Turner (2009), [35],
`http://radar.oreilly.com/2009/04/where-20-preview---darpas-tige.html`

Dies ist nur eine Auswahl der ersten Ideen, wie eine Software per mobiler API die eigenen Truppen unterstützen kann. Von einer Implementierung ist dies natürlich noch weit entfernt. Trotzdem werden schon heutzutage in der U.S. Army Smartphones eingesetzt, die über eine Karte erzeugte Informationen zwischen einzelnen Soldaten synchronisierbar

machen. Bei der dabei eingesetzten Karte handelt es sich zwar noch um eine normale zweidimensionale Luftaufname doch die App implementiert bereits viele Funktionen, die obiges Beispiel auch interessant machen [33]. Die Abbildung 9.5 zeigt einen Screenshot der Software mit dem Namen *Tigr*. Sie ist bereits seit ca. zwei Jahren im Einsatz und scheint sich bewährt zu haben. Mitentwickelt und finanziert wurde sie von der Defense Advanced Research Projects Agency (DARPA), einer Forschungsbehörde des U.S. Verteidigungsministeriums [35].

Eine Kombination aus *Tigr* und der oben beschriebenen Idee würde für den Einsatz völlig neue Informations- und Führungsmöglichkeiten bedeuten.

9.4.3 Ausblick

Bisher werden im professionellen Bereich erst wenige Applikationen auf mobilen Geräten eingesetzt, die die 3D-Schnittstellen nutzen. Einige Apps im Bildungsbereich stellen hier die Ausnahme dar. Sie versuchen nicht real filmbare Sachverhalte in Form von 3D-Animationen darzustellen. Im Militär oder in der Wissenschaft sieht dies schon wieder anders aus. Auf der einen Seite strenge Sicherheitsvorschriften sowie Anforderungen an eine robuste Hardware und auf der anderen Seite riesige Datenmengen scheinen hier bisher fast jeden Einsatz von Smartphones verhindert zu haben. Einzig und allein Spiele nutzen intensiv die Möglichkeiten zur Darstellung photorealistisch gerenderter Bilder auf mobilen Geräten. Sie sind zurzeit die Treiber der Soft- und Hardwareentwicklung auf diesem Gebiet.

Doch langsam beginnen die Entwickler professioneller 3D-Anwendungen für Festrechner, die sich auf mobilen Geräten eröffnenden Einsatzfelder ihrer Software, zu erforschen. *ImageVis3D* ist hierfür das beste Beispiel. Außerdem scheinen auch die Anwender sich nicht länger damit zufrieden geben zu wollen an einen bestimmten Arbeitsplatz gebunden zu sein, sobald etwas komplexere 3D-Szenen dargestellt werden müssen.

Dieses Konsumenten-Interesse kann in vielen Bereichen schon heute durch die Möglichkeiten, die moderne mobile Hardware bietet, befriedigt werden. Kommende Generationen mobiler Grafikprozessoren werden diese Möglichkeiten noch wesentlich erweitern. Einzig an der zur Verfügung stehenden Software mangelt es noch. Die Entwickler professioneller Applikationen müssen sich, so scheint es, noch mehr auf die neue tragbare Hardware einlassen. Ist dies ersteinmal der Fall, so steht einer Verbreitung von 3D-Applikationen unter beruflichen Anwendern nichts mehr im Weg.

Literaturverzeichnis

[1] Akenine-Möller, Thomas; et al. *Real-Time Rendering*, A. K. Peters Ltd., Wellesley, Massachusetts, USA 2008.

[2] Astel, D.; Durnil, D.. *OpenGL ES - Game Development*, Course Technology PTR, Hampshire, United Kingdom 2004.

[3] Blythe, D. (edt.). *OpenGL ES - Common/Common-Lite Profile Specification - Version 1.0.02*, `http://www.khronos.org/registry/gles/specs/1.0/opengles_spec_1_0.pdf`, o.O., 16.11.2011, 2003.

[4] Bender, Michael; Brill, Manfred. *Computergrafik*, Carl Hanser Verlag, München 2006.

[5] Center for Integrative Biomedical Computing - University of Utah. *ImageVis3D - Official Website*, `http://www.imagevis3d.com/`, o.O., 09.12.2011, 2011.

[6] CG2. *CG2 - LIDAR Fusion GEOINT*, `http://www.cg2.com/Products_LIDAR_GEOINT.html`, o.O., 09.12.2011, 2011.

[7] Dawes, A.. *Windows Phone 7 Game Development*, Apress, New York, USA 2010.

[8] Gorchak, M.. *GLU ES - GLU 1.3 partitial port for OpenGL ES 1.x - Project Website*, `http://code.google.com/p/glues/`, o.O., 16.11.2011, 2009.

[9] Hall, C.; Knaus, C. (edt.). *OpenGL ES Safety Critical Profile Specification - Version 1.0 (Annotated)*, `http://www.khronos.org/registry/glsc/specs/opengles_sc_spec_1_0.pdf`, o.O., 20.11.2011, 2004.

[10] Harbour, J. S.. *Microsoft Visual Basic game programming with DirectX*, Premier Press, Indianapolis, USA 2002.

[11] Hawkins, K.; Astle, D.. *OpenGL game programming*, Premier Press, Roseville, Calfornia, USA 2001.

[12] Herman, Gabor T.. *Fundamentals of Computerized Tomography*, Springer, London 2009.

[13] IEEE. *ANSI/IEEE Std 754-2008 - IEEE Standard for Floating-Point Arithmetic*, `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`, New York, USA 16.11.2011, 2008.

[14] Kessenich, J; et al. *The OpenGL Shading Language - Version 4.20*, http://www.opengl.org/registry/doc/GLSLangSpec.4.20.6.clean.pdf, o.O., 17.11.2011, 2011.

[15] Kessenich, J; et al. *The OpenGL ES Shading Language - Version 1.00*, http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdf, o.O., 17.11.2011, 2011.

[16] Khronos Group. *OpenGL - Official Website*, http://www.opengl.org/, o.O., 14.11.2011, 1997.

[17] Khronos Group. *OpenGL ARB and Khronos - News*, http://www.khronos.org/news/press/opengl_arb_to_pass_control_of_opengl_specification_to_khronos_group/, o.O., 09.11.2011, 2006.

[18] Khronos Group. *OpenGL ES - Official Website*, http://www.khronos.org/opengles/, o.O., 09.11.2011, 2007.

[19] Khronos Group. *The Khronos Group - Media Authoring and Acceleration - Official Website*, http://www.khronos.org/, o.O., 09.11.2011, 2011.

[20] Knaus, C.. *OpenGL ES Safety-Critical Profile Philosophy*, http://www.khronos.org/registry/glsc/specs/es_sc_philosophy.pdf, o.O., 20.11.2011, 2004.

[21] Komatineni, S.; et al. *Pro Android 3*, Apress, New York, USA 2011.

[22] Krüger, Jens; et al. *ClearView - An Interactive Context Preserving Hotspot Visualization Technique*, In: IEEE Transactions on Visualization and Computer Graphics. Nr. 5 von September - Oktober 2006, S.: 941 - 948.

[23] Krüger, Jens. *A NEW SAMPLING SCHEME FOR SLICE-BASED VOLUME RENDERING*, http://ivda.cs.uni-saarland.de/fileadmin/slides/VG2010/VG2010.pdf, o.O., 09.12.2011, 2010.

[24] Malizia, A.. *Mobile 3D Graphics*, Springer, London, United Kingdom 2006.

[25] Martz, P.. *OpenGL - Distilled*, Addison Wesley Professional, Boston, Massachusetts, USA 2006.

[26] Munshi, A.; Leech, J. (edt.). *OpenGL ES - Common/Common-Lite Profile Specification - Version 1.1.12 (Full Specification)*, http://www.khronos.org/registry/gles/specs/1.1/es_full_spec_1.1.12.pdf, o.O., 17.11.2011, 2008.

