# Mobile Development for iPhone and Android

A comparison of 3rd party development for the iPhone and Android platforms

P E T E R  G R U N D S T R Ö M


KTH Computer Science
and Communication

Bachelor of Science Thesis
Stockholm, Sweden 2010

# Mobile Development for iPhone and Android

A comparison of 3rd party development for the iPhone and Android platforms

P E T E R  G R U N D S T R Ö M

Bachelor's Thesis in Engineering and Management (15 ECTS credits)
at the School of Industrial Engineering and Management
Royal Institute of Technology year 2010
Supervisor at CSC was Christian Bogdan
Examiner was Stefan Arnborg

URL: www.csc.kth.se/utbildning/kandidatexjobb/teknikmanagement/2010/grundstrom_peter_K10054.pdf

Royal Institute of Technology
*School of Computer Science and Communication*

**KTH** CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

# Abstract

The mobile industry has seen strong development in the last few years, and this has led to that knowledge of mobile development from a few years back is not relevant any more. This puts the industry in a whole new situation. As with every subject this big and versatile it is hard to know where to start especially since there are many platforms and different techniques in use, but which of them are relevant?

My main approach in writing this thesis is to use the technical documentation provided by Apple, the Android Project and to some degree from other developers. These are the documents I used myself when first developing smartphones applications of my own. The knowledge and experience I gathered during the development of these applications will be a large part of this thesis.

Through out this theisis I find that both platforms have their strengths and weaknesses. Android, backed by Google is more technically driven and have better solutions to several hard technical problems. Apple with their iPhone on the other hand have a bigger user base, and has been better at letting developers monetize on their applications. In the end I feel that Android still is the better platform, since the problems it has today is easier to solve then the ones that Apple is facing.

# Contents

**1 Introduction** **1**

1.1 Problem definition . . . . . 1

1.2 Purpose of thesis . . . . . 2

1.3 Delimitations . . . . . 3

1.4 Approach . . . . . 4

**2 Background** **5**

2.1 History of the iPhone . . . . . 5

2.2 History of Android . . . . . 6

2.3 Symbian . . . . . 8

2.4 Windows Mobile . . . . . 8

2.5 RIM BlackBerry . . . . . 9

2.6 Palm webOS . . . . . 9

**3 iPhone OS** **11**

3.1 iPhone OS Architecture . . . . . 11

3.2 iPhone SDK . . . . . 12

3.3 Important Developer Concepts . . . . . 14

3.4 Development Tools . . . . . 16

3.5 Documentation and Developer Resources . . . . . 18
3.6 Distribution . . . . . 19

**4 Android** **21**

4.1 The Android Platform . . . . . 21
4.2 Software Development Kit . . . . . 23
4.3 Native Development Kit . . . . . 25
4.4 Important development concepts . . . . . 25
4.5 Development tools . . . . . 28
4.6 Documentation and Developer Resources . . . . . 30
4.7 Distribution . . . . . 31

**5 Results and Comparisions** **33**

5.1 Result matrix . . . . . 33
5.2 Development Environment and Programming Languages . . . . . 34
5.3 Development Techniques . . . . . 35
5.4 Distribution and Monetization . . . . . 36
5.5 Platform Fragmentation . . . . . 37
5.6 Conclusions and future predictions . . . . . 39

**Bibliography** **41**

# Chapter 1

# Introduction

## 1.1 Problem definition

Because of the strong development the mobile industry has seen the last few years, knowledge of mobile development from a few years back is not relevant any more. As with every subject this big and versatile it is hard to know where to start. There are many platforms and many different techniques in use, but which should one target? I hope to, through this theisis to answear the following questions;

- What alternatives are there when developing for smartphones?
- What are the the strengths and weaknesses of the different smartphone platforms?
- Compare the different smartphone platforms on key areas
  - Development Environment and Programming Languages
  - Development Techniques
  - Distribution and Monetization
  - Platform Fragmentation
- What is the future of smartphones?

I have chosen to focus my comparison to four main areas; Development Environment and Programming Languages, Development Techniques, Distribution and Monetization, and Platform Fragmentation. These are areas that I find necessary for a 3rd party developer to be well versed in, and that are important factors for the success of a platform.

The first area I will focus on is the Development Environment and Programming Languages. This is the base in developing for any platform, mobile or otherwise. I also think that this area will expose differences in philosophy between the two platforms.

The second area, Developer Techniqes, is in some ways related to the first one, since some parts of the development environment and the choice of programming languages determine what development techniques can be used. Another reason to investigate this area is its great interest to anyone trying to get insight in to the world of smartphone development.

One important area if a platform is going to attract a lot of 3rd party developers is if it is possible to make money on that platform. iPhone and Android are in different phases of their development, and it will be interesting to see how this affects distribution and monetization.

Lastly I would like to investigate how Platform fragmentation makes it harder to develop for a platform. I want to study how the situation looks right now, and how it might developing in the future.

There are of course other areas of interest, that might effect 3rd party developers and the success of a platform. But I feel that after comparing these four areas I will be able to give good insight in to the world of smartphone development and make some predictions on where the market is moving.

## 1.2 Purpose of thesis

There are many smartphone platforms out there currently trying to compete for users. Some of these are very successful and have a large market share or an impressive growth rate. Why are these platforms more successful than others? Of course there are several reasons why one platform succeeds and one fail. Traditionally people have been looking at the hardware specifications of a mobile phone when buying it. What applications the phone ran was not as important. Already customers are starting to change this behavior by choosing their mobile phones, not because of its camera, but because of the software it runs. I think this is getting even more common in the future, and I believe that which platforms that will succeed largely depends on the amount of available 3rd party application for that platform.

Through the answers to the questions raised in the problem definition I hope to give the reader a good insight into the world of 3rd party smartphone applications development. The aim is to give an overview of what the different platforms have to offer and what their strength and weaknesses are. I also hope to give good insight

of the technologies and techniques that are relevant and for what purposes they should be used.

## 1.3 Delimitations

I have chosen to limit the subject by only focusing on the iPhone and Android platforms. The main reason for this is that these are the platforms with the strongest growth, and that they are the platforms currently drawing the most attention from developers. The iPhone and Android platforms are the start of a new generation of smartphone operating systems. I feel that they now have matured to a level that makes a comparison of them feasible, and the development rate of the platforms has stabilized at a level where such a comparison would still be relevant for some time after it was written.

It would also have been interesting to study Symbian and Microsoft Windows Mobile further since they control a big part of the smartphone market. The reason that I chose not to study these two platforms more in depth is because of the transition they are facing. Symbian is transitioning from being the world dominant of the smartphone operating systems and is now faced with increased competition and shrinking market share. This at the same time as Nokia is trying to consolidate the earlier fragmented systems based on Symbian into the new Symbian Foundation. I think this is a very interesting move, but there are still no phones running this new platform. This makes it too hard to study, and I do not want to draw conclusions for a platform where not all details are known. The same is true for Microsoft's Windows Mobile. Microsoft has after many years of criticism started developing a whole new mobile platform called Windows Phone 7. The early information that has been made public so far makes it out to be a very interesting platform, but not much is yet known about it and the first phones using the platform will not be released until Q4 2010.

I will also limit myself to study only some parts of the iPhone and Android platforms. In this thesis I focus on 3rd party developers and the knowledge necessary for them to develop and distribute applications for the iPhone and Android platforms. I have chosen not to look at the devices them selves and their specifications. I have also limited the description of the underlying architectures to a level that is relevant for a developer's understanding of the system.

## 1.4 Approach

I started the work on this thesis by advancing my knowledge of developing for iPhone and Android. I had experience with both of them from before, but I felt that I lacked knowledge in some important areas. Since this manly is a comparison of the two different platforms I wanted to develop and distribute an identical application for both platforms. Developing the same application for both systems gave me much knowledge both about them, their differences, and about where to find my sources for this thesis. It also gave me an opportunity to see the differences at each step of the development and distribution cycle.

The application I developed was a travel planner application for Karlstadsbuss. This applications makes use of most important APIs on both platforms. Among these APIs are Maps, Location, the different UI-components and several platform specific APIs for integration, for example APIs for sending ticket SMSs from the application and integrating with the system-wide Android search function.

The majority of the facts in this thesis come from technical papers, reference documentation and other development resources. These are mainly found in the developer section of Apple's and the Android project's homepages. Information about the history of the platforms and other important dates, generally comes from newssources focusing on reporting around these kind of subjects. Some of the news outlets I have used is Wired Magazine, The Wallstreet Journal Online, Engadget and Ars Technica.

# Chapter 2

# Background

## 2.1 History of the iPhone

In January 2007 Steve Jobs, the CEO of Apple, stood on the stage together with Cingular, now AT&T's mobile department, presenting the Apple iPhone. The reactions was unprecedented and the event itself was in many ways the beginning of a paradigm shift in the mobile industry. But the story begins much earlier than that. It begins with Steve Jobs on stage in the same way as he was going to do five years later announcing the iPhone, but this time announcing the first iPod. The iPod was to become one of Apple's biggest successes ever and by 2004 it accounted for 16% of the companies revenue [98].

A few years after announcing the iPod, Apple was beginning to feel the competition from other MP3-player makers as well as cell phone manufacturers who featured built in music players, including Sony Ericsson with their Walkman series. To address this competition Apple teamed up with Motorola and Cingular to create what was to become the Motorola ROKR [2]. The ROKR was supposed to become the music-playing sequel of Motorola's big success story the Motorola RAZOR-series. The collaboration was set up so that Motorola built the entire phone while Apple could focus on writing the music software. The result was a phone that was years behind compared to the competition, that was not seen as aesthetically pleasing and could only store 100 songs [98].

Apple understood that if they were to create a successful music phone, they would have to do it on their own. To be able to create exactly the product he wanted Steve Jobs set out to find an operator in the US who would accept his deal. Verizon declined and Apple ended up back at AT&T [2]. Negotiations were tricky because the of the new situation and Steve Jobs' demands, but they managed to finish an

agreement. The agreement dictated that AT&T would get exclusivity for the device plus a percentage of all music sales to the device. In return Apple would get $10 per customer and month and free hands to create their own product [98].

Around Thanksgiving of 2005 Steve Jobs approached his engineers for the first time with the project. Not yet done with converting Apple's Mac OS X software from running on PowerPC processors to Intel's X86 architecture, they now also had the job of rewriting large parts of it to make it run on an ARM device with limited resources. Apple knew how to make the operating system and interface work but they did not have the competence to design antennas and other mobile components. They decided to still do it on their own, and under extreme secrecy they developed all the necessary equipment. Security was so tight that the people working on the hardware were given dummy software to try on it, and the people working on the software only saw it running on a big black box with a screen [98]. The same was true for the engineers from AT&T who were supposed to test the device against AT&T's network, but were only given a dummy device containing the baseband processor and antenna [2].

The resulting product was expensive, ran only on AT&T's EDGE network, could not search through mails and the browser did not have neither Java nor Flash support. Nothing of this mattered, the customers loved it. Since that day in 2007, all new phones released have been compared to the current version of the Apple's iPhone [98].

Apple has continued to develop their successful concept by releasing new versions of the operating system as well as new versions of the hardware. With the releases of iPhone 3G and later iPhone 3GS, Apple became the de facto standard which every one else is judged against. They also continued to join their success with the iPhone with the iPod by creating the iPod Touch.

