A Comparative Study of the Android and iPhone Operating Systems

COP 5611

Travis Wooley

4/12/2010

Presentation of Each System

Android

Development History

In June of 2003 Andy Rubin founded Android Inc. (Elgin, 2005) His goal "was to design a mobile hand-set platform open to any and all software designers." (Markoff, 2007) In July of 2005 Google purchased Android Inc. for an undisclosed sum of money. (Olsen, 2005) At Google Andy Rubin was named "Director of Mobile Platforms" (Markoff, 2007)

On November 5, 2007 the Open Handset Appliance is announced, along with the open source "Android platform" a "software stack" designed to "significantly lower the cost of developing and distributing mobile devices and services." (Open Handset Alliance, 2007) This Alliance is clearly headed by Google.

On October 21, 2008 Google and the Open Handset Alliance release the source code to the complete Android platform under the Apache license. According to Google Android Product Manager Erick Tseng this was designed to coincide with the October 22, 2008 release of the first "Google phone". The G1 released by T-Mobile. (Boulton, 2008)

Since the October 21st release of the Android platform (OS) there have been three major updates bringing us to version 2.1 in January of this year. (Android Developers, 2010a) Those three major updates have added a wide variety of features including: On-screen keyboard, Video recording, Stereo Bluetooth (Android Developers, April), VPN, Accessibility (Android Developers, 2009), Exchange Support, HTML5 Support (Android Developers, 2010b), etc. Google also believes there are at least 18 phones worldwide made by 8 or 9 manufactures that are using the Android OS. (Richtel, 2009)

Hardware

As of this writing the Nexus One is the newest phone to run the Android OS. The Nexus One uses Qualcomm's 1 GHz Snapdragon processor which also has a 600 MHz DSP and Quad-Band GSM/GPRS/Edge, UMTS, HSPA on board. Built in it has 4 GB of DDR ram and 4 GB of flash storage space, it also comes with a user replaceable 4 GB MicroSD Card, In addition to being a Quad-Band phone the Nexus One comes with a FM Transmitter/Receiver, Wi-Fi (A, B, G & N) and Bluetooth 2.1 + EDR, The screen is a 3.7 Inch OLED that runs at 480X800 and is multi touch, It has a built in 5 megapixel camera with LED flash, accelerometer, compass, GPS, Dual microphone for noise cancelation, External Speaker, Headphone Jack, 1400 mAh battery (iSuppli, 2010) (Galan) (Qualcomm) It appears that only two places have done tear downs of the Nexus One and posted them freely online. Neither of those tear downs contained any information about the bus speed of the snapdragon processor. I suspect that this is due to the snapdragon being a monolithic processor that has not yet released those specs.

Thread Management, Interrupts, Interprocess communication, System Calls

The Android OS has a rather sophisticated threading system that puts a lot of responsibility on the developer to build a thread safe application. By default applications including the UI are single threaded. This means that all "long running" tasks within an application must split off a background thread. (Andriod Developers (a)) However, it's somewhat more complicated than that. The Android OS thinks of applications as being made up of combinations of four building blocks. Those building blocks are Activities, or all of the UI elements of the application, Services, which are threads that perform all of the background work needed for the application to perform its job, Broadcast receivers, which are essentially listeners that allow an application to respond to system or application events, and Content providers, which are data sets that the application has made available to other applications. One of the central reasons for this split is that Android OS encourages applications to reuse components that are provided by other applications, thus reducing the need for duplicated code. (Android Developers (b))

In most cases an application will be thought of based on the activity that is currently being displayed and the infrastructure in the background to accomplish the applications tasks are tied to that activity. However, there are some dangers to that mental model, one of the most prominent being that changing the screen's orientation destroys and recreates the activity that is currently displayed. If the application developer is not careful this process will detach the background worker threads from the UI thread. (Burke, 2009)

