Ausarbeitung zum Thema:

Android™ - Googles Betriebssystem für mobile Plattformen

im Rahmen des Proseminars Softwarearchitekturen, SoSe 2011

von Michaela Rindt
Studiengang: AINF-MW DII

Inhaltsverzeichnis

1. Einleitung

Diese Ausarbeitung beschäftigt sich mit dem Thema „Android“, dem von Google entwickelten Betriebssystem für mobile Plattformen wie Mobiltelefone und Tablet-PCs. Im Vordergrund dieser Ausarbeitung steht die Erläuterung der Softwarearchitektur Androids so wie die Darstellung einiger Grundlagen und Voraussetzungen zur Softwareentwicklung für dieses System. Weiterhin werden die Vor- und Nachteile des Systems erläutert. Abschließend werden marktwirtschaftliche Aspekte aufgeführt und ein Fazit über die Zukunftssicherheit von Android-Projekten und des Betriebssystems gezogen.

2. Softwarearchitektur

Unterhalb einer Android-Applikation (bzw. umgangssprachlich Android-“App“) liegt eine ganze Hierarchie von Komponenten und Softwarearchitekturen, von denen der Endbenutzer im alltäglichen Gebrauch natürlich nichts mitbekommt. Diese Hierarchie mitsamt seinen Komponenten soll im Folgenden erläutert werden.

2.1 Linux Kernel

Auf der untersten Schicht kommt der Linux Kernel in der Version 2.6 zum Einsatz und stellt damit die wesentlichen Treiber für die Hardware, also zum Beispiel dem Keypad und dem Display, so wie Kamera-, WiFi-, Flash Memory-, Audio- und IPC-Treiber[1] bereit. Außerdem regelt der Linux-Kernel auch das Power Management, die Speicher- und Prozessverwaltung, Multitasking, Lastverteilung so wie Sicherheitserzwingungen und I/O-Operationen. Damit wird allen auf Linux aufsetzenden Systemen eine einheitliche API zum Zugriff auf Resourcen bereitgestellt, ohne dass diese Systeme die spezielle Hardware im einzelnen 'kennen' müssen.

Der Bootvorgang des Kernels ist in Assembler und der restliche Part größtenteils in C geschrieben[2]. Der Linux-Kernel war für die Entwicklung von Android deshalb so attraktiv, weil sich die Gelegenheit bot, das von Linux bereitgestellte Mehrbenutzersystem als eine Variante des Multitasking für Applikationen zu verwenden. Gleichzeitig ermöglichte dies sichere und isolierte Umgebungen für einzelne Applikationen. Damit standen für Android schon auf Linux-Kernel-Ebene sinnvolle Sicherheitsfeatures zur Verfügung.

Ein weiterer Grund für den Einsatz des Linux-Kernels war, dass dieser Open-Source ist und daher keine Lizenzgebühren abzutreten sind. Die daher niedrigen Produktionskosten schaffen außerdem eine weite und schnelle Verbreitung von Android-Mobiltelefonen bzw. Tablets und tragen zur Umsatzsteigerung Googles bei.

1 [IPC] „inter-process communication“. Beschreibt den Austausch von Daten zwischen mehreren Threads.
2 [Linux-Kernel]

2.2 “Application Libraries“ und die „Android Runtime“

Auf den Linux-Kernel folgen Application Libraries und die Android Runtime.
Die Android Runtime besteht aus Core Libraries zur Java Programmierung und der Dalvik Virtual Machine. In den Core Libraries sind Zugriffsmöglichkeiten wie Datei- und Netzwerk-Zugriffe, diverse Utilities, Grafiken und Datenstrukturen, uvm. definiert.[3]
Die **D**alvik **V**irtual **M**achine, vom Google Mitarbeiter Dan Bornstein unter der Apache Lizenz 2.0 entwickelt und veröffentlicht, ist eine virtuelle Registermachine speziell für mobile Plattformen. Die DVM unterstützt moderne Prozessoren wie den ARM. Daher ist es möglich, einen für die Java VirtualMachine entwickelten Programmcode in DVM-Bytecode umzuwandeln und sehr ressourcenschonend und schnell zu verarbeiten. Die Abarbeitung ist sogar so effizient, dass auf einem Gerät mehrere DVMs gleichzeitig existieren können.
Diesen Vorteil hat man sich in der Android-Entwicklung zu nutze gemacht, um jedem einzelnen Prozess eine eigene DVM zuzuweisen und damit Multitasking zu ermöglichen. Realisiert wird die Konvertierung des Programmcodes mit dem Programm „dx“ des Android SDKs, welches ein oder mehrere Java-Binärdateien (.class) zusammenfasst, optimiert und anschließend in eine Datei des Dalvik-Executable-Format (.dex) umwandelt.[4]

