Cross platform Mobile Applications Development

Mobile Apps Mobility

Yonathan Aklilu Redda

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

1. **Introduction**  
   - 1.1 Motivation ........................................ 2  
   - 1.2 Objective .................................. 3  
   - 1.3 Research Method .......................... 3  

2. **Mobile Applications Development**  
   - 2.1 Mobile Device Evolution ..................... 5  
     - 2.1.1 Smartphones .......................... 6  
     - 2.1.2 Tablet PCs ......................... 7  
   - 2.2 Classic MAD Challenges ................. 8  
   - 2.3 Mobile Platform ......................... 9  
     - 2.3.1 Types of mobile platforms (Mobile OS) 9  
   - 2.4 Mobile Applications .................. 18  
     - 2.4.1 Types of mobile applications .. 21  
     - 2.4.2 Telco-enriched apps ............ 23  
     - 2.4.3 Mobile Enterprise Applications tools . 24  
     - 2.4.4 Mobile Applications Development tools licensing 24  
     - 2.4.5 Common native features of a mobile platform 25  

3. **Cross platform Mobile Apps Development**  
   - 3.1 Mobile application Development ....... 27  
     - 3.1.1 Native mobile applications development 28  
     - 3.1.2 Cross platform development .... 29  
   - 3.2 The benefits of cross platform MAD .... 29  
   - 3.3 Cross platform system architecture .... 31  
     - 3.3.1 Mobile Web Apps ................ 31  
     - 3.3.2 Abstraction based cross platform system 31  
     - 3.3.3 Native compiling cross platform system 34  
   - 3.4 cross platform MAD tools .......... 34  
     - 3.4.1 Appcelerator Titanium ........ 34  
     - 3.4.2 PhoneGap ........................ 35  
     - 3.4.3 Xamarin .......................... 37  
     - 3.4.4 Rhomobile ........................ 39  
     - 3.4.5 MoSync ........................... 40  
     - 3.4.6 IBM Worklight .................. 41  
     - 3.4.7 Corona .......................... 42  
     - 3.4.8 Marmalade ....................... 43  
     - 3.4.9 Adobe Air ......................... 43  
   - 3.5 Tools comparison framework .......... 46  
     - 3.5.1 Example: How to use the framework 48  
   - 3.6 Sample Applications ............... 51  

4. **Discussion and Evaluation**  
   - 4.1 Main findings .......................... 60  
   - 4.2 Validity of result .................... 64  

5. **Conclusion**  
   - 65
List of Figures

1. Motorola DynaTAC 8000X [32] ........................................... 5
2. Easily carried mobile phone device [92] .......................... 6
3. Android [136], Windows phone 7 [117] and iPhone [73] smartphones . 7
4. ASUS’s latest Quad-core Keyboard dockable tablet PC [43] .... 7
7. iOS system architecture [80] ................................................. 11
10. A reconstructed inner structure of WAC Framework [38] .... 13
12. Windows Phone iconic UI on a Nokia Lumia 800 [159] ....... 15
14. Symbian system architecture [57] ....................................... 17
15. BlackBerry Service Architecture ....................................... 18
17. Mobile apps distribution and development life cycle overview [70] . 19
18. Windows phone market model ........................................... 20
19. Android and iOS performance in the US [48] ..................... 21
20. SMS System [110] ............................................................. 23
22. Native App Development [153] ......................................... 28
23. J2ME [69] ................................................................... 29
25. Abstraction based cross platform overview ........................ 31
26. Appcelerator Titanium architecture [139] ......................... 34
27. PhoneGap architecture [75] ............................................... 37
28. Xamarin architecture for MonoTouch and Mono for Android [155] . 37
29. Rhodes application architecture [75] .................................. 39
30. Rhocnect architecture [120] .............................................. 40
31. Worklight Architecture [154] ............................................ 42
32. Pizza bread selection window ......................................... 51
33. Toppings selection window ............................................ 52
34. Address filling window .................................................. 52
35. Order submission window ............................................. 53
36. Appcelerator Titanium Code sample 1 ............................. 57
37. Appcelerator Titanium Code sample 2 ............................. 58
38. XAML sample code .......................................................... 58
39. Xamarin sample code .................................................... 59