[27] Munshi, A.; Leech, J. (edt.). *OpenGL ES - Common Profile Specification Version 2.0.25 (Full Specification)*, http://www.khronos.org/registry/gles/specs/2.0/es_full_spec_2.0.25.pdf, o.O., 17.11.2011, 2010.

[28] Nischwitz, A.; et al. *Computergrafik und Bildverarbeitung - Alles für Studium und Praxis*, 2.Aufl., Vieweg, Wiesbaden 2007.

[29] Pulli, K.; et al. *Designing graphics programming interfaces for mobile devices*, In: IEEE Computer Graphics and Applications. Nr. 25 von November 2005, S. 66 - 75.

[30] Pulli, K.. *New APIs for Mobile Graphics*, http://www.research.nokia.com/publication/5834/, o.O., 14.11.2011, 2006.

[31] Pulli, K.; et al. *Mobile 3D Graphics with OpenGL ES and M3G*, Verlag, Burlington, Massachusetts, USA. 2008.

[32] Rost, R. J.; et al (1989). *PEX: A Network-Transparent 3D Graphics System*, In: IEEE Computer Graphics and Applications. Nr. 4 von Juli 1989, S.: 14 - 29.

[33] Schwerin, Claire. *Soldier-driven TIGR to continue expansion under Army leadership*, http://www.army.mil/article/66559/, o.O., 09.12.2011, 2011.

[34] Stawarz, C.. *iGLU: OpenGL Utility Library for iPhone OS - Project Website*, http://code.google.com/p/iphone-glu/, o.O., 16.11.2011, 2008.

[35] Turner, James. *Where 2.0 Preview - DARPA's TIGR Project Helps Platoons Stay Alive*, http://radar.oreilly.com/2009/04/where-20-preview---darpas-tige.html, o.O., 09.12.2011, 2009.

[36] Williams, C. G.; Clingerman, G. W.. *Professional Windows Phone 7 Game Development*, Wiley Publishing Inc., Indianapolis, USA 2011.

[37] Zhou, Z; et al. *Windows Phone 7 Programming for Android and iOS Developers*, John Wiley & Sons Inc., Indianapolis, USA 2011.

Kapitel 10

Augmented Reality

Matthias Kuc

Die Augmented Reality (AR) ist noch eine recht neue Technik, die seit Anfang der 1990er Jahre erforscht wird. Doch erst seit einigen Jahren sind die Geräte so klein und leistungsstark geworden, dass es auch nutzbare Anwendungen gibt. Vor allem die Verbreitung von Smartphones in den letzten Jahren hat die Technik deutlich vorangetrieben, da die Plattform Smartphone wie dafür geschaffen zu sein scheint, AR-Anwendungen auszuführen. Mit einem Ortungssensor, einer integrierten Kamera und einem Gyroskop ausgestattet, erfüllen Smartphones alle Voraussetzungen, um fast alle Anwendungen, die die Augmented Reality bietet auszuführen. Die einzige Begrenzung bei einigen Anwendungen ist der zu kleine Bildschirm. Dass die Technik mittlerweile sogar schon in der Werbeindustrie angekommen ist zeigt, dass sie nicht nur etwas für Technikbegeisterte ist, sondern für die breite Bevölkerung. Viele Smartphoneanwendungen bieten auch für das Militär eine Fülle an Möglichkeiten, die nur darauf warten, in den Truppenversuch zu gehen.
Auf den folgenden Seiten sollen die Aspekte der Technik Augmented Reality, vor allem im Zusammenhang mit Smartphones dargestellt werden.

Inhaltsverzeichnis

10.1 Was ist Augmented Reality? 253
10.2 Geschichte der Augmented Reality 254
10.3 Entwicklungsplattformen 255
10.3.1 ARToolKit 256
10.3.2 ARToolKitPlus 256
10.3.3 Studierstube Tracker 256
10.3.4 metaio 257
10.4 Augmented Reality Markup Language (ARML) 257
10.5 Augmented Reality Anwendungen 258
10.5.1 Ortsbezogene Daten 258
10.5.2 Werkzeuge 262
10.5.3 Werbezwecke und Spiele 266
10.5.4 Militärische Anwendungen 269
10.6 Zusammenfassung 271

10.1 Was ist Augmented Reality?

Augmented Reality ist ein englischer Begriff, der in der deutschen Sprache als *erweiterte Realität* bezeichnet wird. Augmented Reality (AR) ist dabei eine Form der Virtual Reality (VR).
Die VR ist eine virtuelle Welt, die von einem Computer in Echtzeit generiert wird. In dieser virtuellen Welt befinden sich keine Dinge aus der realen Welt, sondern alles ist durch einen Computer generiert. Dabei ist es das Ziel, dass der Benutzer der Applikation komplett in diese virtuelle Welt eintritt und mittels Eingabegeräte des Computers einen Avatar steuert, der den Platz des Benutzers in der Welt darstellt. Somit ist es möglich, in der Welt umher zu navigieren. Mit dieser Technik ist es möglich, entweder eine fiktive oder einen Ausschnitt der realen Welt komplett in einer virtuellen Welt abzubilden. Bei dieser Simulation einer virtuellen Welt werden auch die physikalischen Eigenschaften simuliert, wodurch die Welt noch realer scheint und es dem Benutzer noch einfacher fällt, in die virtuelle Welt einzutauchen.
Bei der AR wird die reale Welt als Grundlage genommen und durch weitere Informationen angereichert, also erweitert. Daher hat sich auch der Begriff *erweiterte Realität* durchgesetzt. Hier kommt es nicht darauf an, ein Abbild der realen Welt, oder eine fiktive Welt zu erschaffen, sondern der Benutzer bewegt sich ganz normal in der realen Welt. Das Anreichern der realen Welt geschieht hier durch Informationen oder Hilfen, die dem Benutzer direkt in das Sichtfeld eingeblendet werden. Somit wird die reale Welt durch die Zusatzinformationen erweitert. Die häufigste Art, die Sinneswahrnehmung zu erweitern ist die visuelle Darstellung, wobei etwas in das Sichtfeld des Benutzers eingeblendet wird. Genauso möglich ist es allerdings auch, den Gehörsinn, den Tastsinn oder andere Sinne anzusprechen. Dafür sind allerdings bisher kaum oder keine Anwendungen bekannt.
In der Literatur wird meistens die Definition von Azuma [1] verwendet, in der die drei Hauptbestandteile der Augmented Reality die folgenden Elemente sind:

- Die Virtualität und die Realität sind miteinander kombiniert.
- Die Interaktivität ist in Echtzeit möglich.
- Die 3-D Verortung.

Unter *3-D Verortung* ist hierbei gemeint, dass die eingeblendeten, virtuellen Objekte zu den realen Objekten in der realen Welt, dreidimensional in Bezug stehen. Die hier benutzte Definition von Ronald T. Azuma [1] ist zwar eine in der Literatur und von Fachleuten allgemein akzeptierte Definition, sie kann jedoch nicht als absolut angesehen werden.
Durch Nutzung der Augmented Reality-Technik ist es also beabsichtigt, die reale Welt mit künstlichen (Informations-)Objekten anzureichern. Dabei soll der Schein entstehen, dass reale und künstliche Objekte zusammen existieren.
Nach der allgemeinen Beschreibung der Augmented Reality kann es zu Falscheindrücken kommen, was AR laut der Definition von Azuma [1] bedeutet. Zum Beispiel könnte man denken, dass Filme, in denen Spezialeffekte genutzt werden, um Objekte einzublenden, eine Anwendung der Augmented Reality sind. Die ist allerdings keineswegs so, da hier die interaktive Komponente fehlt.