In more recent history they took the next step and released the iPad, a tablet based on the iPhone architecture. A tablet is something Apple has been trying to create internally for a long time and rumors have always been circulating. It is still to early too say if the iPad will be the same success as the iPhone, but the sales figure of 300'000 units the first day might give a hint of its future [24].

## 2.2 History of Android

Back in 2005 Google bought Android Inc, a small tech company based in Palo Alto. Android Inc's business idea was to develop a Linux based operating system for cell phones and other mobile devices. Part of the idea was to make an operating system not for a certain hardware manufacturer but a system that anyone could license and

put on their device and that was flexible and upgradeable [20].

The vision behind the operating system was to make a smartphone operating system that enabled the user to access the Internet and to take advantage of his or her position and mobility. This was something that Andy Rubin, the CEO of Android Inc, expressed a need for in a 2003 interview with business week when he said that, "there was tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences" [3].

The acquisition of Android sparked rumors of Google entering the mobile market with a gPhone to compete with Apple's iPhone [94]. These rumors continued in 2007 when Google founded the Open Handset Alliance together with mobile operators and handset manufacturers [1]. They also released the first version of the Android SDK to the public at the same time[1]. Google and the Open Handset Alliance have since subsequently released new versions of the SDK which starting from October 2008 is released under the open source Apache License [20].

At the same time as Google and the Open Handset Alliance released the Android SDK as open source they announced the first readily available handset running Android. The phone was the the T-Mobile G1 and is also known as the HTC Dream or Google G1 dev-phone [54]. It featured a capacitive touch screen, a full hardware QWERTY keyboard and was first released to customers in the US together with T-Mobile. Developers could also buy the phone unlocked and rooted through Google's developer program, under the name Google G1 dev phone.

The appearance of new Android phones was slow to start with but picked up pace during 2009. More handsets were released and there are now Android phones from more than seven manufacturers, including HTC, Motorola, LG, Sony Ericsson and Samsung [83]. These phones are available in basically every market [17]. Most of the Android phones are still only available in the the US market, but that is starting to change with the HTC Desire that is only being available in Europe and Asia at launch [22].

In early January 2010 Google fulfilled the rumors of a gPhone, only that its name now is Nexus One. The Nexus One is branded as Google phone but is manufactured by the Taiwanese company HTC that also made the first Android phone, the HTC Dream [53]. The Nexus One is exclusively sold through Google's phone store and is only available in a few countries including USA, UK and Singapore [52]. Sales of the Google Nexus One is lagging behind both the iPhone and other Android phones, but Google feels the need to have a Android phone with the full "Google experience" [12].

## 2.3 Symbian

Symbian is one of the the oldest smartphone platforms still in use. The Symbian history is on one hand a success story, but on the other hand also very fragmented. Symbian started out as a further development of PSION's operating system EPOC. EPOC was in 1997/1998 transformed into a cooperation between PSION and the phone manufacturing companies Ericsson, Motorola and Nokia [11]. PSION sold their part of the joint venture as early as 2004. Four years later Nokia, who already was the biggest stakeholder in Symbian, chose to buy out the rest of the stakeholders [11]. Symbian is right now the most commonly used smartphone operating system with a market share of 45%, but it has started to shrink [8].

Symbian was from the beginning the name of the operating system, until its consolidation to the Symbian Foundation. From the beginning it had three major graphical interfaces; Nokia S60, Ericsson UIQ and MOAP(S) from the Japanese mobile operator NTT [11]. To face the competition from new smartphone platforms, such as Android and iPhone, Nokia choose to create the Symbian Foundation. The goal of the Symbian Foundation is to create a royalty free open source platform for the development of software to mobile phones and other hand held devices [11].

## 2.4 Windows Mobile

Windows Mobile has traditionally been the second biggest smartphone operating system. But since the introduction of Apple's iPhone and Research In Motion's Blackberry it has dropped to a mere fourth place, with a market share of 7.9% [8]. Windows Mobile is today at the version 6.5.3 and is based on the Windows CE platform. Despite its name it is not built on any earlier version of Windows, but is a completely new operating system [60].

Windows Mobile has been facing a lot of problems and criticism during recent years. Complaints range from bad usability to being hard to develop for and for having bad distribution channels. Manufacturers have been trying to work around these problems with varying results. One of the more successful is HTC with their completely reworked user interface Sense [21], while Sony Ericsson's customized interface is generally seen as a failed attempt to make an outdated platform work.

To face these problems Microsoft has created Windows Phone 7. Windows Phone 7 was originally announced under the name Windows Mobile 7 Series during Mobile World Congress 2010 in Barcelona, and is expected to reach consumers some time during the third or forth quarter of 2010. Windows Phone 7 features a completely redesigned user interface, a proper distribution channel and minimum criteria that the manufacturers phones most fulfill [92].

## 2.5 RIM BlackBerry

Another contestant on the smartphone market is BlackBerry by Research in Motion, RIM. RIM revolutionized the mobile market a few years ago when they introduced their BlackBerry platform centered around push mail. The Blackberry is mostly available in the US and is generally considered a business phone. This is apparent when looking at RIM's other services, particularly their BlackBerry Enterprise Server, which aims to integrate the blackberry phones into the companies existing IT-infrastructure and communication services including Microsoft Exchange, Lotus Domino and Novel GroupWise [61]. Blackberry has a big market share in the US and some other countries but is basically non existent in the rest of the world [8].

## 2.6 Palm webOS

Palm was the king of the long lost era of PDA's. With the advance of more capable phones the need to have a second device for PIM-services, including contact list and calender, decreased. Palm has earlier tried to make the leap into the mobile market with their Plam Treo devices, but failed. Now after several years of financial and technological trouble they are back, with what many believe is their last chance at survival. During the summer of 2009 Palm introduced Palm Pre and its smaller sibling Palm Pixe on the US market. Both devices are based around Palms whole new operating system WebOS [58]. Critics were impressed with the device and operating system but sales have been scarce. With only 1.1 percent of the global smartphone market in Q3 2009 palm is still very much a small player [8].

# Chapter 3

# iPhone OS

## 3.1 iPhone OS Architecture

The Core OS layer is the lowest layer of the iPhone operating system. It consists of a stripped down version of the Mac OS X core optimized for mobile devices [36]. The iPhone OS is a Unix system with standard I/O, POSIX threads and BSD sockets. It also contains services for mobile device Power Management, security and Bonjour, which is a system for zero-configuration discovery of devices and services on IP networks [38].

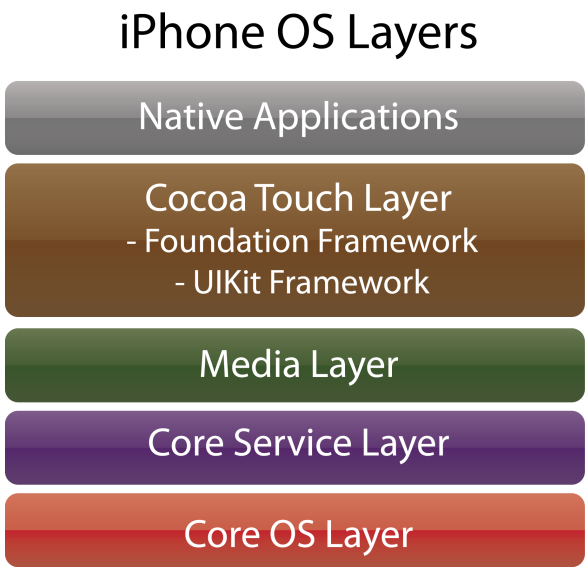


**Figure 3.1.** Schematic picture of the iPhone OS Architecture.

On top of the Core OS layer is the Core Services layer, which provides the fundamental system services that all applications use. One of them is the Core Foundation, which includes a C-based interface for collections made up of lists and dictionaries as well as strings and mutable strings. The Core Service layer also contains frameworks including CoreData which is a framework for a model-view-controller based view on data and Core Location design to exposed location data from sources including GPS, triangulation and WiFi positioning. It also contains the low-level interface towards SQLite and XML parsing [36].

The Media Layer sits in between the lower layers that users never see and the upper layer were the graphical components and applications resides. Most of the functionality in the media layer and above uses the Core Graphics framework for drawing. The Core Graphics framework is some times called Quartz and is the same vector-based drawing API as on Mac OS X. On top of this resides Core Animation that provides a high-level interface for configuring animations and effects, which are then rendered in hardware. The media layer also provides video playback to the application developers. The framework supports the playback of movie files with the .mov, .mp4, .m4v, and .3gp-recordings. Audio can be played and outputted through the Core Audio framework and the OpenGL ES and OpenAL frameworks is regularly used in game development [36].

The layer that 3rd party developers use the most is the Cocoa Touch layer, which can be defined as two sub-layers. The lower sub-level consists of the non-user interface pieces of Cocoa Touch. This is the Foundation Framework, which is a subset of Foundation in Cocoa on Mac OS X, and contains object wrappers of the strings and collections from the Core Services layers as well as some other system services including accessing the file system and networking APIs [27].

The upper portion of Cocoa Touch is the UIKit Framework, which contains the entire Application infrastructure and all the graphical components. The upper part of Cocoa Touch also includes Event handling, Graphics and windowing and Text and web management. The UIKit Framework also enables access for the developer to some of the hardware interfaces including the camera, accelerometer and other sensors on the iPhone [36].

## 3.2 iPhone SDK

The first iPhone that was released did not for the beginning have a SDK for 3rd party development, and the only available applications were the ones developed and pre-installed by Apple. This irritated some users who wished to develop and install homemade applications on the device. In the summer of 2007 something that started out as a way to install custom ringtones [93] evolved with the help of a

iPhone hackers named Jason Merchant in to a way to develop and install 3rd party application applications onto the device [97]. This movement continued to become the iPhone jailbreaking community, which released an unofficial SDK for iPhone and iPod Touch development [100].

The first official SDK announced by Apple was the iPhone SDK 1.2. The 1.2 version was announced on the 6th of March 2008 and was a developer preview sent out to some of Apple's partners. The applications developed with the iPhone SDK 1.2 were at the time mainly for demonstration purposes since there was no way to distribute them at the time [5].

An update to the first official iPhone SDK was released as iPhone SDK 2.0 to 3rd party developers as a part of the iPhone Developer program. The release of 2.0 also coincides with the release of the Apple AppStore for distribution. The release was preceded by a series of beta releases so that developers would have applications ready for release at the launch of the iPhone OS 2.0 for users, which included the AppStore [14].

In the spring of 2009, in preparation for what was going to be the release of the iPhone 3GS, Apple released the first previews of the iPhone 3.0 SDK, which is now considered the baseline of iPhone development. The 3.0 release of the operating system included many new features such as the long awaited copy-and-paste, landscape keyboard in the Mail and Messages applications as well as the possibility to send and receive MMS-messages [13]. Some of these features, for instance the copy-and-paste fuction, reflected similar underlying changes to the SDK. The SDK did not support any form of multitasking for 3rd party applications at the time, but iPhone SDK 3.0 was released with push notifications which provides a way to alert users of new information, even when an application is not currently running. The new SDK also included support for embedding maps directly in applications. Another big new feature in the 3.0 release was the support for in-application purchases. This enabled application developers to charge for extra content inside the application. The iPhone SDK 3.0 also added support for peer-to-peer multiplayer gaming, access for 3rd party developers to the iPod song library as well as OpenGL ES 2.0 [46].