List of Tables

1. Cross-platform mobile apps development tools [150] ........ 2
2. Android device as a minimum requirement [51] ............... 10
3. Android Vs Bada OS comparison [19] ............................. 18
4. Overview of platform providers [71] ............................... 20
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>General Mobile apps category</td>
</tr>
<tr>
<td>6</td>
<td>App development comparison</td>
</tr>
<tr>
<td>7</td>
<td>Mobile web apps vs Mobile apps</td>
</tr>
<tr>
<td>8</td>
<td>Appcelerator Titanium native support</td>
</tr>
<tr>
<td>9</td>
<td>PhoneGap native support</td>
</tr>
<tr>
<td>10</td>
<td>Xamarin native support</td>
</tr>
<tr>
<td>11</td>
<td>MoSync native support</td>
</tr>
<tr>
<td>12</td>
<td>Cross platform tools market penetration</td>
</tr>
<tr>
<td>13</td>
<td>Cross platform tool summarization</td>
</tr>
<tr>
<td>14</td>
<td>Mobile platform support</td>
</tr>
<tr>
<td>15</td>
<td>Tool comparison framework</td>
</tr>
<tr>
<td>16</td>
<td>License cost and developer effort</td>
</tr>
<tr>
<td>17</td>
<td>API and Platform coverage</td>
</tr>
<tr>
<td>18</td>
<td>Technology and cost rating</td>
</tr>
<tr>
<td>19</td>
<td>Pizza ordering app development tool comparison</td>
</tr>
</tbody>
</table>
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 [7], by today’s 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 straightforward 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]

<table>
<thead>
<tr>
<th>Adobe Air</th>
<th>BlackBerry</th>
<th>Gideros Mobile</th>
<th>NS Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marmalad</td>
<td>Blueprint</td>
<td>JwGame Engine</td>
<td>OpenPlug</td>
</tr>
<tr>
<td>aletics</td>
<td>BREW</td>
<td>Jmango</td>
<td>Particle SDK</td>
</tr>
<tr>
<td>AppFurnace</td>
<td>Canappi</td>
<td>July Systems</td>
<td>PhoneGap</td>
</tr>
<tr>
<td>Application Craft</td>
<td>CellSDK</td>
<td>Kony</td>
<td>Rhomobile</td>
</tr>
<tr>
<td>Appcelerator</td>
<td>Celsius</td>
<td>Lazarus</td>
<td>Tiggzi Mobile</td>
</tr>
<tr>
<td>Appception</td>
<td>CloudPact</td>
<td>Meme IDE</td>
<td>Total Cross</td>
</tr>
<tr>
<td>appMobi</td>
<td>CoStore</td>
<td>MobiFlex</td>
<td>Unity</td>
</tr>
<tr>
<td>Aqua</td>
<td>Corona SDK</td>
<td>MobileNationHQ</td>
<td>WebORB IS</td>
</tr>
<tr>
<td>Basic4Android</td>
<td>DragonRAD</td>
<td>Moscrif</td>
<td>WinDev Mobile</td>
</tr>
<tr>
<td>BatteryTech</td>
<td>FeedHenry</td>
<td>MoSync</td>
<td>IBM worklight</td>
</tr>
<tr>
<td>Bedrock</td>
<td>GeneXus</td>
<td>NeoMAD</td>
<td>Fivespark</td>
</tr>
</tbody>
</table>

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
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 \cite{6,44,60}. 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 \cite{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 vendor’s 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].

![Motorola DynaTAC 8000X](image)

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].
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.
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.

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 standpoint 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]

![Android System Architecture](image)

**Figure 5: Android System Architecture** [16]

Android is not only an operating system but also a mobile software toolkit con-
ceived 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]

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

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.

![Android Market share percentage for Q3 2011.](image)
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]. 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].

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].

13
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.

![Figure 11: Windows Device Specification for OEMs](image)

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].
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].
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-
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.

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
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>
<thead>
<tr>
<th>OS Layers</th>
<th>Android</th>
<th>Bada</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Framework</td>
<td>Open API framework</td>
<td>Apps and Widgets</td>
</tr>
<tr>
<td>Libraries</td>
<td>Service Library</td>
<td>C++ library</td>
</tr>
<tr>
<td>Kernel</td>
<td>Kernet</td>
<td>Linux kernel</td>
</tr>
</tbody>
</table>

### Table 3: Android Vs Bada OS comparison [19]

#### 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
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].

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 Microsoft’s Windows Phone marketplace.

