NTNU – Trondheim
Norwegian University of
Science and Technology

Cross platform Mobile Applications Development

Mobile Apps Mobility

Yonathan Aklilu Redda

Master in Information Systems
Submission date: June 2012
Supervisor: Hallvard Trætteberg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

In recent years, the mobile computing sector has been having quite a revolution. Mobile computing devices have shed loads of weight, gone slim, achieved mass popularity and a great market penetration. But one of the challenges that has been part of mobile computing is technology and device fragmentation leaving application developers for mobile phones bewildered. Platform developers, device manufacturers come with so many features and functionalities that it has been difficult to provide developers with an easier means of developing applications and running the application on every mobile device.

To solve that, cross platform tools have been investigated and was found to solve device functionality features, platform and device coverage problems. It was also noted that cross platform development was found to be cheaper and takes less time.

Even though tool selection in cross platform development was difficult, it was suggested that Appcelerator Titanium and Xamarin were picked as a preliminary starting point.

.

Preface

This master's thesis was part of a mandatory course work at the department of computer and information science at the Norwegian University of Science and Technology. The aim of the project was to evaluate cross platform mobile applications development using scientific methods and coming up with a solution on how to select comparatively the best cross platform development tool.

I would like to express my gratitude to my supervisors, to **Hallvard Trætteberg** from NTNU and **Erik Berg** from Telenor for their consistent feedback, support and academic guidance.

Trondheim, June 2012

Yonathan Redda

.

Contents

List of Figures

List of Tables

1 Introduction

In the past a telephone was a chunky device sitting in offices and people's home. Its function was solely for making and receiving calls. The thought of a telephone device storing contacts, displaying caller numbers, forwarding calls, SMS-ing, taking pictures, capturing videos, recording audios was remotely alien to traditional telephone hardware let alone providing connection to the internet and coming off the desk into full mobility. Then Motorola came up with a revolutionary device known as Hefty—or the Brick weighing about 785g with a height of 13 inches [?], by todays standard, a primitive device.

In the 1980s and early 1990s, the concept of mobile phone and actual implementations of mobile services started to gain ground in terms of standards definition and setting up successful commercial services using mobile devices [65, 68]. During this time, the main application of mobile devices was mainly for voice based services. Mobile services began to develop using generations as a means of communicating the changes on successful improvements or functionality extensions [8].

By 2002, mobile phones became completely unrecognizable from their predecessors. They became pocket sized little devices. They evolved into Pagers, PDAs, Palmtops and mobile devices with voice, data and text applications. Mobile phone subscribers number grew astronomically, standing at an astonishing 1.2 billion subscribers across the world [127].

Apart from the usually fancy common mobile apparatus features—including contact details, SMS/Text, multimedia, infrared, Bluetooth modules—mobile devices started to sport a more serious and intelligent applications fueled by service providers desire to make money and unstoppable desire of end users for more of their services [69, 127].

Mobile applications scenarios became as varied as any field could get. The field of mobile computing saw applications from full mobility to full connectivity while on the move to invisible and wearable computing to crystal clear audio streaming to HD video to payment services to location based services to navigational services to mass gaming services to location and context aware to telco-enriched applications of any sort. Little is left one can do on stationary computer but not on mobiles.

Today mobile applications are in a lot of sense quite—with the right and cleverly designed applications—capable of pulling above their weight to compete with traditional stationary computers. A whole raft of applications are springing up everywhere specifically designed for mobile devices. Applications can be developed using a wide variety of tools, methodology and using the traditional applications development life cycle. The number of tools that promise a trouble free, squeaky clean UI, lightning speed development period, and maximum interoperable software product can only be described as staggering. The mobile applications development scene has split into at least two categories—native and cross platform development. In this master's thesis, the focus is going to be on exploring the cross platform mobile applications development field of mobile computing.

1.1 Motivation

The most straight forward method of developing an application will be to target native applications with native thinking, native target and deliberately native technology [25]. Besides it runs faster, most of its ailments are probably fully groped through, well documented idiosyncrasies and often a relatively reliable but expensive professional support.

One of the major roadblocks to native development is the number and variety of platforms and mobile devices in the market—making mobile applications development end up with fragmented development arrangements and costly software development strategies [12].

Cross platform mobile applications is widely believed to provide mobile apps developers with a means for writing once and deploying everywhere. Currently, the market is filled with dizzying array of cross-platform development tools. Though hunting down each one of them is hardly the aim of this master's thesis. Below is a sampler list from the English version Wikipedia that will hopefully indicate the growing need to find a way to comprehend this development pattern.

Table 1: Cross-platform mobile apps development tools [150]

Adobe Air	BlackBerry	Gideros Mobile	NS Basic
Marmalad	Bluepring	IwGame Engine	OpenPlug
alcheMo	BREW	Jmango	Particle SDK
AppFurnace	Canappi	July Systems	PhoneGap
Application Craft	CellSDK	Kony	Rhomobile
Appcelerator	Celsius	Lazarus	Tiggzi Mobile
Appception	CloudPact	Meme IDE	Total Cross
appMobi	CoStore	MobiFlex	Unity
Aqua	Corona SDK	MobileNationHQ	WebORB IS
Basic4Android	DragonRAD	Moscrif	WinDev Mobile
BatteryTech	FeedHenry	MoSync	IBM worklight
Bedrock	GeneXus	NeoMAD	Fivespark

Presumably, Telenor, as one of the stakeholders in this report and Norway's biggest telecoms operator, saw the problem and initiated a research in Mobile apps mobility in collaboration with Norwegian University of Science and Technology (NTNU). Apps developed for mobile devices are often restricted to one platform such as Android, iOS, Blackberry's RIM, Samsung's Bada, Microsoft's Windows Phone 7 and others, the hope is to overcome this limitation through a device-platform independent portable mobile applications.

The potential benefits of writing-once-running-everywhere comes from the cost reduction—in having only one code to write and maintain, and the time reduction—being able to write one code and target multiple devices and platforms, making researching cross-platform mobile applications development worth one's effort and findings.

1.2 Objective

The main objectives—research questions—of this masters thesis in Mobile apps mobility or cross platform mobile applications development are:

- What is the benefit of using cross platform SDKs to develop cross platform and telco-enriched mobile apps?
- Formulate a method to evaluate and select the best cross-platform development tools for a developer
- Evaluate cross platform tools using time, technology, maturity, and cost aspects of mobile apps development.
- What tool best supports cross-platform development?

1.3 Research Method

Approach

While the benefit of developing mobile applications that can run on multiple mobile operating systems is clearly visible in that one needs to maintain a single code base to fix, maintain or later evolve into something newer and better, adopting one cross-platform development tool, to that effect, is not as straight forward as the benefits it might bring [9, 44, 60]. Some of the reason for that has been a growing number of tools' entry into the market in a relatively short period of time and the absence of a reliable comparing mechanism of each one of them. The information about each tool is usually found on websites, tool vendors' webpages, academic literatures and industry whitepapers. The mobile development technology is fairly recent and that it is claimed that the majority of players in the industry are experimenting with different ideas to be able to establish a mature technology [31]. Thus, what this masters thesis does is to employ **literature study and experiment**. The literature study will be applied to:—

- Gather information from different sources about the development tools
- Using the information
  - Document and present cross-platform mobile applications development
  - Explore the advantages and disadvantages of cross platform mobile applications development
  - Formulate a general purpose framework which developers or any stakeholders can use to make a decision when selecting a tool based on technology, programming language, no of native APIs support, no of platforms support, cost and availability of support for the platforms
  - And applying the defined framework to select a cross platform tool and make a sample application.

Since there are several cross-platform mobile application development tools, documenting each one of them by going to each one's source of information might prove difficult. One way of narrowing down what is going to be investigated and minimizing the number of candidates will be to look into tools that:

- At least support Android and iPhone platforms
- Are currently active and being maintained
- Support native feature of the phones such as camera, GPS, and others instead of only running in the web browser of the phone and achieve platform independence.

To guarantee the integrity of the information gathered about each tool, personal webpages, personal blogs and discussion forums will be avoided. The literature study for the task will depend on information that will be collected from each vendors webpage, vendor's technical specifications, published academic literatures and industry whitepapers.

The nature of information that will be collected about each tool will be on:—

- Number of platforms supported
- Number of native features supported
- Programming language used
- Architecture for achieving platform independence
- Whether they are proprietary—incur cost—or open source
- Developer effort and time to develop an application using the novelty of the technology they employ
- Pricing model of the tools (some are free but require fees for support, some have no fee and no support, some are entirely commercial, some are in between)

The evaluation will be done in a tabular framework by integrating developer or stakeholder requirement and later give a score each tool based on the technology, architecture, cost, platform and API support.

A company, for instance, which has an experience developing applications using the technology of Java, obviously, will find it easier to get its developers up and running in cross-platform MAD tools that use Java, in less cost and effort.

Using the experiment method, it will be possible to explore two best candidate cross platforms MAD tools in how each implement the same functionality, user interface, speed of the applications in each tools version, application size they end up with and the perceived running and compilation speed.

Structure of the report

Chapter 2 will discuss on mobile applications development, device evolution and the current status of mobile computing. While Chapter 3 discusses cross platform development tools, how cross platform is achieved, and the benefits of cross platform development and a framework for evaluating tools will be presented in this chapter. Chapter 4 will comprise the discussion and evaluation of this work followed by conclusion.

2 Mobile Applications Development

To help drive mobile applications concepts home, a summarization of key devices, mobile platforms and general challenges of mobile applications development will be quite useful. This section includes a discussion to that effect.

Background

2.1 Mobile Device Evolution

A mobile phone is a telephone hardware that is capable of connecting to a telephone system infrastructure using radio frequencies rather than wired connection [109]. At the beginning, mobile devices were characteristically very prohibitively expensive, big in size, extremely constrained in computational resources and with short battery life [125, 149]

The Motorola DyanTAC 8000X had a price tag of $3,995, 8 hours of standby, 35 minutes of talk, 13 inches tall body, and weighting nearly a kilogram with no display and blind—no camera, but considered a cutting edge technology in 1983. Obviously, this was not designed for a pocket and it was meant to be used in cars.

However, this pioneering device set a revolution that will completely change the mobile device scenario from a terribly inefficient cumbersome object to a must-have device which gave people freedom, continuous presence and ability to work anywhere—the birth of a completely new reality [58, 84].

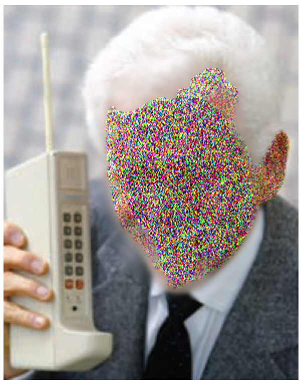


Figure 1: Motorola DynaTAC 8000X [32]

After the millennium, devices started to get smarter and slimmer with decent displays, connectivity options—Bluetooth and infrared sometimes Wi-Fi, and a multitude of fancy features enabling them to be personal devices being carried around 24/7. Currently, mobile subscribers number is estimated to stand at 5.3 billion worldwide [81] on top of having been able to earn being one of the top 10 greatest inventions in the telecommunications industry [41].

Figure 2: Easily carried mobile phone device [92]

2.1.1 Smartphones

A smartphone is a new breed of mobile computing device with advanced computing resources and equipped with technologies that facilitate access to the internet, run web applications, email sync, run and install applications [29].

A typical smartphone will have a touchscreen, access to the internet and an operating system to run apps on [100]. A smartphone packs a whole raft of features such as a still and video camera, a calculator, an alarm, a watch, an mp3 player, a SatNav, and a gaming facility—features that are driving to extinction specialized devices that used to run these features alone [35].

Despite their inconvenient input system and unreliable battery, smartphones are redefining the computer industry and have already sent some field of computing—the rather lackluster, yet-to-shine field of wearable computing—into decline or forced them to realign and rearrange themselves making smartphones as their command center [14].

Multicore processing has now been the dominant pattern of processing architecture for quite a number of years on stationary computers, but in the first quarter of 2011, a number of smartphone manufactures brought to the market an extremely miniaturized smartphone that boast dual-core processor on board, a 1GB memory, a spacious storage capacity up to 32GB, and the ability to play and capture HD Video, connecting to HD monitors, which could make these devices easily pass as a veritable replacement for stationary system units for the majority of consumers—the majority assumption being text processing, browsing, gaming people. After the first dual-core processor smartphone LG Optimus 2X [90], Motorola came once again with a brand new smartphone that had 1GHz dual-core processor, an Adobe Flash support, qHD and a phone that will change into fully functioning PC when connected to a bigger screen via an HDMI port [103]—a clear sign that mobile computing is evolving into much more than smartphone as we know it today.

Figure 3: Android [136], Windows phone 7 [117] and iPhone [73] smartphones

2.1.2 Tablet PCs

What started as a block of stand-alone, bigger screen, stylus dependent electronic e-book reading device evolved into one of the most popular mobile computing device that would later provide wireless internet connections and navigational services just like smartphones [62, 123]. Since their debut to the computing world in 1989 [144], tablet PCs have come a long way thanks to technological breakthroughs especially in better touch based writing to achieve that mystique blurring between paper and digital ink brush.

Figure 4: ASUS's latest Quad-core Keyboard dockable tablet PC [43]

The nature and feel of tablet pcs with human working manner—holding everything in hand and scratching with the other one—has helped them win the hearts and minds of consumers to be one of the most successful products of the past few years. The possibility of free handwriting, imitating actual brush painting, drawing has made them a source of wonder for higher education professionals hoping to introduce them in their class rooms [59, 83].

To become the best mobile computing device, mobile devices have to beat the challenges of small screen, portability, sufficient processing power and seamless connectivity, and tablet pcs seem to have come close to beating these challenges [4].

2.2 Classic MAD Challenges

Inherently, Mobile applications development came with its own challenges that took a decade of effort to eliminate these technological hurdles. The technological breakthrough in mobile devices lead to the proliferation of infinitely variant devices with varying architectures, device capacity and features [69] plus there were, and are still three classic mobile applications development challenges that must be taken into account when conceiving mobile applications either native or cross platform applications. Those are [55, 66, 124]:—

Mobile Device Portability

If an application is to hit it with end users on a mobile device, the mobile device has to be small, and convenient to handle—not one which will be worn around the waist, on arms and head or carried in the bag. Small and convenient devices come with low power due to smaller battery size, constricted display and user interface necessitating alternative input mechanisms such as speech recognition, and smaller storage capacity which might render the device useless when it comes to business and corporate data intensive applications or multimedia content. Another key problem to consider is risk to data due to loss, theft or damage to mobile devices.

Mobility challenges

As mobile devices achieve full mobility, achieving smooth connectivity becomes a major problem. Mobile devices have to search for a connection spot and establish connection as they move dynamically—an operation which could drain the battery fast and a stressful feature for traditional connection model. Address migration, location dependent configuration such as IP address, and service locality starts to get in the way of smooth operations of a connected mobile application.

Wireless communications

Greater bandwidth variation—from high to low network bandwidth, susceptibility to security problems, signal corruption, signal blocking buildings, and natural terrains together bring about an extra overhead in wireless communications. Wireless networks range from Wi-Fi to GPRS to 3G hotspots. A robust mobile applications development process must take into account the above challenges when developing mobile applications.

2.3 Mobile Platform

The definition of mobile platform is better defined from the stand point of traditional operating systems. An operating system is the lower level middle software between the bare hardware and the higher level applications program [131]. Operating systems provide hardware abstraction, driver and networking model, security architecture, process and memory management facilities for optimum utilization of hardware resource [135]. A Mobile platform is the equivalent of traditional operating system for mobile devices—including tablets and smartphones.

The English version Wikipedia defines mobile phone operating systems as "A mobile operating system (Mobile OS) is the operating system that controls a smartphone, tablet, PDA, or other mobile devices. Modern mobile operating systems combine the features of a personal computer operating system with touch screen, cellular, Bluetooth, Wi-Fi, GPS mobile navigation, camera, video camera, speech recognition, voice recorder, music player, near field communication (NFC), personal digital assistant (PDA) and other features" [151]

2.3.1 Types of mobile platforms (Mobile OS)

Android mobile operating system

Android is an open source mobile platform that depends on Linux kernel to provide common operating system services to mobile devices. Android operating system stack provides memory management, process management, network model, driver model, security and an abstraction between mobile hardware and the higher level mobile device applications [16, 67]

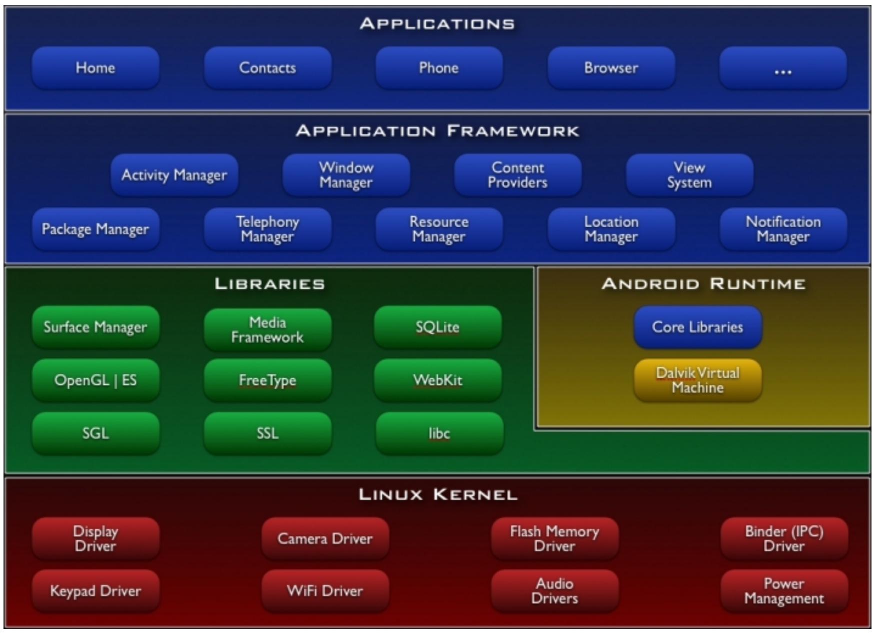


Figure 5: Android System Architecture [16]

Android is not only an operating system but also a mobile software toolkit conceived by telcos and handset manufacturers alliance—Open Handset Alliance and further developed by Google—to specifically target mobile devices with open standards and free distribution [11, 28]. In Android, all applications are considered equal with relatively equal access to most of the core applications and hardware for maximum exploitation of the Android-handset combination [42].

Table 2: Android device as a minimum requirement [51]

Feature	**Minimum Requirement**
Chipset	ARM-based
Memory	128 MB RAM
Storage	Mini or MicroSD
Primary Display	QVGA TFT LCD or larger, 16bit color
Navigation Keys	5-way navigation/ power, camera and volume controls
Camera	2MP CMOS
USB	standard mini-B USB interface
Bluetooth	1.2 or 2.0

At its core, Android is designed for resource poor devices with limited processor power, limited memory, no swap space, battery powered and sandboxed mutually exclusive—each running in their own confined virtual working space—applications using the Dalvik virtual machine architecture (DVM). Each application that runs on Android will be assigned its own DVM which in return provides all the operating system services to each application [26, 33, 51].

Now Android is the most dominant smartphone operating system with a whopping 52.2 percent market share in the 3rd quarter of 2011 [107]. See below Android:52.5, Symbian: 16.9, iOS: 15, RIM: 11, Bada: 2.2, Microsoft: 2.7, Others: 2.5.

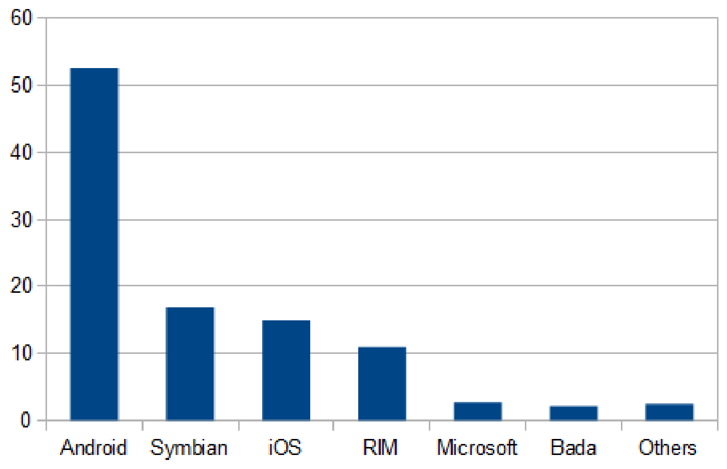


Figure 6: Android Market share percentage for Q3 2011. [107]

Apple iOS mobile operating system

The Apple iPhone hit the market in January 2007 and brought with it the iOS operating system—an operating system filled with goodies that seemed to tickle everyone's fancy. iOS is a heavily guarded proprietary operating system whose fate is tied to a romanticized device in the name of iPhone and other iFamily devices such as iPod touch, and iPad devices. Before the advent of Android, the iPhone proved to be a formidable weapon in the war of smartphone dominance and successfully positioned iOS itself in the top rank in just a short time since its debut, now just trailing behind Symbian and in 2007, it was considered as the invention of the year in five categories among which were being beautiful and defining the future [88, 91, 157].

The iOS with its layered system architecture has since evolved to include advanced features of Voice over IP, multitasking, threading, folders, a unified mailbox and other features [80, 94].

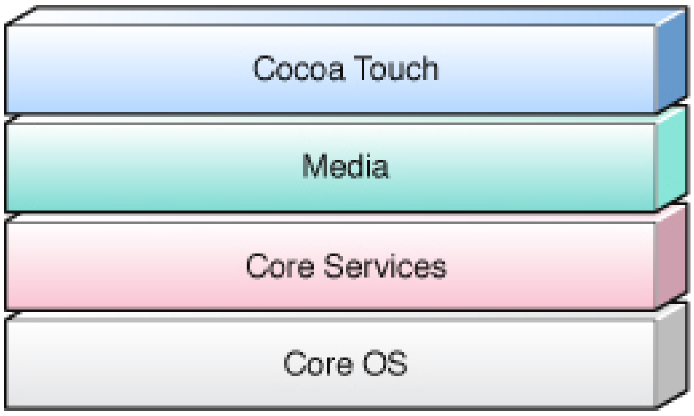


Figure 7: iOS system architecture [80]

iOS provides services that any operating system provides for iFamily devices. In addition to its pioneering technology and innovative design, iOS was considered to be a status symbol making it a market leader in a short period of time. Consequently, other operating systems in the market had to make way for it and die. One of the obstacles to developing applications on iOS is that it necessarily required a license for developers and that the system is closed and that the apps are Apple approved products on the iGlue online store of iTunes app store [61, 160], and not to mention the that fact that the devices come at a premium.