The Android OS makes extensive use of events to handle the interrupts, interprocess communication and system calls needed to make a cutting edge application. A detailed description of how it all works can be found at (Android Developers (c)). However, a brief overview is as follows. An application fires onCreate, onStart and onResume when it initially begins. Than if at any point another window covers any part of the application onPause is fired. It is highly recommended that applications save their state when onPause is fired as they may not get another chance before being killed. After if onPause is fired if the entire application is no longer visible onStop may be called followed by onDestroy as the application is totally closed. However, onStop and onDestroy may not be called if system resources run low. The OS may simple kill the application. This same basic concept of creating events and firing them at appropriate times is the method used by the Android OS to pass data between threads in a single application. If it is necessary to pass information between applications a Content Provider is required. (Android Developers (d)) System calls are a bit more complex. Some of them are done through the same interfaces that are provided to applications. Others are done by querying classes that are built-in, for instance the android.location.Location class which provides access to the current GPS coordinates.

Memory Management

As the Android OS is built on Java it utilizes garbage collection to prevent memory leaks. However, as is true with all languages that include garbage collection that is not an ironclad protection against leaking memory. As a result the Android OS documentation provides some help on how to avoid those situations that would leak memory even with a garbage collector. (Android Developers (e)) However, the low level memory management is handled by the Linux Kernel itself (version 2.6) (Android Developers (f))

Networking Support/Power Management System

As with memory management the low level details are handled by the Linux Kernel. (Android Developers (f)) Of the currently released Android phones there is WiFi A/B/G/N and Bluetooth as the primary networking functionality. As for power management, while the core Linux functionality is used, there have been some optimizations made to increase battery life. In particular the OS will attempt to reduce power consumption at every opportunity, but it also provides some APIs to the developer that allows them to override this behavior if absolutely needed. (Android Developers, 2010c)

Software Development Kits (SDK)

The SDK for the Android OS is made readily available on the Android Developers website, as well as extensive documentation on how to install and use it. A plugin is also provided for the popular IDE Eclipse. (Android Developers, 2010d) Putting applications into the Android Market requires setting up an account paying Google $25, and uploading your application. Further steps may be needed if you wish to sell your application. (Android Market)

IPhone

Development History

Apple's famed secrecy makes any attempt to get a development history quite difficult, and ensures that it will be incomplete. However, what is known is that in February of 2005 Steve Jobs started the top secret negations with Cingular (now AT&T) that eventually lead to AT&T having the US exclusive contract for the iPhone. In early 2006 Apple started the process of revising OSX to become the iPhone OS. With the understanding that they would have to reduce the OSX size to a few hundred megabytes from the several gigabytes it currently was.

Over the next year and a half Apple spent millions setting up testing environments, building the hardware and the software that would make the iPhone. Apple was so obsessed with secrecy that they did not allow the hardware and the software teams to interact. On January of 2007 Steve Jobs announced the iPhone at Macworld. (Vogelstein, 2009) On May 17, 2007 Apple received FCC approval to sell the iPhone. (AppleInsider Staff, 2007) On June 29, 2007 the iPhone went on sale in the US. (Block, 2007)

When the iPhone was released only Web applications were supported, with a directory of over 200 web applications available, (Gonsalves, 2007) on October 17, 2007 bowing to public pressure Steve Jobs announced that Apple was planning on releasing an SDK that would allow for native applications on the iPhone. (Kim, 2007) On March 6 of 2008 Apple released the SDK. (Apple, 2008)

Since its initial release there have been some major milestones in the iPhone OS. Version 2.0 of the software allowed for apps created with the SDK, an app store, and exchange support. (Keizer, 2009) Version 3.0 added copy and paste, phone wide search, stereo Bluetooth, shake to shuffle, voice memo, MMS, etc. (Moren, 2009)

Hardware

The iPhone 3GS is the latest iPhone as of this writing. The processor in it is a 600 MHz ARM Cortex A8, with a 256 kb cache and a Multi Layer AHB/AXI Bus. It has an impressive array of RF capabilities built in. It has UMTS/HSDPA on the 850, 1900 & 2100 MHz wavelengths, GSM/EDGE on the 850, 900, 1800 & 1900 MHz wavelength; it has Wi-Fi B & G, and Bluetooth 2.1 + EDR. It has a 3.5 inch 480X320 multi touch sensitive screen with 163 points per inch. It has a 3 Megapixel camera, built in compass and GPS, 256 MB of RAM, either 16 or 32 GB of internal storage (no external storage) (RapidRepair) It also has a 1219 mAH battery, built in Microphone, speaker, earpiece and headphone jack. (Dr.Wreck, 2009)