Integrierte Libraries auf dieser Ebene sind:

- Bionic, eine angepasste System C library (von BSD abgeleitet) speziell für Linux basierende, mobile Geräte)[5]
- HAL (Hardware Abstraction Libraries), ermöglicht Nutzung und Integration von proprietären Treibern, welche nicht im Linux-Kernel integriert sind (aufgrund der GPL-Konformität)[6]
- Media Libraries zum Wiedergeben und Aufnehmen von Audio, Video und statischen Bildern in bekannten Formaten wie MPEG4, H263, MP3, JPG, uvm.
- Surface Manager, welcher den Zugriff auf das Display-Sub-System regelt und einen nahtlosen Übergang von Composite 2D und 3D Ebenen in mehreren Applikationen ermöglicht.
- LibWebCore, eine Browser-Engine für den Android-Browser und programmintegrierte

3 [Anatomy-of-Android] S. 58
4 [Dalvik]
5 [Anatomy-of-Android] S. 35-40
6 [Anatomy-of-Android] S. 48-53

Web-Darstellungen

- SGL als grundlegende 2D Grafik-Engine
- 3D Libraries, insbesondere OpenGL ES 1.0
- FreeType zum Rendern von Bitmap- und Vektor-Fonts
- SQLite als relationale Datenbank

2.3 Das Android Application Framework

Alle Android-Applikationen haben Zugriff auf die gleichen API-Funktionen, also auch auf jene, die von den Android Standardprogrammen genutzt werden wie beispielsweise „contacts“, „calendar“, „maps“ und viele andere. Die einzelnen Komponenten können also immer wieder verwendet werden. Fähigkeiten und Daten von einzelnen Applikationen können, natürlich unter Einschränkungen gewisser Sicherheitsschranken, wieder von anderen Applikationen aufgegriffen und verwertet werden. Der „Content Provider“, als Teil des Applikation Frameworks, regelt genau diesen Informationsaustausch zwischen den Applikationen.

Das Application Framework beinhaltet weiterhin eine ganze Reihe von verschiedenen „Views“ zur Anzeige. Dazu gehört auch die Darstellung von Listen, Buttons, integrierter Webbrowser etc.

Ein „Resource Manager“ liefert Zugriff auf nicht-codierte Resourcen wie Grafiken und Layout-Dateien während hingegen der „Notification Manager“ es jeder Applikation ermöglicht, Statusinformationen in der Taskbar anzeigen zu lassen.

Eine weitere wichtige Komponente des Applikation Frameworks ist der „Activity Manager“, der den Lebenszyklus von Applikationen regelt und einen Stack von Navigations-Rückschritten 'aus den Applikationen heraus' bereitstellt.[7]

7 [WhatIsAndroid]

2.4 Die Android-Applikation

Eine Android-Applikation wird in Java geschrieben und kann von den frei erhältlichen Android SDK-Tools zusammen mit den benötigten Ressourcen in ein „Android package“ mit dem .apk-Suffix kompiliert werden. Diese apk-Datei dient dann zum Installieren und Ausführen der Applikation.
Nach der Installation befindet sich die Android-Applikation innerhalb einer Sicherheits-Sandbox. Dabei wird die Mehrbenutzer-Fähigkeit von Linux genutzt, jede Applikation als einen anderen identifizierbaren Benutzer mit bestimmten Privilegien aufzufassen. Darüber hinaus, wird jeder Applikation ein Prozess und insbesondere damit eine eigene Dalvik Virtual Machine zugewiesen, wodurch der Applikations-Code isoliert von anderen Applikationen ausgeführt werden kann.
Prozesse werden vom Android-Betriebssystem nur geladen, wenn sie gefordert werden und beendet, sobald die Notwendigkeit erlischt.

Das Android-System verfolgt „*the principle of least privilege*“, bei dem einer Applikation so wenig Zugriffe wie möglich erlaubt werden. Dadurch wird ausgeschlossen, dass die Applikation unerwünscht auf andere System-Komponenten zugreifen kann.