Apple's iOS currently stand third in terms of market share in the third quarter of 2011. See Fig 6

Wholesale Applications Community (WAC)

WAC is a non-profit relatively new endeavor by giant telecom operators such as Telenor among others. Over the past years—with the explosive growth of smartphone use and mobile platforms entry into the market, mobile apps have started to erode the usually lucrative business model of providing mobile services to subscribers for a fee. Today the mobile computing environment is constantly changing and it has turned the table on telecom operators, making them more and more irrelevant into what many call as a dumb pipe. While their mobile network serves as a high way for mobile data traffic, they see no dime in the process. To respond to this growing threat, they got together and concocted WAC—a brand new mobile platform [2, 63]. Note WAC is not a platform rather it is a mobile applications development framework.

Apart from serving as a counteract response, WAC promises to unify the fragmented scenario of mobile services into an open for everyone—everyone being members or adopters of WAC—mobile development platform to offer application developers, device manufacturers, network operators and end users with network as a service (NaaS) mobile cloud computing services [64, 133].

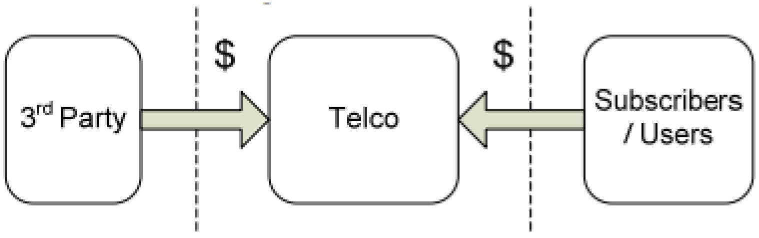


Figure 8: Telco's wish of mobile apps arrangement [2]

WAC is widely believed to furnish new and mature mobile network service APIs which until recently were only used by telecom operators. Call control, messaging, location, payment and profile services are some of the best ones that will be made open to third party developers [1, 133].

The recent WAC 2.x standard has APIs for accelerometer, calendar, camera, contacts, device interaction, device status, file system, geolocation, messaging, orientation, tasks, WebView and address. WAC evolved from Joint Innovation Lab (JIL)—a project conceived by two of the biggest mobile operators in the world which are China Mobil and Vodafone together Verizon Wireless in the US. The WAC not only tries to standardize APIs for accessing native device features from a web app but also confirms to W3C standards. WAC together with its own app store is set to make telecom operators take up arms with market dominant Google [18, 39].

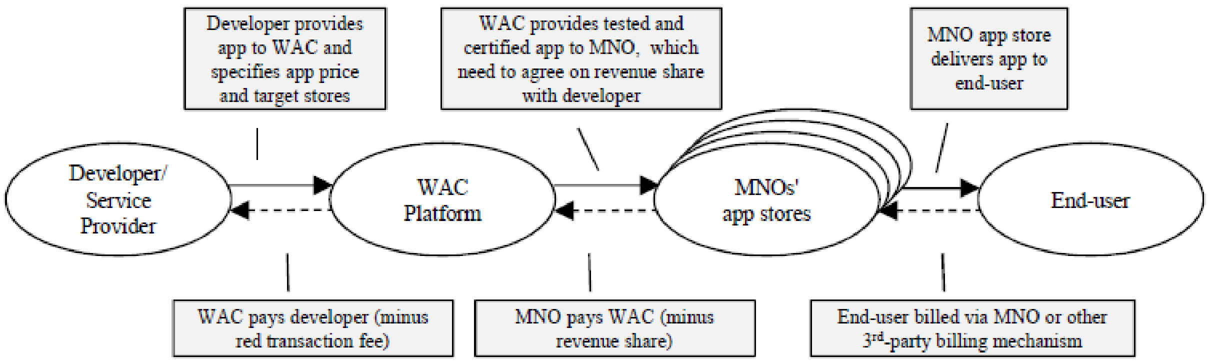

Developer provides app to WAC and specifies app price and target stores
WAC provides tested and certified app to MNO, which need to agree on revenue share with developer
MNO app store delivers app to end-user
Developer/ Service Provider
WAC Platform
MNOs' app stores
End-user
WAC pays developer (minus red transaction fee)
MNO pays WAC (minus revenue share)
End-user billed via MNO or other 3rd-party billing mechanism

Figure 9: Illustration of WAC platform and its business model for telcos [133]

WAC is a cross platform development framework that comes in three flavors—Aplix, Borqs and Obigo which also provide runtime environments to interpret the mobile app—of eclipse based SDK plugins. And using these W3C standard enforcing plugins, a developer can author a native platform agnostic mobile apps in the web technologies of HTML, CSS and JavaScript. WAC has three major system components—the runtime which provides security, privacy and application interpretation, WAC mobile app is a mobile application which is an written using CSS, HTML and JavaScript and device accessing APIs [38, 50]. The key to WACs cross platform deployment is its ability to float on top of a runtime environment, leaving a clean gap from the underlying operating system. Below is a an architecture reconstructed from the WAC developer wiki.

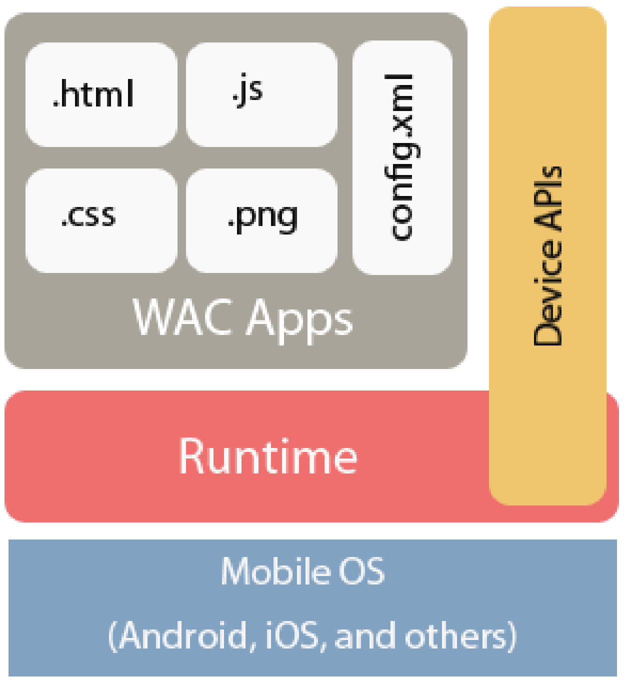


Figure 10: A reconstructed inner structure of WAC Framework [38]

WAC 2.0, supporting HTML5 and multimedia, went commercially live in February 2011 with eight operators and 12000 WAC compliant widgets. In-app billing, user profiling, authentication and being able to change a device without losing apps are considered three of the most valuable additives telecom operators could integrate into WAC 3.0 specification, which itself was expected to be released since September 2011 [36, 37, 78, 79, 158].

Windows Phone 7.5/ Mango

Microsoft, having learnt from its past failures and inefficient mobile computing strategies based on the .NET compact framework, reengineered its approach from the ground up and released a brand new mobile platform—Windows Phone 7, which has since been upgraded to Windows Phone 7.5/ Mango [30].

This time Microsoft seems to have more than prepared for the battle ahead in the smartphone war. Not only has it planned to stave off device fragmentation early on by requiring a strict device specification and design guideline but also packed about 1500 unique APIs as part of its arsenal—which it claims move the stage from OEMs to end user experience. See Fig 11 below. The figure shows how Microsoft wants devices to look like to run its latest Windows Phone platform. Windows Phone also opted for methodical avoidance of multicore processors, kept rapid applications development strategies from the .NET, its apps come with a revocable license, an IE9 based HTML5/ JavaScript support and it seems it has given security some thought by confining each running app in an isolated least privilege host processes [3,30,113]. Strangely enough, Microsoft took the trouble of emphasizing how the start button is to be placed on a device.

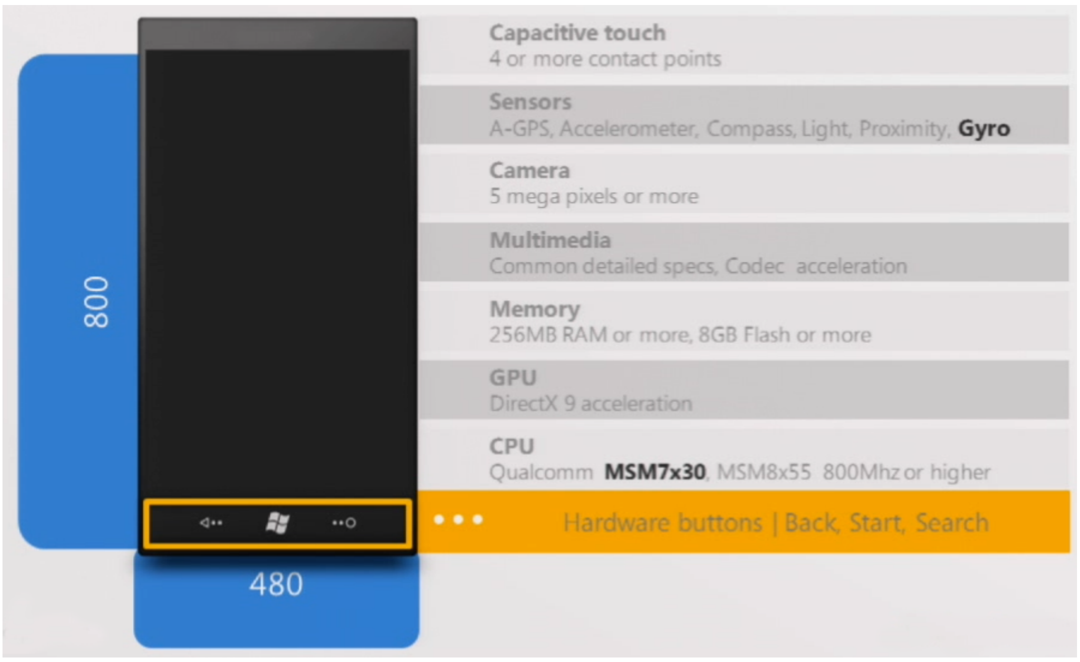

Capacitive touch
4 or more contact points
Sensors
A-GPS, Accelerometer, Compass, Light, Proximity, Gyro
Camera
5 mega pixels or more
Multimedia
Common detailed specs, Codec acceleration
Memory
256MB RAM or more, 8GB Flash or more
GPU
DirectX 9 acceleration
CPU
Qualcomm MSM7x30, MSM8x55 800Mhz or higher
Hardware buttons | Back, Start, Search
800
480

Figure 11: Windows Device Specification for OEMs [3]

With its highly acclaimed tile based simplistic user interface, Windows Phone market share is expected to surpass iOS and Blackberry's RIM between 2013 and 2015, if speculations are anything to heed in this fast shifting technological scenario [77,152].

Breaking a long tradition of being tight-lipped about its technological architecture, Microsoft demonstrated the Windows Phone Platform architecture on a presentation slide labeled Microsoft confidential, in it it depicts that Mango—as it is known by its code name—has two major interests for application developers—the Silverlight framework which is the same as the old windows desktop rapid applications authoring tools using Expression Blend and the XNA framework for game and entertainment developers in addition to the common base class libraries which are the foundations of any windows application. The framework's main development language is C# and anyone with previous .NET development experience can get up and running in no time [3, 87, 105].

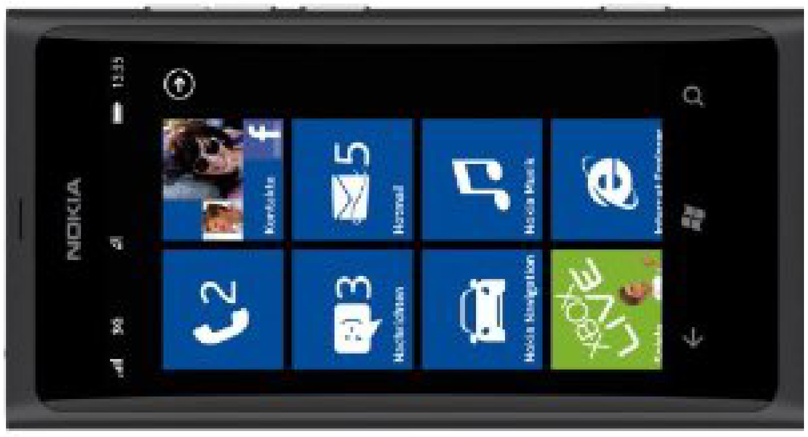


Figure 12: Windows Phone iconic UI on a Nokia Lumia 800 [159]

Symbian

Symbian OS, v6.0, was first seen in the market in early 2001 on Ericsson R380 and Nokia 9210 Communicator mobile devices. Initially, Symbian was conceived as a force for uniting customers, mobile operators and device manufacturers into an open and standardized approach to wireless services development and provision [111, 121].

Symbian was to fulfill certain, at the time, very fundamental requirements in the mobile computing sector. Working on a stand-alone portable device, being future proof, having open, equal and fair licensing terms, and open standards based apps development was some of the central ideas in Symbian. And the system designs were meant to accommodate event-driven, graphical and cross-platform technological patterns with client server model on top of providing facilities like streaming, data persistence together with battery optimization fixtures [101]. During the early days of Symbian, there was too much expectation that Symbian will enable application developers and device manufacturers to come up with innovative network applications and as a result increase mobile service subscribers that big companies—the like of Ericsson, Nokia, Panasonic, Psion, Siemens, Sony-Ericsson, Fujitsu, Kenwood, Sanyo and others with big plan, most notably Nokia—having invested 50 million USD—jumped on board to make it successful [111].
Fig 14 is an overview of Symbians system architecture reflecting its cross-platform tendencies.

Symbian—considered to be one of the most developer hostile and complaints ridden OS [57, 104]—is a light weight mobile operating system written in C++ and uses small system architecture to optimize mobile applications performance on a constrained device [34].

Having floundered for almost a decade with lack of a clear goal and strategy, and its inability to stop device fragmentation, it had to later be elbowed for Windows Phone 7 by its sponsor—Nokia. According to Gartner and IDC, Symbian—in terms of market share—will successfully be out of the mobile computing scene in about 3 years [77, 152].

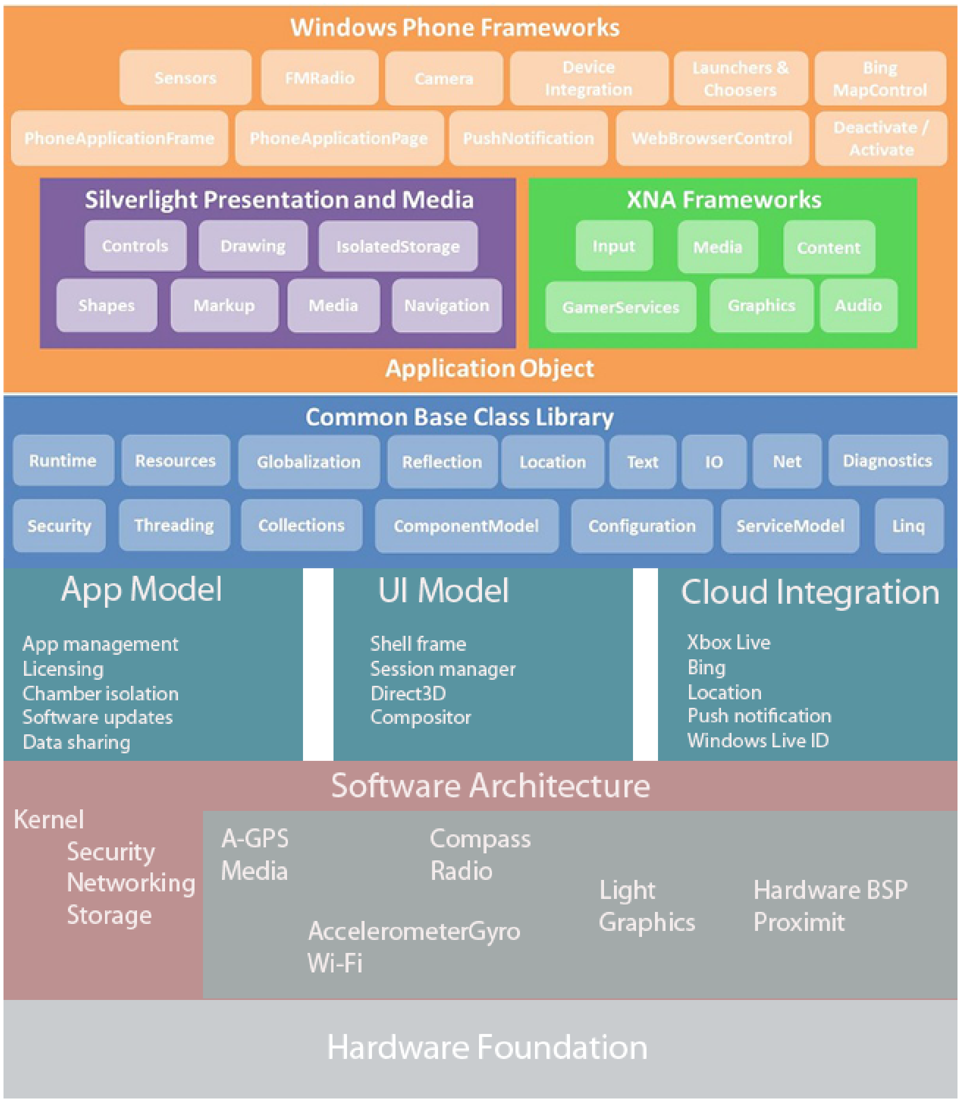


Figure 13: Windows phone architecture [3]

During its golden age and when it ruled the market, the Symbian platform had its own App store and Android Google Play look alike—Nokia Ovi internet service. Ovi was specifically designed for Symbian based smartphones providing app store, navigational maps, media sharing, messaging, gaming, music, contact repository, calendar and file access services [57, 141].

BlackBerry RIM

BlackBerry is a smartphone which is a product of the company RIM—Research In Motion. BlackBerry RIM commands the fourth market share of smartphones [107]. BlackBerry is not a one off operating system enabled device product like Google's Android and Apple's iOS. What is different about BlackBerry smartphones is that it comes as an integrated service between RIM, telecom operators and the smart-

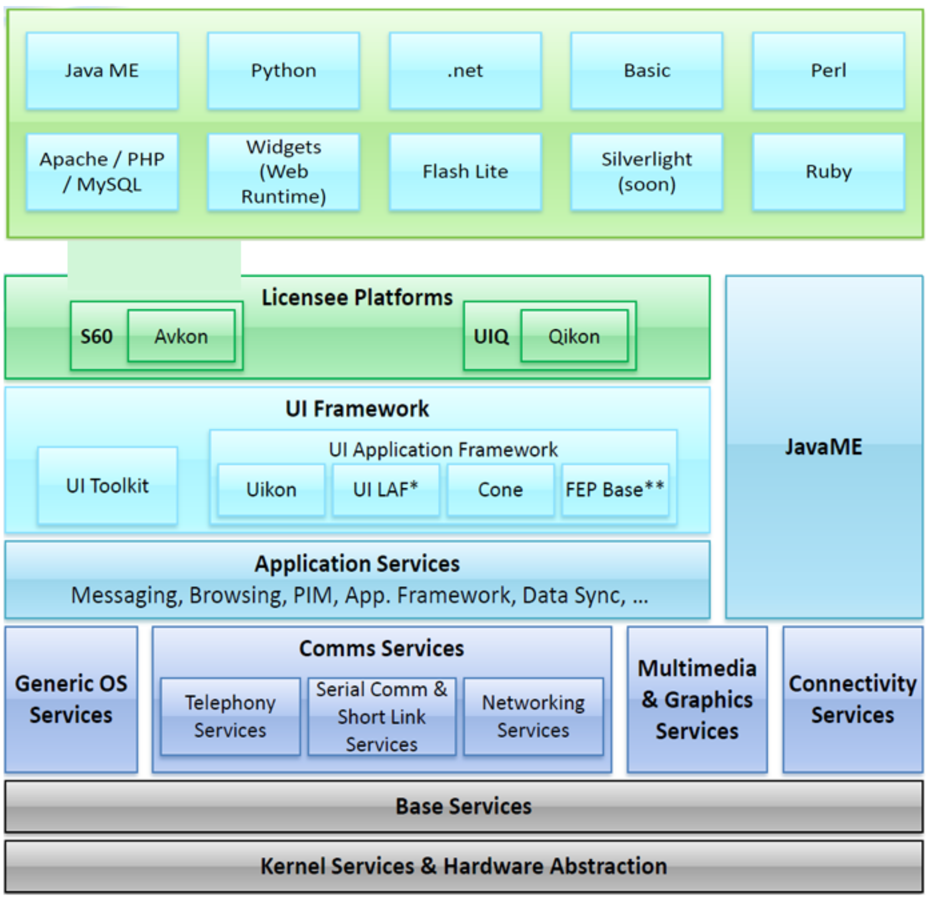


Figure 14: Symbian system architecture [57]

phone hardware. BlackBerry RIM is well-known for its highly secure corporate email communications solutions. BlackBerry smartphones tap into a bigger IT solution infrastructure to help users stay connected through information services such as email, phone, organizers, intranet and multimedia applications [88, 93].

Fig 15 shows the BlackBerry's enterprise solution architecture. BlackBerry smartphones use telecom carriers network to connect to RIM's Network Operating Center (NOC) [23]. The BlackBerry Enterprise Solution (BES) provides a means to access corporate emails and business critical applications. The BES has six major components to render its service to BlackBerry smartphone holders. Those are [22]:

- BlackBerry Enterprise Server — This is the server that integrates with enterprise services to provide messaging and collaboration facilities and also handles all data traffic from the BlackBerry smartphones.
- BlackBerry Mobile Data System — This includes everything from developer tools, administrative services, and BlackBerry Device software that is used to author, deploy and manage applications for the BES.
- BlackBerry Smartphones — The smartphones are one of the components of the enterprise service architecture that provide access to email, MMS, SMS, web and other applications.
- Devices with BlackBerry Connect Software — This is a device with the ability

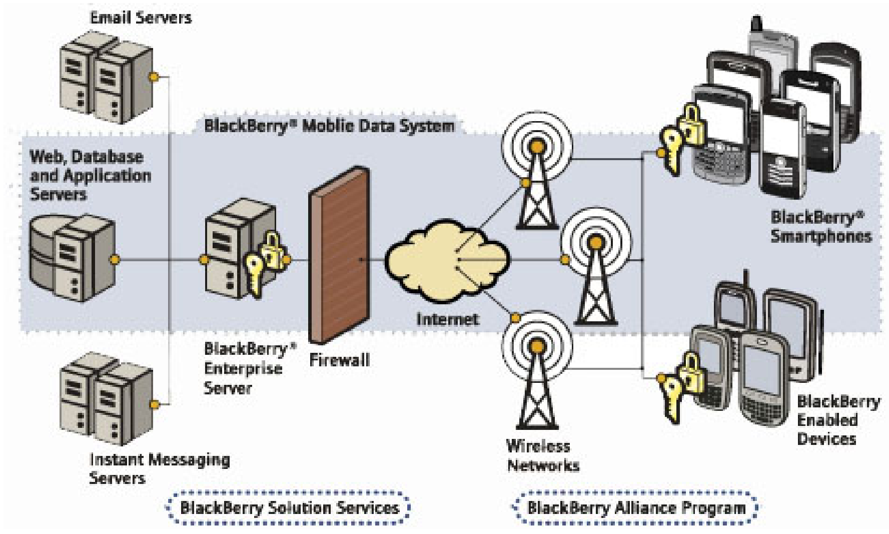

Email Servers
BlackBerry® Mobile Data System
Web, Database and Application Servers
BlackBerry® Enterprise Server
Firewall
Internet
Wireless Networks
BlackBerry® Smartphones
BlackBerry Enabled Devices
Instant Messaging Servers
BlackBerry Solution Services
BlackBerry Alliance Program

Figure 15: BlackBerry Service Architecture

to run the BlackBerry Connect software and maintain a link to the BlackBerry Enterprise Server.