Apple continued its 3.0 series by releasing the iPhone SDK 3.1 in the early fall of 2009. The 3.1 release mainly focused on bug fixes both for users and developers, but some small changes were made. The biggest change was in the CoreAudio-framework that from the 3.0 release onwards allows Bluetooth devices as both input and output. There were also changes to the Camera API and an added failover support for the HTTP Live Streaming [37].

On January 27th 2010 Apple announced and released the first version of the iPhone SDK 3.2. This version is, despite its name, an iPad only release. It adds support for a range of technologies including Popovers, Split Views, Custom Input Views,

External Display Support and Gesture Recognizers that are needed to create iPad applications. iPhone SDK 3.2 features a new framework for text display and text input in the UIKit framework. It also features custom font support and file sharing [47]. Some of these new features will be made available for iPhone developers through the iPhone SDK 4.0 release.

## 3.3 Important Developer Concepts

On the iPhone platform one screen of content is managed by a UIViewController, which usually consists of a root view and one or more sub-views that represent the actual content. If an application only has one screen then the UIViewController is controlled by the application, but in other cases the UIViewController usually have a parent view controller such as a UINavigationController or a UITabBarController. UIViewControllers can either be built programmatically or designed and linked to an Interface Builder file [45].

There are several variations of the UIViewController with UITableViewController being the most common and important. Using the UITableViewController results in a fullscreen list that the developer can fill with content [43]. The list is one of the absolutely crucial design elements on the iPhone, and most application has some part that consist of the users digging down a through hierarchy through the use of lists [14].

UIView is the super-class of graphical components and provides a structure for drawing and handling events. UIView is primarily used in an abstract way but can also be instantiated directly and used as a container to contain other views [44]. The most commonly used subclasses of UIView include components such as UILabel, UITableViewCell, UIButton and UIImageView. Some of these classes are designed to be used directly, for instance the UILabel or UIButton, whilst others can either be used directly or sub-classed, such as in the case of UITableViewCell which represent one cell in an UITableView [42].

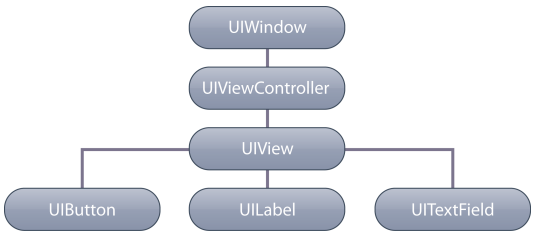


**Figure 3.2.** Hierarchy of a on iPhone application.

To control the navigation between UIViewControllers there is two classes, UINaviga-

tionController and UITabBarController. The UINavigationController automatically adds a navigation bar to the top of the interface that includes information about the current navigation controller such as the title and a back button, which takes the user to the previous view controller. UINavigationController works as a history stack of view controller and a developer can control it by either pushing a new view controller onto it or popping the current, to the user visible, view controller [40].

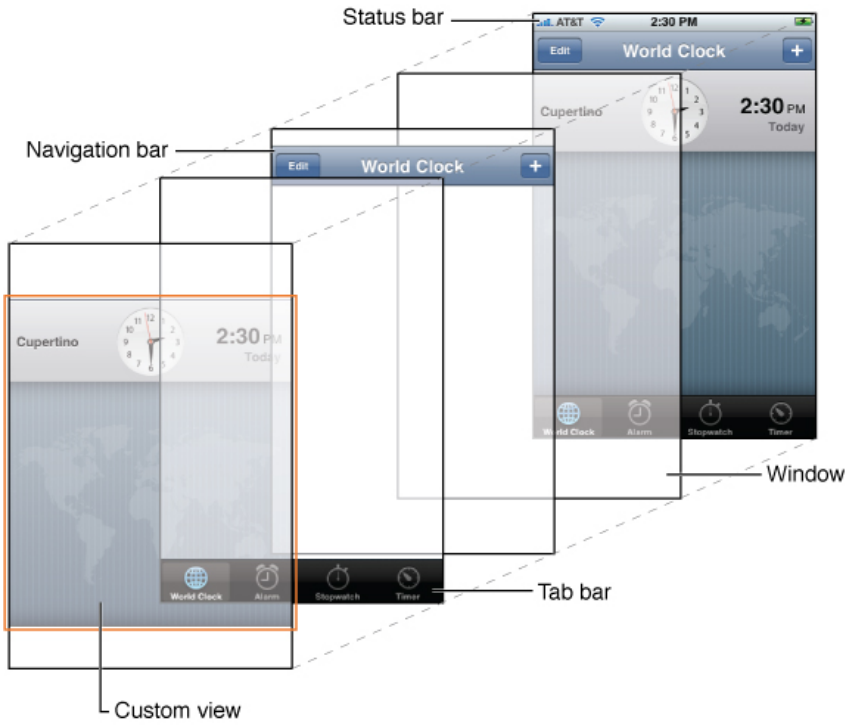


**Figure 3.3.** Schematic image of the different layers of a typical iPhone application.

The other way of controlling navigation between UIViewControllers is with the UITabBarController that adds a bar at the bottom of the screen where the developer can add different tabs. Each of these tabs then represent one view controller. When switching between tabs, the state of the view controller in each tab is preserved [41]. It is also possible to combine the two navigation controllers. The easiest way is to add a UITabBarController and since the UINavigationController in it self is an UIViewController add it as a tab. This gives the effect of having a hierarchal navigation in one or more tabs [14].

## 3.4 Development Tools

All iPhone development is done in XCode, which is Apple's Integrated Development Environment. It was originally built for making Mac OS X applications primarily in Objective-C, but can also be used with programming languages such as AppleScript, Java, Python, Ruby and C/C++. Apple has in the past allowed developers to write games and other applications in tools other than XCode, and other languages such as C#, but now all iPhone development must be made in C, C++ or Objective-C [101]. The XCode IDE only runs on Mac OS X and is together with the iPhone Simulator the main reason that iPhone development is only possible on Intel Mac systems running Mac OS X 10.5 or later [39].

XCode has been around for a long time but XCode 3.1 was the first version that enabled iPhone development [49], and it was release together with the first developer preview of the SDK. XCode has all the features of a modern IDE such as project management, syntax highlighting, code completion, debugging and version control [14]. Also, if used for iPhone development, the XCode IDE is able to remotely debug the application as well as manage packaging of the developed applications for distribution [48].

There has been one more release of XCode since the release of the first XCode version capable of iPhone development. The new version XCode 3.2 was released together with the iPhone SDK version 3.1 and runs on Mac OS X 10.6 codename Snow Leopard [50]. The new version of XCode includes many improvements to the user interface especially when it comes to displaying build errors, both inline and in the build window, as well as to source code navigation. XCode 3.2 also features new functionality such as static analysis of the code and new available compilers including the LLVM version of GCC [50].

Static analysis is a powerful feature that analyses written code to find errors in four different categories; Logic, Memory management, Dead Stores and API usage. Logic errors include use of uninitialized variables and dereferencing of null pointers. The Memory management errors detected includes flaws such as leaking of allocated memory and dead stores errors are raised when a variable is never used. The new static analysis feature of XCode also finds problems involving breaches of the policies imposed by the frameworks and libraries used by the developed application [51].

There are also a number of tools, for instance the Interface Builder, that integrates with XCode to make it easier to build iPhone applications. The Interface Builder is an easy to use tool to build the graphical parts of an iPhone application in a visual way with drag and drop [48]. The integration with XCode also makes it possible to define Outlets and Actions, which connects events from the components defined in the Interface Builder directly to the code of the application [14].

Instruments is a performance analysis tool for Mac and iPhone applications that ships together with XCode. Instruments gather information about disk, memory, and CPU usage in real time, either on a Mac or remotely from a connected iPhone. This makes it possible to find bottlenecks and memory leaks in an application. Another feature in Instruments is that instead of just being able to analyze one aspect such as memory or CPU at the time, it is possible to show the different Instrument tools at the same time in the same view, something that gives developers the ability to see correlations between for instance a spike in CPU usage and a lot of rapid allocations and deallocations of memory [25].

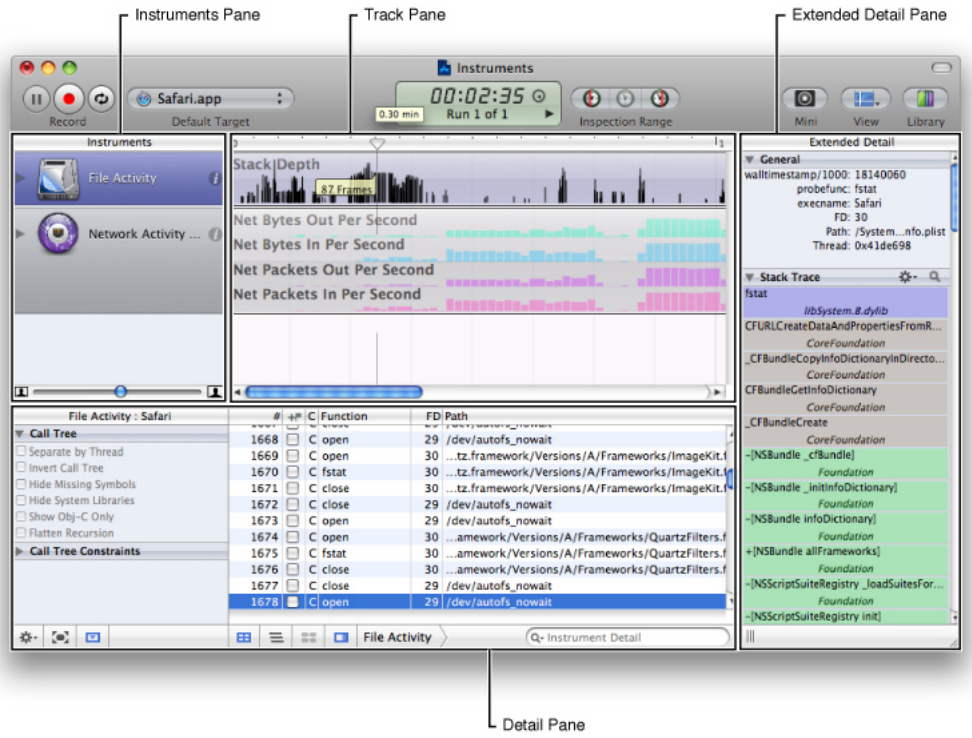


**Figure 3.4.** Example of Instruments running against Safari.

The iPhone Simulator is another tool that makes it easier for developers to test their applications. Since the simulator runs on the same machine as the editor and debugger, the debugging experience get much smoother [48]. The iPhone Simulator is a great time saver, but some times code that works on the simulator does not work on real iPhones. There are differences between the simulator and the actual device related to the fact that the simulator simulates the iPhone API on top of the Mac OS X API instead of emulating the whole device. These differences are mainly around the security model, some small API differences in base classes and that the file system is case sensitive on the iPhone, while it can be either case sensitive or case in-senstive on Mac OS X [64].

## 3.5 Documentation and Developer Resources

3rd party development for the iPhone has been possible since 2008, and this has allowed Apple and the developer community to build up good documentation and developer resources to aid development of iPhone applications.