Thread Management, Interrupts, Interprocess communication, System Calls

The iPhone OS has all of the thread management technologies that are now considered standard. You can spawn threads and synchronize them using all the usual technologies like Mutexes, Read-write locks, Distributed locks, etc. (Apple, 2009b) (Apple, 2009a) However Apple strongly discourages using threads in this manner. They feel that the direct programming of threads is too hard and can be made far more efficient by allowing the OS to handle the thread management. The recommend using operation queues. These are queues that you assign tasks to and the OS handles the necessary thread work to make it happen. This allows the OS to more efficiently handle the thread load and processing of tasks. (Apple, 2009c)

The iPhone OS allows for two different methods of handling interrupts. First there is the UIApplicationDelegate protocol which allows an application to be notified of a variety of activities and take appropriate action. Some of the actions that can be responded to are finished loading, low memory warning, orientation changed, about to be deactivated, etc. (Apple, 2009d) The second method is a lower level construct that can be more complicated to use. It is the NSNotification class. This class allows you to be notified of any interrupts, or other custom activities that occur within the OS. However, to take advantage of this you need to be aware of the activities that you are looking for and trap them specifically. (Apple, 2007)

Interprocess communication within the iPhone OS is handled using custom URL handlers. Essentially the reason for this is the iPhone OS only allows one user application to be active at a time, so user applications have to communicate with each other using custom URLs and then specifically look for them via the UIApplicationDelegate protocol. (Grigsby, 2009)

The iPhone OS bans System calls. In some instances they allow controlled system calls via the libSystem library but for the most part system calls are simply not allowed. (Gerbarg, 2009)

Memory Management

The iPhone OS has no built in garbage collection. The closest thing they have to memory management is auto releasing objects. So in short the application developer is required to manage their own memory. (Kosmaczewski, 2009)

Networking Support

The iPhone OS abstracts away all of the networking functionality. However, they do provide a sample class entitled Reachability that allows an application to determine if a networking resource is available and if so on what type of connection. (Apple, 2008) However, Bluetooth is handled entirely separately from Wi-Fi. Bluetooth is handled through the Game Kit Framework. This framework creates an easy interface to find other Bluetooth devices and to share data with them. (Apple, 2009e)

Power Management System

The iPhone OS makes extensive efforts to optimize the power consumption. It will dynamically clock down the CPU, disable Wi-Fi, and baseband radios, deactivate the GPS functionality, etc. In an effort to help developers Apple also provides extensive recommendations to their developers. (Apple, 2010a)

Software Development Kits (SDK)

Apple provides the only official and supported SDK for the iPhone OS. This SDK requires Mac OS X 10.5.7 or later and an Intel processor. In short to develop for the iPhone you must have an Intel based Mac. However, the SDK that is provided contains and IDE, iPhone simulator, samples, compliers, code analyzers, “and more” (Apple, 2010b)

Comparison

Development Environments

Both development environments provide an IDE and a simulator as well as some debugging tools. IPhone development can only officially be done on a fairly recent Mac in Objective C. Compare this with Android development, which is done in the Eclipse IDE which works on all of the major operating systems, using the Java programming language.

Objective C is for all practical purposes only used for Mac development whereas Java is one of the world’s most popular programming languages. (DedaSys, 2001)

There are also significant differences in how the two controlling companies handle their application stores. Apple is famously controlling over what applications they approve and deny, whereas Google is much more permissive and additionally makes it relatively easy for a user to point their device at another application store. (McAllister, 2008)

Porting difficulty

The impression that I have gotten throughout the process of building this paper is that porting to the iPhone is a mixed bag. The Apple Store only requires a binary file, so it is possible to use other languages and compile them into iPhone apps. However, that does not get you around the difficulties of targeting the iPhone specific resources and limitations. On the other hand if you wish to port the application into Object C, you are essentially porting your application to standard C and then making it compatible with the additions that Apple has made and the iPhone specific functionality.