- BlackBerry Alliance Program — This is a community of software developers that make applications for the BlackBerry enterprise solution.
- BlackBerry Solution Services — This consists of technical support, training and certification programs that will help organizations that need to deploy and use BES.

Samsung Bada

Samsung Bada is another mobile platform which promises a more interactive services using UI controls, Adobe Flash support, in-app-purchasing, and SNS (Social Networking Services) integration. Bada comes with features such as multipoint touch and 3D graphics. In Samsung Bada, applications are developed using C++, Flash and web programming languages [20, 46].

Table 3: Android Vs Bada OS comparison [19]

OS Layers		
Android	Bada	Function
App Framework	Open API framework	Apps and Widgets
Libraries	Service Library	C++ library
Kernel	Kernet	Linux kernel

2.4 Mobile Applications

A mobile application, mobile app, is a software which is written to run on mobile computers. They have the feel of a form based desktop application but with an internet

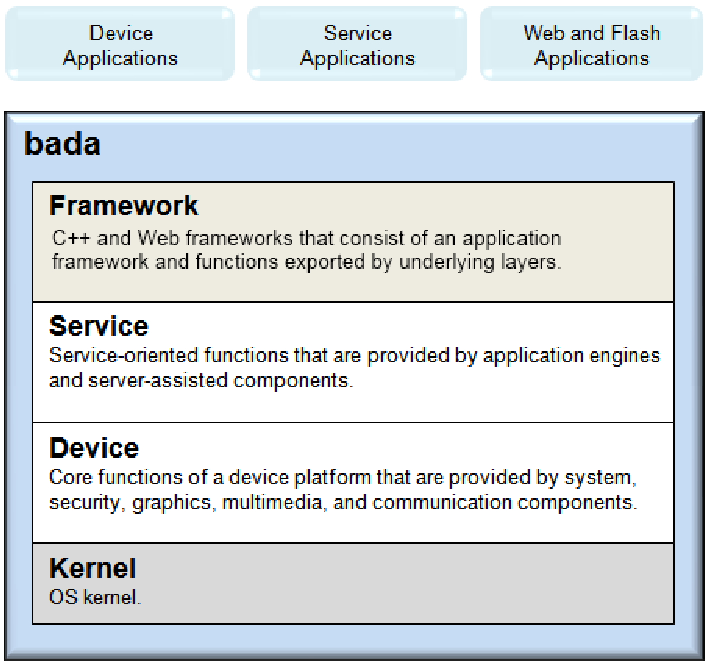

Device Applications
Service Applications
Web and Flash Applications
bada
Framework
C++ and Web frameworks that consist of an application framework and functions exported by underlying layers.
Service
Service-oriented functions that are provided by application engines and server-assisted components.
Device
Core functions of a device platform that are provided by system, security, graphics, multimedia, and communication components.
Kernel
OS kernel.

Figure 16: Bada System Architecture [47]

connection, the applications content come organized in menus, views together with multimedia contents. They can also be offline, batch-and-cache or always connected apps. They are developed using traditional software development methodologies in combination with specialist device emulator or testing simulators. Mobile apps are distributed usually with platform specific market places such as Google Play, Apple's App Store and Windows Phone market. A registered developer publishes a mobile app on a market and users acquire and deploy the apps on their device through the application markets. The figure below describes the application distribution and deployment lifecycle [15, 74, 96].

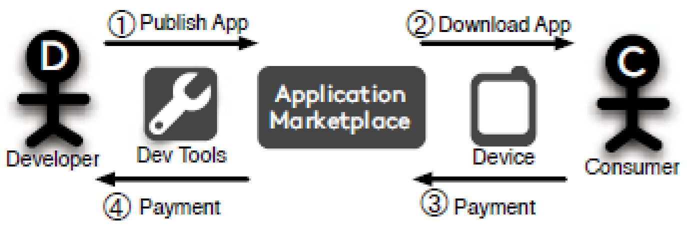


Figure 17: Mobile apps distribution and development life cycle overview [70]

Inside the above model, a number of financial arrangements and application intelligence and other communications take place between third party registered users and processes. The app markets are centralized mobile application platforms providing integrated clearing house between platform owners and application developers as well [70]. Here is a more detailed description from Microsofts Windows Phone marketplace.

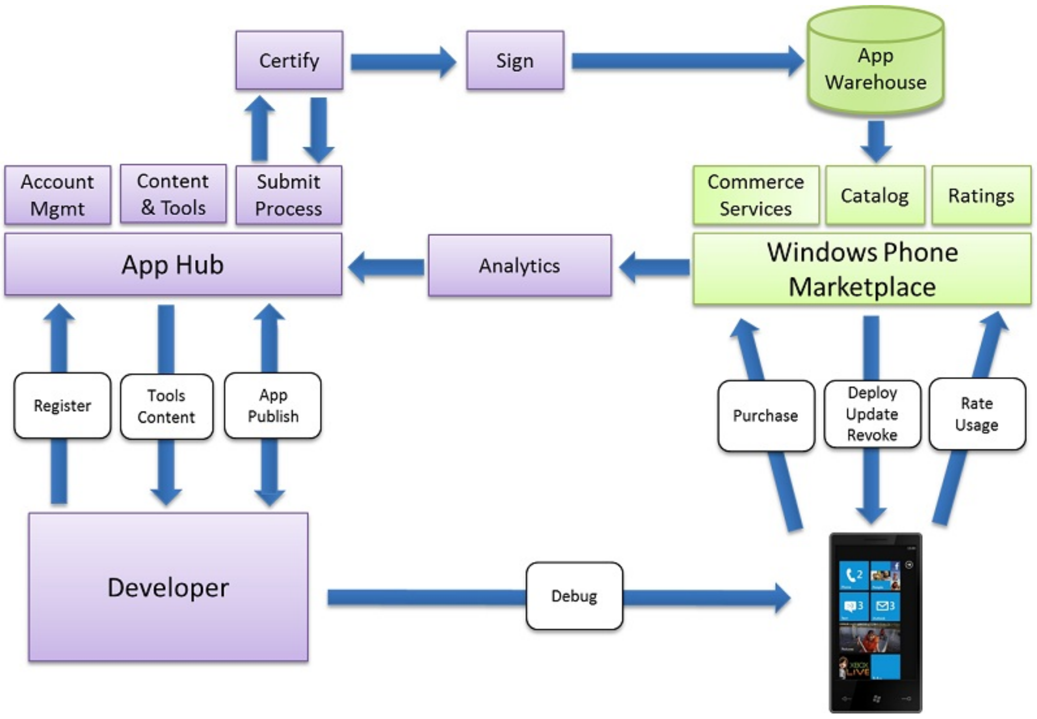


Figure 18: Windows phone market model

Until fairly recently, the mobile industry was a privilege to mobile network operators [72]. They probably had an upper hand in defining the technology or how service was rendered. With the arrival of Apple and Google, they seem be to increasingly sidelined.

Mobile applications are distributed from developers to users through portals that are centralized or decentralized, where centralization provides access to mass market distribution through shops like Google play and AppStore. Full portal integration means having dominion over the entire application distribution process, a good example for this would be Apple's AppStore—no app will be installed on a device without the approval of Apple [71].

Table 4: Overview of platform providers [71]

Platform	**Technology**	**Portal**	**Devices**	**Integration**
Apple	Closed	Centralized	Uniform	Full
Google	Open	Centralized	Various	Portal
Linux	Open	Decentralized	Various	Device
Nokia	Open	Decentralized	Various	Full
RIM	Closed	Decentralized	Various	Full
Windows	Closed	Decentralized	Various	Portal

According to Gartner [107], Android is set to dominate the mobile applications scenario in the near future. Android has been crouching faster and nearer to Apple in terms of mobile apps economy and number of applications on the market [112, 119, 129], and the mobile applications market is assumed to have amassed 9 billion USD in 2011 [71] while Google claimed to have had its 10 billionth app download in December 2011 [24].

Mobile app market monitor Distimo reported that of the major app stores it monitors—including Apple App Store for iPhone and iPad, Apple Mac App Store, BlackBerry App World, GetJar, Google Play, Nokia Ovi Store, Palm App Catalog and Windows Phone 7 Marketplace—iPhone did well than android in the amount of revenue generated while Google performed far better than iPhone in the freemium business model in the US in 2011 [48]. The figure below shows the proportion of revenue generated by freemium apps in the US in November 2011.

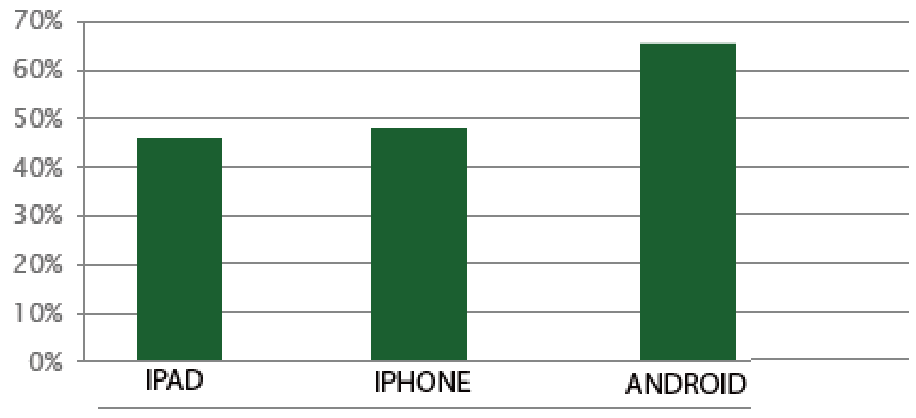


Figure 19: Android and iOS performance in the US [48]

2.4.1 Types of mobile applications

Mobile application can range from purely communication to gaming to entertainment. Despite the smaller user interface, hardware limitations and other constraints, application development is infinitely varied. Google play and Windows marketplace have classified their apps into two major classifications: Games and Apps. In the apps category, there are entertainment, comics, communications, finance, health and fitness, and other. The table below will sample few types of mobile applications by function to users.

Table 5: General Mobile apps category [122]

Mobile Application Tyes		Example from Google play
Communication	Email Clients	Facebook
	IM Clients	
	Mobile web & browsers	
	News	
	Social Network Clients	
Games	Puzzle (Tetris, Sudoku)	Angry Birds
	Cards (Solitaire, Roulette)	
	Action (Doom)	
	Sports (Football, Soccer)	
Multimedia	Graphics viewers	youtube
	Presentation viewers	
	Video players	
	Audio players	
	Streaming players	
Productivity	Calendars	CalenGoo
	Calculators	
	Diary	
	Notepad/ Memo	
	Spreadsheets	
	Directory services	
	Banking & finanace	
Travel	City guide	FlightTrack
	Currency converter	
	Translator	
	GPS/ Maps	
	Itineraries	
	Weather	
Utilities	Profile manager	
	Idle screen	
	Address book	
	Task manager	
	Call manager	
	File manager	

2.4.2 Telco-enriched apps

SMS — Short Messaging Service

SMS is a mobile network services that is used to transfer short, usually a 160 character text, message to other mobile phone users over a cellular network [128]. SMS is an integral feature of GSM, TDMA and CDMA mobile networks [110].

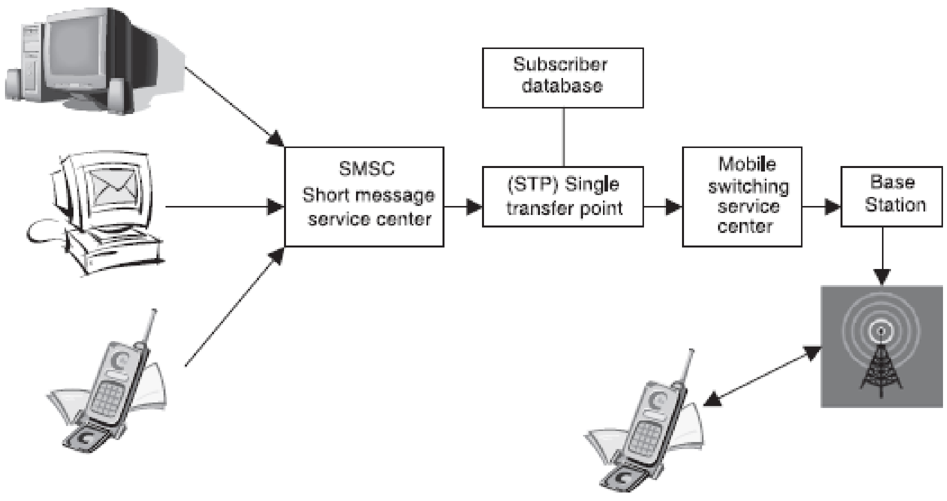

Subscriber database
SMSC Short message service center
(STP) Single transfer point
Mobile switching service center
Base Station

Figure 20: SMS System [110]

SMS is used in business communications such as banking and individual communication [89]. SMS provides a store and forward system—which makes it possible for the message to be stored in a telecom operators infrastructure until the receiver is ready to accept it. SMS can be received at any time even while a phone call is in progress. SMS provides a low cost non-voice communication system [99].

USSD — Unstructured Supplementary Service Data

USSD is a session-oriented and transaction-oriented GSM technology with a shorter response time for menu based interactive applications or could also serve as a trigger for other actions [99, 128]. Unlike SMS, USSD communications are not stored and forwarded. It, instead, sets up a virtual connection between communicating devices [110]. It works based on sending predefined codes such as *150# for requesting current balance on Chess telephone service in Norway.

Network-based Geolocation

Network based location determination is a facility that is provided by cellular network infrastructure the phones use [7]. Location information is computed from time of arrival, time difference of arrival time, angle of arrival, timing advance and multipath fingerprinting. Network-based geolocation helps acquire startup of location information—TTFF, time-to-first-fix—faster than GPS and consumes less power. Network-based location information are claimed to fare far less accurate than GPS and require considerable investment on upgrading cellular networks for such services [49, 126].

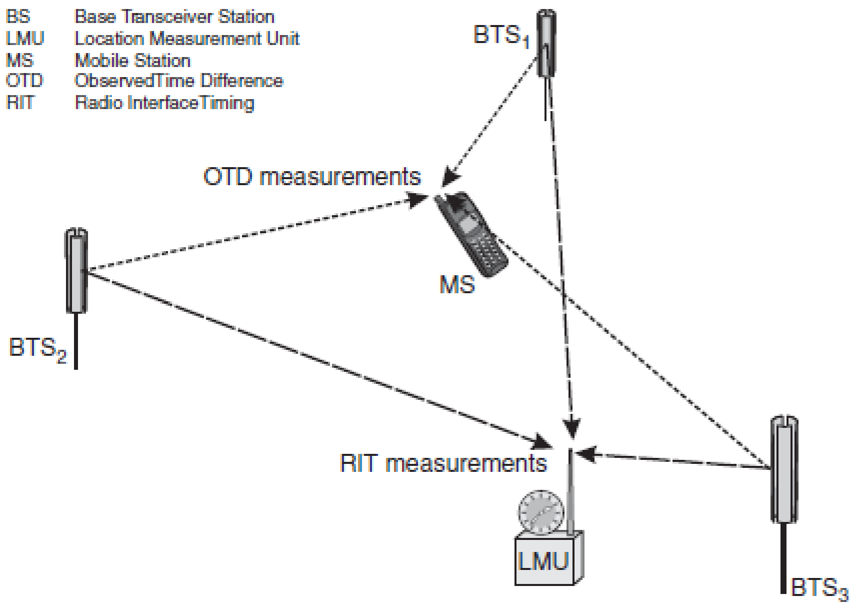

BS Base Transceiver Station
LMU Location Measurement Unit
MS Mobile Station
OTD ObservedTime Difference
RIT Radio InterfaceTiming
BTS$_1$
OTD measurements
MS
BTS$_2$
RIT measurements
LMU
BTS$_3$

Figure 21: Cellular network based positioning [85]

2.4.3 Mobile Enterprise Applications tools

Enterprise application tools are tools that involve the whole mobile applications development life cycle, just like the SDLC in traditional software development life cycles. They are meant to develop enterprise mobile applications that have an extensive backend systems that the mobile apps have to rely on. They include everything from development to integration to publishing to management. These tools come with database connectors, middleware, app hosting mechanisms, push messaging and policy management [98]. Examples of such tools are Rhomobile and Worklight.

2.4.4 Mobile Applications Development tools licensing

One of the factors to consider in mobile applications development is the cost of the tools and technologies in terms of acquisition or licensing cost. The fact that a tool is commercial or properietry essentially means that the tool comes with some sort of cost associated with it. To help avoid the inevitable, free and open source tools come for free but with some inalienable license agreements which will add up to the cost of development of mobile appliations development.

MIT Licensing [76]

MIT licensing allows a developer to copy, use, modify, merge, publish, distribute, sell of software in anyway a developer wants with the condition that the following text is included in the copy of the software. "THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION

OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE“. The only restriction being that the above statement be included in the modifications.

Apache 2.0 Licensing [56]

Apache 2.0 licensing has a different approach in treating rights and patents. Any licensing might be for both using the right or the patented material. Rights once acquired can be used infinitely and any right given in one country is applicable anywhere. This makes rights irrevocable and free given and with no royalty. A developer can use this licensing by including the following structured text.

"Copyright

$$year$$

$$nameofcopyrightowner$$

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License"

2.4.5 Common native features of a mobile platform

There are several native features in a mobile phone. Some of the most common native features supported in a mobile are described below.

Accelerometer

Accelerometer is a feature of a mobile phone that is used to detect the orientation of a mobile device or how the phone is held by a user. It can be used to detect motion in three dimensions [114].

Geolocation

Geolocation is an API that is used to provide or access geographical location of a mobile holder or the mobile device to a computer application. Location information expressed using the earths latitude and longitude are usually collected from Global Positioning System (GPS), IP address, Wi-Fi, Bluetooth, MAC address or GSM/CDMA mobile networks [142].

Barcode

Barcode is an API used to read barcodes through a mobile device camera and interpret the information encoded in the barcode [145].

Calendar

The Calendar API gives access to the native calendar functionality and data of a mobile phone platform [145].

Database

A fast lightweight embedded SQLite database module in a mobile platform. It is used to create, access and store data on the mobile phone and accomplish database management tasks [45, 132].

Media

The media feature enables to access multimedia functionalities of a mobile platform such as playing and recording of audio and video. Access to platform players and multimedia file operations [138].

Network

The network feature of a mobile phone platform provides access to a mobile phone devices cellular and internet connection [114, 138].

Notification

Notification allows to raise audible, visual or vibrational notification [114]

3 Cross platform Mobile Apps Development

3.1 Mobile application Development

In mobile applications development there are certain major tasks whether it is native or cross platform mobile applications development.

Develop

In mobile applications development, one of the main tasks is the development process. The development process involves having a profound insight into the software users, the target platform, the programming languages and the tools. There are also methodologies or frameworks that could potentially be considered to manage and plan the process. MOWAHS framework is one of them. MOWAHS framework helps in eliciting requirements for a mobile system, helps plan an architecture, how and when each functionality in a mobile system will be implemented [143]. When it comes to the authoring tools for mobile applications, the choice can be so varied and might need a careful selection on condition that a developer has a target platform, a user platform and technology in mind. The development goal can be native or cross platform. Mobile applications development can follow a web based rendition using HTML5, CSS, JavaScript. C++, Java are two other authoring languages one uses to develop mobile applications. During mobile applications development, it is common to use IDEs (integrated development environments), debuggers, testing modules (device emulators or simulators), and source control tools [98, 116].

Integrate

Integration stage is where the mobile application could be made to work with enterprise IT facilities, cloud infrastructure and native device capabilities. Integration will help it extend its functionalities with extra features that depend on another system. This extra facilities might be in-app billing, analytics tools, diagnostic tools, social networking facilities and storage [98, 154].

Build

Building is where application compiling, conversion, optimization of the application code into executable binaries or packages occur. A lot of cross platform applications optimization for individual platforms are accomplished in this stage [98, 116].

Publish

Publishing is putting apps on stores like Apple's app store and Google's Google Play. It is the process of making it accessible for download with or without fee for interested user [98, 115, 116].

Manage

Management facilities such as remote installation or removal, presenting users with the right version of application for their device, push messaging, data flow management, app performance data gathering are provided. Some examples of cross platform development tools that also provide management facilities are Appcelerator

Titanium, Worklight, Rhomobile [98, 116].

There are two major ways of developing applications for mobile devices. Those are native and cross platform development. In each development process different techniques are applied to achieve the same result whether one develops using one method or the other. Native features that are most commonly utilized to develop mobile applications are accelerometer, camera, compass, contacts, calendar, database, file, geolocation, media, network, notification services, and storage.

3.1.1 Native mobile applications development

Native application development involves developing software specifically designed for a specific platform, hardware, processor and its set of instructions . The advantages of native applications development are predictable performance, streamlined support, native UI, native API, coherent library updates. Native development requires platform specific SDK, programming language and it comes at a cost of developing different software for different platforms [44].

Native apps are executable binary files of an application that will be installed into the mobile device without the need for other abstraction layering to the operating system. They are able to call built in functionalities such as contacts, the dialer, and integrated emails directly. Despite the fact that native applications development requires platform specific skill and expertise, this strategy delivers a higher quality user experience than other mobile application development methods. Native apps are also best distributed through an app store [154].