10.2 Geschichte der Augmented Reality

Der Begriff *Augmented Reality* geht auf einen Artikel aus dem Jahr **1992** zurück[3]. In diesem wird der Begriff das erste Mal überhaupt verwendet und die Vorteile gegenüber VR dargestellt. Hierbei wird beschrieben, dass weniger Rechenaufwand für AR im Vergleich zu VR benötigt wird, da nicht so viele Bildpunkte gerendert werden müssen. Allerdings fällt in dem Artikel auch schon auf, dass die Ortung des Blickfeldes sehr genau sein muss, um das zusätzliche, projizierte Bild auf einen Punkt oder eine Fläche in der realen Welt abzubilden. Im Speziellen wurde auf die Nutzung von Heads-Up Displays (HUDs) (siehe Abbildung 10.1) eingegangen, die eine bessere Projektion der Informationen in das Sichtfeld des Benutzers ermöglichen, da es keine Bildschirmränder gibt und somit der Eindruck entsteht, die Objekte würden direkt in die reale Welt projiziert werden.

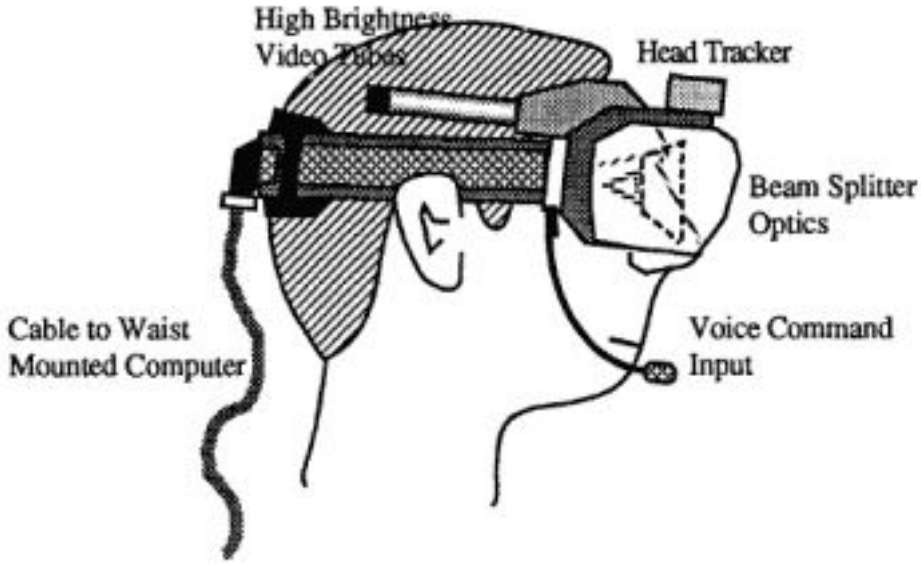

Abbildung 10.1: Skizze eines Heads-Up Displays
Quelle: Christian Doppler Laboratory for Handheld Augmented Reality,
`https://www.icg.tugraz.at/~daniel/HistoryOfMobileAR/`

Im Jahr **1993** ging dann das Global Positioning System (GPS) in Betrieb, wodurch eine genaue Standortbestimmung möglich wurde. Durch die Ortung eines Gerätes sind viele der heutigen Funktionen von AR-Anwendungen erst möglich geworden, die ohne das System undenkbar wären.
1996 wurden das erste Mal 2-D-Matrix Marker für das Kameratracking verwendet. Diese Marker sind kleine zweidimensionale Barcodes, die durch Kameras gut erkennbar sind. Mit Hilfe dieser Marker ist es für den Computer möglich, sich in einem Raum zu orientieren und bestimmte Stellen wieder zu erkennen. Die Marker werden dann im endgültig dargestellten Bild meist ausgeblendet oder überblendet. Auch hat man dazu die Ecken eines Bereiches begrenzt, der in der Ausgabe durch ein Bild überlagert wurde.
Ein Jahr später, im Jahr **1997** wurde das erste mobile AR-System fertig gestellt. Dabei ist der Begriff *mobil* nicht mehr mit dem heutigen Stand der Technik vergleichbar. Damals galt es als mobil, ein HUD zu tragen, eine Steuereinheit in den Händen zu halten, die so groß wie ein Laptop ist und auf dem Rücken als Rucksack einen Computer zu tragen

(siehe Abbildung 10.2).

Abbildung 10.2: Mobiles AR-System aus dem Jahr 1997
Quelle: Christian Doppler Laboratory for Handheld Augmented Reality,
`https://www.icg.tugraz.at/~daniel/HistoryOfMobileAR/`

Im Jahr **1999** kommt das erste Mobiltelefon mit eingebautem GPS-Empfänger auf den Markt, womit der erste Schritt für eine Entwicklung von mobilen AR-Anwendungen getan ist. Außerdem beginnt in diesem Jahr die Entwicklung des *ARToolKit*. Mit diesem Toolkit soll es von nun an einfacher sein, AR-Anwendungen zu entwickeln. Mehr dazu in Abschnitt 10.3.1.
Der letzte elementare Schritt, AR-Anwendungen auf mobilen Geräten einzuführen ist die Marktreife des ersten Mobiltelefons mit eingebauter Kamera im Jahr **2000**. Damit ist es nun möglich, die Welt im Sichtfeld der Kamera darzustellen und Objekte einzublenden. Allerdings lassen die ersten AR-Anwendungen für die breite Bevölkerung noch auf sich warten, da die Rechenleistung in Mobilgeräten noch nicht ausreicht, um die Objekte an sich und dazu auch noch korrekt darzustellen.

10.3 Entwicklungsplattformen

Um AR-Anwendungen, sowohl für Computer, das Web, als auch für Smartphones zu entwickeln, werden meistens spezielle Entwicklungsplattformen genutzt, da die Entwicklung eines eigenen AR-Kerns sehr aufwändig und damit teuer ist. Seit den Anfängen der Nutzung von AR werden diese Toolkits entwickelt und angeboten, damit die Technik möglichst weite Verbreitung findet und vielen Nutzern zur Verfügung steht. Nachfolgend sind eini-

ge Entwicklungsplattformen aufgelistet, die man zurzeit, teilweise kostenlos und teilweise kostenpflichtig, nutzen kann.

10.3.1 ARToolKit

Das *ARToolKit* wird seit 1999 entwickelt und ist der Pionier unter den AR-Toolkits. Mittlerweile wird diese Entwicklungsumgebung unter der Gnu General Public License (GPL) angeboten und ist somit frei zum Entwicklen und Testen verfügbar. Eine Lizenz ist also nur bei kommerzieller Nutzung der Anwendung nötig. Seit 2007 ist allerdings kein Update mehr erschienen, was dieses Toolkit eher nicht für aktuelle Entwicklungen empfiehlt. Es gibt zwei bekannte Anwendungen, die auf Basis des *ARToolKit* entwickelt wurden:
Die Anwendung *Mixare* (mix Augmented Reality Engine), ein AR-Browser, der zum einen auch als Grundlage für eigene Entwicklungen genutzt werden kann, wodurch eine neue Anwendung erstellt wird, zu dem zum anderen aber auch in einer eigenen Anwendung verlinkt werden kann, so dass Geodaten von einer Anwendung an den Mixare-Browser weitergeleitet werden. Mixare ist auch unter der GPL veröffentlicht. Nutzbar ist Mixare unter den Plattformen android und iOS
Eine weitere Anwendung, die auf dem ARToolKit basiert und daher ebenso unter der GPL veröffentlicht ist, ist *AndAR*. AndAR ist eine Anwendung, die ausschließlich für die android-Plattform entwickelt wurde. Mit dieser Anwendung ist es möglich, 3-D-Objekte mit Hilfe eines Markers, der sich im Bild der Kamera befinden muss, zu platzieren. Dieses Objekt bleibt dann auch dort platziert. Eine eigene Erstellung von 3-D-Objekten für diese Anwendung ist möglich.