Figure 16: Bada System Architecture [47]

Figure 17: Mobile apps distribution and development life cycle overview [70]
Until fairly recently, the mobile industry was a privilege to mobile network operators \cite{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 \cite{71}.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Platform & Technology & Portal & Devices & Integration \\
\hline
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 \\
\hline
\end{tabular}
\caption{Overview of platform providers \cite{71}}
\end{table}

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

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>
<thead>
<tr>
<th>Mobile Application Types</th>
<th>Example from Google Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td></td>
</tr>
<tr>
<td>Email Clients</td>
<td>Facebook</td>
</tr>
<tr>
<td>IM Clients</td>
<td></td>
</tr>
<tr>
<td>Mobile web &amp; browsers</td>
<td></td>
</tr>
<tr>
<td>News</td>
<td></td>
</tr>
<tr>
<td>Social Network Clients</td>
<td></td>
</tr>
<tr>
<td>Games</td>
<td></td>
</tr>
<tr>
<td>Puzzle (Tetris, Sudoku)</td>
<td>Angry Birds</td>
</tr>
<tr>
<td>Cards (Solitaire, Roulette)</td>
<td></td>
</tr>
<tr>
<td>Action (Doom)</td>
<td></td>
</tr>
<tr>
<td>Sports (Football, Soccer)</td>
<td></td>
</tr>
<tr>
<td>Multimedia</td>
<td></td>
</tr>
<tr>
<td>Graphics viewers</td>
<td>youtube</td>
</tr>
<tr>
<td>Presentation viewers</td>
<td></td>
</tr>
<tr>
<td>Video players</td>
<td></td>
</tr>
<tr>
<td>Audio players</td>
<td></td>
</tr>
<tr>
<td>Streaming players</td>
<td></td>
</tr>
<tr>
<td>Productivity</td>
<td></td>
</tr>
<tr>
<td>Calendars</td>
<td>CalenGoo</td>
</tr>
<tr>
<td>Calculators</td>
<td></td>
</tr>
<tr>
<td>Diary</td>
<td></td>
</tr>
<tr>
<td>Notepad/ Memo</td>
<td></td>
</tr>
<tr>
<td>Spreadsheets</td>
<td></td>
</tr>
<tr>
<td>Directory services</td>
<td></td>
</tr>
<tr>
<td>Banking &amp; finance</td>
<td></td>
</tr>
<tr>
<td>Travel</td>
<td></td>
</tr>
<tr>
<td>City guide</td>
<td>FlightTrack</td>
</tr>
<tr>
<td>Currency converter</td>
<td></td>
</tr>
<tr>
<td>Translator</td>
<td></td>
</tr>
<tr>
<td>GPS/ Maps</td>
<td></td>
</tr>
<tr>
<td>Itineraries</td>
<td></td>
</tr>
<tr>
<td>Weather</td>
<td></td>
</tr>
<tr>
<td>Utilities</td>
<td></td>
</tr>
<tr>
<td>Profile manager</td>
<td></td>
</tr>
<tr>
<td>Idle screen</td>
<td></td>
</tr>
<tr>
<td>Address book</td>
<td></td>
</tr>
<tr>
<td>Task manager</td>
<td></td>
</tr>
<tr>
<td>Call manager</td>
<td></td>
</tr>
<tr>
<td>File manager</td>
<td></td>
</tr>
</tbody>
</table>
2.4.2 Telco-enriched apps

SMS — Short Messaging Service

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

![SMS System Diagram]

SMS is used in business communications such as banking and individual communication. SMS provides a store and forward system—which makes it possible for the message to be stored in a telecom operator’s 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.

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. Unlike SMS, USSD communications are not stored and forwarded. It instead, sets up a virtual connection between communicating devices. 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. 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.
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. 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 propriety 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 applications development.

MIT Licensing

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 earth’s 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.

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.

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.

![Native App Development](image-url)

Figure 22: Native App Development
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].

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 plat-
forms. 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 than 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.

![Abstraction based cross platform overview](image)

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 \textsuperscript{[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 \textsuperscript{[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 \textsuperscript{[54]}.