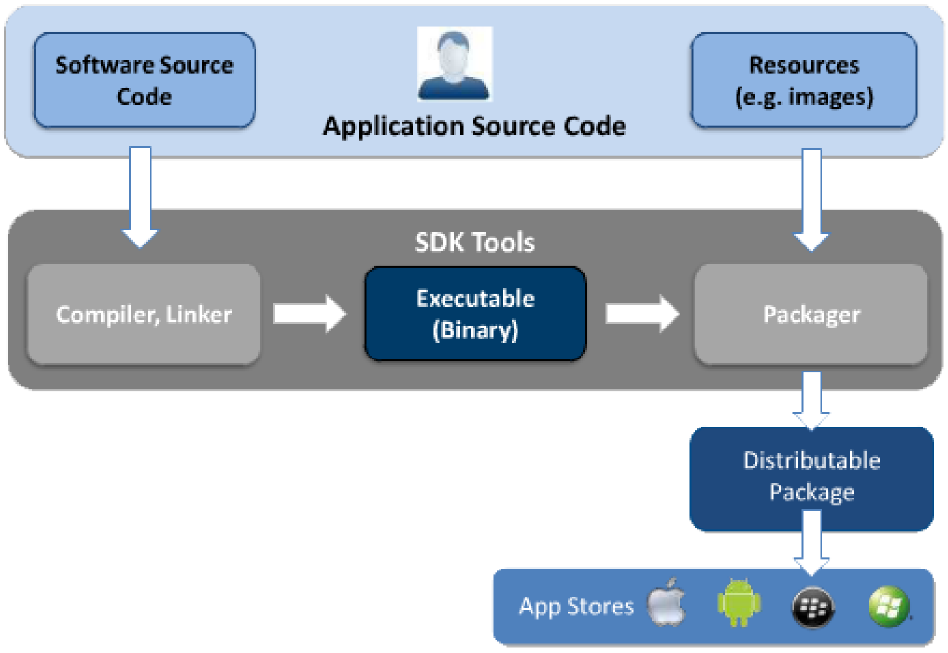


Figure 22: Native App Development [153]

3.1.2 Cross platform development

Sun Microsystems tried to bring the idea of write-once-run-anywhere to popularity in the 1990s. With the explosion of multiple devices like pagers, PDAs, set-to-boxes, DVD players with different manufacturers, Sun Microsystems came up with a Java 2 Micro Edition [69].

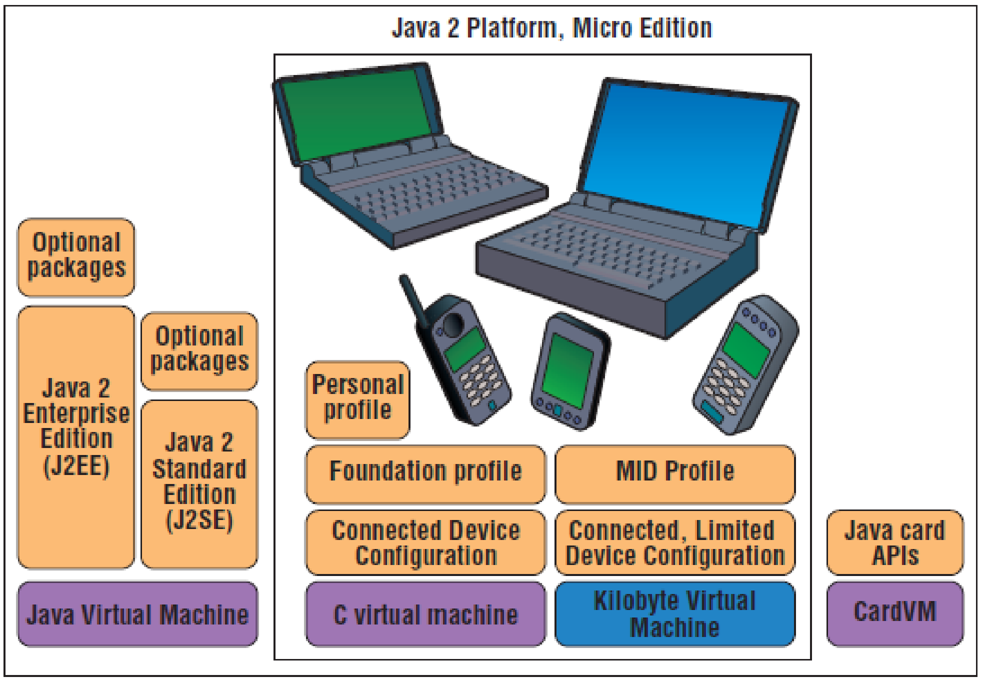

Java 2 Platform, Micro Edition
Optional packages
Java 2 Enterprise Edition (J2EE)
Optional packages
Java 2 Standard Edition (J2SE)
Java Virtual Machine
Personal profile
Foundation profile
MID Profile
Connected Device Configuration
Connected, Limited Device Configuration
C virtual machine
Kilobyte Virtual Machine
Java card APIs
CardVM

Figure 23: J2ME [69]

The plan in Java ME was to provide a common and shared platform layering environment onto which hardware manufacturers and application developers work on. Java ME promised a standardized application development environment for all sort of mobile devices at the time. Due to lack of standardized implementation of the idea, the idea itself became a problem to software developers and device manufacturers [21, 40]. To achieve cross platform application development, specification and implementation were split into separate entities. And Java ME tried to support multiple platform development using configurations and profiles but this did not do much to help developers from implementing applications using various versions to meet java specification(JSR) for multiple devices [60].

One thing that is common to all smartphones is understanding JavaScript which is what currently most cross platform tools are exploiting to develop multiplatform running mobile apps. The cross platform tools depend on HTML, CSS and JavaScript together with native accessing wrapper codes [9, 44].

3.2 The benefits of cross platform MAD

Cross platform development promises lower cost and time of development since it produces one code base to maintain and write targeting multiple devices and platforms. The figure below shows the trend for native to cross platform development cost and time factors.

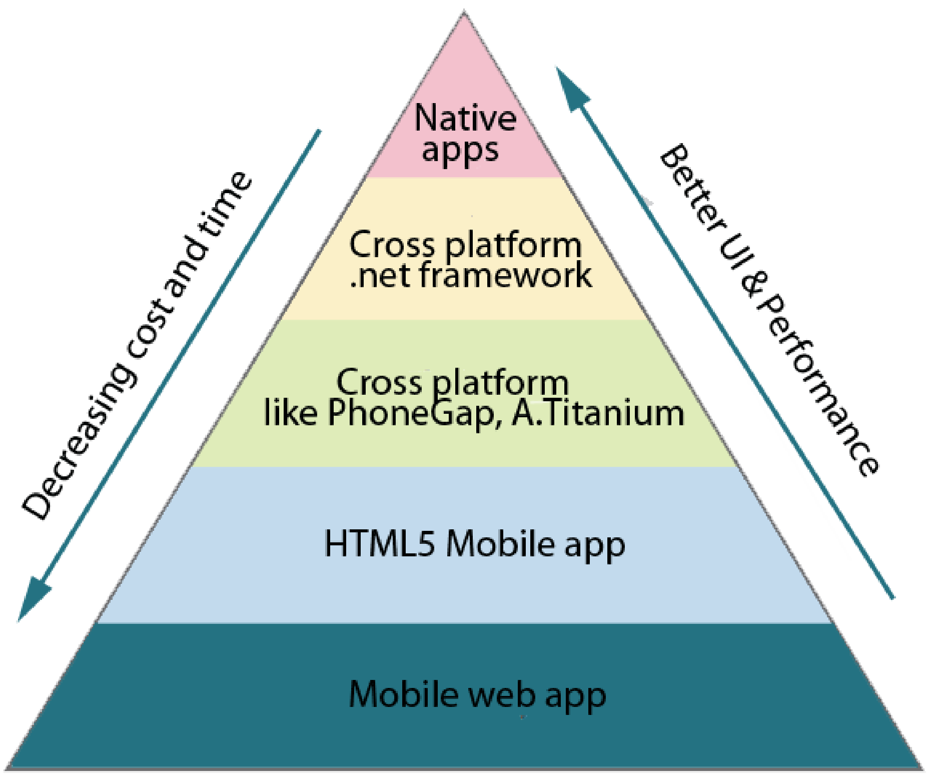


Figure 24: Native vs. Cross platform development [12]

[60] claims that cross platform development might inhibit future development in the field since it might be focused only on making it easy for developers than on innovation. Some noteworthy benefits of cross platform development are: [9, 60]

- Cross platform development is mainly based on using HTML5 technologies and developers in web site development might be able to leverage their experience
- Learning web technologies might be less challenging that learning an all new platform
- Devices come in all sorts of technology and feature that through cross platform development it might be possible to develop an application that works well on each one of them with less effort
- Cross platform mobile apps written using one SDK, operating System and development environment requiring less financial investment
- Cross platform apps try best to emulate natively written applications
- Cross platform apps might exploit well the native controls
- Cross platform development might help reduce cost in human resource, time resource, and development period
- Cross platform might help enable developers reach mass device through their hardware and platform abstraction features

- Democratization of mobile application development in which everyone will be able to develop without the constraints of proprietary and closed systems.

3.3 Cross platform system architecture

Developing an application that can run on multiple platforms such as iOS, Android, RIM, Symbian, WP7 and Bada or others might be a difficult task considering the number of technological implementations that comes with each platform. But efforts have been made to make cross platform running application a possibility. Some of the ways to implement cross platform mobile applications are outlined below.

3.3.1 Mobile Web Apps

These are standard web applications specifically designed to target mobile devices. But the application is built purely using HTML, CSS, JavaScript and other similar technologies that are used to build a website application for the desktop browsing experience. This type of mobile app runs entirely in the confinements of the mobile browsers. It accesses no native features of the mobile phone. These sorts of application are accessed by entering the addresses of the website on the mobile phone browsers address bar. Examples of such application are Wikipedia Mobile -http://en.m.wikipedia.org/, Facebook Mobile http://m.facebook.com, and BBC News mobile (http://www.bbc.co.uk/mobile/index.html) They are capable of running on any sort of device that comes with a browser [6].

3.3.2 Abstraction based cross platform system

The second form of developing a cross platform mobile application is using an abstraction model. These are a combination of both the web technologies and native API accessibility feature of a mobile device. These applications are written mostly in HTML/ CSS/ JavaScript with an architecture that abstracts device or platform variation for uniform device feature accessibility.

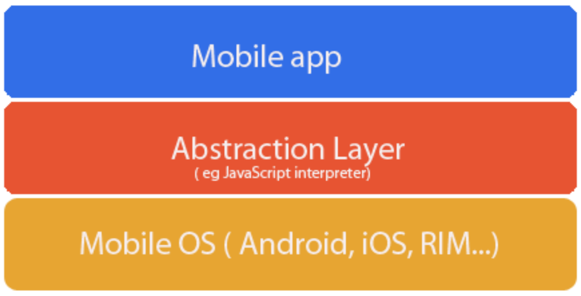

Mobile app
Abstraction Layer
(eg JavaScript interpreter)
Mobile OS (Android, iOS, RIM...)

Figure 25: Abstraction based cross platform overview

The software code is customized to run inside the specific mobile platforms WebView or JavaScript interpreter. But what makes hybrid mobile apps development different is that there is no need to start a browser to use the apps, there will be an icon which will start up the apps, and most of all the possibility of access to native features of the phone through APIs of the specific platform using JavaScript. WAC development falls under this category. The diagram below illustrates that the mobile application lies on top of an abstraction layer that negotiates native API calls to the operating system [98].

Although these forms of apps provide access to native features, full API accessibility and faster performance and some of the latest UI technologies support might not be guaranteed [53].

Abstraction model based cross platform applications are usually developed using tools and frameworks that help in developing applications with a platform independence goal such as web technologies [54].

Table 6: App development comparison [154]

	Device Access	Speed	Devt Cost	AppStore	Approval Process
Native	Full	Very fast	Expensive	Yes	Mandatory
Hybrid	Full	Native speed	Reasonable	Yes	Low overhead
Web	Partial	Fast	Reasonable	No	None

The mobile platforms' APIs give JavaScript access to the devices features and sensors and help it mimick the devices native interface. According to J. Dehlinger and J. Dixon, abstraction based cross platform—HTML5 tools, like PhoneGap, try to create a near native application for multiple platforms, but this does not allow for rich features that have access to the devices APIs and is a technological solution rather than a software engineering solution that allows reuse of engineering assets [82]. The table below will summarize what a mobile websites(mobile web apps) and abstraction based mobile apps are.

Table 7: Mobile web apps vs Mobile apps [6]

	Mobile website	Mobile app
Audience	Any mobile browser	needs special device
User experience	depends on bandwidth, technology, site capacity	robust user experience
Graphics & effects	limited by bandwidth, site performance	
Access to Hardware	limited	Unlimited (camera, GPS etc)
Ease of development	uses web technologies	native code, cross platform codes
Development resource	build once deployed everywhere	build for specific Mobile OS
Development cost	less expensive	more expensive
Ease & speed of implementation	published as a website	require app store
Distribution	internet	download and installation
Installation	No installation	download and installed
Updates & maintenance	easy	requires an app store
Search optimization	standard search	app store search
connectivity	required	can be used offline

3.3.3 Native compiling cross platform system

Native compiling cross platform mobile applications are applications that are built using a specifically designed tool and programming language and that the code base will be directly changed into the processors machine code—currently most are ARM based processors. In cross platform mobile applications, different types of technological approaches were deployed to achieve the desired portability. A framework in case of Rhomobile, runtime in case of WAC, JavaScript interpreter and other forms of layering between the operating system, the machine and the mobile applications. But in native compiling cross platform applications, the application code is directly changed to the right operating system and to the bare machine execution instructions. This in effect makes native compiling applications faster, fairly predictable, and have light system foot print [86, 95, 106, 146].

3.4 cross platform MAD tools

3.4.1 Appcelerator Titanium

Appcelerator Titanium is a cross platform mobile application development tool. It claims to use open web standards, architecture and follows open source initiatives. According to the Appcelerator Titanium's website, it provides 5000 device and mobile operating system APIs to create native like applications. Using the tool, a developer might be able to develop applications that will be able to run on tablet pcs, smartphones and desktops.

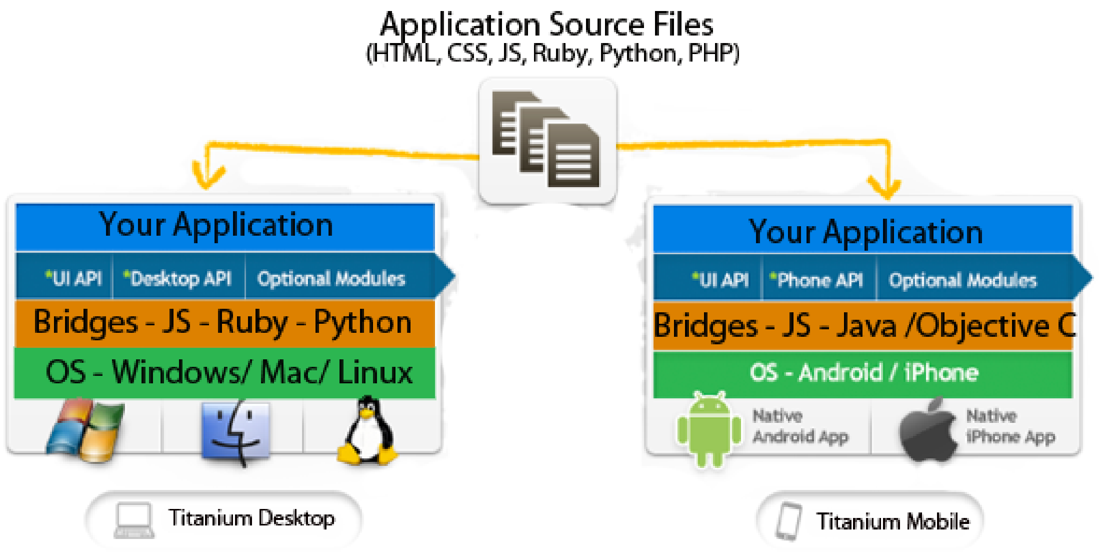


Figure 26: Appcelerator Titanium architecture [139]

Titanium Mobile SDK relies on JavaScript, HTML and CSS to enable the development of cross platform mobile apps. Titanium Mobile SKD webpage writes that 1.5 million developers are using it worldwide, more than 25000 apps have been developed with it and big brands like Reuters, eBay, Cisco are using the tool.

Titanium Mobile SDK comes with its own IDE. Titanium Studio is an eclipse like IDE which is used to write and test applications together with and Android Emulator. Appcelerator titanium supports database, media, geolocation, contacts, notification and many other native features of a smartphone.

Titanium enables web developers to create native mobile, desktop, and tablet applications using open web technologies such as JavaScript, HTML and CSS. One uses JavaScript APIs for UI(native), JavaScript for scripting [139].

Streaming an audio clip can take between 50- 500 lines of code of Objective-C or Java, in Appcelerator Titanium a developer can simply write one line of JavaScript to point to a URL location to start clip streaming [137]. Appcelerator Titanium has two components in its framework, UI API and Phone API. The former handles mappings to native UI from cross platform codes while the later deals with mappings to native feature of the phone such as database, file system and network.

Table 8: Appcelerator Titanium native support [139]

Appcelerator Titanium native API support			
	Android	iOS	BlackBerry
Accelerometer	✔	✔	✔
Analytics	✔	✔	✘
Barcode	✘	✘	✘
Calendar	✔	✘	✘
Camera	✔	✔	✔
Compass	✔	✔	✔
Contacts	✔	✔	✔
Database	✔	✔	✔
File	✔	✔	✔
Geolocation	✔	✔	✔
Map	✔	✔	✔
Media	✔	✔	✔
Network	✔	✔	✔
Notification	✔	✘	✘
SMS	✔	✔	✔

3.4.2 PhoneGap

PhoneGap is a cross platform mobile applications development tool. It uses HTML5, JavaScript and CSS3 web technologies to develop cross platform mobile applications that exploit native features of a mobile device [10, 17, 114].

PhoneGap supports the largest number of platforms of all other tools—iOS, Android, BlackBerry, webOS, WP7, Symbian and Bada. PhoneGap developed apps are hybrid apps that are neither pure web apps nor native apps. PhoneGap is an open source mobile apps development tool. With PhoneGap, one can develop an abstraction based cross platform applications using web technologies and wrap the code in native accessing system architecture of an application [108, 114].

PhoneGap uses jQuery JavaScript library in its development framework and made it easier to build jQuery Base mobile apps to native feature accessing applications. It supports accelerometer, camera, compass, contacts, file, geolocation, media, network, notification (alert, sound, vibration) and storage [27, 114].

Table 9: PhoneGap native support [114]

PhoneGap native support						
	Android	iOS	BlackBerry	WP7	Symbian	Bada
Accelerometer	✔	✔	✔	✔	✔	✔
Analytics	✘	✘	✘	✘	✘	✘
Barcode	✘	✘	✘	✘	✘	✘
Calendar	✘	✘	✘	✘	✘	✘
Camera	✔	✔	✔	✔	✔	✔
Compass	✔	✔	✘	✔	✘	✔
Contacts	✔	✔	✔	✔	✔	✔
Database	✔	✔	✔	✔	✔	✘
File	✔	✔	✔	✔	✘	✘
Geolocation	✔	✔	✔	✔	✔	✔
Map	✘	✘	✘	✘	✘	✘
Media	✔	✔	✘	✔	✘	✘
Network	✔	✔	✔	✔	✔	✔
Notification	✔	✘	✘	✔	✔	✔
SMS	✔	✔	✔	✔	✔	✔

Building apps in PhoneGap could involve authoring the application and then submitting the developed application to PhoneGap build service which will return a platform optimized application ready for distribution [118]. The build service of PhoneGap comes at a price from Adobe Systems. The diagram below illustrates PhoneGap's system of achieving cross platform.

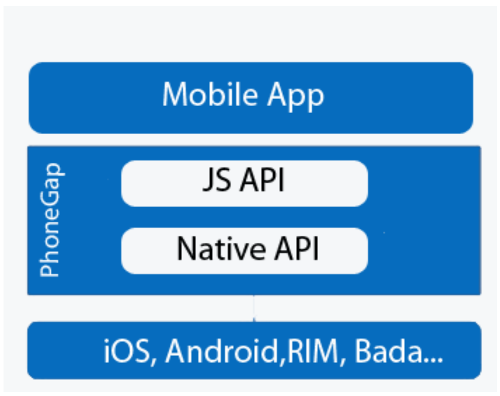


Figure 27: PhoneGap architecture [75]

3.4.3 Xamarin

Xamarin is a commercial cross platform mobile applications development tool. Xamarin enables to develop applications for iOS and Android using .NET framework and C# programming language. Application development using Xamarin comes in two forms as MonoTouch for iOS and Mono for Android [52, 156].

Mobile applications development using Xamarin is claimed to bring the benefit of sharing codes between platforms, using existing .NET expertise, easy access to native APIs, the niceties of rich IDE—Visual Studio in case of Android—and developing applications using strongly typed programming along with garbage collecting feature of .NET framework. Xamarin IDE comes as MonoDevelop for android and as a Visual Studio plug-in component for windows and Mac OSX and as monotouch for Mac OSX only.

Xamarin follows a native compiling cross platform system. The codes written for both iOS and Android are mapped directly to native codes. Xamarin claims to have 1:1 mapping for 4000 C# and 1700 C# classes for both Android and iOS respectively.

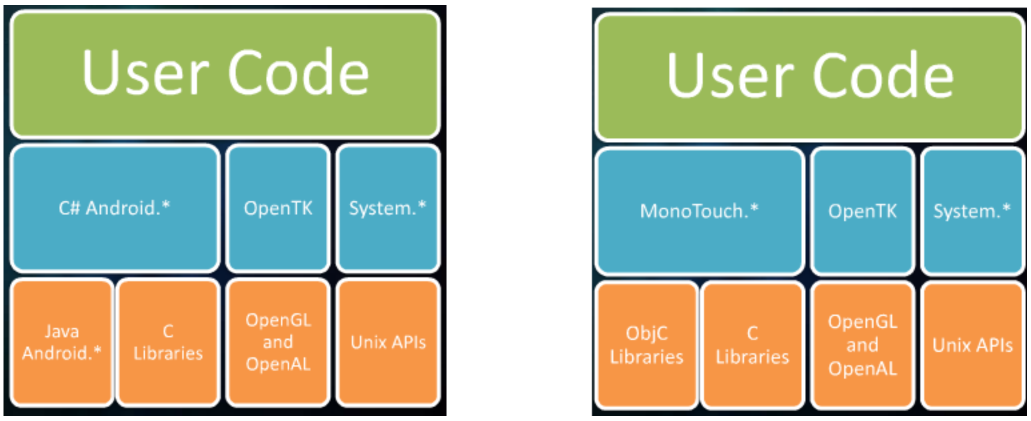


Figure 28: Xamarin architecture for MonoTouch and Mono for Android [155]

Xamarin uses the .NET Base Class Library (BCL) which is a collection of Microsofts .NET framework libraries and it uses LINQ—a feature of .NET framework that extends the power of C# to handle data source querying [106, 148].