Porting applications to the Android is pretty easy if your application started out in Java. You just have to make the adjustments required to handle Android specific functionality. However, if your application was initially written in a different language the challenge could be much greater, especially if your application was initially written in C or C++. Games are frequently written in C or C++ for performance reasons, and the Java backend for Android is often labeled as to slow for serious game development.

Virtualization

As far as I can tell there is no virtualization technology available for the iPhone OS. Since Apple has been very clear that they will not approve any applications that run interpreted code (Mike, 2008), and has not yet released their own virtualization solution this is not a surprise.

The Android OS is one of the many virtualized Operating Systems that is supported VMware's Mobile Virtualization Platform (VMware).

In short this is a case of what Operating Systems the big virtualization companies have been able to modify to work with. Since the iPhone is very tightly locked down to a particular set of hardware and Apple is very controlling they will probably not ever be virtualized by anyone other than themselves. However, more open Operating Systems like Android already have been virtualized.

Reliability

I'm afraid I was unable to find any quality resources about the reliability of the iPhone OS vs. the Android OS. It seems that people are much more interested in the reliability of the hardware and the supporting network infrastructure than they are in the Operating System itself. Having, said that I can say that in my experience the iPhone OS is reasonably stable, however it does require rebooting once every few weeks. I would guess that this is at least in part due to memory leaks that accumulate over time. However, in this case the lack of multitasking serves it very well.

The Android OS is by its multitasking nature more susceptible to processes conflicting with each other and causing system instability. Judging by the experience of my colleague who has used both the original Android phone and the brand new Nexus One this is true. He had to reboot every couple of days with his first phone. On his newer Nexus One he only has to reboot every few weeks, putting it on par with my iPhone experience.

Security

It is surprisingly difficult to compare the security of the iPhone OS to that of the Android OS. The iPhone only allows one user application to run at a time which in theory allows it to be much more secure. In practice however there have been a number of recent exploits for it (Kumparak, 2010) (Brian, 2010)and the code that protects application data from being stolen has been criticized for being very lax (Messmer, 2010). Additionally, the restriction to one user application at a time has prevented any antivirus programs from working on the iPhone.

By comparison Android's multitasking in theory makes it more open to attack. However, its security model is also much better. Android was able to take advantage of the years of research in locking down the Java Virtual Machine (JVM) in protecting their OS. Also the multitasking nature of the Android OS has allowed antivirus programs to be written for it (Gohring, 2008). At the moment I would say that the two operating systems are on roughly equal footing with regards to security. However, iPhone is more popular and is thus more targeted.

Strong Points

The iPhone OS clearly put a very large emphasis on the user experience, and this is one area where Apple has always excelled. The emphasis on the user makes the iPhone a joy to use. Also the iPhone OS is extremely popular right now. This means that the majority of mobile applications are currently being created for the iPhone OS. This has created an environment where nearly everything that allows you to do nearly everything you would want to on a mobile device through an iPhone application. Also, the iPhone has all of its storage space in a single space. This means that applications and the operating system all use the same partition. On most other "phone" operating systems there is a tiny primary partition and everything else is expected to be installed on memory cards, but most applications are written under the assumption that they will be run on the internal memory only. Not having to perform that juggling process is extremely valuable.

The Android OS has put a premium on being customizable. Thus there are virtually no limits with what you can do with an Android device. If the hardware can handle what you wish to do you can probably write an Android application to do it. The ability to run tasks in the background is a huge plus. There are many things that can only be accomplished by having two or more applications running at the same time and Android allows that. Also, the Android OS allows for memory cards. This means that you are not limited to the internal storage space that comes with the device. It is possible to fill a memory card up with applications and another with music, and switch them out as needed

Weak Points

The iPhone OS does not allow for multiple user applications to be run at the same time. This removes a number of very useful functionalities from the platform. Developing for the iPhone is also limited to just the Mac platform. This makes the entry cost to develop iPhone applications very high for anyone who does not already own the necessary hardware. Also, developing for the iPhone requires learning a language that is not valuable outside of the Apple community. In short developing for the iPhone requires learning a large number of Apple specific items.