Apple supplies many different developer resources to help developers get up to speed with iPhone development. One of these is the Getting Started Videos available through iTunes University. Videos are divided in three sections based on the level of the videos; Essential Videos, Advanced Videos and Foundation Videos. Essential Videos is a good start and include videos that introduces a programmer not familiar with iPhone development to the new experience it is to develop on a mobile a device in general and an iPhone in particular. The Advance Videos takes the guides to the next level and delivers a more in-depth view on still quite broad subjects. Lastly the Foundation Videos goes in-depth on specific subject such as multi-touch, game development for the iPhone or how to master the Interface Builder [27].

As a more conventional means of documentation Apple also provides a big selection of Getting Started Documents and Guides. The Getting Started Documents covers larger areas of the iPhone development such as Network and Internet, Graphics and Animations, and Data Management. These documents then links to other guides or parts of the reference library were the necessary knowledge for that subject can be found. The guides are more detailed and covers subjects including View Controller programming, Core Data Model, Data Migration and Low-level File Management [33].

Apple also provides Coding How-To's and Sample Code to help development. The Coding How-To's are set up more as a FAQ, where short regular questions such as "How do I format dates and numbers?" or "How do I write to an SQLite database?" gets answered with a section of code and/or a link to the iPhone Reference manual or one of the many Guides of Getting Started Documents provided [32].

The Sample code section provides whole projects including code and other resources for some sample applications. These applications often cover one or two topics such as in the case of MapCallouts or TableSearch, or it covers the process of using one of the frameworks, for instance the UICatalog example that shows more or less every view component available to the developer in action [35].

To solve more detailed problems of topics not covered in the other documentation sections there is the iPhone Reference Library. The iPhone Reference Library is very similar to a normal reference library and covers every class and protocol (Protocols are the Objective-C equivalent of Java or C++ Interfaces Java or C++ Interfaces). The iPhone Reference Library gives a good overview of the functionality of every class and protocol as well as all the methods associated with it. It describes the

functionality of every method as well as its method signature [35].

For those cases when a developer needs extra help, or if it is unclear which solution is the best, there are the Apple Developer Forums. The Apple Developers Forum is only available to paying members of the iPhone Developers Program. The forum is divided in different sections where developers can discuss problems and ideas with each other, but also get in contact with Apple to get their view on an issue [34].

Some of the official Apple documentation including the Getting Started Documents and the iPhone Reference Library is available freely to the public. Apple has also released a very small subset of all the Coding How-To's and Coding Samples, but to get access to all the official documentation and developer resources such as the Apple Developer Forums, one needs to be a member of the Apple iPhone Developer Program [14].

## 3.6 Distribution

Only members of the iPhone Developer Program are allowed to distribute their applications through the Apple AppStore. There are two variations of the iPhone Developer Program. The cost of the normal iPhone Developer Program is a yearly fee, which currently is $99 and at this point is translated to SEK 795. The normal version of the developer program enables distribution of free and paid applications through the AppStore and can be bought either as a company or an individual, but the price and the content of the program is the same [29]. The other variant of the developer program is the iPhone Developer Enterprise Program, which is more expensive at $299 a year and does not enable distribution in the AppStore. The enterprise program is made for large corporations that want in-house distribution of iPhone software. To be eligible for the enterprise program, a company must have more than 500 employees and be registered with the US government through the DUNS program [28]. Common for the two variations of the iPhone developer program is that they grant access to the iPhone Dev Center Resources, the iPhone SDK, pre-releases of the iPhone OS and development tools and the Apple developer forums [30]. There is also a variation of the iPhone Developer Program for Universities that gives access to the iPhone SDK, all the tools and the Apple developer forums, but it does not give permission to do any kind of distribution more than for basic device testing [31].

The main channel for distribution on the iPhone is the AppStore. The AppStore was first released with the version 2.0 of the iPhone OS and now features more than 185 000 applications in 20 categories that have been downloaded in total more than 4 billion times [24]. In the AppStore there are the concept of both free and paid applications. Paid applications can be priced at 85 different price tiers going

from $0.99 to $999 [26]. As of the 6th of April 2010 73% of all iPhone applications were paid applications, with an average price of $3.55. Games are the most popular category on the iPhone with about 56% of all the applications. It also features a lower average price, in the vicinity of the $1.99 price tier [7].

With the release of iPhone OS 3.0 Apple introduced one more way for developers to monetize on applications in the AppStore through in-application purchases, which made it possible for developers to charge the user for extra material in the application after installation. Some examples of this is the "SVD Reseguide"-application that is free but charges the user for every travel guide he or she downloads [63]. There is also VinVin.se that is free to use but charges a subscription fee for access to the wine reviews [99].

Before a 3rd party developed application can be released through the Apple AppStore it has to go through the approval process. This process has been accused of being slow, anti-competitive and arbitrary. Although the process is faster now than it was in the beginning, it still takes approximately 4-5 working days to get an application approved, given that it does not break any of the rules in the iPhone Developers Agreement [14]. Apple has also been accused of being anti-competitive when it rejects applications that they feel duplicate functionality provided by them. One of the most public examples of this is Apple's rejection of the iPhone version of Google Voice [57], that they felt duplicated functionality already in the iPhone OS. Some of Apple's decisions have been seen as highly arbitrary, such as in the recent case of Apple banning Pulitzer Prize winning cartoonist Mark Fiore's application because his satire "ridicules public figures", which they felt was a violation of the iPhone Developer Program License Agreement [59]. This decision has since then been withdrawn due to the bad PR it has been generating for Apple, and they have now asked Mark Fiore to re-submit his application [95].

# Chapter 4

# Android

## 4.1 The Android Platform

Android is more than the name of the operating system running on Android phones. The Android platform consists of several well-defined layers spanning from the Linux kernel to the built in applications that are shipped with every device. The Android platform is to this date available on ARM and x86 platforms, but could in theory be ported to any hardware platform supported by Linux [91].

The Android operating system is built on top of the Linux Kernel. The kernel, starting at version 2.6.27 for Android 1.0, is heavliy modified and not available in the normal Linux kernel, the changes are instead distributed by the Open Handset Alliance. The modified Linux kernel provides a software abstraction over the hardware and provides drivers for smartphone hardware such as the screen, camera and 3g-modem as well as power management and low-level security. 3rd party developers never directly touch this part of the Android platform, only manufacturers do [91].

Android also consists of a range of Libraries used by different components in the platform. These include well-known libraries such as libc, SQLite as a database, FreeType for font rendering, WebKit for html rendering as well as libraries for media playback and 2D/3D drawing. These libraries are abstracted away by the SDK or used directly by 3rd party developers through the NDK [91].

The Android Runtime is made up of two parts, the Software Developer Kit, and the Virtual machine that executes applications written with the SDK. The SDK is exposed to the 3rd party developer as a group of Java frameworks and classes. The virtual machine is called Dalvik and is similar to a Java Virtual Machine with the difference that it does not run Java byte code directly, but instead the Dalvik

bytecode that the Java bytecode is converted to during compilation time. The Dalvik Executable format (.dex) is optimized for memory footprint made to run on the Dalvik VM, which is register-based and runs each application in a new instance of the VM [91].

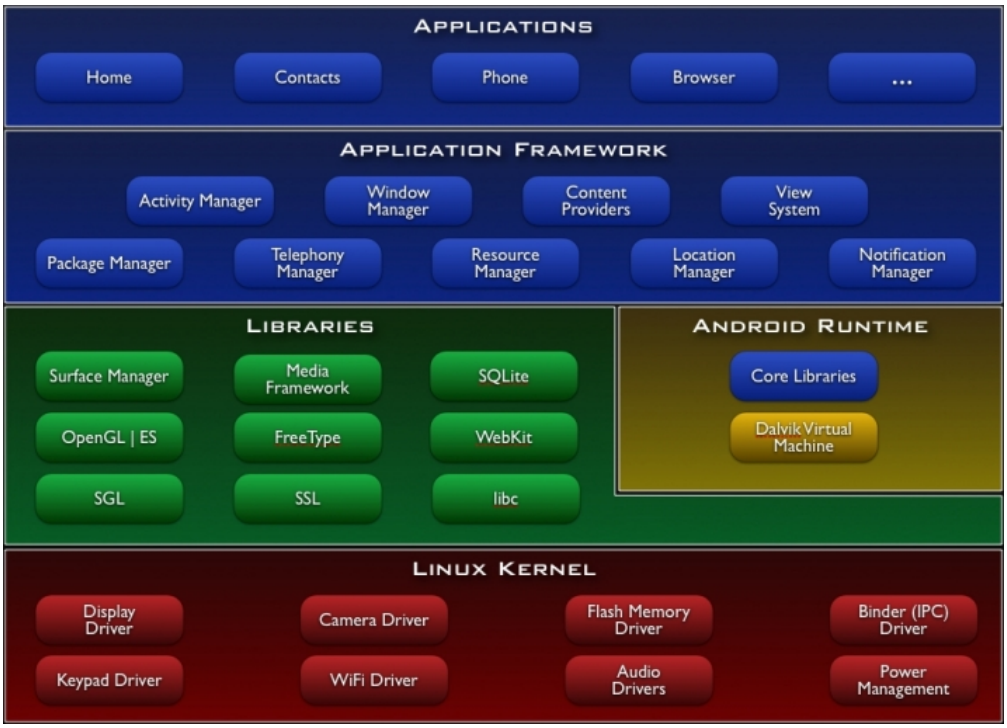


**Figure 4.1.** Schematic picture of the Android platform.

In Android the Application Framework is the name for the higher-level APIs and services that a 3rd party developer can use to build an application. 3rd party developers have access to the same frameworks as the core applications. The base in the Application Framework is an Activity Manager that controls the lifecycle of the applications and provides a common navigation backstack. The Application Framework provides a rich set of views including lists, text boxes, buttons and menus that can be used to build applications. It also provides a set of Content Providers that enable sharing of information between applications. Applications can for instance, if it has the right permissions, access the information about the users contacts. The Application also include a Resource Manager that provides access to non-code resources such as localized strings, graphics and layout files.

On top of the Android Platform there is a set of Applications provided by the Android Project. These applications usually include Home Screen, Contacts, Phone, Messaging and Browser. Because of how the Android platform is built, these can be replaced by either the hardware manufacturer or a 3rd party developed application

distributed through for example the Android Market.

## 4.2 Software Development Kit

Most Android applications, both pre-installed and 3rd party, are written in Java. A normal Java compiler first compiles the code into Java byte code, which is then transformed into Dalvik byte code, .dex files, that runs in the Dalvik VM on any Android device. There are differences between normal Java and Dalvik byte code, but this is normally not an issue for a 3rd party developer. It is possible to run Java optimization frameworks such as Proguard, but some times this generates Java bytecode that can not be converted to dex-bytecode. These problems are usually solvable by removing a few of the optimizations that does not play well with Dalvik [14].

The Android SDK also has a different set of APIs from the normal Java environment. Google has chosen to not include J2ME, an older version of Java for mobile phones, and to only include a subset of the normal Java APIs in the Android SDK. The important parts of normal Java APIs are included, including java.nio, java.lang and java.util, but virtually all classes related to normal Java GUI code including AWT or SWING are removed. These are instead replaced with Android specific GUI and system classes. The SDK also include Khronos OpenGL ES framework for 3D rendering, the Apache Harmony HTTP components [6] and W3C XML parsing components [87].

Since the first stable release of the Android SDK back in 2008 there have been 6 subsequent releases from 1.1 to 2.1, which has led to platform fragmentation becoming a problem. The different internal market shares can be seen in figure 4.2, which show that the SDK version 1.0 that was replaced with 1.1 and which in turn quickly was replaced by 1.5 are not used anymore.