According to Xamarin, mobile applications developed using its Mono for Android tool build 40% faster, have 70% smaller system footprints, and faster installation than Java due to small size of the applications [155].

Xamarin is being adopted by over 600 developers everyday and 3M, AT&T, Monster jobs, HP, Cisco and Microsoft are some of its users [155]. The table below illustrates native features Xamarin supports.

Table 10: Xamarin native support [156]

Xamarin native support		
	Android	iOS
Accelerometer	✔	✔
Analytics	✘	✘
Barcode	✘	✘
Calendar	✔	✔
Camera	✔	✔
Compass	✘	✘
Contacts	✔	✔
Database	✔	✔
File	✔	✔
Geolocation	✔	✔
Map	✔	✔
Media	✔	✔
Network	✔	✔
Notification	✔	✔
SMS	✔	✘

3.4.4 Rhombile

Rhomobile is a Motorola Solutions owned company that brought a cross platform mobile applications development tool that relies on Model-View-Controller(MVC) system architecture of programming applications using HTML, CSS and JavaScript and Ruby. Rhomobile supports iOS, RIM, Windows Mobile and Android.

Rhomobile comes with a Rhodes framework for building locally running web applications that are device optimized and be able to work with transactional enterprise applications such as Oracle CRM on demand, Microsoft Dynamics CRM, and SalesForce.

The Rhomobile development tool comes with three integrated tools. Those are Rhodes, RhoConnect, RhoHub and RhoGallery. Rhodes is an open source framework which is used to build native applications using HTML, CSS, JavaScript and Ruby. It supports GPS, calendar, camera, push, barcode, Bluetooth and near field communication (NFC).

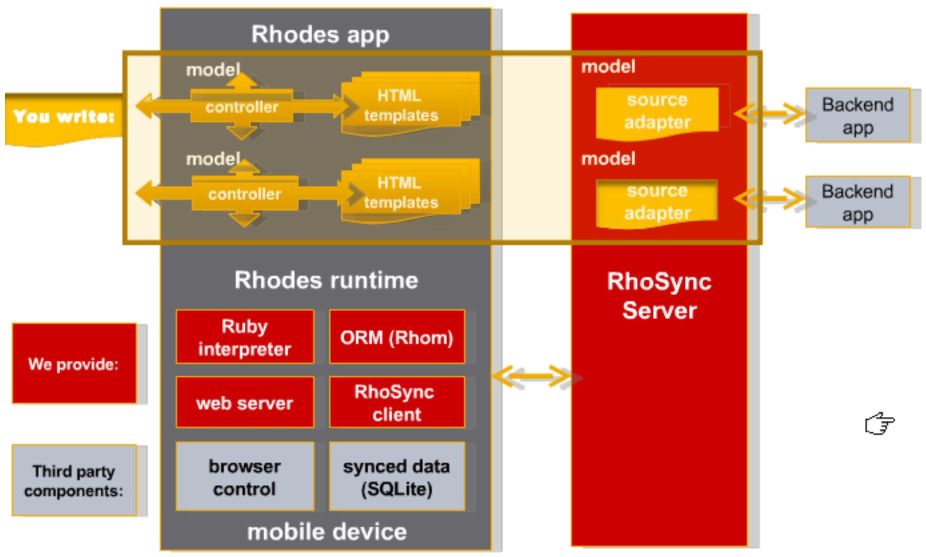


Figure 29: Rhodes application architecture [75]

RhoConnect provides a connection path to a backend enterprise data source. This will enable mobile apps to access data between local and backend systems using web services technologies of SOAP, REST, XML and JSON.

RhoHub enables to write applications online without the need for installing the RhoStudio SDK and provides git like source control for programmer collaboration and online build while RhoGallery is the equivalent of an app store.

Rhodes applications are composed of models, views and controllers. The views are HTML, CSS and JavaScript webpages executed by a WebView on the phone and a local lightweight server running on the phone and the controllers and the models are parts that are written using the programming language Ruby.

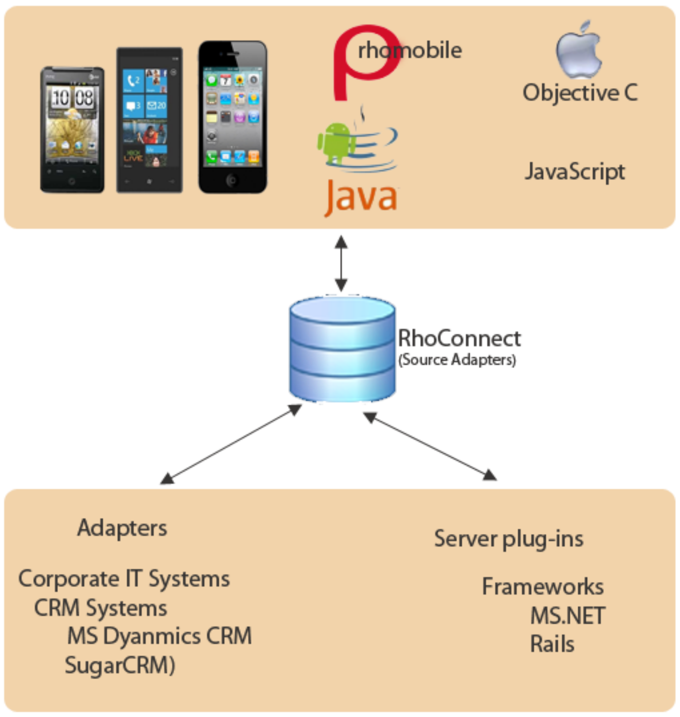


Figure 30: Rhoconnect architecture [120]

3.4.5 MoSync

MoSync is an open source cross platform mobile applications development tool. It enables one to develop native like cross platform mobile applications using C/C++, HTML5 and JavaScript.

Using MoSync, a developer can develop an application using a single codebase but target multiple platforms. MoSync supports iOS, Android, RIM, JavaME, Symbian, and Windows Phone. Application development using MoSync takes place in three forms. One of them is using HTML5 and JavaScript referred to as WebUI applications, the other one is HTML5 and JavaScript code but accessing native applications while the third method is using C++, JavaScript and HTML5 which, MoSync claims, to be real native applications [75, 102].

MoSync produces MoSync intermediate language using a C++ compiler that will be put together with all the application sources and MoSync libraries into a pipe tool. A pipe is a custom C++ compiler which outputs MoSync intermediate language. This is fed into the pipe tool, along with the application resources and the MoSync libraries. The pipe tool builds the codes, analyzes, optimizes and outputs either C/C++ source code, MoSync bytecode or Java bytecode. This is then packaged with the appropriate runtime for a specific platform [147].

Table 11: MoSync native support [145]

MoSync native support					
	Android	iOS	BlackBerry	WP7	Symbian
Accelerometer	✘	✘	✘	✘	✘
Analytics	✘	✘	✘	✘	✘
Barcode	✘	✘	✘	✘	✘
Calendar	✔	✔	✔	✔	✔
Camera	✔	✔	✔	✔	✔
Compass	✔	✔	✔	✔	✔
Contacts	✔	✔	✔	✔	✔
Database	✔	✔	✔	✔	✔
File	✔	✔	✔	✔	✘
Geolocation	✘	✔	✔	✔	✔
Map	✔	✔	✘	✘	✘
Media	✔	✔	✘	✔	✘
Network	✔	✔	✔	✔	✔
Notification	✔	✔	✘	✔	✔
SMS	✔	✘	✔	✔	✔

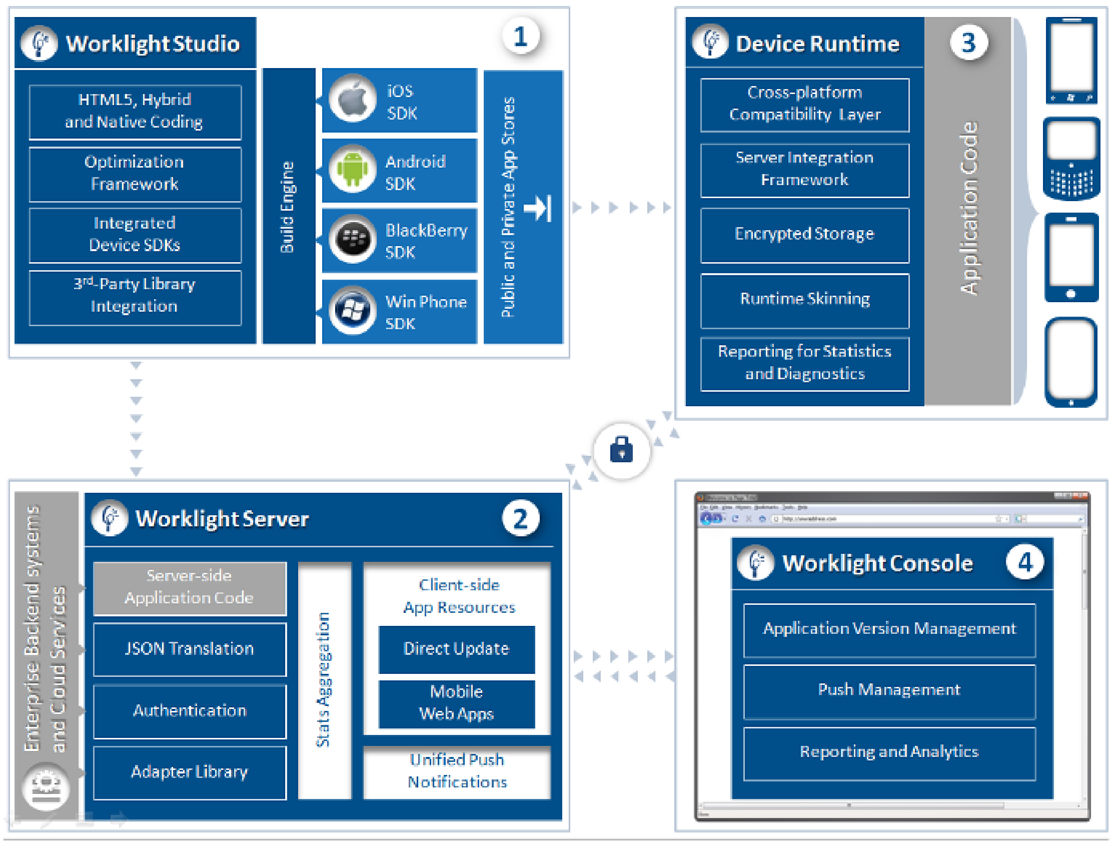
3.4.6 IBM Worklight

Worklight is a proprietary cross platform mobile applications development tool. It is used to develop mobile web apps or abstraction based mobile apps using HTML5, CSS3 and JavaScript. Like Rhomobile, the main focus of Worklight is enterprise applications with backend systems like servers and databases. Worklight SDK comes with four components as Worklight Studio, Worklight Device runtime, Worklight Server and Worklight Console [98, 154].

Worklight Studio is an Eclipse based IDE which consists of facilities to develop Worklight standard web or native applications and includes third party and backend connectivity facilities. Worklight uses PhoneGap components to wrap its JavaScript code for native device accessing capabilities. Worklight applications, usually targeting enterprise users and developers, are not published on app stores rather they are published on an internal server or private application hosts. Fig 31 illustrates the components of Worklight.

Worklight Device runtime component provides server integration framework, secure server connectivity, authentication, remote disable, notification and cross platform compatibility whereas Worklight Server provides direct access to backend transactional data access and Worklight Console handles application version management, analytics, and administrative dashboards [154].

Worklight supports iOS, Android, RIM and Windows Phone 7. Worklight together with PhoneGap's JavaScript-to-native device API interpreter provides an en-



Worklight Studio
1
HTML5, Hybrid and Native Coding
Optimization Framework
Integrated Device SDKs
3rd-Party Library Integration
Build Engine
iOS SDK
Android SDK
BlackBerry SDK
Win Phone SDK
Public and Private App Stores
Device Runtime
3
Cross-platform Compatibility Layer
Server Integration Framework
Encrypted Storage
Runtime Skinning
Reporting for Statistics and Diagnostics
Application Code
Enterprise Backend systems and Cloud Services
Worklight Server
2
Server-side Application Code
JSON Translation
Authentication
Adapter Library
Stats Aggregation
Client-side App Resources
Direct Update
Mobile Web Apps
Unified Push Notifications
Worklight Console
4
Application Version Management
Push Management
Reporting and Analytics

Figure 31: Worklight Architecture [154]

terprise applications development platform [98].

Accoring to Worklight's website, it has been used in large number of applications in healthcare, energy, financial, media and hospitality industries.

3.4.7 Corona

Corona is a cross platform mobile applications development tool that is used to author games and apps. Mobile applications development in Corona are written using the programming language Lua. The Lua code is compiled into an intermediate bytecode for a native runtime abstraction [97].

Corona supports only iOS and Android. Corona is a commercial development tool and the SDK comes with mobile app templates, API libraries, sample code, a debugger, a simulator but the SDK does not have a full IDE with GUI based tools. It uses a text editor like SDK [97, 98].

Corona follows the abstraction based cross platform approach. According to Corona's website, developers can build an application tasks such as animation with less code and ten times faster and with a rich multimedia experience [97].

3.4.8 Marmalade

Marmalade is a cross platform mobile applications development tool that follows either abstraction-based or native compiling style of cross platform mobile applications development. Using marmalade a developer can author an application using HTML5, CSS3 and JavaScript or a fully native compiling, yet portable cross platform applications using C++ [95]. Marmalade is most suitable to develop rich HTML5 apps and cross platform and high performance 2D and 3D games using C++.

Marmalade supports iOS, BlackBerry, Symbian, and Bada. Marmalade uses Microsoft Visual studio and Xcode as its integrated development environment (IDE). Marmalade is commercial tool which makes it difficult to know the details of its API support and documentations [146].

3.4.9 Adobe Air

Adobe Air is a runtime based cross platform mobile applications development tool. It uses HTML, JavaScript, ActionScript, Flex, Adobe Flash Professional, and Adobe Flash Builder for development of mobile applications that run on platforms and devices of Android, BlackBerry, iOS devices, and personal computers [5].

Adobe Air is a proprietary tool and it employs abstraction based cross platform technology. Adobe Air runtime is believed to be deployed on over a billion devices, which comes as second after JME. Adobe Air is most known for its technology to create creative and fancy looking user interfaces and for apps that require rich multimedia games and mobile apps. Adobe Air combines a number of technologies within it, most notably Adobe Flex and ActionScript to program applications, in addition to the drag-and-drop user interface designing strategy of Adobe [60, 98].

Adobe Air negotiates with the platform specific Adobe Air runtimes to access the native APIs of the phones. Through its runtime, Adobe Air accesses the network, database and other APIs.

Summary

In this document a number of cross platform mobile application tools have been investigated. Each tool was described based on modes of cross platform technology implementation such as just mobile web, abstraction based and native compiling cross platform. The technologies each tool depends on to develop applications such as web technology—HTML, CSS and JavaScript—or a general purpose programming language such as C++ or C# or Lua have been discussed. The tables below will summarize the main similarity, difference, weakness, strength, licensing and other key information about each tool.

Table 12: Cross platform tools market penetration [13, 97, 98, 114–116, 116, 120, 138, 145, 146, 153, 154, 156]

	Developer	SDK Downloads	Number of Apps	Sample Apps	License
Appcelerator Titanium	1600000	250000	35000	NBC iPad, MyTravel	Apache 2.0
PhoneGap	NA	600000	NA	Wikipedia, Netflix	MIT
Rhomobile	NA	100000	NA	SuperTrainerHQ	MIT
MoSync	20000	180000	NA	MoSync Reload	GPL/Commercial
Xamarin	NA	NA	NA	Monster Stack 2	Commercial
Worklight	NA	NA	NA	NA	Commercial
Corona	10000	100000	6000	Bubbleball	Commercial
Marmalade	50000	NA	NA	Plants vs Zombies, Backbreaker 2	Commercial
Adobe Air	3000000	NA	NA	eBay, BBC iPlayer	Commercial

Table 13: Cross-platfrom tool summarization [13, 97, 98, 114–116, 116, 120, 138, 145, 146, 153, 154, 156]

	Best for apps of	Strength	Weakness	Cross platform model	Programming Language
Appcelerator Titanium	Communications, Productivity, Travel	Free, MPSC, CDS, Rich documentation, native UI access, JSON	Supports only iOS and Android	AB	HTML, CSS, JS, PHP, Ruby, Python
PhoneGap	Travel	Free, MPSC, MPS	Runs confined in WebView	AB	HTML, CSS, JS
Rhomobile	Productivity	Free, MPSC, CDS	code interpreted, requires Ruby skill	AB	HTML, Ruby
MoSync	Communications, Games	Produces native code	-	AB/ NC	C/C++, HTML, JS
Xamarin	Communications, Games, Multimedia, Productivity, Travel	Produces native code	Commercial and supports only iOS and Android	NC	C#
Worklight	Productivity	MPSC, CDS, MPS	-	AB	HTML, CSS, JS
Corona	Games, Multimedia	2D and 3D game	Limited to game apps	AB	Lua
Marmalade	Games, Multimedia	2D and 3D game	Limited to game apps	NC	C++
Adobe Air	Communications, Games, Multimedia, Productivity, Travel	Rich multimedia content distribution	large application size, requires flash player	AB	HTML, CSS, JS, ActionScript
KEY:	MPSC Multiple Platform with single code base				
	MPS Multiplatform support				
	CDS Cloud data syncing				
	AB Abstraction based cross platform system				
	NC Native Compiling cross platform system				

Table 14: Mobile platform support [95,97,102,106,114,120,134,138,154]

Cross platform development tools mobile platform support						
	iOS	Android	WP7	RIM	Symbian	Bada
Appcelerator Titanium	✔	✔	✘	✔	✘	✘
PhoneGap	✔	✔	✔	✔	✔	✔
Rhomobile	✔	✔	✔	✔	✘	✘
MoSync	✔	✔	✔	✔	✔	✘
Xamarin	✔	✔	✘	✘	✘	✘
Worklight	✔	✔	✔	✔	✘	✘
Corona	✔	✔	✘	✘	✘	✘
Marmalade	✔	✔	✘	✔	✘	✔
Adobe Air	✔	✔	✘	✔	✘	✘

3.5 Tools comparison framework

Cross platform mobile applications come in so many forms. Each tool might approach cross platform mobile applications development in so many ways. Some come as runtime implementations or interpreter abstraction layers and when it comes to licensing some are completely free and open source while some are proprietary and commercial. The level of native feature support of a mobile device varies widely that much. As seen in the previous sections, the level native support some tools provide is wider and some of the other ones also provide a limited support but might also provide a special feature that a particular developer might be keen to utilize.

For a company engaged in cross platform mobile applications development or for a developer in this business, choosing a single tool can be a tricky one. With so many choices, features and conditions to review before making any decision, landing one tool and going cross platform could prove to be a challenging experience. After having seen several tools, picking one is not as straight forward.

What this section does is to provide a developer or a developing company with a framework through which they can compare the tools using the following criteria so they could be able to make a sane decision. Every tool comes with

Level of Platform support

This will help a company identify a tool that will help it reach the maximum target audience through the number of platforms a tool supports. A company or a developer that wants to reach Android users should consider or rate a tool that has the best support for Android and similarly for others.

Level of device feature set support

Once a tool is selected or primed to be selected, the next evaluation will be the level of native feature the tool provides for the platform it supports and how good it supports them. Or considering some unique feature one tool provides others can't such as cloud data syncing, social network integration.

Programming language the tools uses

Learning a new programming language can make a developer less productive and as a result could potentially be a cost to a company sponsoring the developer or the development, plus learning a new language from the ground up might probably require a lot of effort and time. Developers must take into account the development language of the tools in their tool screening sessions. A company which has been in web development will find it easier to adopt tools like Appcelerator Titanium, Worklight or PhoneGap while a company with Java, C++ or C# development background might find it easier to adopt tools like MoSync, Xamarin, Marmalade.

Tool license

A developer or a developing company should take into account the cost associated with using the tools. Some are entirely free to use, some are free but require acknowledging the license agreement and some of them are commercial and require payment to use them or publish the applications. A developer should be able to review the options using the tool comparison framework.

In the framework proposed, developers must find ways of rating their needs by putting values of 1 and 0 or between 1 to 3, see section 3.5.1, for native feature support and platform support they are interested in and sum the total together. Selecting the highest scoring tool will more likely satisfy their requirements. The table below will be used to rate native support and platform support of each tool during a selection process.

Table 15: Tool comparison framework

Score	Native feature support					CPDT	Platforms						Score
	Geolocation	Camera	SMS	Contact	DB		Android	iOS	WP7	BlackBerry	Symbian	Bada	
						A.Titanium							
						PhoneGap							
						Rhomobile							
						MoSync							
						Xamarin							
						Worklight							
						Corona							
						Marmalade							
						Adobe Air							
KEY:													
CPDT	Cross platform development tool												

In the following table, developers could compare the cost of using each tool once they have prescreened one tool and they can also list the specific language each tool uses and rate the level of effort and taxiing effort to adopting one of the languages in their development stage. For instance, a developer can rate a programming language for the tool zero if they have already been using that language and if the development language of the tool is new to them they might rate it five which means it is going to require a lot of effort and time to start using it.

Table 16: License cost and developer effort

CDPT	Cost/ License	Programming Language					Score
		HTML5	Java	C++	Ruby	PHP	
Appcelerator Titanium							
PhoneGap							
Rhomobile							
MoSync							
Xamarin							
Worklight							
Corona							
Marmalade							
Adobe Air							

3.5.1 Example: How to use the framework

To help show as to how one uses the framework designed in this masters thesis, a small mock up company with mock needs is presented below. This could indicate how a developer should go about selecting a cross platform tool.

Assumption: A small company called Easter wants to put its online software to mobile devices. The company has been developing webpages for the last 8 years using web technologies of HTML, CSS, JavaScript and PHP. Its software engineers are all experts in web authoring technologies. Now they plan to reach a minimum audience of those on Android and iOS in a minimum cost. Its mobile applications must use the SMS and Geolocation feature of mobile devices. The company is interested in a cross platform tool that is used for productivity or travel apps. This is how the company selects a certain cross platform mobile applications development tool.

Rating API needs

- 3 — Very important {SMS, Geolocation}
- 2 — Important {Contacts}
- 1 — Nice to have {File}
- 0 — Doesn't matter or not supported {Media, Compass, Barcode, DB}

Platform Support

- 3 — Very important {Android, iOS}
- 2 — Important {BlackBerry}
- 1 — Nice to have {WP7}
- 0 — Doesn't matter or not supported {Symbian, Bada}

The next step will be to estimate the cost and effort associated with selecting a tool.