10.3.2 ARToolKitPlus

Eine Weiterentwicklung des ARToolKits ist das *ARToolKitPlus*. Dieses Framework wurde auf Basis des ARToolKit entwickelt und um viele Funktionen erweitert. Allerdings war es nie für die Öffentlichkeit bestimmt und wurde nur auf Anfrage vieler User veröffentlicht. Es gibt also keinen Support zu diesem Toolkit. ARToolKitPlus ist, da es auf dem ARToolKit basiert auch unter der GPL veröffentlicht. Die Entwicklung des ARToolKitPlus wurde allerdings im Jahr 2006 eingestellt, da das zugehörige Projekt beendet wurde.

10.3.3 Studierstube Tracker

Die Softwarebibliothek *Studierstube Tracker* wurde auf Grundlage des ARToolKitPlus entwickelt, basiert aber nicht auf dessen Code. Die Bibliothek ist eine komplette Eigenentwicklung des Herstellers und daher auch nicht unter einer OpenSource-Lizenz verfügbar. Die Hauptanwendung dieser Bibliothek ist das Tracken von 2-D-Markern und die anschließende Platzierung von Objekten. Die Bibliothek ist derzeit nutzbar auf den Plattformen Windows XP, Windows CE & Windows Mobile, Symbian, Linux, MacOS und iOS.

10.3.4 metaio

Die Firma metaio GmbH mit Sitz in München bietet verschiedene Software Development Kits (SDKs) an, mit denen es möglich ist, auf verschiedenen Plattformen, AR-Anwendungen zu entwickeln. Unter den Referenzen der Firma sind größtenteils Werbeanwendungen zu finden, die die Hauptzielgruppe der Firma zu sein scheint.
Für die Entwicklung von Anwendungen drei SDK zur Verfügung:

- **PC SDK**
  Mit dem PC SDK ist es möglich, AR-Anwendungen für den PC zu entwickeln. Dabei werden die Betriebssysteme Windows XP, Windows Vista und Windows 7 unterstützt. Derzeit ist das SDK zwar unter der 64-bit-Plattform lauffähig, eine native 64-bit-Anwendung ist derzeit in Entwicklung. Diese SDK unterstützt Face-Tracking, Marker-Tracking, 2-D-Bildtracking, 3-D-Objekttracking und Infrarottracking. Dieses SDK ist Modular aufgebaut und kann somit an alle Bedürfnisse angepasst werden und trotzdem noch eine handliche Dateigröße haben.
  Das PC SDK ist kostenpflichtig erhältlich.

- **Web SDK**
  Mit dem Web SDK ist es möglich AR-Anwendungen als Browser-Plugins zu entwickeln. Diese Plugins basieren auf Adobe Flash, daher muss auf dem Gerät, das die Anwendungen ausführen soll das Flash-Plugin installiert sein. Unterstützt werden somit alle Plattformen, die die Flash-Technik unterstützen. Die Erstellung von Flash-Plugins mit der AR-Technik soll sehr einfach sein, da sie auf Schablonen basiert. Das Videobild wird über eine Webcam aufgenommen und dann virtuelle Objekte eingefügt. Trotz Nutzung eines Plugins, ist der Vollbildmodus möglich.
  Das Web SDK ist kostenpflichtig erhältlich.

- **Mobile SDK**
  Im Vergleich zu den anderen beiden angebotenen SDKs ist das Mobile SDK kostenlos erhältlich, lediglich der Support ist kostenpflichtig. Unterstützt werden vom Mobile SDK die Plattformen android und iOS. Das Rendering in der entstandenen Anwendung wird von der integrierten Unity3D-Engine übernommen, wodurch eine sehr aktuelle Technik genutzt wird. Außerdem wird das 2-D und 3-D-Tracking von Objekten unterstützt, das Marker-Tracking ist ebenso möglich. Eine Besonderheit beim Mobile SDK ist die Unterstützung des Gyroskop des Smartphones, wodurch die Schwerkraft realistisch in der Anwendung simuliert werden kann.

10.4 Augmented Reality Markup Language (ARML)

Die Augmented Reality Markup Language (ARML) ist eine Extensible Markup Language (XML)-basierte Auszeichnungssprache zum einfachen und schnellen Erstellen von AR-Inhalten. Die Sprache wurde im Jahr 2009 durch die Firma Wikitude entwickelt, die einen eigenen AR-Browser anbietet (siehe Kapitel 10.5.1). Eine Dokumentation mit Beispielen

zur Entwicklung von Inhalten in dieser Sprache, die bisher nur von dem AR-Browser wikitude darstellbar sind, ist auf der Seite `http://www.openarml.org` zu finden. Zurzeit wird gerade die ARML 2.0 entwickelt, die einige Verbesserungen gegenüber der ersten Version bieten soll. Dies geschieht unter Aufsicht des Open Geospatial Consortium (OGC). Dieses Konsortium ist ein Zusammenschluss von 420 Unternehmen, Forschungsinstituten, Universitäten und öffentlichen Einrichtungen. Das erste offizielle *ARML 2.0 Standards Working Group Meeting* unter der Leitung von Wikitude fand am 19. September 2011 in den USA statt. Ziel der Version 2.0 ist die Standardisierung der ARML und deren Weiterentwicklung. Dabei sollen einige Funktionen hinzukommen, wie zum Beispiel verschiedene AR-Tracking-Methoden und Bilderkennungskomponenten. Verbesserungen sollen zudem bei der dynamischen Modifizierung von virtuellen Objekten, dem Event Handling, der 3D-Visualisierung und der audio- beziehungsweise haptischen Repräsentation stattfinden [4].

10.5 Augmented Reality Anwendungen

In diesem Kapitel sind einige Beispielanwendungen aufgeführt und präsentiert. Dabei soll ersichtlich werden, welche Möglichkeiten diese Technik, vor allem auf dem Smartphone bietet. Allerdings sind hier nicht nur Smartphone-Anwendungen, sondern auch Anwendungen für Computer dargestellt, da denkbar wäre, eine ähnliche Funktionalität auch auf dem Smartphone zu implementieren. Bei der Betrachtung der Anwendungen wurde auch eine potentielle militärische Nutzung der jeweiligen Technik erwogen und wird bei jeder Anwendung einzeln betrachtet.
Aufgrund der Übersichtlichkeit wurden die Beispielanwendungen in vier Kategorien eingeteilt.

10.5.1 Ortsbezogene Daten

Die erste Gruppe von AR-Anwendungen verwendet ortsbezogene Daten. Dabei werden in das Kamerafeld des Smartphones verschiedene Daten eingeblendet, die auf Orte hinweisen.

Wikitude

Die Anwendung *Wikitude*, die von der gleichnamigen Firma entwickelt wird, versteht sich als AR-Browser. Für diesen Browser können Inhalte selbst mit Hilfe der ARML erstellt werden (siehe Kapitel 10.4). Außerdem können beim Starten der Anwendung verschiedene vorhandene Datenbanken genutzt werden, um Geoinhalte anzuzeigen. Beispielsweise können Wikipedia-Inhalte angezeigt werden, die sich in der Nähe befinden (siehe Abbildung 10.3), genau so wie Inhalte der Online-Community Qype (siehe Abbildung 10.4). Ebenso verfügbar sind Inhalte der Dienste Panoramio (Fotografien, die in der Nähe aufgenommen wurden), Youtube (Videos in der Nähe) und Twitter. Diese Inhalte werden mit Sprechblasensymbolen auf Höhe des Horizontes angezeigt und können mit einer Berührung ausgewählt werden, so dass die Inhalte dann gelesen werden können. Die Anwendung

ist sowohl vertikal, als auch horizontal nutzbar (siehe Abbildungen 10.3 und 10.4).