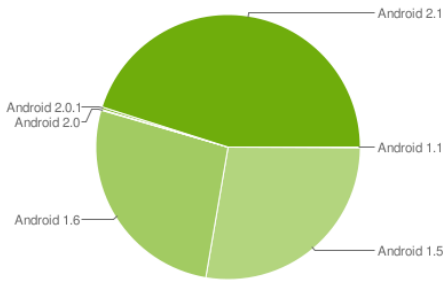


**Figure 4.2.** Percentage of active devices running a given version of the Android platform. Data collected during two weeks ending on June 1, 2010

Today Android 1.5 SDK is the baseline when it comes to Android development. With 27.6% of the total number of Android devices it is too large of a market to ignore. Apart from all the changes to the user experience and to the pre-installed applications, 1.5 also meant large changes for developers. It was a big leap forward for Android as a development platform, with big changes to basically all Android specific APIs [77]. The changes included some new elements in the UI framework, added the ability to create homescreen widget with the AppWidgets framework, improvements to the Media framework and the ability to write new and replace the existing Input Method Engines [66].

The Android 1.6 SDK was in many ways a minor update when it came to the user interface. Apart from the new Android Market and Camcorder UI, most changes were to the API and the development environment. Android SDK version 1.6 was the first version of the SDK that enables applications to run and look good on different resolutions and screen densities. Developers can since 1.6 specify what types of screens their application is supported on. 1.6 also included an extension to the search API making it possible for developers to include search results from their application in the global search. For instance if a developer develops a news application, the articles can be searched at the same time as the web. Something else that was new in the 1.6 release was the new gesture framework that enables developers to create, store, load and map gestures to actions. This was released together with a tool, Gesture Builder, which helps developer create the gestures. The 1.6 release also contains support for CDMA in the telephone stack as well as a new version of the Linux kernel, 2.6.29 [67].

Phones carrying the Android 2.0 SDK started shipping in November 2009 and had many new features to the user experience. Among these new features were Quick Contact that provides instant access to a contact's information and different methods of communication. Android 2.0 also featured a better camera application that provides effects, digital zoom and flash support. A better virtual keyboard and an improved browser UI were also included in 2.0 [69]. For developers Android 2.0 SDK provides a Bluetooth API that enables developers to turn Bluetooth on and off, discover and pair to devices, and use different Bluetooth services including OBEX or A2DP. This version of the SDK also enables developers to write sync adapters and to write/use modules in the centralized account manager API. This for instance lets users provide their twitter login credentials once for all twitter enabled applications [68].

The Android 2.1 SDK have been shipping starting with the Google Nexus One from the 5th of January 2010 [55]. Android 2.1 SDK is a minor platform release with some small changes in the telephony stack, Webkit- and UI-framework. Another feature of the Android 2.1 SDK was Live Wallpapers, which enables developers to create wallpapers that changes or responds to events such as touch events, orientation or what time of day it is [70].

## 4.3 Native Development Kit

Apart from the Software Development Kit, SDK, which enables developers to write applications using the Java programming language the Android platform also provide a Native Development Kit starting at Android 1.5 SDK. The Native Development Kit enables developers to write parts of their applications in languages, such as C and C++ that complies into native byte code. It is still not possible to develop whole applications using the NDK, but in certain situations, including games, signal processing or physics simulation application developers can take advantage of the increased performance that comes with running native code. This release of the NDK supports CPU architectures using the ARMv5TE machine instruction set and higher. The NDK consists of a set of compilers, a way to embed the compiled code in Android applications as well as headers for included libraries that will be supported in all future versions of the android platform. Among these libraries are libc, libm, libz, JNI headers and libraries for OpenGL ES support [84].

The Android NDK, Revision 1 was released together with the Android 1.5 SDK in June 2009. It includes compiler support (GCC) for ARMv5TE instructions, including Thumb-1 instructions as well as system headers for stable native APIs, documentation, and sample applications [84].

In September 2009 the Android NDK, Revision 2 was released. Compatible with the Android 1.6 SDK, the second revision of the Android NDK featured support for the OpenGL ES 1.1 API and some more documentation on how to build parts of an application as native code [84].

In March 2010 the third revision, Android NDK, Revision 3, was released to the public. This release is compatible with any device running Android 2.0 SDK or later. Android NDK, Revision 3 includes a new set of compilers, GCC 4.4.0, as well as other changes to the NDK tool chain. It also includes OpenGL ES 2.0 support that had been lacking from both the NDK and the SDK so far. The biggest advantage of OpenGL ES 2.0 is its support for Vertex and Fragments shaders. OpenGL ES 2.0 is the same version that is used in the iPhone OS, and running the same version will ease the process of porting games between the two platforms [84].

## 4.4 Important development concepts

Android includes some new development concepts to tackle the problem of writing applications for devices with limited resources. It also includes some interesting techniques to make it possible to have a higher layer of integration between applications than other mobile platforms.

The most important development concept on Android are Activities, which are the graphical presentation layer for an Android application, and in the typical case there is one Activity for every screen [9]. Usually the window that the Activity draws to fills the screen, but it might be smaller and float on top of other windows [78]. To show its content and to interact with the user an Activity uses a hierarchy of Views and Layouts. The base Activity can have one root View or Layout but there are also specialized Activities such as ListActivity or MapActivity.

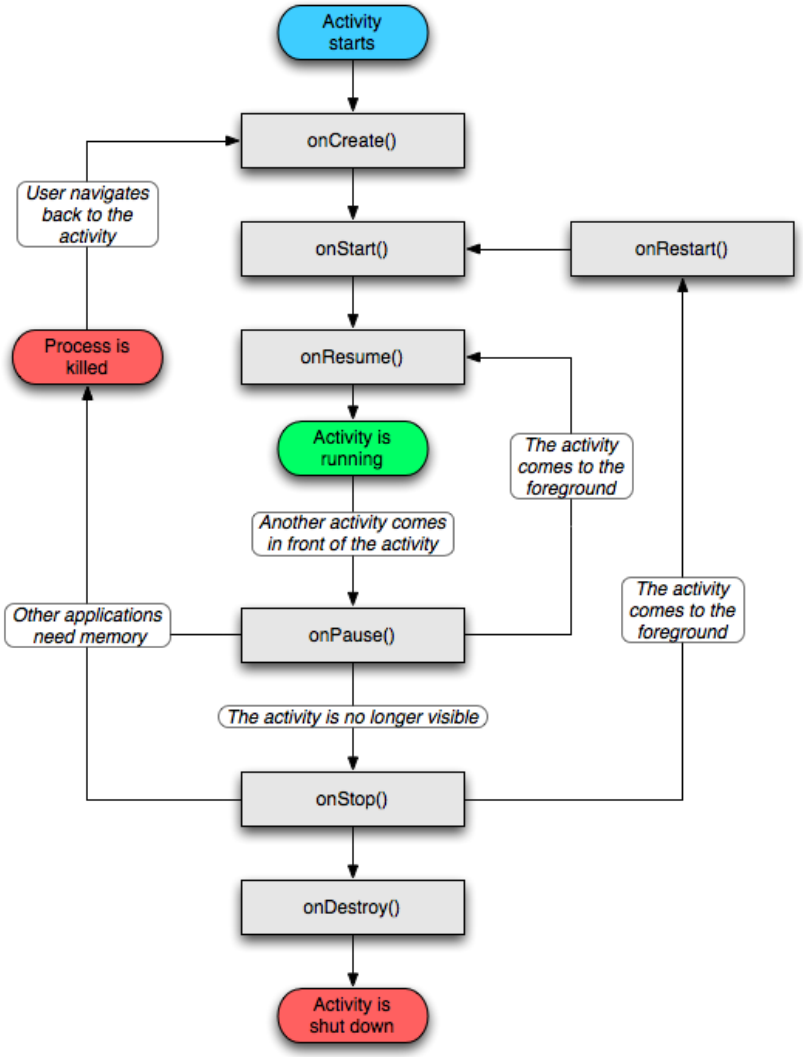


**Figure 4.3.** The Android Activity lifecycle.

One aspect that differentiates an Android Activity from the normal concept of a Window is its lifecycle, see 4.3. The Activity lifecycle is designed around require-

ments and limitations of the mobile devices it is supposed to run on. The lack of possibility to swap Activities out of memory creates the need for an Activity to be able to recreate itself after being removed from memory. This is solved with Bundles to which Activities can store their necessary data when they get destroyed and read it back when they are recreated [65].

The second basic concept in Android programming is Intents. An Intent is a specification of what action a developer wants to perform. A simple example of this is a request such as "I want to start Activity A" but it can also be a more general request, for instance "I want to open this PDF file" or "I want to send a SMS". To listen to this more general intentions, application developers can set up Intent Filters that are designed to show the rest of the system what capabilities a certain Activity have [85].

Intents is not limited to making request but can also broadcast information to the rest of the system. One example of this is the music-player that broadcasts every time a new song is being played. Some of these broadcasts are standardized or de facto standardized to help integration between applications. This enables a developer to for example build a home screen widget that downloads and shows the lyrics of the current song without actually knowing anything about the application playing the song [14]. To listen for these kinds of Intents a developer can set up Broadcast receivers whose only purpose is to listen to and react to these kinds of broadcasts [85].

The Android platform is designed from the ground up to be multitasking. One big part of this is to be able to run several applications at once, but equally important is to be able to run backgrounds Services. Services do not have a visual user interface and are designed to run in the background for an indefinite period of time. Services can perform tasks such as play music or download data in the background. A Service can when the task it is set to do is finished choose to communicate this to the user either by a notification or by launching an Activity [85].

All Android applications run sandboxed in separate virtual machines, and one consequence of this security model is that applications can not share memory or files directly. Instead they use Content providers to let other applications access their data. The Content provider can indirectly give access to data from files, databases or memory. There is also a set of Content providers provided by system applications [79].

A developer must provide an AndroidManifest.xml to tie these different components together and to define the capabilities of an application. The Android manifest includes information about all the Activities, Services, Intent Filters, Broadcast Receivers and Content providers that an application provides. The manifest file also provides all the meta-information about the application such as name, icon, supported languages, supported screen-sizes, required hardware and minimum SDK

version. Additionally the AndroidManifest.xml file is where the developer defines all the permissions the application needs to function properly. Permissions be given to allow starting the SMS application, accessing the Internet, to get the current position from the GPS or start a phone call. Instead of asking every time one of these permissions is needed they are instead presented to the user at install time [76].

## 4.5 Development tools

Android development is, as other development in the Java programming language, often done in a development environment such as Eclipse. It is possible to do development in other IDE's including Netbeans [56], but the official development environment is Eclipse. Eclipse is an open source development platform originally built by IBM in 2001 but is now maintained by the non-profit foundation The Eclipse Foundation [10].

Google's Android Development Tools, ADT, is a set of Eclipse plugins made to ease Android development. ADT enables developers to create Android projects with all the necessary files and build configurations, as well as simplifying the process of building, testing and signing an application before release. In the day-to-day work the ADT plugin enables control of an attached device or emulator. These controls includes setting breakpoints, changing the capabilities of the device, logging, emulating positions or incoming calls etc [82].