Table 17: API and Platform coverage

Score	Native feature support					CPDT	Platforms						Score
	Geoloc	Camera	SMS	Contact	DB		Android	iOS	WP7	BlackBerry	Symbian	Bada	
8	3	0	3	2	0	A.Titanium	3	3	0	0	0	0	6
8	3	0	3	2	0	PhoneGap	3	3	1	2	0	0	9
*	*	*	*	*	*	Rhomobile	3	3	1	2	0	0	9
5	0	0	3	2	0	MoSync	3	3	1	2	0	0	9
8	3	0	3	2	0	Xamarin	3	3	0	0	0	0	6
*	*	*	*	*	*	Worklight	3	3	1	2	0	0	9
*	*	*	*	*	*	Corona	3	3	0	0	0	0	6
*	*	*	*	*	*	Marmalade	3	3	0	2	0	0	8
KEY:													
CPDT Cross platform development tool													
* no available information													

Rating cost

- 3 — New technology (learning can be costly in terms of time, effort and technology)
- 2 — Easy to re-adjust existing resource (Human or technology)
- 1 — small cost
- 0 — Free or no cost

Table 18: Technology and cost rating

CDPT	Cost		Technology				Score
	Organizational	License	JavaScript	HTML5	CSS	PHP	
A. Titanium		0	0	0	0	0	0
PhoneGap		0	0	0	0	0	0
Rhomobile		2	0	0	0	0	2
MoSync		2	0	0	0	*	2
Xamarin		3	*	*	*	*	3
Worklight		3	0	0	0	*	3
Corona		3	*	*	*	*	3
Marmalade		3	0	0	0	*	3
Key							
* not supported							
Organizational cost might vary from human resource to technology cost							

From table 17 and table 18, developers can see that in API coverage Appcelerator Titanium, PhoneGap, MoSync and Xamarin had an equal score but when it comes to platform coverage PhoneGap, Rhomobile, MoSync and Worklight seemed to provide the widest coverage of platforms. But our company needs to consider other factors such as cost, license, novelty of technology for its operations. In technology, since the company Easter had been involved in web technologies, it saw that adopting cross platform tools that depended on web technologies might be less costly to adopt. PhoneGap is a purely web technology based cross platform mobile applications development tool and the cost and technology cost score for it is either zero or very small and it fulfills the needs of the company. It handles both the important API coverage and provides wide platform support. Using the above reasoning and the company Easters need, the tool selected has to be PhoneGap

3.6 Sample Applications

After having looked at several cross platform mobile applications development tools, it might be useful to select at least two of the tools and make an application as an experiment to show that the nature and make of the best tools we selected. In this simple experiment, the tools will be investigated to show how they implement the same functionality. Number of code lines, how a tool implements user interfaces, application size, compilation speed, how user controls such as buttons are constructed, the symmetry between simulation and actual device implementation of the applications—testing on simulation and running on actual devices should not cause misplacement of the user controls.

The sample application for this case study is a pizza cross platform mobile application. The applications provide facilities to users for selecting pizza type, selecting toppings type, filling delivery address and finally sending the order to a pizza shop. The applications will have four windows

Pizza selection

This is where a user selects a pizza bread. The pizza bread choices are Hand Made, Natural, Pan Crust, Stuffed Crust, Thin and Crispy Crust and they will be arranged in some sort of horizontally scrollable gallery.

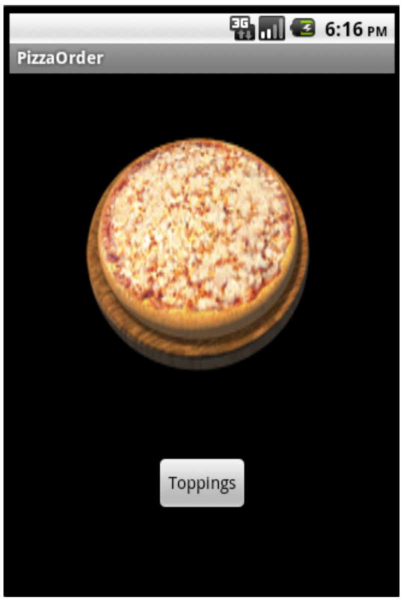


Figure 32: Pizza bread selection window

Toppings selection

After pizza selection, the application will transition to this window so users can select the toppings of their preference. The available choices for topping are Bacon, Beef, Italian Sausage and Grilled Chicken.

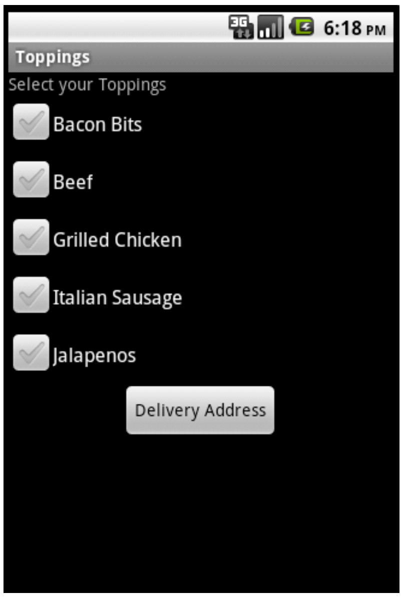
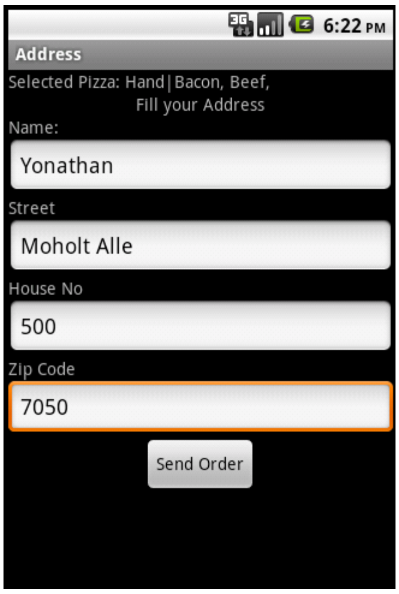


Figure 33: Toppings selection window

Address filling

Having selected the pizza bread and the toppings, a user can proceed and fill the delivery address. The address is composed of name of ordering user, street, house no, zip code or any other preferred addressing system. The addressing window has three textbox bars.

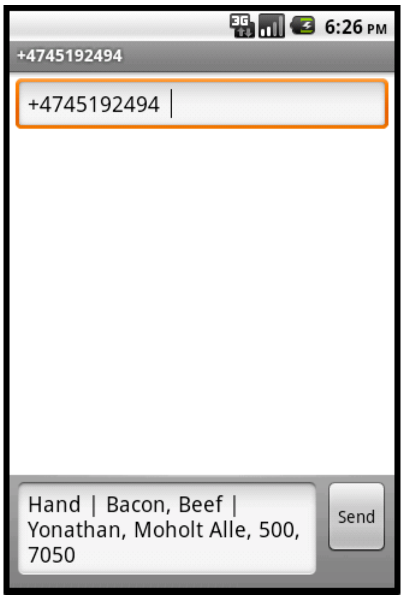


Selected Pizza: Hand|Bacon, Beef,
Fill your Address
Name: Yonathan
Street: Moholt Alle
House No: 500
Zip Code: 7050

Figure 34: Address filling window

Order submission

This is the last window in the application where a user gets to send the order via SMS to a pizza shop using the address embedded in the application.



Hand | Bacon, Beef | Yonathan, Moholt Alle, 500, 7050

Figure 35: Order submission window

Previous sections of this document have explained and documented the various features, the strength and weakness of several tools—Appcelerator Titanium, PhoneGap, Xamarin, Rhomobile, MoSync, Marmalade, Worklight, Adobe Air and Corona. Some of these tools are meant for general mobile application development while the other are specialized mobile applications development tools specifically optimized either for games or other forms of mobile application categories. Some are also open source while other are proprietary and commercial.

For this sample application, Appcelerator Titanium and Xamarin were selected based on the following criteria:

- These two tools were designed for general mobile application development purposes
- Appcelerator Titanium is open source and Xamarin is commercial. This is to sample how open source and commercial tools deal with cross platform mobile application development tasks.
- Appcelerator Titanium is an abstraction based cross platform implementation while Xamarin is native compiled based cross platform implementation. This provides a good opportunity to see how these two strategies work in a cross platform mobile application development processes
- Appcelerator Titanium uses web technologies such as HTML5, CCS, and JavaScript and might serve as representative of all other cross platform tools that use the same technology while Xamarin uses C# as its authoring tools. C# is a multipurpose core programming language from Microsoft.

- Both Appcelerator Titanium and Xamarin support many native features of a mobile phone such as SMS, Camera and others.
- They both support Android and iOS fully and this makes them equivalent candidates for comparison.

In the applications developed, certain features of both tools were explored. Those features were how both of these tools implemented user controls like buttons, text boxes, windows, and overall application size. The other observation that was taken into account was how many lines of code one needs to write to achieve the same feature in both tools. One native API feature, SMS, is explored in the applications and comparison were made how similar the codes of each tools are to native authoring codes, i.e. code syntax. Mobile applications once developed using a certain SDK and tested on an emulator might look perfect and when deployed on devices they might appear different, this is also one of the things that will be explored.

Appcelerator Titanium application

Appcelerator Titanium uses web technologies and applications developed using this tool depend on HTML5, CSS3 and JavaScript codes. In the sample pizza ordering sample application, all the codes are JavaScript. The windows, the scroll view, the text box, the checkbox and buttons in this application were constructed using JavaScript. While this can give a developer the freedom to construct everything from the ground up any user interface component, the reality is that one needs to write several lines of code to just implement the smallest of user interface component such as checkbox. And the placement of the user controls on an application window took a lot of time and required dividing the screen geometrically. One needs to consider the screen size of a device as to where and how to put user controls. When making applications using Appcelerator Titanium, a developer must take into account the portrait and landscape view of a screen, plus the width and height of the screen of an application. Constructing user interface components using Appcelerator Titanium requires one to use images in file formats .jpeg or .png which might possibly have contributed to higher application size, slower compilation and slower running experience. There might be a possibility also that some devices might not support certain image formats. Images of buttons, checkboxes, text boxes for user interface have to be authored in graphics editing software like Adobe Photoshop, this might require additional cost as license for extra supporting software. The code syntax similarity between Objective-C for iOS and Java for Android is found to be little in comparison to Appcelerator Titanium—almost no similarity. This dissimilarity might require a developer—with a background in either only Java or Objective-C—to learn JavaScript anew to work with Appcelerator Titanium. See Fig 36 and Fig 37.

Appcelerator has advantages, though, especially for a developer with a strong background in web development. Appcelerator is entirely dependent on web technologies except on a few cases that it uses PHP and other server side technologies. For an application where system foot print in processor demand, storage and memory is not a concern, Appcelerator seems to be perfect for general purpose applications like the Pizza ordering application. The look and feel of every component of the user interface of an application can be drawn in any form the developer wants to do it.

The SMS support provided by Appcelerator was found to be flawless. It brought the phones native SMS interface with the messages from the Pizza ordering application in the SMS message body and all the user has to do was to press the button send.

The Appcelerator Titanium Pizza ordering application was taken from [140] and customized to fit the context of this master's thesis while the Xamarin version was written from the ground up.

Xamarin Application

Xamarin uses C# as application authoring language. In the Pizza ordering application, Xamarin seemed to give the best result than Appcelerator Titanium. The application size was by far smaller than Titanium, the user controls appeared where they were meant to be either in a simulator or on an actual device. When deploying the Pizza ordering application to an actual device, everything looked the same and there were no need to tweak or adjust the positions and placements of the user controls in the application.

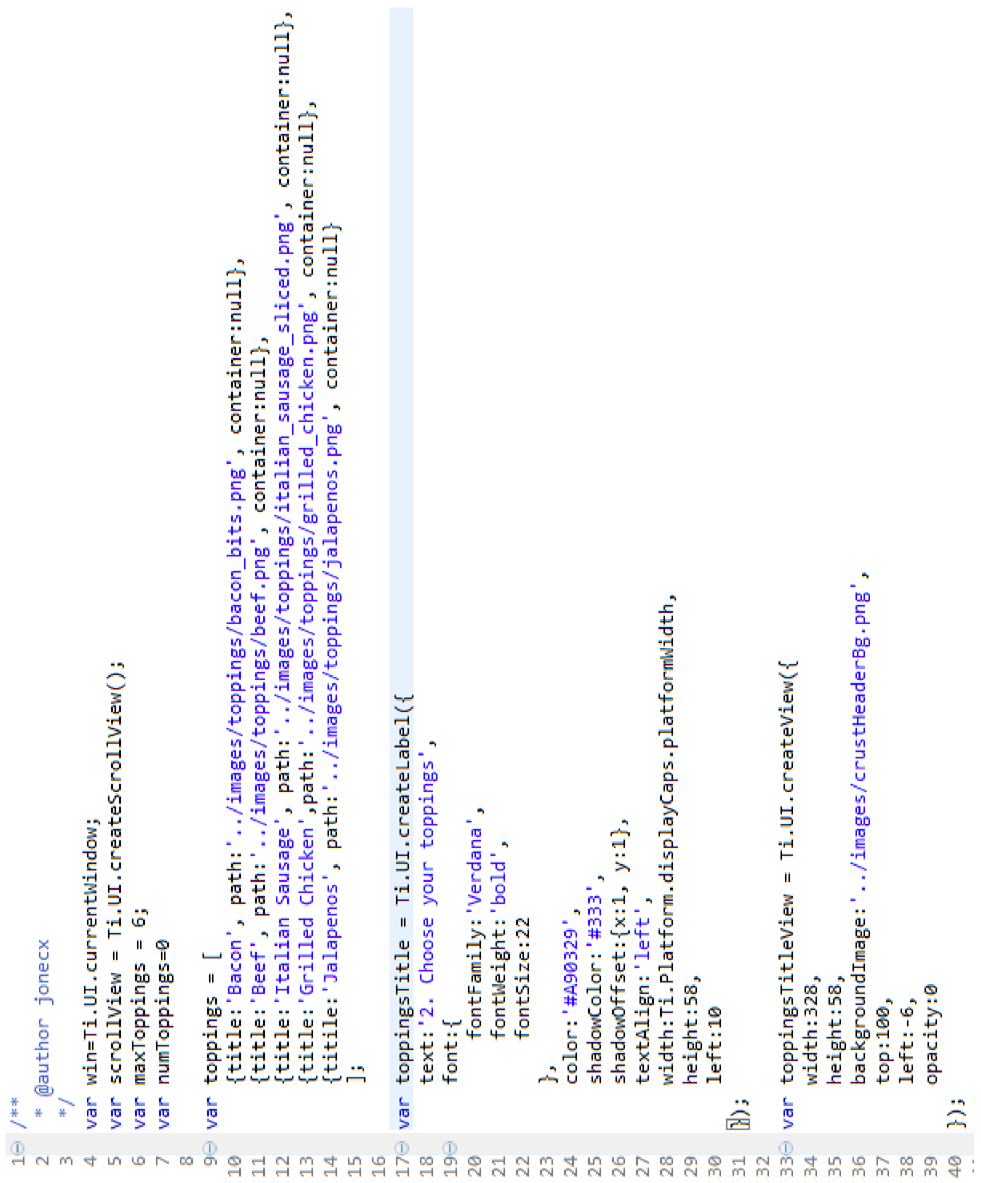
Xamarin also required significantly fewer code lines to implement user controls such as Textbox, checkbox, scrollview, windows and buttons than Appcelerator Titanium. Xamarin uses XAML (eXtensible Application Markup Language) [130]. XAML is a sort of XML syntax that is used by Android for UI implementation, by Silverlight for Windows Phone, and as Windows Presentation Foundation (WPF) for desktop and mobile applications. XAML is XML like structured code that is used to define user interfaces such as buttons, checkboxes, textboxes and other user controls. See Fig 38. Xamarin uses XAML to define user controls. Using XAML enables Xamarin developers to design a predictable, clean and standard user controls. Placing UI controls was easier in Xamarin than in Appcelerator Titanium.

Xamarin is used to make general purpose cross platform mobile applications. Xamarin also compiles faster and runs without any loading or updating interface elements as that was quite visible in Appcelerator Titanium. Xamarin comes with a lot of support and documentation. The code syntax of C# in Xamarin shared so many similarities with Adroid Java. Xamarin's intent declaration and handling is indistinguishable from Android's Java. See Fig 39. Developers who come from C++, Java and C# should find adopting Xamarin effortless. Defining classes, control flows, function definition, intent definition and others using C# appeared to be indistinguishable from native authoring tools such as Java for android. The Pizza Ordering application version written in Xamarin had generally a smaller size and runs faster and produced robust user interface.

The table below summarized the differences that were identified in the way the two tools were used to realize the Pizza ordering cross platform mobile application.

Table 19: Pizza ordering app development tool comparison

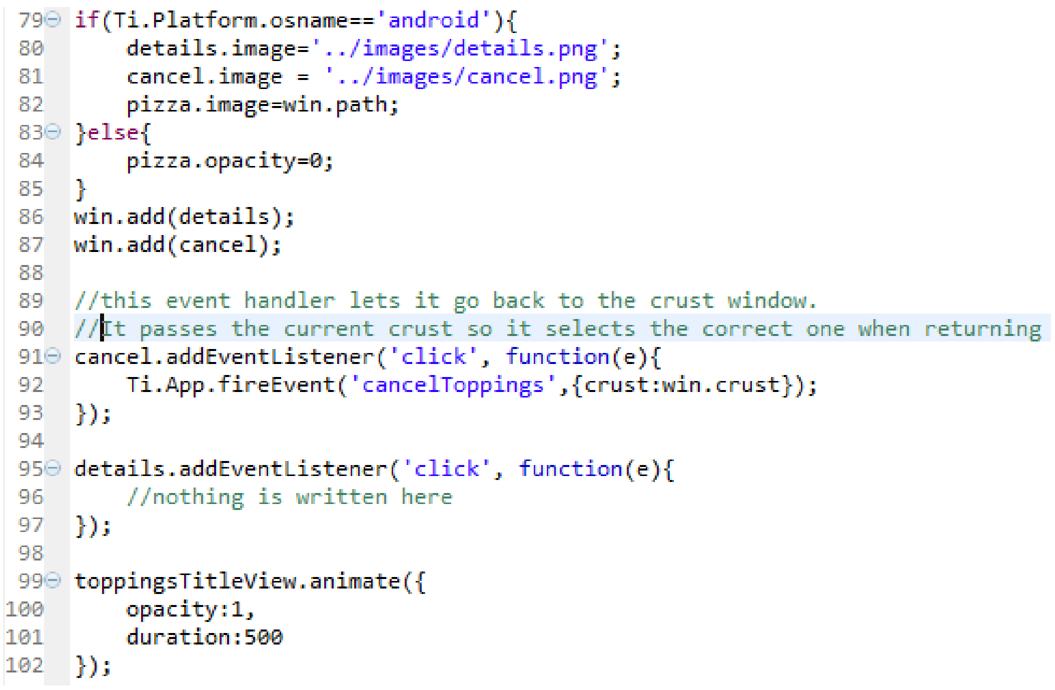
		Appcelerator Titanium	Xamarin
code lines	Window (toppings)	221	89
	Textbox	13	2
	Checkbox	6	2
	Button	7	2
	Scrollview	9	4
User control implementations	Window	JS, Titanium Code	XAML and C#
	Textbox	JavaScript, Titanium Code and Images	
	Checkbox		
	Button		
	Scrollview	JS, Titanium Code	
Perceived speed	Compilation	slower	faster
	Running	slower	faster
Code Type		JS, Titanium Code	C#
Code Syntax Similarity		JavaScript	Java, C++
SMS support		yes	yes
Application Size		22.8MB	2.28MB
Test result as expected		No	Yes



```
1  /**
2   * @author jonecx
3   */
4  var win=Ti.UI.currentWindow;
5  var scrollView = Ti.UI.createScrollView();
6  var maxToppings = 6;
7  var numToppings=0
8
9  var toppings = [
10     {title:'Bacon', path:'../images/toppings/bacon_bits.png', container:null},
11     {title:'Beef', path:'../images/toppings/beef.png', container:null},
12     {title:'Italian Sausage', path:'../images/toppings/italian_sausage_sliced.png', container:null},
13     {title:'Grilled Chicken',path:'../images/toppings/grilled_chicken.png', container:null},
14     {titile:'Jalapenos', path:'../images/toppings/jalapenos.png', container:null}
15     ];
16
17 var toppingsTitle = Ti.UI.createLabel({
18     text:'2. Choose your toppings',
19     font:{
20         fontFamily:'Verdana',
21         fontWeight:'bold',
22         fontSize:22
23     },
24     color:'#A90329',
25     shadowColor:'#333',
26     shadowOffset:{x:1, y:1},
27     textAlign:'left',
28     width:Ti.Platform.displayCaps.platformWidth,
29     height:58,
30     left:10
31 });
32
33 var toppingsTitleView = Ti.UI.createView({
34     width:328,
35     height:58,
36     backgroundImage:'../images/crustHeaderBg.png',
37     top:100,
38     left:-6,
39     opacity:0
40 });
```

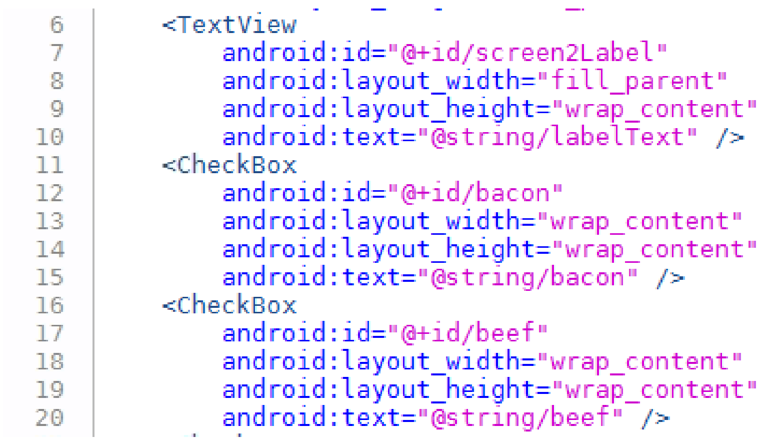
Figure 36: Appcelerator Titanium Code sample 1

In the figure above, line 4, 5, 17, and 33 are Titanium specific codes.