Falls aber der Informationsaustausch zwischen Applikationen doch gewünscht wird, so ist es möglich, zwei Applikationen die gleiche Benutzer-ID zuzuweisen. Alternativ können Privilegien zum Zugriff auf andere System-Komponenten wie beispielsweise „contacts“, „SMS messages“ bei der Installation der Applikation vom Endbenutzer erfragt werden.[8]

8 DevGuide: Application Fundamentals

3. Softwareentwicklung für Android

Die Softwareentwicklung für Android setzt grundlegende Kenntnisse über die verfügbaren Komponenten des Android-Systems voraus. Insbesondere auch deshalb, weil diese Komponenten einen Einstiegspunkt für die eigene Applikation sein und auch den Lebenszyklus dieser steuern können. Im Folgenden sollen die wichtigsten Komponenten grob skizziert werden.

3.1 Aktivitäten

Eine Aktivität stellt ein User-Interface zu einer Applikation auf einem Bildschirm da (unter Android können bis zu 5 virtuelle Bildschirme genutzt werden). Eine Aktivität der E-Mail Applikation ist zum Beispiel die Darstellung der eingegangenen E-Mails. Eine andere wiederum zum Beispiel das Erstellen einer neuen E-Mail.
Diese Aktivitäten sind unabhängig von einander und können auch, sofern die E-Mail Applikation den Zugriff gewährt, von anderen Applikationen aufgerufen werden. Beispielsweise ist es aus der Kamera-Applikation möglich, die Aktivität zum Erstellen einer neuen E-Mail zu öffnen, um ein gerade erstelltes Foto per E-Mail zu versenden.[9]

3.2 Services

Ein Service dient dazu, langwierige Operationen im Hintergrund auszuführen oder Aufgaben für nicht lokale Prozesse zu erledigen. Dabei stellt der Service keine Benutzeroberfläche bereit.

Ein Service kann daher zum Beispiel dazu dienen, Musik im Hintergrund laufen zu lassen, oder Daten aus dem Internet zu laden, ohne dabei weitere Aktivitäten oder die Nutzung von weiteren Applikationen für den Benutzer zu blockieren. Andere Komponenten wie zum Beispiel Aktivitäten können Services starten oder auch sich an diesen Service „binden“, um eine Interaktion mit diesem herzustellen.[10]

9 [DevGuide] Application Fundamentals: Application Components: Activities
10 DevGuide:Application Fundamentals: Application Components: Services

3.3 Content Provider

Der Content Provider verwaltet den gemeinsamen Zugriff auf Applikations-Daten. Dabei ist es unerheblich, wo die Daten gespeichert sind: ob im Datei-System, in einer SQLite Datenbank, im Internet oder in anderen persistenten Speichern auf die eine Applikation Zugriff hat. Mithilfe des Content Providers können Applikationen die Daten auswählen und auch verändern, sofern der Content Provider dieses Privileg erlaubt.

Ein weiterer Sinn hinter den Content Providern ist, den Zugriff auf private Daten für andere Applikationen zu sperren, denn nur über diesen werden die Zugriffe erst ermöglicht.[11]

3.4 Broadcast Recievers

Ein Broadcast Reciever ist eine Komponente, die auf systemweit gesendete Ankündigungen reagiert. Die meisten Ankündigungen entstehen durch das Betriebssystem, wie beispielsweise die Mitteilung über einen niedrigen Akkustand. Aber nicht nur das Betriebssystem, sondern auch Applikationen können Mitteilungen senden, um zum Beispiel alle anderen Applikationen, die auf die geteilten Daten Zugriff haben, über Änderungen an diesen zu informieren.

Broadcast Recievers haben keine Benutzeroberfläche, sondern können lediglich eine Benachrichtigungsanzeige in der Status Bar erstellen, um den Benutzer auf etwas hinzuweisen. Ein Broadcast Reciever kann darüber hinaus auch einen Service initiieren als Antwort auf ein bestimmtes Ereignis.[12]

3.5 Aktivierung von Komponenten

11 [DevGuide] Application Fundamentals: Application Components: Content providers
12 [DevGuide] Application Fundamentals: Application Components: Broadcast recievers

Die obigen Komponenten werden durch eine asynchrone Nachricht aktiviert, einen sogenannten „Intent“. Diese „Intents“ binden individuelle Komponenten während der Laufzeit aneinander. Ein „Intent“ beschreibt mit anderen Worten die Absicht oder die Anfrage einer Applikation, eine andere Komponente zu aktivieren oder Informationen von ihr zu verlangen.