The Android OS is already having problems with the fact that applications are not always portable across devices, with a number of developers complaining about how hard cross device compatibility is (Gruman, 2010). Also, Android has run into the problem of having to juggle a tiny amount of internal memory. This is most notable on the brand new Nexus One which has only 192 MB of that is available to install applications on, and most applications can only be installed on the internal memory (Latif, 2010). It is also worth noting that the user interface of the Android OS is not nearly as nice and polished as the iPhone OS. If feels much rougher and is not nearly intuitive. This is one area where Apple has just done a much better job.

Works Cited

Andriod Developers (a). (n.d.). *Painless Threading*. Retrieved March 21, 2010, from http://developer.android.com/resources/articles/painless-threading.html

Android Developers (b). (n.d.). *Application Fundamentals*. Retrieved March 21, 2010, from http://developer.android.com/guide/topics/fundamentals.html

Android Developers (c). (n.d.). *Activity*. Retrieved March 21, 2010, from http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

Android Developers (d). (n.d.). *Content Providers*. Retrieved March 21, 2010, from http://developer.android.com/guide/topics/providers/content-providers.html

Android Developers (e). (n.d.). *Avoiding Memory Leaks*. Retrieved March 21, 2010, from http://developer.android.com/resources/articles/avoiding-memory-leaks.html

Android Developers (f). (n.d.). *What is Android*. Retrieved March 21, 2010, from http://developer.android.com/guide/basics/what-is-android.html

Android Developers. (April, 2009). *Android 1.5 Platform Highlights*. Retrieved February 21, 2010, from http://developer.android.com/sdk/android-1.5-highlights.html

Android Developers. (2009, December). *Android 1.6 Platform Highlights*. Retrieved February 21, 2010, from http://developer.android.com/sdk/android-1.6-highlights.html

Android Developers. (2010b, January). *Android 2.0 Platform Highlights*. Retrieved February 21, 2010, from http://developer.android.com/sdk/android-2.0-highlights.html

Android Developers. (2010a, January). *Android 2.1, Release 1*. Retrieved February 14, 2010, from http://developer.android.com/sdk/android-2.1.html

Android Developers. (2010d). *Dowload the Android SDK*. Retrieved March 21, 2010, from http://developer.android.com/sdk/index.html

Android Developers. (2010c). *PowerManager*. Retrieved March 21, 2010, from http://developer.android.com/reference/android/os/PowerManager.html

Android Market. (n.d.). *Getting Started*. Retrieved March 21, 2010, from http://market.android.com/support/bin/topic.py?topic=15866

Apple. (2008, March). *Apple March 6 Event*. Retrieved February 21, 2010, from http://www.apple.com/quicktime/qtv/iphoneroadmap/

Apple. (2009c, August 2009). *Concurrency and Application Design*. Retrieved March 28, 2010, from http://developer.apple.com/iPhone/library/documentation/General/Conceptual/ConcurrencyProgrammingGuide/ConcurrencyandApplicationDesign/ConcurrencyandApplicationDesign.html#//apple_ref/doc/uid/TP40008091-CH100-SW1

Apple. (2009e, May 26). *Game Kit Framework Reference*. Retrieved March 28, 2010, from http://developer.apple.com/iphone/library/documentation/GameKit/Reference/GameKit_Collection/GameKit_Collection.pdf

Apple. (2010b, February 2). *iPhone SDK 3.1.3*. Retrieved March 28, 2010, from http://www.apple.com/downloads/macosx/development_tools/iphonesdk.html

Apple. (2007, April 02). *NSNotification Class Reference*. Retrieved March 28, 2010, from http://developer.apple.com/iphone/library/documentation/Cocoa/Reference/Foundation/Classes/NSNotification_Class/Reference/Reference.html

Apple. (2008, August 09). *Reachability*. Retrieved March 28, 2010, from http://developer.apple.com/iphone/library/samplecode/Reachability/