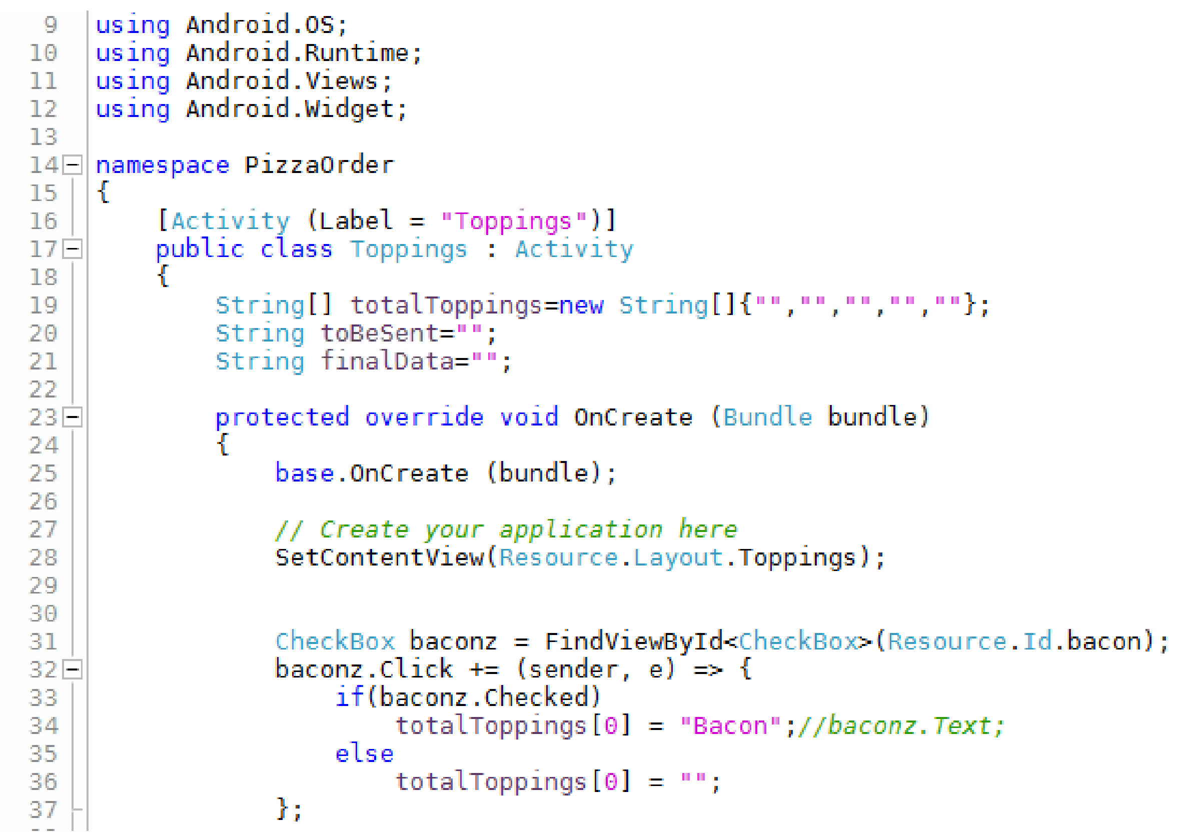
```
if(Ti.Platform.osname=='android'){
    details.image='../images/details.png';
    cancel.image = '../images/cancel.png';
    pizza.image=win.path;
}else{
    pizza.opacity=0;
}
win.add(details);
win.add(cancel);

//this event handler lets it go back to the crust window.
//It passes the current crust so it selects the correct one when returning
cancel.addEventListener('click', function(e){
    Ti.App.fireEvent('cancelToppings',{crust:win.crust});
});

details.addEventListener('click', function(e){
    //nothing is written here
});

toppingsTitleView.animate({
    opacity:1,
    duration:500
});
```

Figure 37: Appcelerator Titanium Code sample 2



```
<TextView
    android:id="@+id/screen2Label"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/labelText" />
<CheckBox
    android:id="@+id/bacon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/bacon" />
<CheckBox
    android:id="@+id/beef"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/beef" />
```

Figure 38: XAML sample code



```
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace PizzaOrder
{
    [Activity (Label = "Toppings")]
    public class Toppings : Activity
    {
        String[] totalToppings=new String[]{"","","","",""};
        String toBeSent="";
        String finalData="";

        protected override void OnCreate (Bundle bundle)
        {
            base.OnCreate (bundle);

            // Create your application here
            SetContentView(Resource.Layout.Toppings);


            CheckBox baconz = FindViewById<CheckBox>(Resource.Id.bacon);
            baconz.Click += (sender, e) => {
                if(baconz.Checked)
                    totalToppings[0] = "Bacon";//baconz.Text;
                else
                    totalToppings[0] = "";
            };
```

Figure 39: Xamarin sample code

4 Discussion and Evaluation

4.1 Main findings

Using literature study and experimental methods, it was tried to find out the benefits of using cross platform mobile applications development and formulate evaluation method for each tool so developers could assess cost-time-technology trade off associated with each tool and identify tools that best support cross platform mobile applications development in least cost and trouble for developers. The main findings are summarized in this section.

Mobile technology fragmentation

In line with the documentations in earlier sections, it is clear that the technologies of mobile computing and cross-platform technologies are fraught with several key technological features that could make decisive selection of tools as hard.

As can be seen from section 2, Mobile devices come from different device manufacturers with various mobile platform and technologies. The mobile device has a different set of hardware such as screen size, camera, connectivity, storage and processor. Clearly, the only way one might be able to accommodate myriad devices and platforms is to adopt cross platform development. Using cross platform, a developer will be able to write one code base and be able to deploy the applications, at least with some modification, on several platforms and mobile devices. Not only cross platform tools have to accommodate different device features, requirements and technologies but also has be able to meet the three classical mobile applications development challenges outlined in section 2.2. All this elements make selection of cross platform mobile applications development tools a difficult task.

Mobile platforms

Mobile platforms come in various forms some closed some open and completely free. Mobile platforms come also with their own app store which might be centralized or decentralized or which works only with specific devices to operate. See table 4. iOS, BlackBerry RIM and Windows Phone 7 are most known for being closed and proprietary while Android comes as an open technology with its own app store—Google Play. WAC follows a rather different approach compared to other platforms in that it needs a reliable device manufacturer to include the runtime. Although WAC has been supported by big telecom operators, it has not been able to make any effect on the market and it was decided to sideline it in the discussion. WAC seems to lack the right stamina to break into popularity partly because it does need a close collaboration and willingness of device manufacturers to install WAC runtime in their device. It is not only a tool but also a framework. See section 2.3

While every mobile computing industry player fighting ferociously for survival or dominance, expecting the industry to streamline—bringing forth standard devices to make developers and sponsors predicament more comprehensible—can be a bit of a naivety. Currently, Android and iOS make up major market share percentage. see Fig 6 and Fig 19 shows freemium revenue performance in the United States. For maximum target audience, considering Android and iOS coverage as a requirement might be advisable—leaving the detail to developers. Having said that, a developer aiming to target multiple platforms and market shares have to consider a cross platform development that at least covers Android and iOS.

Referring to table 1, we can see that there are several cross platform tools that a developer has to consider. A cross platform development tool that targets additional platform such as BlackBerry's RIM and Windows Phone 7 in addition to Android and iOS might come as a blessing depending on a developers needs and target audience. Undoubtedly, cross platform tools like Appcelerator Titanium, PhoneGap, Rhomobile, MoSync, and Worklight provide a considerable benefit in covering multiple platforms, device functionality features and in effect Resulting in a multi-platform telco-enriched app possibility.

Cross platform scenario

One of the findings was that in addition to multiple platform coverage and offering support for maximum number of mobile device feature functionality, cross platform development tools benefit developers in keeping one code base. A single code base based application development will reduce cost of development, maintenance, upgrade, support and management of application codes. This will help developers pool their resources to develop one code and be able to deploy it in as many platforms as possible. Fig 24 and section 3.2 show the benefits of adopting cross platform instead of native mobile applications development. However, native applications produce better user experience, run faster even if they are expensive to develop. See Fig 24

As can been seen from table 14 and table 13, there are several cross platform tools in the market and each one of them has their own unique features that makes them indispensable and at the same time lack some key functionality that a developer might really want to exploit. Deciding on one killer-solve-everything cross platform development tool seems unrealistic expectation. Developers must plan meticulously what they expect from a tool, must know in advance their target audience and platform aim, must have a clear idea of what they intend to exploit in the device features, and must decide how they want to implement using them. This provides developers with a draft tool selection guidelines.

From table 13, one can see that cross platform tools like Appcelerator Titanium, Worklight and all other tools that employ web authoring technologies such as JavaScript, HTML and CSS should really provide a clear advantage to a company or developer with a website development experience that was dependent on these technologies. Cross platform tools that use web authoring technologies might not have introduced a radically brand new technology, all they do is access device functionalities like SMS and Camera using JavaScript callable APIs that are provided with the cross platform tools. And cross platform tools like Corona, Marmalade, and Xamarin are tools that use programming languages of Lua, C/C++ and C#. Developers must consider cost of adopting new technology, new programming language, and the programming language that meets their needs best. Cross platform tools give a developer a greater access to the power of the mobile device to be able to write applications that go deep into the capabilities of the device than the usually superficial web based technology using tools like Appcelerator Titanium and PhoneGap.

As explained in section 3.1, mobile applications development has to go through five common steps—namely develop, integrate, build, publish and manage stages. How each tool provides these services varies widely. In addition to that cross platform mobile application development occurs as abstraction based, native compiling system architectures or as mobile web apps. The developer must be able to pick one of these architectures based on his or her needs.

Cross platform tools

According to the summarization in table 13, Web technology using cross platform tools are excellent for making communications, productivity, travel and utilities mobile applications. Great examples of these tools are Appcelerator Titanium and PhoneGap followed by Rhomobile, Adobe Air. Rhomobile requires Ruby skills, though and ActionScript is one of the requirements for Adobe Air. Being too reliant on Adobe Air might be a little short-sighted since it has garnered significant complaints for being too big, buggy and draining battery power. Section 3.3 shows that native compiling cross platform development architecture is considered to make the most out of the mobile hardware and run faster while abstraction based tools require some form of layering to access device feature sets of a mobile device. See table 19. Tools such as Corona, Xamarin, Marmalade and MoSync are great for developing applications that need to use the raw power of the mobile device for animation, games and processor intensive applications. These tools give mobile developers sheer power to make mobile devices obey their instructions. Above all that, Corona, Marmalade and Xamarin codes are converted into machine specific codes by-passing some sort of layering which in return makes them run faster. In our experiment in earlier section 3.6, the pizza application written using Xamarin visibly runs faster and seems to require less developer effort to bring about the desired interface and application functionalities.

There are also tools that are best for corporate applications that provide good integration to existing enterprise IT infrastructures with big database and business intelligence reports. Worklight and Rhomobile are two of the best tools for applications that depend on integration of existing data warehouse and syncing of business reports together with the possibility of extending them with third party libraries and software services. See section 3

Cost model

Section 3.4.9 sums up the time, level of effort, licensing and technology of the cross platform tools cost that add up to as a result of adopting one of the tools. Considering time to proper developing—taxiing to initialization of formal development—depends on how familiar are the developers to the technologies the tools might bring. For instance, Appcelerator Titanium, even though its syntax is dominated by JavaScript, it brings with it some unique lines of codes developers have to acquaint themselves with. Time and effort of users depends on the technology and the programming language the cross platform tools come with. Another key issue that could potentially have an effect on the cost of using the tools is licensing. See table 12. Appcelerator Titanium, PhoneGap, Rhomobile offer free or arrangements that have free components while Corona, Xamarin, Marmalade, Adobe Air are closed and commercial tools that require license fees to use for authoring or publishing. MoSync is in between free and commercial. However, cost of using tools might indirectly come from device and platform coverage of the tools. The more platform and device the tools support, the less the cost of cross platform development endeavor.

The trade off

Selecting the best cross platform development tool that provides the best cross platform support and cross platform development experience is a matter of trade off developers has to go through. **Appcelerator Titanium** supports two platforms—iOS and Android—fully, BlackBerry support is for a fee and Windows Phone 7 support is in the making. Appcelerator provides the most coverage of device feature functionality sets, integration with social networks such as Facebook, Twitter and has support for data syncing with bigger enterprise IT infrastructures. See section 3.4.1. Appcelerator is excellent for a great majority of mobile apps category with an exception of Games and animation intensive mobile apps. The level of documentation provided by the Appcelerator Company for Titanium is excellent. There are examples and code documentation for each API it provides plus a sample application that covers most of the important APIs of the tool. It has a huge user base and the fact that it depends on web technologies makes it a tool in touch with current technology trends, if not future proof.

Xamarin, as described in section 3.4.3, is a powerful cross platform development tool that produces native machine code for mobile apps. C# is an object oriented powerful programming language and a developer can bring the power of C# for the desktop to mobile devices. Besides, Xamarin can be used for any categories of mobile applications—including communications, games, multimedia, productivity, travel and utilities.

In the sample application section 3.6, it has been possible to find out that **Appcelerator Titanium** truly depends on web technologies of JavaScript and some syntax of its own to author applications. It is found to be representative of a class of cross platform mobile application development tools that depend on web technologies. Appcelerator Titanium is an abstraction based cross platform mobile applications development tool and can best represent PhoneGap, Rhomobile, Worklight, Adobe Air and partly MoSync and Marmalade. Anything that can be developed using PhoneGap, Rhomobile, Worklight, can be best done using Appcelerator Titanium. Even though MoSync and Marmalade are capable of and best suited for native compiling mobile applications development, Appcelerator Titanium can be used be used instead if the application did not require animation and processor demanding tasks in a game. In table 13, it is clear that nine of the tools evaluated only two of them use native compiling cross platform model. While MoSync uses both cross platform architecture models and native compiling architecture, and except Xamarin, Corona and marmalade, the rest of the tools depend on purely web technologies of JavaScript, CSS, HTML and some scripting languages such as PHP. Xamarin well represents native compiling cross platform MAD tools but one thing to note about Xamarin would be that it is a commercial product and has associated cost to use it. While native compiling cross platform mobile applications development tools such as Marmalade, MoSync and Corona are best for games and rich multimedia mobile applications, **Xamarin** beats them all in that Xamarin is best for developing any sort of mobile applications from productivity to communication to games and to multimedia applications.

Developers should look deep into their needs before selecting a cross platform tool. The tool comparison framework provided in section 3.4.9 will help a lot in finding out developer needs, technology, cost and other issues of cross platform development. Case by case subjective decision to select the best tools might result in better decision than a strict recommendation of criteria for a decision. This will help to incorporate recent developments in the ever changing cross platform technologies.

As one can see in the application zip attachment to this thesis and the comparison in table 19, Appcelerator titanium built applications had the biggest size and required a verbose code to produce a window with user controls while Xamarin managed to do exactly the same task with less code and better code organization possibilities. Compilation and running the same application was slower in case of Appcelerator Titanium than in Xamarin.

4.2 Validity of result

The research method that was applied was literature study based on gathering information from various sources—sources like the tools websites and scientific publications about the tools together with two simple sample experimental applications using Appcelerator Titanium and Xamarin. It has been possible to identify quite important points about each tool in how they cover mobile platforms, what sort of device functionality features they support and how they enable to author different categories of mobile applications. In the sample applications development using Appcelerator Titanium and Xamarin, it was possible to see that Xamarin produced excellent interfaces and that it was more flexible to work with Xamarin than Appcelerator Titanium. Writing codes in Xamarin was rather well organized. Developers can put codes in classes and objects in separate files. Working with Appcelerator Titanium is mostly in JavaScript and its code is more verbose than Xamarin. User control implementation in Xamarin is done using XAML. User controls are defined using XAML and are stored separately from the C# code while in Appcelerator everything is in the same file. See Table 19. Appcelerator Titanium built pizza ordering application had the bigger file size than Xamarin—10 times bigger. This is due to the fact that Appcelerator Titanium uses image files of .jpg, .png and others to generate the user controls. In comparison, Xamarin is a self-contained tool providing every facility to author mobile applications. With Appcelerator Titanium, a developer needs some other supporting tools such as Adobe Photoshop to have background coloring, button designing and the like.

Despite the fact that several key information have been collected, as one can see in the previous paragraph, the main point to note here is that all of the information about the tools—device functionality feature set, platform support, the technology the tools use is gathered from the tools' vendors websites and publication. Care should be taken when using this thesis. The ideal approach would have been to make a sample application using each one of these tools and be able to confirm what they wrote and how well they meet what the tools claim. Following a more practical experimental approach might help to refine the result further. Experimental approach will probably help in verifying the device coverage, device feature functionality coverage, third party library integration, cloud service integration and other claims vendors make.

Another point to note would be that studying cross platform tools is like a moving target. They keep changing so fast that it is difficult to have a clear cut opinion about each one of them at any given time. During the writing of this thesis, Appcelerator Titanium had three updates bringing in new functionalities, new supports and new syntax and APIs. While Xamarin had about three in the last 2 months and the syntax of the Pizza ordering application had to be changed. This requires developers to have a constant vigilance to stay abreast of the developments in cross platform tools. It is incredibly hard to imagine how cross platform development tools evolve in the next six to eight months.

5 Conclusion

The cross platform development technological ground is constantly shifting with each tool introducing new functionalities every time—making what holds three months from now fairly unpredictable. During the writing of this master's thesis, Appcelerator Titanium and Xamarin has gone through major changes each time each introducing changes in their syntax and functionalities. This makes drawing out a clearly defined selection framework a daunting task. The best recommendation for any developer is to really know the technological needs and to keep oneself up-to-date of the changes in the technology and considering each tool using the tool comparison framework in the thesis to make a decision on selection of a cross platform tools.

Cross platform development is an actively developing young technology. It could be sometime before the dust settles and a tools comes out as a winner. Considering each tool based on its merit for a mobile application need and target audience seems the only sensible option available right now.

But there are two system architecture of cross platform development—native compiling and abstraction based. Xamarin might be an ideal tool to start with native compiling while with abstraction based cross platform development, Appcelerator Titanium might be a good starting point.

References

[1] Aepona February 2010. Network as a service and mobile cloud computing http://www.aepona.com/wp-content/uploads/2010/10/Aepona-White-Paper-NaaS-MCC-Feb-2010.pdf, Mar 2012.

[2] SATNAC 2010. Mobile cloud computing: Embracing network as a service http://www.satnac.org.za/proceedings/2010/papers/progress/GutierrezMar 2012.

[3] Microsoft Developer Network Channel 9. The future of the web is at mix11 - video presentation, windows phone architecture: Deep dive, http://channel9.msdn.com/events/MIX/MIX11/DVC19, Mar 2012.

[4] Joyram Chakraborty A. Ant Ozok, Dana Benson and Anthony F. Norcio. A comparative study between tablet and laptop pcs: User satisfaction and preferences. *International Journal of Human-Computer Interaction*, 24(3):329–352, 2008.

[5] Adobe. Adobe air 3 http://www.adobe.com/products/air.html, April 2012.

[6] Adobe. Adobe air http://www.adobe.com/newsletters/inspire/february2012/articles/article7/, April 2012.

[7] I.K. Adusei, K. Kyamakya, and K. Jobmann. Mobile positioning technologies in cellular networks: an evaluation of their performance metrics. In *MILCOM 2002. Proceedings*, volume 2, pages 1239 – 1244 vol.2, oct. 2002.

[8] Dharma Prakash Agrawal and Qing-An Zehg. *Introduction to Wireless and Mobile Systems*. Cenage Learning, Connecticut, USA, 2011.

[9] Niyaz Noor Ali and Hussain Mansoor. Cross platform mobile application development framework, 2011. http://ieeepkhi.org/studentseminar/doc/Cross

[10] Sarah Allen, Vidal Graupera, Lee Lundrigan, Sarah Allen, Vidal Graupera, and Lee Lundrigan. Phonegap. In *Pro Smartphone Cross-Platform Development*, pages 131–152. Apress, 2010. 10.1007/978-1-4302-2869-1_8.

[11] Open Handset Alliance. Android overview, sept 2011 http://www.openhandsetalliance.com/android_overview.html, Mar 2012.

[12] AllianceTek. Comparison chart for mobile applications development, 2011 http://www.alliancetek.com/downloads/article/comparison-chart-for-mobile-app.pdf, April 2012.

[13] almcode. Comparison: App inventor, droiddraw, rhomobile, phonegap, appcelerator, webview, and aml http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml/, May 2012.

[14] Oliver Amft and Paul Lukowicz. From backpacks to smartphones: Past, present and future of wearable computers. *IEEE Perv Comput*, 8(3):8–13, July–September 2009. Wearable Computing Department.

[15] Android. Android https://developer.android.com/index.html, Mar 2012.

[16] Android. What is android? http://developer.android.com/guide/basics/what-is-android.html, Mar 2012.

[17] ANUBAVAM. Phonegap developer http://www.anubavam.com/phonegap-developer, April 2012.

[18] S.Z. Asif. *Next Generation Mobile Communications Ecosystem: Technology Management for Mobile Communications.* John Wiley & Sons, 2011.

[19] Bada. The basic architecture and ui comparison between bada and android http://developer.bada.com/documentation/docView.do?docID=D000001372&&searchCategoryID=C April 2012.

[20] Bada. What is bada http://bada.com/whatisbada/index.html, April 2012.

[21] Judith Bishop and Nigel Horspool. Cross-platform development: Software that lasts. *Computer*, 39(10):26–35, October 2006.

[22] BlackBerry. Blackberry enterprise solution architecture, http://us.blackberry.com/ataglance/solutions/architecture.jsp, Mar 2012.

[23] BlackBerry. Programming the blackberry with j2me, http://www.oracle.com/technetwork/java/index-139239.html#1, Mar 2012.

[24] Google Official Blog. 10 billion android market downloads and counting http://googleblog.blogspot.com/2011/12/10-billion-android-market-downloads-and.html, April 2012.

[25] S Blom, Matthias Book, Volker Gruhn, Ruslan Hrushchak, and Andr K. Write once, run anywhere a survey of mobile runtime environments. *2008 The 3rd International Conference on Grid and Pervasive Computing Workshops*, 0:132–137, 2008.

[26] Dan Bornstein. Dalvik virtual machine internals. Google I/O 2008, Juni 2008.

[27] Brad Broulik and Brad Broulik. Easy deployment with phonegap. In *Pro jQuery Mobile*, pages 227–247. Apress, 2011. 10.1007/978-1-4302-3967-3_10.

[28] Ed Burnette. *Hello, Android: Introducing Google's Mobile Development Platform.* Pragmatic Bookshelf, 2nd edition, 2009.

[29] Nina Kirstine Busk. Smartphone etiquette http://ninakirstineis.me/wp-content/uploads/BMMS-Nina.pdf, Feb 2012.

[30] R. Cameron. *Pro Windows Phone App Development.* Apress Series. Apress, 2011.

[31] Giovanni Camponovo and Yves Pigneur. *Business models analysis applied to mobile commerce.* 2003.

[32] A. Charlesworth. The ascent of smartphone. *Engineering Technology*, 4(3):32 –33, february 14 2009.

[33] Erika Chin, Adrienne Porter Felt, Kate Greenwood, and David Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 239–252, New York, NY, USA, 2011. ACM.