Table 6: App development comparison \textsuperscript{[154]}

<table>
<thead>
<tr>
<th></th>
<th>Device Access</th>
<th>Speed</th>
<th>Devt Cost</th>
<th>AppStore</th>
<th>Approval Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native</td>
<td>Full</td>
<td>Very fast</td>
<td>Expensive</td>
<td>Yes</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Hybrid</td>
<td>Full</td>
<td>Native speed</td>
<td>Reasonable</td>
<td>Yes</td>
<td>Low overhead</td>
</tr>
<tr>
<td>Web</td>
<td>Partial</td>
<td>Fast</td>
<td>Reasonable</td>
<td>No</td>
<td>None</td>
</tr>
</tbody>
</table>

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 \textsuperscript{[82]}. The table below will summarize what a mobile websites(mobile web apps) and abstraction based mobile apps are.
<table>
<thead>
<tr>
<th>Comparison</th>
<th>Mobile website</th>
<th>Mobile app</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>Any mobile browser</td>
<td>needs special device</td>
</tr>
<tr>
<td>User experience</td>
<td>depends on bandwidth, technology, site capacity</td>
<td>robust user experience</td>
</tr>
<tr>
<td>Graphics &amp; effects</td>
<td>limited by bandwidth, site performance</td>
<td></td>
</tr>
<tr>
<td>Access to Hardware</td>
<td>limited</td>
<td>Unlimited (camera, GPS etc)</td>
</tr>
<tr>
<td>Ease of development</td>
<td>uses web technologies</td>
<td>native code, cross platform codes</td>
</tr>
<tr>
<td>Development resource</td>
<td>build once deployed everywhere</td>
<td>build for specific Mobile OS</td>
</tr>
<tr>
<td>Development cost</td>
<td>less expensive</td>
<td>more expensive</td>
</tr>
<tr>
<td>Ease &amp; speed of implementation</td>
<td>published as a website</td>
<td>require app store</td>
</tr>
<tr>
<td>Distribution</td>
<td>internet</td>
<td>download and installation</td>
</tr>
<tr>
<td>Installation</td>
<td>No installation</td>
<td>download and installed</td>
</tr>
<tr>
<td>Updates &amp; maintenance</td>
<td>easy</td>
<td>requires an app store</td>
</tr>
<tr>
<td>Search optimization</td>
<td>standard search</td>
<td>app store search</td>
</tr>
<tr>
<td>connectivity</td>
<td>required</td>
<td>can be used offline</td>
</tr>
</tbody>
</table>
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.

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]

<table>
<thead>
<tr>
<th>Appcelerator Titanium native API support</th>
<th>Android</th>
<th>iOS</th>
<th>BlackBerry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Analytics</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Barcode</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Calendar</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Camera</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Compass</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Contacts</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Database</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>File</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Map</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Media</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Network</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Notification</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SMS</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

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]

<table>
<thead>
<tr>
<th>PhoneGap native support</th>
<th>Android</th>
<th>iOS</th>
<th>BlackBerry</th>
<th>WP7</th>
<th>Symbian</th>
<th>Bada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Analytics</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Barcode</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Calendar</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Camera</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Compass</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Contacts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Database</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>File</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Map</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Network</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SMS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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.
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.
Xamarin uses the .NET Base Class Library (BCL) which is a collection of Microsoft’s .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]

<table>
<thead>
<tr>
<th>Xamarin native support</th>
<th>Android</th>
<th>iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Analytics</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Barcode</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Calendar</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Camera</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Contacts</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Database</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>File</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Map</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Network</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SMS</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
3.4.4 Rhomobile

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 Sales-Force.

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](image)

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.
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

<table>
<thead>
<tr>
<th>MoSync native support</th>
<th>Android</th>
<th>iOS</th>
<th>BlackBerry</th>
<th>WP7</th>
<th>Symbian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Analytics</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Barcode</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Calendar</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Camera</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Compass</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Contacts</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Database</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>File</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Map</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Media</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Network</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Notification</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>SMS</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

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-
terprise applications development platform.

According 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.

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.

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.