Für einen Broadcast Reciever definiert ein solcher „Intent“ beispielsweise die Ankündigung, einen Benachrichtigungs-String „Akkustand niedrig“ bei niedrigem Akkustand zu erzeugen.

Der Content Provider hingegen kann nicht durch Intents aktiviert werden. Die Aktivierung muss stattdessen über den sogenannten „ContentResolver“ erfolgen. Der „Intent“ wird in diesem Fall an den „ContentResolver“ gesendet. Sofern das System die Anfrage für „OK“ befindet, werden die Anfragen an den Content Provider vermittelt und die gefragte Komponente aktiviert beziehungsweise Anfragen an diese weitergeleitet. Der ContentResolver dient dabei als abstrahierende Sicherheitsschicht zwischen Content Provider und anderen Komponenten.[13]

3.6 Die Manifest-Datei

Die Manifest-Datei jeder Applikation beschreibt ihre eigenen Komponenten. Mithilfe dieser Datei erkennt dann das Android-System, welche Komponenten es zu jeder Applikation gibt. Erst wenn alle Komponenten erfasst sind, kann das System die Applikation starten.
Die Manifest-Datei muss im XML-Format vorliegen. In ihr werden alle benötigten Zugriffsprivilegien definiert, beispielsweise der Zugriff aufs Internet oder der Lese-Zugriff auf die Kontakte.

Weiterhin wird in der Manifest-Datei das minimale API-Level für die Applikation festgelegt, also welche Android-Version mindestens erforderlich ist.

Darüber hinaus werden hier noch benötigte Hardware- und Software-Features definiert wie zum Beispiel ob eine Kamera oder Bluetooth Services etc. benötigt werden.[14] Einem Android-Gerät, welches nicht über eine Kamera verfügt, wird die Applikation im „Android-Market“ gar nicht erst zur Verfügung gestellt. Die Filterung nach Voraussetzung, ob Hard-/Software oder API-

13 [DevGuide] Application Fundamentals: Application Components: Activating Components
14 [DevGuide] Application Fundamentals: The Manifest File

Level/-Features erfolgt also schon über den „Android-Market“ mithilfe der Manifest-Datei einer Applikation.[15]
Oft werden auch Zugriffe auf andere Libraries wie beispielsweise die Google Maps Library, benötigt. Auch diese müssen in der Manifest-Datei angegeben werden.[16]

In der Manifest-Datei stehen außerdem noch „Intent-Filters“. Diese geben die verfügbaren Aktivitäten an, die nach außen hin für bestimmte Komponenten sichtbar sind. Beispielsweise von einer E-Mail Applikation die „sende“-Aktivität.[17]

3.7 Applikations-Ressourcen

Ressourcen einer Applikation wie Bilder, XML-Templates, Übersetzungs-Dateien und Konfigurations-Dateien für unterschiedliche Android-Versionen oder Ansichten können in den Unterordner „/res“ der Applikation und auch in weiteren Unterordern dazu abgelegt werden.

Dies ermöglicht auch das leichte Erweitern und Austauschen von solchen Ressourcen, ohne im Programmcode etwas ändern zu müssen. Die SDK-Build-Tools weisen jeder Ressource eine einzigartige ID zu, auf die man im Programmcode Zugriff hat. Über den Unterordner „drawable“ werden Bilder erkannt und über den Unterordner „values-fr“ zum Beispiel werden Übersetzungsdateien für die französische Sprache erkannt.[18]

15 [DevGuide] Application Fundamentals: The Manifest File: Declaring application requirements
16 [DevGuide] Application Fundamentals: The Manifest File
17 [DevGuide] Application Fundamentals: The Manifest File: Declaring component capabilities
18 [DevGuide] Application Fundamentals: Application Resources