Apple. (2009b, May 22). *Synchronization*. Retrieved March 28, 2010, from http://developer.apple.com/iPhone/library/documentation/Cocoa/Conceptual/Multithreading/ThreadSafety/ThreadSafety.html#//apple_ref/doc/uid/10000057i-CH8-SW1

Apple. (2010a, February 24). *The Core Application Design*. Retrieved March 28, 2010, from http://developer.apple.com/iPhone/library/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ApplicationEnvironment/ApplicationEnvironment.html#//apple_ref/doc/uid/TP40007072-CH7-SW56

Apple. (2009a, May 22). *Thread Management*. Retrieved March 28, 2010, from http://developer.apple.com/iPhone/library/documentation/Cocoa/Conceptual/Multithreading/CreatingThreads/CreatingThreads.html

Apple. (2009d, November 17). *UIApplicationDelegate Protocol Reference*. Retrieved March 28, 2010, from http://developer.apple.com/iphone/library/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/Reference/Reference.html

AppleInsider Staff. (2007, May 17). *News Flash: Apple iPhone receives FCC approval*. Retrieved February 2010, 21, from http://www.appleinsider.com/articles/07/05/17/news_flash_apple_iphone_receives_fcc_approval.html

Block, R. (2007, June 3). *iPhone release date confirmed: yours on June 29th*. Retrieved Feburary 21, 2010, from http://www.engadget.com/2007/06/03/iphone-release-date-confirmed-yours-on-june-29th/

Boulton, C. (2008, October 21). *Google Open-Sources Android on Eve of G1 Launch*. Retrieved February 14, 2010, from http://www.eweek.com/c/a/Mobile-and-Wireless/Google-Open-Sources-Android-on-Eve-of-G1-Launch/

Brian. (2010, March 25). *iPhone Security Breached?* Retrieved March 28, 2010, from Mobile Whack: http://www.mobilewhack.com/iphone-security-breached/

Burke, E. M. (2009). *A Simple Android App and a Threading Bug*. Retrieved March 21, 2010, from http://jnb.ociweb.com/jnb/jnbJan2009.html

DedaSys. (2001, March 21). *Programming Language Popularity*. Retrieved MArch 28, 2010, from http://langpop.com/

Dr.Wreck. (2009, June 19). *iPhone 3Gs – Teardown and Analysis*. Retrieved February 2010, 2010, from http://www.phonewreck.com/2009/06/19/iphone-3gs-teardown-and-analysis/

Elgin, B. (2005, August 17). *Google Buys Android for Its Mobile Arsenal*. Retrieved February 14, 2010, from http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm

Galan, W. (n.d.). *Nexus One Teardown*. Retrieved February 28, 2010, from http://www.ifixit.com/Teardown/Nexus-One-Teardown/1654/1

Gerbarg, L. (2009, August 22). *iPhone - is it possible to make system call*. Retrieved March 28, 2010, from stack overflow: http://stackoverflow.com/questions/1315939/iphone-is-it-possible-to-make-system-call

Gohring, N. (2008, November 7). *Google Android Antivirus Available, and Unnecessary?* Retrieved March 28, 2010, from http://www.csoonline.com/article/460867/Google_Android_Antivirus_Available_and_Unnecessary_

Gonsalves, A. (2007, October 11). *Apple Launches iPhone Web Apps Directory*. Retrieved Feburary 21, 2010, from http://www.informationweek.com/news/hardware/mac/showArticle.jhtml?articleID=202401732

Grigsby, D. (2009, January 27). *Apple Approved iPhone Inter-process Communication*. Retrieved March 28, 2010, from http://www.mobileorchard.com/apple-approved-iphone-inter-process-communication/

Gruman, G. (2010, February 22). *Google Android's self-destruction derby begins*. Retrieved March 28, 2010, from InfoWorld: http://infoworld.com/d/mobilize/google-androids-self-destruction-derby-begins-863

iSuppli. (2010, January 08). *Google Nexus One Carries $174.15 Materials Cost, iSuppli Teardown Reveals*. Retrieved February 28, 2010, from http://www.isuppli.com/News/Pages/Google-Nexus-One-Carries-$17415-Materials-Cost-iSuppli-Teardown-Reveals.aspx