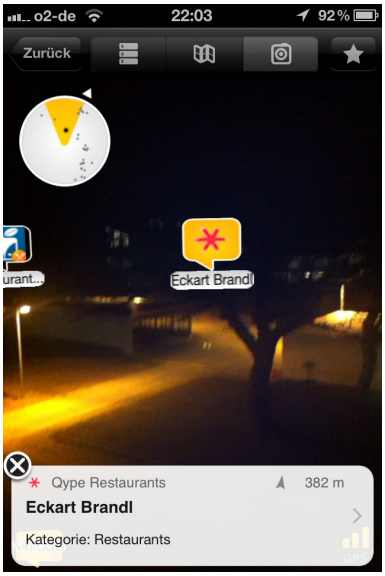

Abbildung 10.3: Screenshot wikitude vertikal Qype-Daten

Abbildung 10.4: Screenshot wikitude horizontal Wikipedia-Daten

AcrossAir

Die Anwendung *AcrossAir* ist der vorherigen Anwendung Wikitude sehr ähnlich. Hier ist es allerdings nicht möglich, eigene Inhalte einzufügen. Dafür ist die Anwendung auf verschiedene Dienste festgelegt, die eine gute Auswahl bieten, wenn man in einer fremden Stadt etwas sucht, wie zum Beispiel ein Restaurant. Der Vorteil dieser Anwendung ist, dass die einzelnen Punkte nach ihrer Entfernung sortiert angezeigt werden. Dabei sind die einzelnen Punkte, die in der gleichen Richtung liegen nicht, wie bei Wikitude auf dem Horizont angeordnet, sondern übereinander. Dabei sind die weiter entfernten Punkte weiter oben, die sehr nahen Punkte weiter unten (siehe Abbildung 10.5). Somit kann man direkt schon durch das Betrachten der Anwendung die grobe Entfernung abschätzen. Zur Präzisierung steht die Entfernung noch an dem jeweiligen Objekt.

Ein kleines Extra der Anwendung ist die Möglichkeit, den Standort des eigenen Autos zu speichern, um es beim nächsten Mal nicht suchen zu müssen (siehe Abbildung 10.6).
Im militärischen Umfeld würde sich diese Anwendung besonders in Verbindung mit einer Offline-Datenbank auf dem Smartphone zu Speicherung wichtiger Wegpunkte eignen. Außerdem könnten wichtige Punkte vermerkt sein, dass ein Durchschlagen im Notfall möglich wird. Ebenso nützlich wären die nächstgelegenen militärischen Stützpunkte und sanitätsdienstlichen Einrichtungen.

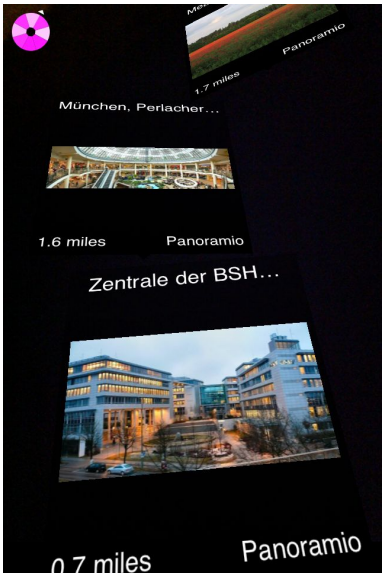

Abbildung 10.5: Screenshot AcrossAir Panoramio-Daten

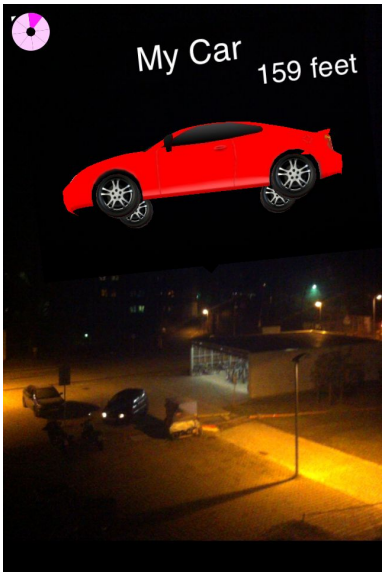

Abbildung 10.6: Screenshot AcrossAir Standort eingenes Auto (selbst gesetzt)

CarFinder

Die Anwendung *CarFinder* bietet die Funktionalität, die in der Anwendung AcrossAir als Zusatz integriert ist. Der Benutzer kann die Position seines Autos in der Anwendung

abspeichern, um dann auf dem Weg zurück zum Auto nicht umher irren zu müssen. Dazu kann der Benutzer das Smartphone parallel zum Boden halten, um dann die Richtung mittels Pfeil angezeigt zu bekommen (siehe Abbildung 10.7). Befindet man sich dann in der Nähe, ist es ratsam, das Smartphone aufrecht zu halten, um dann ein Zielkreuz angezeigt zu bekommen, wo sich das Auto befindet (siehe Abbildung 10.8).

Abbildung 10.7: Screenshot CarFinder Richtung

Abbildung 10.8: Screenshot CarFinder Auto

Plane Finder AR

Mit der Anwendung *Plane Finder AR* lassen sich Flugzeugbewegungen am Himmel identifizieren. Dazu greift die Anwendung auf eine Datenbank der Firma *Pinkfroot* zu. Daher lassen sich auch nur die Flugzeuge identifizieren, die in dieser Datenbank sind. Außerdem

müssen die Flugzeuge den ADS-B-Standard unterstützen, mit dem die Positionsdaten von Flugzeugen über eine Basisstation an einen Pinkfroot-Server gesendet werden. Von dieser Anwendung gibt es eine kostenlose Version, in der die Flugnummer und Registrierungsnummer angezeigt werden (siehe Abbildung 10.9 und 10.10). Außerdem gib es eine kostenpflichtige Version, bei der noch etliche Zusatzdaten angezeigt werden, wie zum Beispiel Geschwindigkeit, Abflug- und Ankunftsflughafen und weitere.
Im militärischen Umfeld würde sich diese Anwendung besonders dazu anbieten, eigene und feindliche Flugobjekte zu identifizieren. Wenn die Aufklärungsdaten elektronisch übermittelt würden, könnte der Fliegerabwehrschütze eigene von fremden Flugzeugen einfacher unterscheiden.

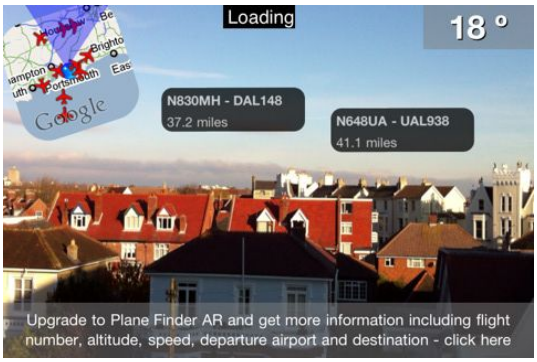

Abbildung 10.9: Screenshot PlaneFinder U.S.-Luftraum
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/gb/app/plane-finder-ar-free/id406950665?mt=8&ign-mpt=uo%3D4`

Abbildung 10.10: Screenshot PlaneFinder U.S.-Luftraum
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/gb/app/plane-finder-ar-free/id406950665?mt=8&ign-mpt=uo%3D4`

10.5.2 Werkzeuge

In dieser Gruppe sind Anwendungen für Smartphones sortiert, die den Benutzer bei einer bestimmten Aufgabe unterstützen und dabei nicht auf ortsbezogene Daten zugreifen.