4. Unterschied zu anderen mobilen Plattformen:

4.1 Vorteile

- Applikationen können Daten über einen ContentProvider teilen: Das bedeutet einen nahtlosen Übergang für den Benutzer, hohe Benutzerfreundlichkeit, auch nicht zuletzt durch Multitasking
- Komponenten können von mehreren Applikationen genutzt werden, daher besteht systemweit eine hohe Wiederverwendbarkeit.
- Daraus folgt, dass es nicht nur einen Einstiegspunkt für Applikationen, sondern immer mehrere gibt. Also keine einzelne main()-Methode.
- Applikation laufen in separaten Prozessen und können durch die einhergehenden Zugriffsprivilegien pro Benutzer/Applikation nicht einfach auf Daten anderer Applikationen zugreifen. Dazu muss erst ein „Intent“ via Sicherheitsschicht, dem ContentResolvers, gesendet und die Erlaubnis und daraus resultierenden Daten angefragt werden.
- Android und die zugehörige SDK-Tools (z.B. Debugger, Emulator, etc.) sind Opensource und damit für jeden frei zugänglich.
- Die Entwicklung / Lizenzierung von Android-Applikationen ist nicht kostenpflichtig
- Es existieren auch kostenlose Tools / IDEs zur Entwicklung, daher besteht die freie Wahl der IDE. Herausragend in seiner Funktionalität ist das ADT Plugin für Eclipse.[19]
- Es ist kein Android-Gerät zum Entwickeln notwendig da ein Emulator existiert. Eine Entwicklung mit Gerät wird aber empfohlen.
- Falls doch ein Android-Gerät zum Testen gewünscht wird, muss unter Windows lediglich ein OEM USB Treiber des Herstellers installiert werden, sofern es sich nicht um ein ADP (Android Developer Phone wie das Nexus S) handelt, für den spezielle USB Treiber von Google bereitgestellt werden. Unter Mac OSX und Linux muss in der Regel kein extra Treiber installiert werden.[20]
- Der Android-Market wird nicht zensiert, wie beispielsweise beim Kontrahenten Apple. Jeder kann kostenlose und kostenpflichtige Applikationen erstellen und anbieten
- Die Zukunftsaussichten bezüglich der Weiterentwicklung durch Google sind gut und

19 [eclipse-ADT]
20 [OEM-usb]

werden schon allein durch die wachsende Zahl von Mobiltelefonherstellern, die auf Android umsteigen, deutlich.

4.2 Nachteile

- Der soeben genannte freie Market, welcher nicht zensiert wird, kann auch dazu genutzt werden, schädliche Software zu verbreiten. Dies ist bereits ein paar mal passiert, allerdings nicht ohne dass Google per entfernter Löschung der Applikationen bei betroffenen Mobiltelefonen eingegriffen hat.[21]

Google verlässt sich dabei allerdings auf Meldungen aus der Community und überwacht den Market nicht selbst.

- Für Smartphones mit älteren Android-Versionen liegt nicht immer ein Update zur nächst höheren Android-Version vor, da sie unter Umständen durch Hardware-Beschränkungen nicht dazu geeignet sind. Sie sind folglich auch nicht kompatibel mit Applikationen für neuere Android-Versionen. Das hat wiederum zur Folge, dass Entwickler, sofern Sie es den wollen und sich auch mit den Unterschieden der einzelnen Android-Versionen auskennen, Anpassungen oder extra Konfigurationen schreiben müssen.

Allerdings ist eine Erweiterung um Ressourcen, wie oben erläutert, relativ schnell erledigt und in Zukunft wird die noch vor einem Jahr weite Spanne zwischen älteren Mobiltelefonen mit Versionen wie 1.6 und 2.0 nicht mehr so groß sein. Grund dafür ist, dass es mittlerweile weniger Android 1.6 Geräte im Gebrauch gibt, und so ziemlich die meisten neueren Handys über die notwendige Hardware-Leistung verfügen, um mindestens Android 2.1 zu verwenden.

Die Sprünge in der Entwicklung von 2.1 zu 2.3 sind zudem nicht mehr so enorm wie von 1.6 zu 2.0. Bleibt allerdings abzuwarten, ob Gerüchten zufolge Google tatsächlich eine Anpassung der 2.x Versionen an die 3.x Version für Tablets anstrebt.[22]

- Definitiver Nachteil bezüglich der Sicherheit von Benutzerdaten auf dem Mobiltelefon ist das Gewähren von Zugriffsrechten bei der Installation durch den Benutzer selbst. Wenn also eine Applikation zu schädlichen Zwecken programmiert wurde und der Benutzer bedenkenlos einfach alle angefragten Privilegien erteilt, ohne sich im Vorfeld mit Bewertungen, Rezensionen etc. auseinanderzusetzen, ist der Nutzer zunächst nicht geschützt. Hier wird sich also auf den Verstand des Benutzers verlassen.