When developing for Android a developer can either test applications on an attached device or an emulator. In comparison with iPhone development, Android makes it very easy to test an application on different devices. Binaries can be distributed as-is to any device for testing. Developers can also set a device in debug mode which enables access to the debug console output of the device and to set breakpoints. Development can be done on any consumer or development device as long as it is set to debug mode, under Setting -> Applications -> Development -> USB debugging, and the Android USB driver is installed [83].

The Android project also provides an emulator, which is very different from the iPhone simulator in that it is a real hardware emulator. The emulator is based on the QEMU project and emulates an ARMv5 device with a 16-bit color screen [73]. One of the big advantages of having a hardware emulator is that it gives developers the ability to run different builds of the operating system. It is for instance possible to run the real system images from device manufacturers such as HTC. This is very handy since a developer can see how an application performs on different customizations of Android [4].

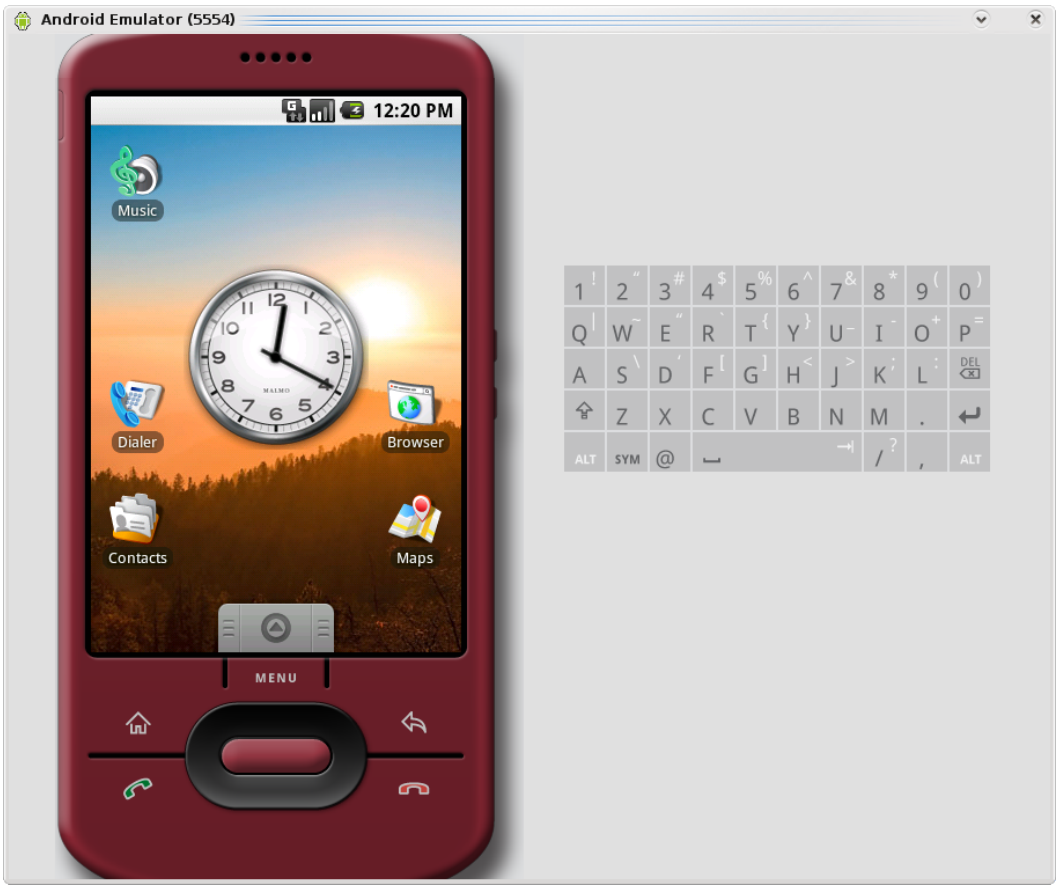


**Figure 4.4.** Android running on the emulator.

The emulator runs an Android Virtual Device, AVD, which is a profile that defines the hardware and software capabilities of an emulated device. Software wise a developer can specify any downloaded SDK version or any other device image that he or she might want to test on. A developer can specify if the virtual device should have the Google libraries such as maps installed or not, as well as specifying the device RAM size, hardware keyboard support, camera support, GPS support, screen resolution and screen density. The AVDs are all managed through the ADT plugin in Eclipse [75].

The Android development environment includes command line tools such as adb that enables detection and control of all attached devices and emulators. It also enables installation of 3rd party application onto a device or emulator as well as opening a shell on the device [75]. The development environment also includes a tool called TraceView that makes it possible to measure the performance of an application. A developer can get detailed information on how much memory and

CPU the application uses as well as which methods that most time is spent executing or most objects are allocated in [89].

## 4.6 Documentation and Developer Resources

The main documentation and resource hub for Android is the developer section of the Android homepage. The natural starting point when developing for Android is the Developer Guide, which goes through the all the major concepts of Android, from the basics of what it is to publishing of applications and best practices. It also covers different framework topics including the fundamentals of application lifecycles, user interfaces, location and maps, data storage and security [81].

As with the Java programming language Android provides a good reference manual over the APIs. The reference manual is also available at the Android developer site and has in-depth coverage of all the Android specific classes, but also covers the classes pulled in from W3C and the Apache project as well as the classes from Java Standard Edition [87].

The Android community also has a series of developer resources such as tutorials, sample code, technical articles and FAQs for developers to use. The number of tutorials are a bit limited, but the technical articles provide a good stepping stone when venturing into a new area of Android [88]. The sample code section also has a growing list of examples that can be used to study certain techniques [86].

The Android community also provides a range of different forums for developers that might not find an answer to their questions or wish to discuss a subject [80]. Beginning and entry-level questions are directed to Stack Overflow, which is a "Collaboratively edited question and answer site for programmers" [80]. On the site there is a special sponsored tag for Android that consolidates all questions related to Android [62].

For immediate and expert-level questions as well as question regarding certain topics, the Android community provides a range of mailing lists. There is one mailing list for development questions, one for general discussions around Android, one for NDK based development and two regarding Android security concerns [80]. To keep the traffic down to a manageable level it is asked that anyone who has a question tries to find the answer elsewhere before posting it on one of the mailing lists [71]. There is also an Android Market Help Forum for help with Android Market related questions such as publishing and payouts [74].

On the Android developers website there is a number of educational videos available that range from official videos about each release, which shows off the new stuff, to highly technical videos about the platform. Google has also published videos

from conferences such as Google I/O as well as developer tips and interviews with developers [90].

Google also publish an Android developers blog where they post about the highlights of new releases, write-ups of new APIs and other news worthy items [72].

## 4.7 Distribution

The main distribution channel on the Android platform is the by Google managed Android Market, which is distributed with near all Android-running devices. A user needs to have a Google account to be able to use the Android Market, and to be allowed to publish applications in the Android market a developer needs an Android Market Developer account. This account is available to individuals and companies for a one-time fee of $25. This gives them the ability to upload applications to the Android Market [16]. An Android Market publisher can choose to distribute their applications for free or a price, and in the case of paid applications 70% of the price goes to the developer while Google keeps the other 30% [19].

Android Market is available in 46 countries [17] but only 13 of them support paid applications, [17] and only developers from 9 countries are allowed to publish paid applications [18]. Paid applications are mainly available in key markets such as the United States, United Kingdom, Germany and early Android adopting markets including Austria and Japan [17]. However, it is still missing from the majority of the European markets, for instance Scandinavia, the Baltic States and large parts of eastern and southern Europe. Google have earlier said that paid applications should have been rolled out to more markets in October 2009 but that promise did not materialize and now there is no new public roadmap for when the remaining countries will get paid applications [23].

On Android there is also the possibility to distribute applications outside of the Android Market. Developers can publish applications on their homepage or send the application directly to users [15]. The possibility to install applications outside of the Android Market has spawned a range of different 3rd party markets. The biggest of these alternative markets is SlideME, which offers some advantages over the normal Android Market such as enabling paid applications on all markets and giving the developer 100% of the income from an application sales [96].

# Chapter 5

# Results and Comparisions

## 5.1 Result matrix

| | Android | iPhone |
|---|---|---|
| Development Environment and Programming Languages | | |
| - Language | Java, C/C++ | Objective-C/C/C++ |
| - Development Environment | Eclipse | X-Code |
| - Emulator | Good | Good |
| - Interface desiger | OK | Good |
| - Performance Tool | Bad | Good |
| Development Techniques | | |
| - Multitasking | Yes | Future |
| - Integration between applications | Yes | Limited |
| Distribution and Monetization | | |
| - Free applications | Worldwide | Worldwide |
| - Paid applications | 13 Markets | Worldwide |
| Platform Fragmentation | | |
| - SDK (Active Versions) | 3 | 1 |
| - Screen sizes | Different | One |
| - CPU-speed | Different | Different |
| - Keyboard | Software or hard-ware | Only software |

**Figure 5.1.** The results in this table is thourder discussed in the comming sections.

## 5.2 Development Environment and Programming Languages

I have chosen to start by comparing iPhone development in XCode and Android development in Eclipse. XCode is in many ways a good IDE, and is well integrated into the Mac environment. Eclipse on the other hand, is a de facto industry standard that many developers already are proficient with. It also has more in common with other well known IDEs such as Visual Studio. The biggest advantages of the Android development environment are the easy project management and the possibility to use existing plugins for Eclipse to connect to a range of different revision control systems including git and svn. Other strong advantages are refactoring and error reporting.

A comparison between XCode and Eclipse is also in many ways a comparison between Objective-C and Java. Objective-C is in the same way as XCode an pretty obscure product specific to the mac world. Java on the other hand is one of the most commonly used modern programming languages. The nature of Java with the help of Eclipse gives a much more enjoyable development environment, and Java is over all a more modern programming language, with features such as garbage collection. Objective-C on the other hand uses more obscure constructs such as having to write @"this is a string" instead of "this is a string". Objective-C is not a language I would use outside of iPhone development.

XCode's advantages are the external tools made to help building iPhone applications. The Interface Builder and performance analysis tools are much better and feel more integrated than the Android counterpart ADT, which does not feel well integrated and well polished. The corresponding tool to the Interface Builder on the Android platform does not have a user friendly interface and lack functionality to link actions in the interface to methods in the code, a feature that the Interface Builder has. In my experience it was easier to write the interface XML for Android on my own, rather than to use the supplied tool. The performace tool for iPhone development, Instruments, is equaly superior to the corresponding Android tool.

It seems that the time Google did not spend on their interface and performance tool they spent on the emulator. Much of the advantages of the Android Emilator comes from that it is built on QEMU and actually emulates a real phone. The iPhone Simulator on the other hand runs as a normal application and this can give the developer strange error as those I listed in the Developer Environment section of the iPhone chapter. The Android Emulator also has much functionality that is missing from the iPhone Simulator, such as in the Android Emulator being able to simulate a call or text message from a specified number, change the position returned by the emulators fake gps as well as limiting the network bandwidth and latency for the emulated phone to simulate real conditions.

## 5.3 Development Techniques

The development teqniques used in iPhone and Android is very different from normal applications on the desktop. It is easy to produce good looking applications on the iPhone, which follows the iPhone guidelines. View controllers and navigation controllers works well and creates a good workflow for developers, and easy to use applications for users. Apple has designed most of their APIs around the Model View Controller design pattern. One example is the location manager, which has a corresponding protocol that is registered and then get callbacks every time the location change. The use of MVC makes it easy to write good applications and developers does not have to poll different APIs for the current information. One of the downsides of iPhone development is that it is impossible in many ways for applications to integrate with the system and to communicate with each other. It is possible to open another application if it has registered an URI-handle, but this is only one way communications, and it is impossible to send files of larger amounts of data between applications.