Figure 31: Worklight Architecture
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

<table>
<thead>
<tr>
<th>Developer</th>
<th>SDK Downloads</th>
<th>Number of Apps</th>
<th>Sample Apps</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appcelerator Titanium</td>
<td>1600000</td>
<td>250000</td>
<td>35000</td>
<td>NBC iPad, MyTravel, Apache 2.0</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>NA</td>
<td>600000</td>
<td>NA</td>
<td>Wikipedia, Netflix, MIT</td>
</tr>
<tr>
<td>Rhomobile</td>
<td>NA</td>
<td>100000</td>
<td>NA</td>
<td>SuperTrainerHQ, MIT</td>
</tr>
<tr>
<td>MoSync</td>
<td>20000</td>
<td>180000</td>
<td>NA</td>
<td>MoSync Reload, GPL/Commercial</td>
</tr>
<tr>
<td>Xamarin</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Monster Stack 2, Commercial</td>
</tr>
<tr>
<td>Worklight</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA, Commercial</td>
</tr>
<tr>
<td>Corona</td>
<td>10000</td>
<td>100000</td>
<td>6000</td>
<td>Bubbleball, Commercial</td>
</tr>
<tr>
<td>Marmalade</td>
<td>50000</td>
<td>NA</td>
<td>NA</td>
<td>Plants vs Zombies, Backbreaker 2, Commercial</td>
</tr>
<tr>
<td>Adobe Air</td>
<td>3000000</td>
<td>NA</td>
<td>NA</td>
<td>eBay, BBC iPlayer, Commercial</td>
</tr>
</tbody>
</table>
Table 13: Cross-platform tool summarization

<table>
<thead>
<tr>
<th>Appcelerator Titanium</th>
<th>Best for apps of</th>
<th>Strength</th>
<th>Weakness</th>
<th>Cross platform model</th>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Communications, Productivity, Travel</td>
<td>Free, MPS, CDS, Rich documentation, native UI access, iOS</td>
<td>Supports only iOS and Android</td>
<td>AB</td>
<td>HTML, CSS, JS, PHP, Ruby, Python</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>HTML</td>
<td>Free, MPS, MPS</td>
<td>Runs confined in WebView</td>
<td>AB</td>
<td>HTML, CSS, JS</td>
</tr>
<tr>
<td>Rheincode</td>
<td>Productivity</td>
<td>Free, MPS, CDS</td>
<td>code interpreted, requires Ruby skill</td>
<td>AB</td>
<td>HTML, Ruby</td>
</tr>
<tr>
<td>Mobile</td>
<td>Communications, Games</td>
<td>Produces native code</td>
<td>-</td>
<td>AB, NC</td>
<td>C/C+++, HTML, JS</td>
</tr>
<tr>
<td>Xamarin</td>
<td>Communications, Games, Multimedia, Productivity, Travel</td>
<td>Produces native code</td>
<td>Commercial and supports only iOS and Android</td>
<td>NC</td>
<td>CF</td>
</tr>
<tr>
<td>Worklight</td>
<td>Productivity</td>
<td>MPS, CDS, MPS</td>
<td>-</td>
<td>AB</td>
<td>HTML, CSS, JS</td>
</tr>
<tr>
<td>Corona</td>
<td>Games, Multimedia</td>
<td>2D and 3D game</td>
<td>Limited to game apps</td>
<td>AB</td>
<td>Java</td>
</tr>
<tr>
<td>Gamebryo</td>
<td>Games, Multimedia</td>
<td>2D and 3D game</td>
<td>Limited to game apps</td>
<td>NC</td>
<td>C++</td>
</tr>
<tr>
<td>Adobe Air</td>
<td>Communications, Games, Multimedia, Productivity, Travel</td>
<td>Rich multimedia content distribution</td>
<td>Large application size, requires flash player</td>
<td>AB</td>
<td>HTML, CSS, JS, ActionScript</td>
</tr>
</tbody>
</table>

**KEY:**
- **MPSC** Multiple Platform with single code base
- **MPS** Multipurpose support
- **CDS** Cloud data syncing
- **AB** Abstraction based cross-platform system
- **NC** Native Compiling cross-platform system
Table 14: Mobile platform support

<table>
<thead>
<tr>
<th>Cross platform development tools mobile platform support</th>
<th>iOS</th>
<th>Android</th>
<th>WP7</th>
<th>RIM</th>
<th>Symbian</th>
<th>Bada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appcelerator Titanium</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rhomobile</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MoSync</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Xamarin</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Worklight</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Corona</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Marmalade</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Adobe Air</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Score</th>
<th>Native feature support</th>
<th>CPDT</th>
<th>Platforms</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Geolocation</td>
<td>Camera</td>
<td>SMS</td>
<td>Contact</td>
</tr>
<tr>
<td></td>
<td>A. Titanium</td>
<td>PhoneGap</td>
<td>Rhomobile</td>
<td>MoSync</td>
</tr>
</tbody>
</table>