21 [remote-removal]
22 [3.x-for-smartphones?]

5. Wirtschaftliche Aspekte bezüglich Android

Im Mai 2010 stieg die Zahl der aktivierten Android-Smartphones auf 100 Millionen.[23] Zum jetzigen Zeitpunkt werden täglich rund 500.000 Android Smartphones aktiviert und damit sogar 100.000 Geräte mehr als im Mai noch prognostiziert.[24] Mit mehr wie 400.000 „Apps“ im „Android Market“[25] befindet sich Google auf der Überholspur zu seinem Konkurrenten Apple, dessen „AppStore derzeit 425.000 „Apps“ besitzt.[26] Google verfügt also aktuell schon über eine enorme Marktmacht mit weiterhin großem Wachstumspotenzial, und das nicht zuletzt, weil Google Teil und auch Gründer der „Open Handset Alliance“ ist. Die OHA ist ein Zusammenschluss von 80 Firmen, darunter namenhafte wie Samsung, LG, HTC, Motorola und Nvidia, welcher sich um die Entwicklung, Förderung und Ausweitung von offenen Standards bemüht.[27]

Diesem Zusammenschluss und dem scheinbar unaufhaltsamen Wachstum der Marktstellung Googles im Smartphone-Sektor könnte sich lediglich das CPTN Konsortium in die Quere stellen. Dem CPTN, angeführt von Microsoft, gehören mittlerweile die Firmen Apple, Oracle und die EMC Corperation an.[28] Nach der Übernahme von Java-Entwickler SUN im Jahr 2010 durch Oracle klagte Oracle Google an, diversen Java-Code kopiert und für die Entwicklung der Dalvik Virtual Machine benutzt zuhaben. Laut Klageerwiderung Googles habe Oracle Beweise gefälscht und selbst keinerlei Urheberrechtsverletzungen begangen. Sofern keine Einigung stattfindet, wird im November 2011 voraussichtlich der Prozess beginnen.[29] Bleibt abzuwarten, was dies für Auswirkungen auf die Android-Entwicklung haben wird. Genauso könnte eine tatsächliche Partnerschaft zwischen Microsoft und Nokia den Smartphone-Sektor in Zukunft beeinflussen.[30] Auch aktuelle Berichte über mögliche Übernahmen des Blackberry Herstellers RIM durch andere Smartphone-Hersteller zeigen, dass für die nächste Zeit noch einige wirtschaftliche Veränderungen anstehen.[31]

23 [100mioAndroids]
24 [500000Daily]
25 [androlib]
26 [apple]
27 [OHA]
28 [CPTN]
29 [OracleVSGoogle]
30 [M$&Nokia]
31 [RIM]

6. Fazit

Das Android-Betriebssystem ist ein stabiles und umfangreiches System, welches durch seine Vielzahl unterschiedlicher Applikationen in seiner Funktionalität erweitert werden kann. Dadurch, dass die Applikations-Entwicklung frei ist, wird ein großes Entwickler-Publikum angezogen, was langzeitlich gesehen die Auswahl der Applikationen erweitern und vor allem auch viele Alternativen zu bestehenden Applikationen bieten wird. Durch kontinuierliches Hinzufügen neuer Applikationen im Market werden Entwickler von bereits vorhandenen Applikationen dazu ermuntert, die eigene Applikation immer weiter zu verbessern, wenn sie denn „das beliebteste App“ haben wollen. Ein breites, kontinuierlich wachsendes Spektrum von Applikationen muss also nicht unbedingt den Markt 'überschwemmen', wie man es vielleicht vermuten würde, sondern kann auch stattdessen die Qualität der Applikationen im Gesamten anheben, was einen enormen Vorteil für den Endnutzer darstellt.

Lediglich der Sicherheitsaspekt bezüglich der Verteilung von Schadsoftware über den Market ohne jegliche Kontrolle bleibt derzeit ein Manko. Auch wenn Google angibt, betroffene Smartphones von Schadsoftware per Remote-Removal zu befreien, ist erst einmal eine Rückmeldung von Nutzern notwendig. Meiner Meinung nach wird sich Google sicherlich, sofern diese Vorfälle Überhand nehmen, eine besser geeignete Strategie einfallen lassen, während die entfernte Löschung eine vorübergehende Lösung ist.