Android has a more open approach to development, and all APIs are available to all developers unlike on the iPhone platform where Apple has many hidden APIs for their own and their partners applications. The main development concept in the Activities, which I find very useful. The design of Activities is centered around the requirements of mobile devices, and it is good that Activitiescan be stopped if the device is running out of memory, and then restored when the user returns to that Activity. The same situation on the iPhone would result in the application being terminated. Another very useful and innovative development techniques is the use of Intents, which is the easiest wayto integrate applications that I have seenon any platform.

One example where intents and intent filters are very powerful, both for users and developers is the Facebook application. On iPhone if one want to take a photo and share it then he or she must first take the photo, then start the Facebook application, find the photo and then share it. Another ways is to launch the Facebook application and use their camera interface, but this will only post your photo to Facebook, not save it in your camera roll. On Android, the same action would involve openening the camera application, takeing the photo and the pressin the share icon. This will show a lost of all applications that can share images, like mail and Facebook. Clicking on Facebook will start the share photo Activity from the Facebook application where the user can add a caption. This is not just a good feature for users, but is also very useful for developers since it creates a many-to-many relation for sharing images. Any application can send an intent saying “I want to share a photo.” and any application can define intent filter to answer “I can share your photo.”. It is then up to the user to chose which service to use. This kind of integration is a natural part of the Android platform because all applications are

written with the same frameworks and that there are no hidden APIs.

One of the more debated differences between iPhone and Android is their stance on multitasking. Both platforms have always been able to run several applications at once, the question has been if running several 3rd party applications at once should be allowed. Apple has traditionally said no to this with the excuse that it drains battery-life, while it has always been an integral part of how Android works. Apple has lately changed their mind, and the version 4.0 of the iPhone OS will allow running several 3rd party applications at the same time. There are some technical differences in how this is done, but the experience will be the same. Something the iPhone lacks even in version 4.0 is the possibility to run 100% background applications, such as Android Services.

## 5.4 Distribution and Monetization

To be able to develop for either iPhone or Android a developer needs to be a part of the corresponding developer program. Apple's developer program is much more expensive, $99 per year compared to $ 25 for perpetuity with the Android development program. The Android program is also much easier to apply for, it is basically a normal website registration process followed by a credit card charge. Applying to Apple's iPhone development program takes several weeks and includes, sending a copy the companies certificate of registration to Apple by fax.

Development for the Android platform can start even before purchasing the Android development program. iPhone development can only be done locally and not tested on devices before being an approved iPhone developer. When developing for iPhone there is a certificate or key for everything. One to be able to compile the code for devices, one key per device that should be able to run the code and a distribution certificate for the AppStore. On the Android platform the only key is the distribution key, that is a normal jarsigner key which is commonly used in Java development. All the administrative tasks when doing iPhone development is very cumbersome. Keys constantly needs to be generated or regenerated because their early expiry dates.

On one hand Android has easier ad-hoc distribution, but fails in comparison on Application Store distribution. Google still have not made it possible for users outside the major markets to buy paid-applications. Google have earlier said that this should have been done during the fall of 2009 but have since that announcement not given any insight on when it will be possible. This makes it hard for developers to monetize on applications, there are fewer countries that it is possible to sell applications from. Apple on the other hand has from day one of the AppStore provided paid applications for all users through their already established iTunes

payment channel. Apple also invented the in-application purchase model where developers can sell content through their applications. This has givven birth to a range of different payment models including subscriptions or one time charges for extra content.

Another way of making money on an application is Advertising. There are several 3rd party solutions for embedding advertising. On Android these works pretty ok. Since the multi tasking makes the content of the ad open in front of the running application, and you can always go back. On the iPhone this has not been possible which have made users hesitant to click on ads. To solve this Apple has also announced that in the 4.0 version of the platform, said to be released during the summer of 2010, it will be possible to their own solution iAd. iAds open in front of the running applications, but does not close it as 3rd party solutions does. These ads will be sold by Apple and the profit will be shared at 60/40.

## 5.5 Platform Fragmentation

The Android Platform has had an extra ordinary development pace since it launch, and is at its 6th major version. Three of this versions make up almost all of the devices running Android, and this has created problems for developers who wants to create software using the latest features, but have been held back because they do not want to exclude many of the Android users. One of the main reasons for this problem is that new releases have to go through the phone manufactures modifications and quality assurance testing before sent out to the users phones. This has in the past often taken longer than users and developers have hoped for. The underlaying reason for this is that manufacturers has not been able to keep up with Google's release schedule, which is much faster than whats normal in the industry. Apple has been spare of this problem so far since they are the only hardware manufacturer that produce smartphones running the iPhone OS, this makes it possible for them to push updates as they get released.

Google is planning some changes to the coming releases of the Android platform, which will try to solve this problem by transition many of the components and pre-installed applications to the Android Market. This will probably start with the release of version 2.2, codename Fro-yo, and the version after that, codename Gingerbread. This has already been done with the Google Maps application, and the result is that the users earlier versions of Android can update to the same version as those running the latest version. Google has said that this process of moving components to the Android Market will not be limited to Applications but will also cover system components such as input methods.

Another move that will help with the platform fragmentation on Android is that

Google is said to start slowing down the release pace. The APIs on Android have now reached feature parity with the other smartphone platforms, and the developer focus is now switching towards new features. I think that this slower release pace together with many components moving to the Android Market will be an effective approach to platform fragmentation. There will always be some platform fragmentation on Android since the updates always have to be checked by the manufactures, but in the future the number of active versions will reduce and the difference between them will not be as large as today.

Apple has, as mention earlier, been spared from the problem of device fragmentation in the past. iPhone users have always gotten their updates to new versions directly for free. iPod Touch users have been charged $ 99 for every major update, i.e. 1.0 -> 2.0 and 2.2 -> 3.0, but they have been a minority and most of the users have paid this fee since they otherwise been very limited to which applications they could download.

The oldest of the Apple iPhone has been around for a while now, and is hardware-wise not necessarily design for the features that Apple wants to add to the operating system. Apple has been under great pressure from users and developers to start supporting multitasking. They have now recognized this criticism and added multitasking in the new version of the iPhone OS, version 4.0, which is said to be released during the summer of 2010. iPhone OS version 4.0 is the first version that will not be available to all past and current devices. Apple has said that the first version of the iPhone and iPod Touch have reached their end of life, and that they will not be updated due to hardware constraints. Also the iPhone 3G and the second version of the iPod Touch will be updated to iPhone OS 4.0, but will be subjects of some limits because of their small amount of memory. These versions will not be able to take advantage of the multitasking, so the only versions of the iPhone that will fully be able to take advantage of the new version of the operating system is the iPhone 3GS and the iPhone "4G" rumored to be released this summer.

This will create a whole new situation in the iPhone developments, which have earlier been able to take for granted that all users are running the latest major version of the operating system a few month after its release. This will no longer be true when users of the old iPhone and iPod Touch are excluded from the new versions. Another difficult situation will face the developers who want to take advantage of the multitasking features of the new version since only the latest versions of the iPhone and iPod Touch will be access this functionality. The iPad is also problematic, since it runs another versions of the iPhone OS. Right now it runs version 3.2 which is a iPad only release, and it will not be updated to 4.0 until sometime during the fall of 2010. All of this together will from the summer of 2010 and onwards create fragmentation on the iPhone platform. It is to early to tell what impact this will have, but I think that it will at least create disappointed users and headaches for developers in a similar way as the platform fragmentation on Android

has.

## 5.6 Conclusions and future predictions

It is my opinion that Android is a better development platform, much because of its stronger focus on technology. I think that the technology choices such as programming languages and the different techniques feels more natural for a mobile device, that those done by Apple. Android has however some shortcomings when it comes to platform fragmentation and distribution. I think it is hard to justify why Google has not made it possible to buy and sell applications through the Android Market in more than a few countries, and I think this limits the growth of Android. Google seems to have the platform fragmentation problem under control, but they still have not given any idea of when paid applications will be released to the rest of the world.

iPhone is a more mature platform when it comes to distribution and monetization, and these areas seems to in many areas come before technology on Apple's priority list. The technology that developer uses today to build iPhone applications works, but it is older than that of Android, and I think that at some point developers will start to protest against all the rules and limitations set by Apple. Apple has been good at keeping away platform fragmentation, but the oldest devices running iPhone OS is now starting to be too limited hardware-wise and have reached their end of life, which will cause problems.

It is always hard to do predictions about the future, but I think that the big momentum around Android as a development platform will continue, and that it some time in the next two years will pass Apple and their iPhone in both market share and number of 3rd party applications. iPhone will also continue to grow as a platform, but at a slower rate, and together they will take a large part of the smartphone market share. The whole new Windows Phone 7 and Symbian Foundation will also be two large actors in the future and it is my prediction that these four platforms will be those that have the absolute majority of the smartphone market during the coming years.

# Bibliography


[1] Open Handset Alliance. Industry leaders announce open platform for mobile devices. http://www.openhandsetalliance.com/press_110507.html. (2010-04-03).

[2] Nick Wingfield Amol Sharma and Li Yuan. Apple coup: How steve jobs played hardball in iphone birth. *Wall Street Journal Online*, February 2007.

[3] Gareth Beavis. A complete history of android: Everything you need to know about google's mobile operating system. (2010-04-03).

[4] Kumar Bibek. Android market on emulator. http://tech-droid.blogspot.com/2009/11/android-market-on-emulator.html. (2010-04-06).

[5] Ryan Block. Live from apple's iphone sdk press conference. `http://www.engadget.com/2008/03/06/live-from-apples-iphone-press-conference/`. (2010-04-11).

[6] Scott Delap. Google's android sdk bypasses java me in favor of java lite and apache harmony. (2010-04-05).

[7] Distmo. Special ipad report - apple app store - ipad and iphone. *Distmo Report*, April 2010.

[8] Practical E-commerce. Chart of the week: Google's android mobile os will outpace the iphone, others. `http://www.practicalecommerce.com/articles/1575-Chart-of-the-Week-Google-s-Android-Mobile-OS-Will-Outpace-the-iPhone-Others.` (2010-02-15).

[9] Mihai Fonoage. *Android - An Overview*. February 2009.

[10] Eclipse Foundation. About the eclipse foundation. http://www.eclipse.org/org/. (2010-04-06).

[11] Symbian Foundation. The history of symbian. http://www.symbian.org/about-us/history-symbian. (2010-02-15).

[12] Priya Ganapati. Google nexus one sales off to slow start. (2010-04-03).

[13] Kent German. A very early review of iphone os 3.0. http://reviews.cnet.com/8301-19512_7-10205643-233.html. (2010-04-11).

[14] Peter Grundström. Personal experience.

[15] Android Market Help. Distributing apps outside android market. http://market.android.com/support/bin/answer.py?answer=142471. (2010-04-18).

[16] Android Market Help. Registration. http://market.android.com/support/bin/answer.py?answer=113468. (2010-04-18).

[17] Android Market Help. Supported locations for distributing applications. http://market.android.com/support/bin/answer.py?answer=138294. (2010-04-03).

[18] Android Market Help. Supported locations for merchants. http://market.android.com/support/bin/answer.py?answer=150324. (2010-04-18).