Keizer, G. (2009, March 2008). *FAQ: What iPhone 2.0 means to you*. Retrieved February 21, 2010, from http://www.computerworld.com/s/article/print/9067438/FAQ_What_iPhone_2.0_means_to_you?taxonomyName=Mobile+and+Wireless&taxonomyId=15

Kim, A. (2007, October 17). *Steve Jobs Announces 3rd Party SDK for iPhone for February 2008*. Retrieved Feburary 21, 2010, from http://www.macrumors.com/2007/10/17/steve-jobs-announces-3rd-party-sdk-for-iphone-for-february-2008/

Kosmaczewski, A. (2009, January 28). *10 iPhone Memory Management Tips*. Retrieved March 28, 2010, from http://akosma.com/2009/01/28/10-iphone-memory-management-tips/

Kumparak, G. (2010, February 3). *Potentially nasty new iPhone security flaw discovered*. Retrieved March 28, 2010, from Mobile Crunch: http://www.mobilecrunch.com/2010/02/03/potentially-nasty-new-iphone-security-flaw-discovered/

Latif, L. (2010, January 25). *Google Nexus One smartphone*. Retrieved March 28, 2010, from the Inquirer: http://www.theinquirer.net/inquirer/review/1588402/google-nexus-one-smartphone

Markoff, J. (2007, November 4). *I, Robot: The Man Behind the Google Phone*. Retrieved February 14, 2010, from
http://www.nytimes.com/2007/11/04/technology/04google.html?_r=1&hp=&pagewanted=all

McAllister, N. (2008, September 25). *SDK shoot-out: Android vs. iPhone*. Retrieved March 28, 2010, from InfoWorld: http://www.infoworld.com/print/61074

Messmer, E. (2010, February 5). *Researcher: Apple iPhone security, privacy claims exaggerated*. Retrieved March 28, 2010, from Macworld:
http://www.macworld.com/article/146153/2010/02/iphone_security.html

Mike. (2008, March 07). *iPhone SDK restrictions*. Retrieved March 28, 2010, from
http://mcdevzone.com/2008/03/07/iphone-sdk-restrictions/

Moren, D. (2009, June 17). *Hands On With Apple's IPhone 3.0 Software Update*. Retrieved February 21, 2010, from http://www.pcworld.com/printable/article/id,166845/printable.html

Olsen, S. (2005, August 17). *Google buys Android*. Retrieved Feburary 14, 2010, from
http://news.cnet.com/8301-10784_3-5837102-7.html

Open Handset Alliance. (2007, November 5). *Industry Leaders Announce Open Platform for Mobile Devices*. Retrieved February 14, 2010, from http://www.openhandsetalliance.com/press_110507.html

Qualcomm. (n.d.). *The Snapdragon Platform*. Retrieved February 28, 2010, from
http://www.qctconnect.com/products/snapdragon.html

RapidRepair. (n.d.). Retrieved February 21, 2010, from iPhone 3G S Comparison Chart:
http://www.rapidrepair.com/guides/iphone-3g-s-repair/iphone-3g-s-comparison-chart.html

Richtel, M. (2009, May 27). *Google: Expect 18 Android Phones by Year's End*. Retrieved February 14, 2010, from http://bits.blogs.nytimes.com/2009/05/27/google-expect-18-android-phones-by-years-end/

Soules, L. (n.d.). *iPhone 3GS Teardown*. Retrieved February 2010, 2010, from http://www.ifixit.com/Teardown/iPhone-3GS-Teardown/817

VMware. (n.d.). *VMware MVP (Mobile Virtualization Platform)*. Retrieved March 28, 2010, from http://www.vmware.com/products/mobile/features.html

Vogelstein, F. (2009, January 9). *The Untold Story: How the iPhone Blew Up the Wireless Industry*. Retrieved Feburary 21, 2010, from http://www.wired.com/gadgets/wireless/magazine/16-02/ff_iphone?currentPage=all