DanKam: Colorblind Fix

Die Anwendung *DanKam: Colorblind Fix* ist für Menschen mit einer Farbfehlsichtigkeit oder auch Rot-Grün-Blindheit bestimmt. Viele, die Probleme mit der Unterscheidung verschiedener Rot- und Grüntöne haben sind nicht vollständig Rot- oder Grünblind, sondern haben nur eine Sehschwäche. Für diese Personen ändert diese Smartphone-Anwendung die Farbdarstellung, so dass Farbunterschiede wieder besser wahrgenommen werden können. In Abbildung 10.11 ist ein Bild eines Hemdes mit angepassten Farben dargestellt. Besonders schwierig ist es für Personen mit Farbsehschwäche, passende und unpassende Farben zu erkennen, wie zum Beispiel bei der Auswahl der passenden Krawatte zum Hemd. Dabei soll diese Anwendung nun unterstützen. Entwickelt wurde die Anwendung von dem IT-Sicherheitsexperten Dan Kaminsky.

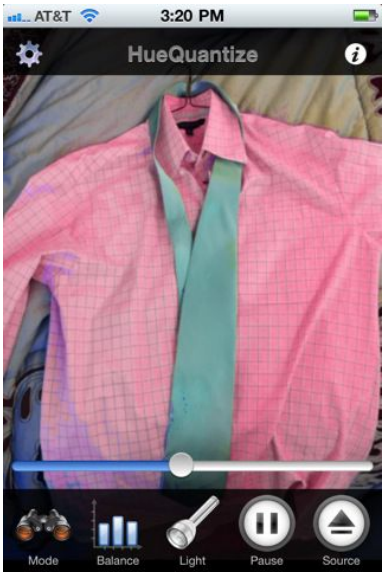

Abbildung 10.11: Screenshot DanKam: Colorblind Fix
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/de/app/dankam-colorblind-fix/id406739331?mt=8`

Magic Plan

Mit der Anwendung *Magic Plan* lässt sich auf sehr einfache Art und Weise ein Grundriss eines oder mehrerer Zimmer erstellen. Dazu wird in der Smartphoneanwendung ein Zimmer angelegt und dann werden mit der eingebauten Kamera die Ecken (siehe Abbildung 10.12) und die Außenseiten der Türen und Durchgänge erfasst (siehe Abbildung 10.13). Dabei ist es wichtig, dass der Benutzer an einer Stelle in dem Zimmer steht, von wo aus alle Ecken sichtbar sind. Einzelne Ecken können auch durch Schränke verstellt sein, da es in der Anwendung Hilfslinien gibt. Die erste Ecke wird doppelt aufgenommen. Anhand der Daten des Gyroskops werden die Abmessungen des Zimmers berechnet. Dies kann durch eine Kalibrierung präzisiert werden. So lassen sich alle Zimmer eines Hauses erfassen und jeweils an den Türen miteinander verbinden. Das Resultat ist ein Grundriss eines ganzen Hauses (siehe Abbildung 10.14). Die Anwendung ist kostenlos, der Export der Pläne ist jeweils kostenpflichtig und wird pro Etage berechnet.

Im militärischen Bereich wäre es mit dieser Technik möglich, schnell und einfach Skizzen zu erstellen, die dann elektronisch vorlägen und einfach versendet werden könnten. Dies könnte eine bessere Planungsgrundlage für die jeweils übergeordnete Führung bedeuten. Auch die Einweisung der eigenen Truppen könnte an dieser Skizze geschehen. Auch ist diese Technik nicht auf Grundrisse beschränkt, sondern es könnten auch Punkte im Gelände markiert werden, deren Position dann übertragen werden könnte.

Abbildung 10.12: Screenshot Magic Plan

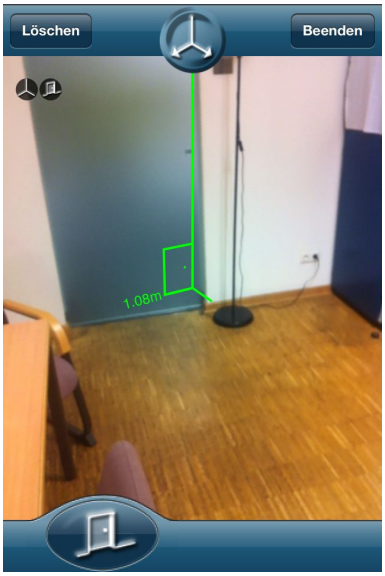

Abbildung 10.13: Screenshot Magic Plan

Theodolite Pro

Die Anwendung *Theodolite Pro* ist speziell für den Outdoor-Einsatz gedacht. Dabei werden im Sichtfeld der Kamera etliche nützliche Informationen angezeigt, wie zum Beispiel

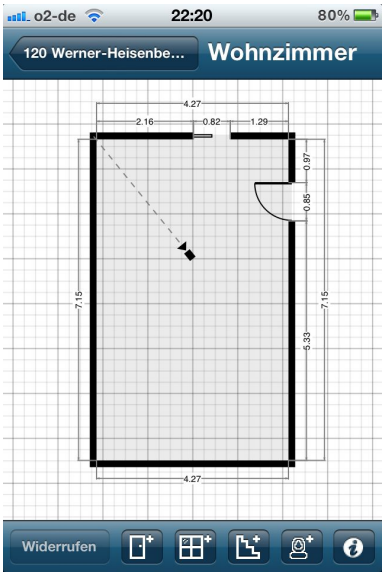

Abbildung 10.14: Screenshot Magic Plan

Kompassdaten, GPS-Daten, Neigungsdaten. Diese Ansicht kann auch in der Anwendung direkt als Foto gespeichert werden und versendet werden (siehe Abbildung 10.15). Zusätzlich ist ein Zoom möglich. In einer weiteren Ansicht kann die GoogleMaps-Karte angezeigt werden, auf der die aktuelle Position mit Blickrichtung angezeigt wird (siehe Abbildung 10.16). Außerdem ist es möglich, die Höhe eines Gegenstandes in der Landschaft zu ermitteln, wenn die Entfernung dazu bekannt ist. Die geschieht durch Markieren des obersten und untersten Punktes. Genau so gibt es noch weitere Verfahren zur Vermessung.
Im militärischen Bereich wäre diese Anwendung gut dafür geeignet, Lagemeldungen abzusetzen, indem ein mit Daten angereichertes Foto versendet wird. Ebenso ist die Technik der Vermessung in der Landschaft zum Abschätzen von Entfernungen interessant.

Abbildung 10.15: Screenshot Theodolite Pro
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/us/app/theodolite-pro/id339393884?mt=8`

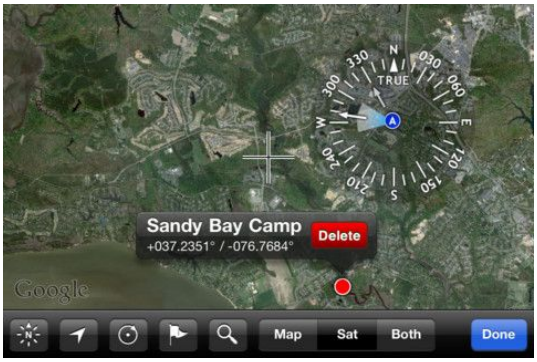

Abbildung 10.16: Screenshot Theodolite Pro
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/us/app/theodolite-pro/id339393884?mt=8`

10.5.3 Werbezwecke und Spiele

In dieser Kategorie geht es um den Einsatz der AR-Technik in der Werbe- und Spieleindustrie. Hierbei sind viele Anwendungen, die nicht nur auf dem Smartphone zu finden sind, entstanden, um den Kunden von dem jeweiligen Produkt zu überzeugen.

Lego Digital Box

Die nächste AR-Anwendung ist zwar keine Smartphone-Anwendung, ist aber ebenso als Anwendung für Smartphones denkbar. Die *Lego Digital Box* ist ein Terminal, was in Spielzeuggeschäften aufgestellt wird mit Bildschirm und Kamera. Wenn man nun eine Verpackung eines Lego-Produktes in die Kamera hält, erscheint das darin befindliche Spielzeug schon fertig aufgebaut als 3-D-Objekt auf der Packung. Man kann sich also den aufgebauten Inhalt schon vor dem Kauf betrachten. Durch Drehen und Verändern des Abstandes von der Kamera kann man sich jedes kleine Detail des Spielzeuges ansehen (siehe Abbildung 10.17).
Die Technik ist dabei ebenso als Smartphone-Anwendung denkbar. Durch den kleinen Bildschirm des Smartphones könnte es nur schwer werden, das Modell genauso detailgetreu darzustellen, wie am Terminal.