Die Entwicklung von Applikationen wird dem Programmierer mit einer sehr guten Dokumentation der Komponenten, Services und Zusammenhänge und einer umfangreichen und regelmäßig aktualisierten API erleichtert. Auch dass die Entwicklung von allen Plattformen aus erfolgen kann, ist ein besonderer Vorteil im Vergleich zum Beispiel zu iOS, da Apple einen Mac und eine kostenpflichtige Entwickler-Lizenz zur Entwicklung von iPhone-Apps voraussetzt.

Während die Schlacht um die beste marktwirtschaftliche Stellung der Smartphone-Hersteller noch nicht ausgefochten ist, bleibt jedoch eines sicher: Nämlich dass infolge des Konkurrenzdruckes unter den Betreibern der Endnutzer in den Genuss von kontinuierlich verbesserten Geräten und Software-Features kommt. Fast täglich werden neue Funktionen, Erweiterungen oder Geräte, die mit Android ausgestattet sind, in den Nachrichten angekündigt. Auch auf Apple liegt nun ein stärkerer Leistungsdruck, um seine Marktanteile nicht an Google

zu verlieren. Bleibt nur die Frage, wann auch Facebook im Smartphone-Kampf mitmischen wird, nicht nur zuletzt, um Google Kontra zu seinem neuen sozialen Netzwerks „google+“ zu geben, welches schon jetzt in seiner Entwicklungsphase als würdiger Nachfolger Facebooks erscheint.

Alles in allem, und auch aus eigener Erfahrung, ist Android ein stabiles, benutzerfreundliches und gleichzeitig auch gut konfigurierbares System, welches durch die rasanten Entwicklungsprozesse seitens Google mit immer wieder neuen Funktionen und Erweiterungen überrascht und überzeugt.

7. Quellenverzeichnis

1 [IPC] Stand 30.06.2011

http://en.wikipedia.org/wiki/Inter-process_communication

2 [Linux-Kernel] Stand: 01.06.2011

http://de.wikipedia.org/wiki/Linux_Kernel

3, 5, 6 [Anatomy-of-Android] S. 58, Stand 01.06.2010, PDF: Presentation-Slides

http://sites.google.com/site/io/anatomy--physiology-of-an-android

4 [Dalvik] Stand 01.06.2011

http://de.wikipedia.org/wiki/Dalvik_Virtual_Machine

7 [WhatIsAndroid] Stand 01.06.2010

http://developer.android.com/guide/basics/what-is-android.html

9-18 [DevGuide] Stand 01.06.2011

http://developer.android.com/guide/topics/fundamentals.html

19 [eclipse-ADT] Stand 01.06.2011

http://developer.android.com/sdk/eclipse-adt.html

20 [OEM-usb] Stand 01.06.2011

http://developer.android.com/sdk/oem-usb.html

21 [remote-removal] Stand 22.06.2011

http://www.heise.de/security/meldung/Google-loescht-Android-App-auf-Smartphones-aus-der-Ferne-Update-1028907.html

22 [3.x-for-smartphones?] Stand 22.06.2011

http://phandroid.com/2011/02/03/exclusive-android-ice-cream-details-bits-of-honey-but-not-the-full-comb/

23 [1mioAndroids] Stand 29.06.2011

http://googleblog.blogspot.com/2011/05/android-momentum-mobile-and-more-at.html

24 [500000Daily] Stand 29.06.2011

http://www.andro-phones.com/android-devices-545.html

25 [androlib] Stand 29.06.2011

http://www.androlib.com/appstats.aspx

26 [apple] Stand 29.06.2011

http://www.apple.com/iphone/apps-for-iphone/

27 [OHA] Stand 29.06.2011

http://en.wikipedia.org/wiki/Open_Handset_Alliance

28 [CPTN] Stand 29.06.2011

http://en.wikipedia.org/wiki/CPTN_Holdings

29 [OracleVSGoogle] Stand 29.06.2011

http://www.golem.de/1106/84591.html

30 [M$&Nokia] Stand 29.06.2011

http://www.golem.de/1102/81342.html

30 [RIM] Stand 29.06.2011

http://uk.reuters.com/article/2011/06/21/tech-us-rim-employees-idUKTRE75K70O20110621