**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 taxing 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.
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

<table>
<thead>
<tr>
<th>Score</th>
<th>Native feature support</th>
<th>CPDT</th>
<th>Platforms</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Geoloc</td>
<td>Camera</td>
<td>SMS</td>
<td>Contact</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>CDPT</th>
<th>Cost</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Organizational</td>
<td>License</td>
</tr>
<tr>
<td>A. Titanium</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rhomobile</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>MoSync</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Xamarin</td>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>Worklight</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Corona</td>
<td>3</td>
<td>*</td>
</tr>
<tr>
<td>Marmalade</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

**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.

![Pizza bread selection window](image)

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.
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.
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.

![Order submission window](image)

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

<table>
<thead>
<tr>
<th></th>
<th>Appcelerator Titanium</th>
<th>Xamarin</th>
</tr>
</thead>
<tbody>
<tr>
<td>code lines</td>
<td>Window (toppings)</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Textbox</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Checkbox</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Button</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>ScrollView</td>
<td>9</td>
</tr>
<tr>
<td>User control implementations</td>
<td>Window</td>
<td>JS, Titanium Code</td>
</tr>
<tr>
<td></td>
<td>Textbox</td>
<td>JavaScript, Titanium Code and Images</td>
</tr>
<tr>
<td></td>
<td>Checkbox</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ScrollView</td>
<td>JS, Titanium Code</td>
</tr>
<tr>
<td>Perceived speed</td>
<td>Compilation</td>
<td>slower</td>
</tr>
<tr>
<td></td>
<td>Running</td>
<td>slower</td>
</tr>
<tr>
<td>Code Type</td>
<td>JS, Titanium Code</td>
<td>C#</td>
</tr>
<tr>
<td>Code Syntax Similarity</td>
<td>JavaScript</td>
<td>Java, C++</td>
</tr>
<tr>
<td>SMS support</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Application Size</td>
<td>22.8MB</td>
<td>2.28MB</td>
</tr>
<tr>
<td>Test result as expected</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
In the figure above, line 4, 5, 17, and 33 are Titanium specific codes.

```javascript
/**
 * @author joneix
 */

var win=Ti.UI.currentWindow;
var scrollView = Ti.UI.createScrollView();
var maxToppings = 6;
var numToppings=8

var toppings = [
    {title:'Bacon', path:'../images/toppings/bacon_bits.png', container:null},
    {title:'Beef', path:'../images/toppings/beef.png', container:null},
    {title:'Italian Sausage', path:'../images/toppings/italian_sausage_slices.png', container:null},
    {title:'Grilled Chicken',path:'../images/toppings/grilled_chicken.png', container:null},
    {title:'Jalapenos', path:'../images/toppings/jalapenos.png', container:null}];

var toppingsTitle = Ti.UI.createLabel({
    text: '1. Choose your toppings',
    font: {
        fontFamily: 'Verdana',
        fontSize: 22
    },
    color: '#003333',
    shadowColor: '#3333',
    shadowOffset: {x:1, y:1},
    textalign:'left',
    width:Ti.Platform.displayCaps.platformWidth,
    height:58,
    left:10
});

var toppingsTitleView = Ti.UI.createView({
    width:328,
    height:58,
    backgroundImage:'../images/crustHeaderB.png',
    top:100,
    left:-6,
    opacity:0
});
```
```javascript
79 if(Ti.Platform.osname==='android'){
80     details.image='..//images/details.png';
81     cancel.image = '../images/cancel.png';
82     pizza.image=win.path;
83 }else{
84     pizza.opacity=0;
85 }
86     win.add(details);
87     win.add(cancel);
88     //this event handler lets it go back to the crust window.
89     cancel.addEventListener('click', function(e){
90         Ti.App.fireEvent('cancelToppings',{crust:win.crust});
91     });
92     details.addEventListener('click', function(e){
93         //nothing is written here
94     });
95     toppingsTitleView.animate({
96         opacity:1,
97         duration:500
98     });

Figure 37: Appcelerator Titanium Code sample 2

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


67


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


73


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