Aurasma

Die nächste Anwendung ist wieder eine Smartphone-Anwendung. Ziel der Anwendung *Aurasma* ist es, statische Werbeanzeigen in gedruckten Medien dynamisch werden zu lassen. Dazu soll das Smartphone des Lesers eingesetzt werden. Nach dem Starten der Anwendung auf dem Smartphone richtet man das Bild auf die gewünschte Anzeige, die mit einem kleinen Symbol gekennzeichnet ist. nach einigen Sekunden, in denen die Daten geladen werden, startet ein Film, der genau auf die Anzeige überlagert wird. In Abbildung 10.18 ist dabei das Ausgangsbild einer Anzeige auf dem Computerbildschirm gezeigt, da keine Zeitung mit einer solchen Anzeige erreichbar war. Die Funktionalität ist aber keineswegs

Abbildung 10.17: Lego Digital Box
Quelle: boingboing.net,
`http://gadgets.boingboing.net/2009/01/22/lego-digital-box-aug.html`

eingeschränkt. Nach einigen Sekunden fängt das Bild an, sich wie von Zauberhand zu bewegen. Das abgebildete Fahrzeug fährt bei diesem Beispiel aus dem ursprünglichen Bild in eine andere Landschaft. Kleine Bewegungen beim Halten des Smartphones werden dynamisch angepasst, so dass es scheint, als bewege sich das Bild in der Zeitung, oder bei diesem Beispiel auf dem Computermonitor (siehe Abbildung 10.19).

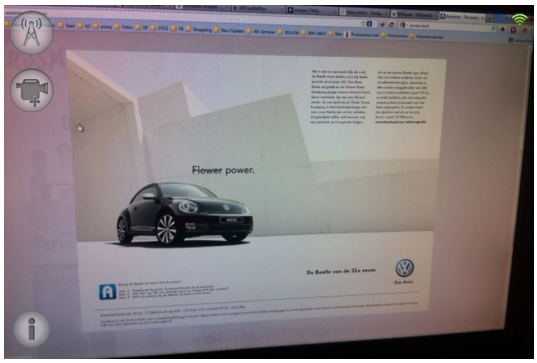

Abbildung 10.18: Screenshot Aurasma Originalansicht

Tissot Reality

Die Firma Tissot bietet eine AR-Anwendung an *(Tissot Reality)*, die nur auf dem Computer per Flash-Plugin zur Verfügung steht. Eine Portierung der Anwendung auf das Smartphone wäre aber ebenso denkbar. Hier soll der potentielle Kunde das Aussehen einer Armbanduhr am eigenen Handgelenk prüfen können, wie er das ja beim traditionellen Juwelier auch kann. Dazu muss eine Pappschablone um das Handgelenk gelegt und befestigt werden (siehe Abbildung 10.20). Auf dieser Schablone ist ein Marker aufgedruckt, auf den dann im Bild, das über die Webcam aufgenommen wird, der jeweils gewünschte Armbanduhrtyp projiziert wird (siehe Abbildung 10.21). Durch diese Technik ist es möglich, mehr Kunden in den Online-Kauf zu locken. Gerade bei so wertvollen Dingen, wie Schmuck, Uhren und Brillen, wo es ganz besonders auch darauf ankommt, wie dieses

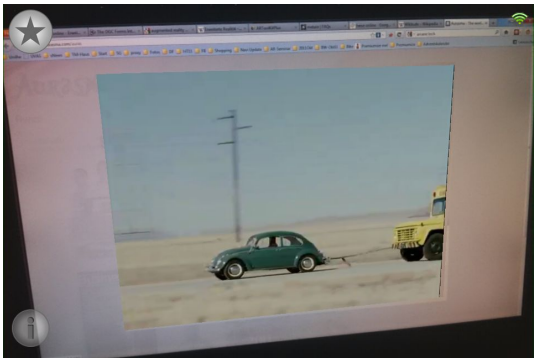

Abbildung 10.19: Screenshot Aurasma überlagertes Bild

Objekt jeweils zum Kunden passt, besteht beim Kunden ein großes Hemmnis beim Kauf, da man ohne Anprobieren sonst nicht abschätzen konnte, wie es jeweils zu einem passt. Das ist nun in Ansätzen möglich.

Abbildung 10.20: Tissot Reality Schablone
Quelle: The Mobile Message-Blog,
`http://themobilemessage.com/2011/07/augmented-reality-freshens-shopping-experience/`

Abbildung 10.21: Tissot Reality Bild
Quelle: CNET-News, `http://news.cnet.com/8301-17938_105-20004279-1.html`

Spiele

Eine weitere Anwendungsmöglichkeit für die AR-Technik stellen Spiele dar, bei denen Objekte in die reale Welt eingeblendet werden. In Abbildung 10.22 ist zum Beispiel das Spiel *iButterfly* zu sehen. Ziel des Spiels ist es, möglichst viele verschiedene Schmetterlingsarten

zu finden. Diese werden, bezogen auf Ort und Zeit eingeblendet und können dann durch eine Bewegung des Smartphones eingefangen werden. Eine weitere Art des Spiels ist *AR Basketball*, bei dem ein Basketballkorb auf einen Schreibtisch (siehe Abbildung 10.23) oder etwas Ähnliches aufgesetzt wird, wo dann der Spieler virtuelles Basketball in der realen Welt spielen kann. Eine weitere Anwendung von AR für Actionspiele gibt es beim Spiel *Sky Siege 3D* (siehe Abbildung 10.24). In diesem ist es möglich, in der eigenen Wohnung einen Hubschrauber zu bekämpfen. Dieser wird virtuell in den Wohnraum eingeblendet. Durch Bewegen des Smartphones muss dieser erst einmal gesucht werden und kann dann bekämpft werden. Eine ähnliche Technik wird bereits bei den U.S.-Streitkräften zum Training verwendet (siehe Kapitel 10.5.4), allerdings nicht auf Basis von Smartphones, da dies zu unrealistisch für die Ansprüche des Militärs ist.

Abbildung 10.22: Screenshot iButterfly

10.5.4 Militärische Anwendungen

Für die Anwendung der AR-Technik, vor allem auch in Smartphones gibt es im militärischen Bereich viel Potential für Anwendungsmöglichkeiten. Dieses Potential ist an den Anwendungsbeispielen in den vorhergehenden Kapiteln teilweise erwähnt. Allerdings ist zu prüfen, ob es sinnvoll ist, diese Technik jeweils zu benutzen und ob diese Technik auch die Anforderungen des Militärs erfüllt. Konkrete Anwendungen im Militär sind eher selten, so nutzen beispielsweise die U.S.-Streitkräfte diese Technik beim Training gegen virtuelle Gegner in realen Umgebungen (siehe Abbildung 10.25). Dies geschieht mit Hilfe von HUDs und per Rucksack aufgeschnallten Laptops (siehe Abbildung 10.26). Da diese Geräte relativ sperrig sind, wäre es mit heutiger Technik möglich, das Ganze in einer kleinen Box unterzubringen. Dadurch könnte weiterhin die normale Ausrüstung getragen werden.

Ein weiteres Anwendungsgebiet für die AR-Technik beim Militär ist die Zielangabe für indirektes Feuer, also Artillerie oder Flugzeugangriffe. Hier könnten Beobachter Ziele mit

Abbildung 10.23: Screenshot AR Basketball
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/us/app/arbasketball/id393333529?mt=8`

Abbildung 10.24: Screenshot Sky Siege 3D
Quelle: Apple iTunes AppStore,
`http://itunes.apple.com/us/app/sky-siege-3d/id349892759?mt=8`

einem Fernglas, in das die AR-Technik eingebaut ist, festlegen. Smartphones sind dafür wegen der fehlenden Genauigkeit wohl eher ungeeignet. Dies könnte dann Aussehen, wie in Abbildung 10.27.