[34] M. Cinque, D. Cotroneo, Z. Kalbarczyk, and R.K. Iyer. How do mobile phones fail? a failure data analysis of symbian os smart phones. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 585 –594, june 2007.

[35] BBC Click. Will smartphones replace all other gadgets? http://news.bbc.co.uk/2/hi/programmes/click_online/9637184.stm, Mar 2012.

[36] cnet.com. Carriers try outflanking app stores with wac, http://reviews.cnet.com/8301-13970_7-20031936-78.html, Mar 2012.

[37] Wholesale Applications Communicty. Press and events,http://www.wacapps.net/press-releases, Mar 2012.

[38] Wholesale Application Community. Wac wiki, http://ext.wacapps.net/web/wac/wiki/-/wiki/Developer

[39] Wholesale Applications Community. The wac core specification defines the core requirements and base apis for a widget engine http://specs.wacapps.net/core/index.html#deviceapisobject-interface, Mar 2012.

[40] Andreas Constaninou. Mobile operating systems - the new generation - the competitive landscape of handset operating systems, interface frameworks and application execution environments, vision mobile 2006 http://www.visionmobile.com/rsc/researchreports/Mobile_Operating_Systems_The_New_Generation.p April 2012.

[41] J.P. Conti. The 10 greatest communications inventions. *Communications Engineer*, 5(1):14 –21, february-march 2007.

[42] Kevin Cording. A quick introduction to android, sept 2011 http://melpc.org/Downloads/Talks/A%20quick%20introduction%20to%20ANDROID.pdf, Mar 2012.

[43] ASUS Corporation. Asus eee pad transformer prime, paired and primed for perfection. http://eee.asus.com/en/eeepad/transformer-prime/features/, Mar 2012.

[44] D. Dern. Writing small [tools and toys]. *Spectrum, IEEE*, 47(6):14 –15, june 2010.

[45] Android Developers. Sqlite database http://developer.android.com/reference/android/database/sqlite/ April 2012.

[46] Bada developers. Architecture of bada http://developer.bada.com/library/help, April 2012.

[47] Bada developers. Overview of bada http://developer.bada.com/library/help, April 2012.

[48] Distimo. Full year 2011 http://www.distimo.com/publications/, April 2012.

[49] G.M. Djuknic and R.E. Richton. Geolocation and assisted gps. *Computer*, 34(2):123 –125, feb 2001.

[50] Carlos Duarte and Ana Paula Afonso. Developing once, deploying everywhere: A case study using jil. *Procedia Computer Science*, 5(0):641 – 644, 2011. The 2nd International Conference on Ambient Systems, Networks and Technologies $ANT - 2011$ / The 8th International Conference on Mobile Web Information Systems $MobiWIS2011$.

[51] David Ehringer. The dalvik virtual machine architecture http://davidehringer.com/software/android/The_Dalvik_Virtual_Machine.pdf, Mar 2012.

[52] eWeek. Xamarin delivers monotouch 5.2 for iphone and ipad app development http://www.eweek.com/c/a/Application-Development/Xamarin-Delivers-MonoTouch-52-for-iPhone-and-iPad-App-Development-366462/, April 2012.

[53] Mark Power Center for Educational Technology and JISC CETIS A Briefing Paper Interoperability Standards. Mobile web apps http://wiki.cetis.ac.uk/images/7/76/Mobile_Web_Apps.pdf, April 2012.

[54] Forbes. Mobile web app vs. native app? itś complicated http://www.forbes.com/sites/fredcavazza/2011/09/27/mobile-web-app-vs-native-app-its-complicated/, April 2012.

[55] George H. Forman and John Zahorjan. The challenges of mobile computing. *Computer*, 27:38–47, April 1994.

[56] Apache Software Foundation. Apache license version 2.0 hhttp://www.apache.org/licenses/LICENSE-2.0.html, April 2012.

[57] Symbian Foundation. General symbian os topics http://symbianresources.com/tutorials/general.phpmobileos, Mar 2012.

[58] Louis Galambos and Eric John Abrahamson. *Anytime, Anywhere: Entrepreneurship and the Creation of a Wireless World.* Cambridge University Press, New York, NY, USA, 2002.

[59] Monica J. Garfield. Acceptance of ubiquitous computing. *Information Systems Management*, 22(4):24–31, 2005.

[60] D. Gavalas and D. Economou. Development platforms for mobile applications: Status and trends. *Software, IEEE*, 28(1):77 –86, jan.-feb. 2011.

[61] Mark H. Goadrich and Michael P. Rogers. Smart smartphone development: ios versus android. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 607–612, New York, NY, USA, 2011. ACM.

[62] Robert Godwin-Jones. Emerging technologies: E-books and the tablet pc. *Language Learning Technology*, 7(1):4–8, January 2003.

[63] P. Golding. *Connected Services: A Guide to the Internet Technologies Shaping the Future of Mobile Services and Operators.* John Wiley & Sons, 2011.

[64] Vnia Gonalves, Pieter Ballon, and Vrije Universiteit Brussel. *Mobile operators and mobile web applications : the evolution from SaaS to PaaS models*, pages 1 – 14. 2009.

[65] Groupe Speciale Mobile $GSMA$. Mobie phone, Feb 2012.

[66] Abhishek Kumar Gupta. Challenges in mobile computing. In *Proceedings of the second national conference on Challenges and Opportuinities in Information Technologty*, PCOIT '02, pages 86–90, RIMT-IET, Mandi Gobindgarh, 2008.

[67] Sharon P. Hall and Eric Anderson. Operating systems for mobile computing. *J. Comput. Sci. Coll.*, 25(2):64–71, December 2009.

[68] Richard Harper. People versus information: The evolution of mobile technology. In *Proceedings of 5th International Symposium on Mobile HCI*, pages 1–18. Springer-Verlag, 2003.

[69] Sumi Helal. Pervasive java. *IEEE Pervasive Computing*, 1:82–85, 2002.

[70] Adrian Holzer and Jan Ondrus. Trends in mobile application development. In Cristian Hesselman, Carlo Giannelli, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin *Sherman* Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Mobile Wireless Middleware, Operating Systems, and Applications - Workshops*, volume 12 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 55–64. Springer Berlin Heidelberg, 2009. 10.1007/978-3-642-03569-2_6.

[71] Adrian Holzer and Jan Ondrus. Mobile application market: A developers perspective. *Telematics and Informatics*, 28(1):22 – 31, 2011.

[72] Adrian Holzer and Jan Ondrus. Mobile application market: A mobile network operators perspective. In Raj Sharman, H. Raghav Rao, T. S. Raghu, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, Clemens Szyperski, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski, editors, *Exploring the Grand Challenges for Next Generation E-Business*, volume 52 of *Lecture Notes in Business Information Processing*, pages 186–191. Springer Berlin Heidelberg, 2011. 10.1007/978-3-642-17449-0_19.

[73] Apple Inc. Identifying iphone models http://km.support.apple.com/library/APPLE/APPLECARE_AI
iphone_4-side_front_dimensions-001-en.png, Mar 2012.

[74] Apple Inc. ios developer program https://developer.apple.com/programs/ios/, Mar 2012.

[75] Manuel Palmieri Inderjeet Singh. Comparison of cross-platform mobile development tools http://www.idt.mdh.se/kurser/ct3340/ht11/MINICONFERENCE/FinalPapers/ircse11_submission_1
April 2012.

[76] Open Source Initiative. Mit licensing http://www.opensource.org/licenses/mit-license.php, April 2012.

[77] IDC International Data Corporation. Idc forecasts worldwide smartphone market to grow by nearly 502011, http://www.idc.com/getdoc.jsp?containerId=prUS22762811, Mar 2012.

[78] IntoMobile. Wac 2.0 specification announced; 8 operators already connected to the platform, http://www.intomobile.com/2011/02/16/wac-20-specification-operators/, Mar 2012.

[79] IntoMobile. Wac 2.0 specification announced; 8 operators already connected to the platform, http://www.intomobile.com/2011/02/16/wac-20-specification-operators/, Mar 2012.

[80] Apple: iOS Developer Library. ios technology overview, oct 2007 https://developer.apple.com/library/ios/#DOCUMENTATION/Miscellaneous/Conceptual/iPhoneOS
CH4-SW5, Mar 2012.

[81] ITU. The world in 2010 - facts and figures, Feb 2012.

[82] J. Dixon J. Dehlinger. Mobile application software engineering: Challenges and research directions http://www.mobileseworkshop.org/papers/7-Dehlinger_Dixon.pdf, April 2012.

[83] Richard L Kerns John E Andereson, Paul H Schwager. The drivers for acceptance of tablet pcs by faculty in a college of business. *Journal of Information Systems Education*, 17(4):429, Winter 2006.

[84] L. Kleinrock. Nomadicity: Anytime, anywhere in a disconnected world. *Mobile Networks and Applications invitedpaper*, 1(4):351–357, January 1996.

[85] A. Küpper. *Location-based services: fundamentals and operation.* John Wiley, 2005.

[86] Tribal Labs. Cross platform mobile app development http://webinos.org/crossplatformtools/, May 2012.

[87] J. Liberty and J. Blankenburg. *Migrating to Windows Phone.* Apress Series. Apress, 2011.

[88] Feida Lin and Weiguo Ye. Operating system battle in the ecosystem of smartphone industry. In *Information Engineering and Electronic Commerce, 2009. IEEC '09. International Symposium on*, pages 617 –621, may 2009.

[89] Johnny Li-Chang Lo, Judith Bishop, and Jan H. P. Eloff. Smssec: An end-to-end protocol for secure sms. *Computers & Security*, 27(5-6):154–167, 2008.

[90] Smartphones Magazine. Lg optimus 2x meet the first dual core smartphone in the world? http://smartphonespc.com/lg-optimus-2x-meet-the-first-dual-core-smartphone-in-the-world/, Mar 2012.

[91] Time Magazine. Invention of the year oct 2007 http://www.time.com/time/specials/2007/article/0,28804,1677329_1678542_1677891,00.html, Mar 2012.

[92] Time Magazine. A photographic history of the cell phone, Feb 2012.

[93] Qusay H. Mahmoud and Allan Dyer. Integrating blackberry wireless devices into computer programming and literacy courses. In *Proceedings of the 45th annual southeast regional conference*, ACM-SE 45, pages 495–500, New York, NY, USA, 2007. ACM.

[94] David Mark, Jeff LaMarche, and Jack Nutting. *Beginning iPhone 4 Development: Exploring the iOS SDK.* Apress, Berkely, CA, USA, 1st edition, 2011.

[95] Marmalade. Cross platform mobile applications and games development http://www.madewithmarmalade.com/marmalade, May 2012.

[96] Microsoft. Application platform overview for windows phone http://msdn.microsoft.com/en-us/library/ff402531$v = vs$.92.aspx, Mar 2012.

[97] Ansca Mobile. Corona http://www.anscamobile.com/corona/, April 2012.

[98] Vision Mobile. Cross-platform developer tools 2012 http://www.visionmobile.com/rsc/researchreports/VisionMobile_Cross-Platform_Developer_Tools_2012.pdf, April 2012.

[99] MobileIN. Short message service http://www.mobilein.com/sms.htm, April 2012.

[100] Mobilosoft. What is a smartphone? http://mobilosoft.com/blog/what-is-a-smartphone/, Feb 2012.

[101] B. Morris. *The Symbian OS architecture sourcebook: design and evolution of a mobile phone OS*. Symbian Press. J. Wiley & Sons, 2007.

[102] MoSync. Mosync home http://www.mosync.com/, April 2012.

[103] Motorola. Motorola atrix 4g http://www.motorola.com/Consumers/US-EN/Consumer-Product-and-Services/Mobile-Phones/Motorola-ATRIX-US-EN, Mar 2012.

[104] Collin Mulliner. Exploiting symbian, symbian exploitation and shellcode development, 25th chaos communication congress, berlin, germany. In *Fraunhofer-Institute for Secure Information Technology SIT, Darmstadt, Germany*, 2008.

[105] Microsoft Developer Network. Application platform overview for windows phone, http://msdn.microsoft.com/en-us/library/ff402531$v = vs$.92.aspx, Mar 2012.

[106] Microsoft Developer Network. Linq http://msdn.microsoft.com/en-us/library/bb397926.aspx, April 2012.

[107] Gartner Newsroom. Gartner says sales of mobile devices grew 5.6 percent in third quarter of 2011; smartphone sales increased 42 percent http://www.gartner.com/it/page.jsp?id=1848514, Mar 2012.

[108] Sigma Noblis. Bridging the mobile app gap http://www.noblis.org/NewsPublications/Publications/TechnicalPublications/SigmaJournal/Docume
Vol 11, Number 1, October 2011 Last accessed: April, 2012.

[109] Cambridge Dictionaries Online. History http://www.gsma.org/history/, Feb 2012.

[110] T. Ozkul and A. Al Homoud. Communication protocol for monitoring a large number of remotely distributed hazardous material detection devices. *Computer Standards amp; Interfaces*, 25(5):553–561, 2003.

[111] Nokia White paper. Nokia and symbian os, 2002. http://nds2.ir.nokia.com/NOKIA_COM_1/About_Nokia/Press/White_Papers/pdf_files/symbian_net.p
Mar 2012.

[112] PCWorld. Android market hits 450k apps, challengers abound http://www.pcworld.com/article/250765/android_market$_hits\_450k\_apps\_challengers\_abound.html$, A

[113] Microsoft Windows Phone. Windows phone: How-to, http://www.microsoft.com/windowsphone/en-us/howto/wp7/start/whats-new-in-windows-phone.aspx, Mar 2012.

[114] PhoneGap. How phonegap works http://phonegap.com/about, April 2012.

[115] Tribal: Medical Mobile Project. Cross platform mobile development http://www.mole-project.net/images/documents/deliverables/WP4_crossplatform_mobile_development_March2011.pdf, March 2011.

[116] Adam M. Christ Noblis publication. Mobile application: Bridging the mobile app gap http://www.noblis.org/NewsPublications/Publications/TechnicalPublications/SigmaJournal/Docume
March 2011.

[117] REDMONDPIE. Windows phone 7 versus android http://cdn.redmondpie.com/wp-content/uploads/2010/02/ Windows-Phone7vsAndroid2.jpg, Mar 2012.

[118] refulz web developerś blog. Build cross platform phone applications with phonegap http://php.refulz.com/build-cross-platform-phone-applications-with-phonegap-1/, April 2012.

[119] CNET Reviews. Report: Android app market growing faster than iphone apps http://reviews.cnet.com/8301-13970_7-20032228-78.html, April 2012.

[120] Rhomobile. The universal mobile application integration server http://www.rhomobile.com/products/rhoconnect/, April 2012.

[121] J. Sales. *Symbian OS Internals: Real-time Kernel Programming.* Symbian Press. John Wiley & Sons, 2006.

[122] Jane Sales. *Symbian OS internals, Real-Time Kernel Programming.* John Wiley and Sons, 2005.

[123] Jane Sales. *Building Tablet PC Applications.* O'Reilly Media, Inc., 2009.

[124] M. Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, PODC '96, pages 1–7, New York, NY, USA, 1996. ACM.

[125] M Satyanarayanan. Pervasive computing: vision and challenges. *Ieee Personal Communications*, 8(4):10–17, 2001.

[126] A.H. Sayed, A. Tarighat, and N. Khajehnouri. Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. *Signal Processing Magazine, IEEE*, 22(4):24 – 40, july 2005.

[127] J.H. Schiller. *Mobile communications.* Addison-Wesley, 2003.

[128] S. Schwiderski-Grosche and H. Knospe. Secure mobile commerce. *Electronics Communication Engineering Journal*, 14(5):228 – 238, oct 2002.

[129] Lookout Mobile Security. App genome report http://www.pcworld.com/article/250765/android_market$_h$*its_450k_apps_challengers_abound.html, Ap*

[130] Silverlight Show. Wp7 for iphone and android developers - introduction to xaml and silverlight http://www.silverlightshow.net/items/WP7-for-iPhone-and-Android-Developers-Introduction-to-Xaml-and-Silverlight.aspx, May 2012.

[131] A. Silberschatz, P.B. Galvin, and G. Gagne. *Operating system concepts.* Operating System Concepts. J. Wiley & Sons, 2005.

[132] SQLite. About sqlite http://www.sqlite.org/about.html, April 2012.

[133] K. Stanoevska-Slabeva and T. Wozniak. Opportunities and threats by mobile platforms: The *new* role of mobile network operators. In *Intelligence in Next Generation Networks ICIN, 2010 14th International Conference on*, pages 1 –6, oct. 2010.

[134] Adobe Systems. Adobe air 3 http://demand.assets.adobe.com/en/downloads/guides/6408.guide.Mobil April 2012.

[135] A.S. Tanenbaum. *Modern operating systems.* GOAL Series. Pearson Prentice Hall, 2008.

[136] WIRED DO TCOM. Htc's aria: Android phone that can be had for a song http://www.wired.com/images/productreviews/2010/06/pr_htc_aria_f.jpg, Mar 2012.

[137] Technoholik. Titanium: an app making tool for web developers http://technoholik.com/news/mobile/apps/titanium-an-app-making-tool-for-web-developers/80, April 2012.

[138] Appcelerator Titanium. Api documentation http://docs.appcelerator.com/titanium/2.0/index.html#/*api*, *April*2012.

[139] Appcelerator Titanium. Getting started with titanium studio https://wiki.appcelerator.org/display/tis/Getting+Started+with+Titanium+Studio, April 2012.

[140] Mobile Tuts+. Introduction to cross platform develeopment with appcelerator http://www.adobe.com/newsletters/inspire/february2012/articles/article7/, April 2012.

[141] Virpi Kristiina Tuunainen, Tuure Tuunanen, and Jouni Piispanen. Mobile service platforms: Comparing nokia ovi and apple app store with the iisin model. *Mobile Business, International Conference on*, 0:74–83, 2011.

[142] W3C. Geolocation api specification http://dev.webinos.org/deliverables/wp3/Deliverable32/wiki$t3-2$API_investigations.html, April 2012.

[143] Alf Wang, Carl-Fredrik Srensen, Heri Ramampiaro, Hien Le, Reidar Conradi, and Mads Nygrd. Using the mowahs characterisation framework for development of mobile work applications. In Frank Bomarius and Seija Komi-Sirvi, editors, *Product Focused Software Process Improvement*, volume 3547 of *Lecture Notes in Computer Science*, pages 111–127. Springer Berlin / Heidelberg, 2005. 10.1007/11497455$_1$2.

[144] Bekkering E. Schimdt M. Johnston A. Warkentin, M. Proposed study of end-user perceptions regarding tablet pcs. In *Proceedings of the Information Resources Management Conference*, pages 430–431. Idea Group Publishing, 2004.

[145] Webinos. Barcode api http://www.mosync.com/, April 2012.

[146] Webinos. Cross platform development tools http://webinos.org/crossplatformtools/, May 2012.

[147] webinos. Mosync http://webinos.org/crossplatformtools/mosync/, April 2012.

[148] webinos. Xamarin — monotouch and mono for android http://webinos.org/crossplatformtools/xamarin-monotouch-and-mono-for-android/, April 2012.

[149] Andrew Wheen and Andrew Wheen. The mobile revolution. In *Dot-Dash to Dot.Com*, Springer Praxis Books, pages 163–173. Springer New York, 2011. 10.1007/978-1-4419-6760-2_12.

[150] Wikipedia. Mobile application development, 2011 http://en.wikipedia.org/wiki/Mobile_application_development, April 2012.

[151] Wikipedia. Mobile operating systems http://en.wikipedia.org/wiki/Mobile_operating_system, Mar 2012.

[152] WMPoweruser. Gartner predicts windows phone 7 will overtake blackberry in 2013, overtake iphone in 2015, http://wmpoweruser.com/gartner-predicts-windows-phone-7-will-overtake-blackberry-in-2013-overtake-iphone-in-2015/, Mar 2012.

[153] Worklight. Native, web or hybrid mobile app development http://www.slideshare.net/WorkLightInc/native-web-or-hybrid-mobile-app-development-webinar, April 2012.

[154] Worklight. Products overview http://www.worklight.com/product/overview, April 2012.

[155] Xamarin. Mono for android 4.0 introduction http://www.slideshare.net/Xamarin/mono-for-android-4-0-introduction, April 2012.

[156] Xamarin. Xamarin documentation http://www.xamarin.com, April 2012.

[157] Jonathan A. Zdziarski. *Iphone open application development.* O'Reilly, first edition, 2008.

[158] Pospischil Gnther Zeiss Joachim, Davis Marcin. The role of wac in the mobile apps ecosystem, ftw, telecommunications research center, vienna, austria http://cdn.intechopen.com/pdfs/24902/InTech-The_role_of_wac_in_the_mobile_apps_ecosystem.pdf, Mar 2012.

[159] The Amazing Zones. Nokia lumia 800 windows mango smartphone review, http://www.theamazingzones.com/wp-content/uploads/2012/02/nokia-lumia-800-review.jpg, Mar 2012.

[160] Dino A Dai Zovi. *Apple iOS 4 Security Evaluation.* 2010.

Appendices

Acronyms

MRC Maximal Ratio Combining

API Application Programming interface

BAP BlackBerry Alliance Program

BCS BlackBerry Connect Software

BMDS BlackBerry Mobile Data System

BS BlackBerry Smartphone

BCL Base Class Library

BS Base Station

BSS BlackBerry Solution Services

CDMA Code Division Multiple Access

CPMAD Cross Platform Mobile Applications Development

CPU Central Processing Unit

CRM Customer Relations Management

DVM Dalvik Virtual Machine

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile

GUI Graphical User Interface

HD High Definition

HDMI Hight Definition Multimedia Interface

IDE Integrated Development Environment

IE9 Internet Explorer 9

IP Internet Protocol

JIL Joint Innovation Lab

JME Java Mobile Edition

JS JavaScript

JSR Java Specification Request

LCD Liquid Crystal Display

LMU Location Measurement Unit

MAD Mobile Applications Development

MOWAHS Mobile Work Access Heterogeneous Systems

MP Megapixel

MS Mobile Station

MVC	Model-View-Controller
NaaS	Network as a Service
OEM	Original Equipment Manufacturer
OS	Operating System
OTD	Observed Time Difference
PDA	Personal Digital Assistant
QVGA	Quarter Video Graphics Adapter
RAM	Random Access Memory
RIM	Research In Motion
RIT	Radio Interface Timing
SDK	Software Development Kit
SDLC	Software Development Life Cycle
SMS	Short Messaging Service
SNS	Social Networking Services
TDMA	Time Division Multiple Access
TFT	Thin Film Transistor
TTFF	Time-to-first-fix
UI	User Interface
URL	Uniform Resource Locator
USSD	Unstructured Supplementary Service Data
VOIP	Voice Over IP
WAC	Wholesale Applications Community
WP7	Windows Phone 7
XAML	Extensible Application Markup Language