[19] Android Market Help. Transaction fees. http://market.android.com/support/bin/answer.py?answer=112622. (2010-04-18).

[20] Simon Hill. History of android: First applications, prototypes & other events. (2010-04-03).

[21] HTC. Htc hd mini overview. http://www.htc.com/europe/product/hdmini/overview.html. (2010-04-21).

[22] HTC. Htc presentation at mwc 2010.

[23] Erik Hörnfeldt. 3 Sverige Facebook Fanpage. (2010-04-18).

[24] Apple Inc. Apple presentation of iphone os 4.0. http://www.apple.com/iphone/preview-iphone-os/. (2010-04-16).

[25] Apple Inc. Developer tools technology overview - apple developer. http://developer.apple.com/technologies/tools/. (2010-04-11).

[26] Apple Inc. Information from itunes connect. iTunes Connect. (2010-04-16) Restricted Access.

[27] Apple Inc. iphone application development fundamentals. (2010-04-12).

[28] Apple Inc. iphone developer enterprise program - apple developer. http://developer.apple.com/programs/iphone/enterprise/. (2010-04-16).

[29] Apple Inc. iphone developer program - apple developer. http://developer.apple.com/programs/iphone/. (2010-04-16).

[30] Apple Inc. iphone developer program benefits. http://developer.apple.com/programs/iphone/distribute.html\#compare. (2010-04-16).

[31] Apple Inc. iphone developer university program - apple developer. http://developer.apple.com/programs/iphone/university/. (2010-04-16).

[32] Apple Inc. iphone os reference library - coding how-to's. http://developer.apple.com/iphone/library/navigation/index.html?section=Resource+Types&topic=Coding%20How-Tos. (2010-04-12).

[33] Apple Inc. iphone os reference library - getting started. https://developer.apple.com/iphone/library/navigation/index.html?section=Resource%20Types&topic=Getting%20Started. (2010-04-12).

[34] Apple Inc. iphone os reference library - reference. http://developer.apple.com/iphone/library/navigation/index.html?section=Resource%20Types&topic=Reference. (2010-04-12).

[35] Apple Inc. iphone os reference library - sample code. http://developer.apple.com/iphone/library/navigation/index.html?section=Resource%20Types&topic=Sample%20Code. (2010-04-12).

[36] Apple Inc. iphone os technology overview: iphone os technologies. http://developer.apple.com/iphone/library/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html/. (2010-04-12).

[37] Apple Inc. iphone sdk release notes for iphone os 3.1. http://developer.apple.com/iphone/library/releasenotes/General/RN-iPhoneSDK-3/index.html. (2010-04-11).

[38] Apple Inc. Networking - bonjour. http://developer.apple.com/networking/bonjour/index.html. (2010-04-12).

[39] Apple Inc. Tools for iphone os development. https://developer.apple.com/iphone/library/referencelibrary/GettingStarted/URL_Tools_for_iPhone_OS_Development/index.html. (2010-04-11).

[40] Apple Inc. Uinavigationcontroller class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UINavigationController_Class/Reference/Reference.html. (2010-04-14).

[41] Apple Inc. Uitabbarcontroller class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UITabBarController_Class/Reference/Reference.html. (2010-04-14).

[42] Apple Inc. Uitableviewcell class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UITableViewCell_Class/Reference/Reference.html. (2010-04-14).

[43] Apple Inc. Uitableviewcontroller class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UITableViewController_Class/Reference/Reference.html. (2010-04-13).

[44] Apple Inc. Uiview class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UIView_Class/Reference/Reference.html. (2010-04-13).

[45] Apple Inc. Uiviewcontroller class reference. http://developer.apple.com/iPhone/library/documentation/UIKit/Reference/UIViewController_Class/Reference/Reference.html. (2010-04-13).

[46] Apple Inc. What's new in iphone os: iphone os 3.0. http://developer.apple.com/iphone/library/releasenotes/General/WhatsNewIniPhoneOS/Articles/iPhoneOSv3.html. (2010-04-11).

[47] Apple Inc. What's new in iphone os: iphone os 3.2. http://developer.apple.com/iphone/library/releasenotes/General/WhatsNewIniPhoneOS/Articles/iPhoneOS3_2.html. (2010-04-11).

[48] Apple Inc. Xcode - developer tools technology overview - apple developer. http://developer.apple.com/technologies/tools/xcode.html. (2010-04-11).

[49] Apple Inc. Xcode 3.1 feature overview. http://developer.apple.com/iphone/library/documentation/DeveloperTools/Conceptual/WhatsNewXcode/10-Articles/xcode_3_1.html. (2010-04-11).

[50] Apple Inc. Xcode 3.2 feature overview. http://developer.apple.com/iphone/library/documentation/DeveloperTools/Conceptual/WhatsNewXcode/10-Articles/xcode_3_2.html. (2010-04-11).

[51] Apple Inc. Xcode project management guide: Analyzing code. http://developer.apple.com/iphone/library/documentation/DeveloperTools/Conceptual/XcodeProjectManagement/220-Analyzing_Code/static_analysis.html. (2010-04-11).

[52] Google Inc. Availability in your country and language : Place an order - nexus one help. http://www.google.com/support/android/bin/answer.py?answer=166508. (2010-04-03).

[53] Google Inc. Nexus one phone - feature overview & technical specifications. http://www.google.com/phone/static/en_US-nexusone_tech_specs.html. (2010-04-03).

[54] HTC Inc. T-mobile unveils the t-mobile g1 - the first phone powered by android. http://www.htc.com/www/press.aspx?id=66338&lang=1033. (2010-04-03).

[55] Rob Jackson. Nexus one now available... for verizon/vodafone too (soon)! http://phandroid.com/2010/01/05/nexus-one-now-available-for-verizonvodafone-too-soon/. (2010-04-05).

[56] Kenai. Android plugin for netbeans. http://kenai.com/projects/nbandroid/. (2010-04-06).

[57] Jason Kincaid. Apple is growing rotten to the core: Official google voice app blocked from app store. (2010-04-16).

[58] Stephen Lawson. Palm pre launch is high-stakes gamble. (2010-04-21).

[59] Laura McGann. Mark fiore can win a pulitzer prize, but he can't get his iphone cartoon app past apple's satire police. (2010-04-18).

[60] Microsoft. Welcome to windows ce 5.0. http://msdn.microsoft.com/en-us/library/ms905511.aspx. (2010-04-21).

[61] Research In Motion. Business solutions at blackberry.com. http://na.blackberry.com/eng/solutions/. (2010-04-21).

[62] Roman Nurik. Hello, stack overflow! http://android-developers.blogspot.com/2009/12/hello-stack-overflow.html. (2010-04-05).

[63] Karin O'Mahony and Ola Henriksson. Svd:s reseguider - nu i din iphone. *Svenska Dagbladet Online*. (2010-04-19).

[64] Stack Overflow. iphone device vs. iphone simulator. `http://stackoverflow.com/questions/380062/iphone-device-vs-iphone-simulator`. (2010-04-11).

[65] The Android Project. Activity. http://developer.android.com/reference/android/app/Activity.html. (2010-04-10).

[66] The Android Project. Android 1.5 platform. http://developer.android.com/sdk/android-1.5.html. (2010-04-05).

[67] The Android Project. Android 1.6 platform. http://developer.android.com/sdk/android-1.6.html. (2010-04-05).

[68] The Android Project. Android 2.0 platform. http://developer.android.com/sdk/android-2.0.html. (2010-04-05).

[69] The Android Project. Android 2.0 platform highlights. http://developer.android.com/sdk/android-2.0-highlights.html. (2010-04-05).

[70] The Android Project. Android 2.1 platform. http://developer.android.com/sdk/android-2.1.html. (2010-04-05).

[71] The Android Project. Android developers. http://groups.google.com/group/android-developers. (2010-04-05).

[72] The Android Project. Android developers blog. http://android-developers.blogspot.com. (2010-04-05).

[73] The Android Project. Android emulator. http://developer.android.com/guide/developing/tools/emulator.html. (2010-04-06).

[74] The Android Project. Android market help. http://www.google.com/support/forum/p/Android+Market. (2010-04-05).

[75] The Android Project. Android virtual devices. http://developer.android.com/guide/developing/tools/avd.html. (2010-04-06).

[76] The Android Project. The androidmanifest.xml file. http://developer.android.com/guide/topics/manifest/manifest-intro.html. (2010-04-10).

[77] The Android Project. Api differences between 2 and 3. http://developer.android.com/sdk/api_diff/3/changes.html. (2010-04-05).

[78] The Android Project. Application fundamentals. http://developer.android.com/guide/topics/fundamentals.html. (2010-04-10).

[79] The Android Project. Content providers. http://developer.android.com/guide/topics/providers/content-providers.html. (2010-04-10).

[80] The Android Project. Developer forums. http://developer.android.com/resources/community-groups.html. (2010-04-05).

[81] The Android Project. The developer's guide. http://developer.android.com/guide/index.html. (2010-04-05).

[82] The Android Project. Developing in eclipse, with adt. http://developer.android.com/guide/developing/eclipse-adt.html. (2010-04-06).

[83] The Android Project. Developing on a device. http://developer.android.com/guide/developing/device.html. (2010-04-03).

[84] The Android Project. Download the android ndk. http://developer.android.com/sdk/ndk/index.html. (2010-04-05).

[85] The Android Project. Intents and intent filters. http://developer.android.com/guide/topics/intents/intents-filters.html. (2010-04-10).

[86] The Android Project. List of sample apps. http://developer.android.com/resources/samples/index.html. (2010-04-05).

[87] The Android Project. Package index. http://developer.android.com/reference/packages.html. (2010-04-05).

[88] The Android Project. Technical articles. http://developer.android.com/resources/articles/index.html. (2010-04-05).

[89] The Android Project. Traceview: A graphical log viewer. http://developer.android.com/guide/developing/tools/traceview.html. (2010-04-06).

[90] The Android Project. Videos. http://developer.android.com/videos/index.html. (2010-04-05).

[91] The Android Project. What is android? http://developer.android.com/guide/basics/what-is-android.html. (2010-04-03).

[92] Emil Protalinski. Windows phone 7 series to have three chassis. (2010-04-21).

[93] Thomas Ricker. iphone hackers: "we have owned the filesystem". `http://www.engadget.com/2007/07/10/iphone-hackers-we-have-owned-the-filesystem/`. (2010-04-11).

[94] Daniel Roth. Google's open source android os will free the wireless web. *Wired Magazine: 16.07*. (2010-04-03).

[95] Ryan Singel. Bad pr forces apple to reconsider banning prize-winning satirist. (2010-04-18).

[96] SlideMe. Frequently asked questions. http://slideme.org/faq. (2010-04-18).

[97] Joshua Topolsky. First third-party "game" app appears for iphone. `http://www.engadget.com/2007/08/06/first-third-party-game-app-appears-for-iphone/`. (2010-04-11).

[98] Fred Vogelstein. The untold story: How the iphone blew up the wireless industry. *Wired Magazine: Issue 16.2*, September 2008.

[99] Carolina Werner. Alla viner i din iphone.

[100] Ben Willson. The unofficial iphone sdk: Guide to writing native iphone applications. `http://reviews.cnet.com/8301-19512_7-10115160-233.html`. (2010-04-11).

[101] Chris Ziegler. Apple's iphone lockdown: apps must be written in one of three languages, adobe in the hurt locker. (2010-04-11).