Eine schon vorhandene Software für das Miltär ist die Software *TIGR*, die von der Firma *General Dynamics C4 Systems* entwickelt wurde. Informationen lassen sich darüber leider keine finden, da diese Software für das Militär entwickelt wurde. Allerdings basiert sie auf der Software, die zivil *GeoSuite* heißt. GeoSuite ist eine Command and Control Software, mit der sich bestimmte Aktionen planen und auch mit anderen Nutzern teilen lassen. Dies entspricht der militärischen Lagekarte. In der AR-Software TIGR werden dann diese Daten in das Sichtfeld des Soldaten eingeblendet, bzw. in die Smartphonekamera.
Zur Planung eines Einsatzes kann die Technik AR ebenso sehr gut verwendet werden. Da Flugzeugeinsätze im dreidimensionalen Raum stattfinden, ist es immer schwierig, das Verhalten in nur zwei Dimensionen zu zeigen, also auf normalen Landkarten. Dazu gibt es

Abbildung 10.25: Training U.S.-Streitkräfte
Quelle: Livingston, M. [2], S. 45

eine Software, die eine dreidimensionale Landschaft in das Bild einer Kamera einfügt. Dies geschieht mit einem Marker, der auf einer Platte befestigt ist, die man beliebig drehen kann. So ist es möglich, die Landschaft frei zu drehen, während der geplante Einsatz als Animation dargestellt wird. Damit ist es möglich, sich dieses Szenario von allen Seiten zu betrachten und dadurch auch besser zu verstehen. Ein weiterer Vorteil dieser Technik ist, dass man per Finger etwas anzeigen kann und dazu kein extra Eingabegerät benötigt. Ebenso können weitere 3-D-Objekte zur Laufzeit in die Landschaft eingefügt werden, wie zum Beispiel Pfeile. In Abbildung 10.28 ist ein Screenshot eines Videos dargestellt, das diese Anwendung zeigt.

10.6 Zusammenfassung

Alles in allem lässt sich sagen, dass die Technik der Augmented Reality (AR) mittlerweile sehr ausgereift ist und an einem Punkt angekommen ist, wo sie in vielen Anwendungsgebieten eingesetzt werden kann. Seit der ersten Erwähnung im Jahr 1992 hat sich die Technik so weiter entwickelt, dass mittlerweile selbst ein Smartphone über genügend Rechenleistung verfügt, um AR-Anwendungen flüssig und problemlos auszuführen. Dass die Technik mittlerweile schon in der Werbeindustrie angekommen ist zeigt, dass diese Technik nicht nur etwas für Technikbegeisterte ist, sondern für die breite Bevölkerung. Durch ein Angebot von guten Entwicklungsplattformen ist es mittlerweile auch einfach möglich, für einen beliebigen Anwendungszweck eine passende AR-Anwendung zu erstellen.
Viele Smartphoneanwendungen bieten auch für das Militär eine Fülle an Möglichkeiten, die nur darauf warten, auf die Anforderungen des Militärs geprüft zu werden. Einzelne Versuche der Nutzung von AR im militärischen Bereich sind vorhanden und scheinen vielversprechend für eine künftige Nutzung dieser Anwendungen in den Streitkräften.
In Zukunft wird es wahrscheinlich auch möglich sein, Eingaben über das Kamerabild mit

Abbildung 10.26: Training U.S.-Streitkräfte
Quelle: Livingston, M. [2], S. 45

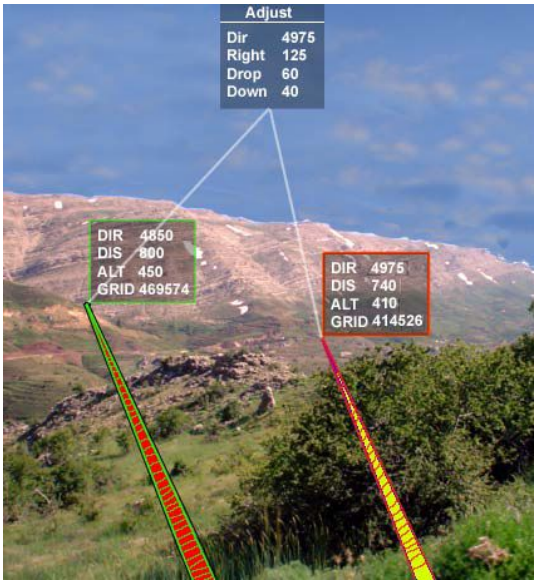

Abbildung 10.27: Militärische Anwendungen der AR
Quelle: Livingston, M. [2], S. 9

Handgesten zu machen, wie es bereits bei Microsoft Kinect möglich ist. Dies bedeutet erneut einen großen Sprung und weitere spannende und interessante Anwendungsmöglichkeiten im Zusammenspiel mit der AR-Technik.

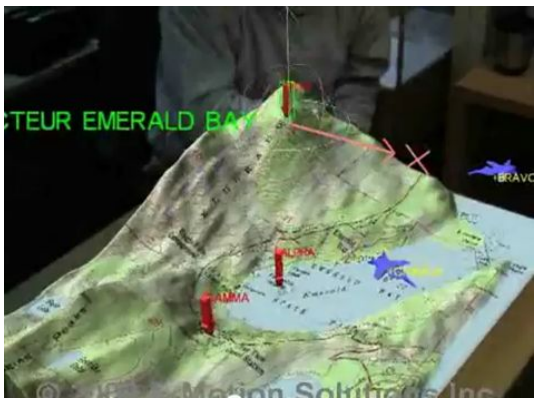

Abbildung 10.28: Screenshot Topografische 3-D-Karte
Quelle: Arcane Technologies inc. `http://www.youtube.com/watch?v=FdC-PRmNcRs`

Literaturverzeichnis

[1] Azuma, Ronald T. *A Survey of Augmented Reality*, in Presence: Teleoperators and Virtual Environments 6, Issue 4 (August 1997), S. 355-385.

[2] Livingston, Mark A. *Augmented Reality: Applications and Current Research Issues*, in Information Science & Technology Colloquium Series (Herbst 2010).

[3] Caudell, Tom P. & Mizell, David W. *Augmented Reality: An Application of Heads-Up Display Technology to Manual Manujfactoring Proceesses*, in Proceedings of 1992 IEEE Hawaii International Conference on Systems Sciences (1992), S. 659-669.

[4] Klass, Christian. *ARML 2.0 - Auf dem Weg zum Augmented-Reality-Standard*, auf `http://www.golem.de/1109/86487.html` (15.09.2011).

Abkürzungsverzeichnis

2D	zweidimensional.
3D	dreidimensional.
API	Application Programming Interface, zu deutsch Programmierschnittstelle.
AR	Augmented Reality.
ARB	Architecture Review Board.
ARML	Augmented Reality Markup Language.
CAD	Computer Aided Design.
DARPA	Defense Advanced Research Projects Agency.
GLU	OpenGL Utility Library.
GPL	Gnu General Public License.
GPS	Global Positioning System.
GPU	Graphics Processing Unit.
HUD	Heads-Up Display.
IEEE	Institute of Electrical and Electronics Engineers.
OpenGL	Open Graphics Library.
OpenGL ES	Open Graphics Library for Embedded System.
OpenGL SC	OpenGL ES Safety Critical Profile.
PC	Personal Computer.
PDA	Personal Digital Assistant.
RIM	Research in Motion.
SDK	Software Development Kit.
SGI	Silicon Graphics Incorporated.
VR	Virtual Reality.

XML	Extensible Markup Language.