ATTAINING UNIFORMITY IN USER INTERFACES ACROSS MOBILE PLATFORMS - A DEVELOPER'S PERSPECTIVE

by

Deepak Karthikeyan Rajendran

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
November 2012

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**ATTAINING UNIFORMITY IN USER INTERFACES ACROSS MOBILE PLATFORMS - A DEVELOPER'S PERSPECTIVE**" by Deepak Karthikeyan Rajendran in partial fulfilment of the requirements for the degree of Master of Computer Science.

Dated: 19-Nov-2012

Supervisor: _____

Co-supervisor: _____

Reader: _____

DALHOUSIE UNIVERSITY

DATE: 19-Nov-2012

AUTHOR: Deepak Karthikeyan Rajendran

TITLE: **ATTAINING UNIFORMITY IN USER INTERFACES ACROSS MOBILE PLATFORMS – A DEVELOPER'S PERSPECTIVE**

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: MCSc CONVOCATION: May YEAR: 2013

_____

Signature of Author

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Mobile Application Development encompasses disparate facets such as architectural design, development and deployment, integration with existing web applications and business issues. Cross-platform mobile application development is one of the significant emerging areas in Mobile Application Development. Cross-platform technique can be approached by two ways: Cross-compilation for each mobile platform and porting a single code base to multiple platforms by leveraging platform oriented capabilities. Cross-compilation can have significant deployment implications, constraining functionality to be delivered through Application Programming Interfaces. Hybrid Mobile Application technique, written with web technologies, is one of the alternatives for porting an application to multiple platforms by utilizing the device's browser engine. Even though applications are deployed across platforms using hybrid approach, user interfaces lack consistency across platforms. In order to provide unified user interfaces across platforms, this thesis work proposes a solution of a hybrid mobile application approach by combining two cross-platform mobile application frameworks. Issues and elucidations are technically delineated in developer's perspective.

LIST OF ABBREVIATIONS USED

SDK	-	Software Development Kit
API	-	Application Programming Interface
IDE	-	Integrated Development Environment
HTML	-	HyperText Markup Language
CSS	-	Cascading Style Sheet
XML	-	Extensible Markup Language
XSL	-	Extensible Stylesheet Language
XAML	-	Extensible Application Markup Language
MWI	-	Mobile Web Initiative
COM	-	Component Object Model
MIDP	-	Mobile Information Device Profile
CLR	-	Common Language Runtime
JVM	-	Java Virtual Machine

ACKNOWLEDGEMENTS

Working on the Master's Thesis has been a wonderful and often innovative experience. I am deeply grateful to my Professors Dr. Morven Gentleman, Dr. Vlado Keselj. You have been patient and encouraging in times of new concepts and methods. I am very thankful to Dr. Morven Gentleman for his valuable suggestions and his influential ideas to select and to approach compelling research problems. You have listened to my ideas and discussions with you frequently led to key insights. Furthermore, I am very grateful to my Reader Dr. Srinivas Sampalli, for insightful comments both in my course works and in this thesis, for his support, and for many motivational thoughts.

I am indebted to my friends for making the time working on my Master's degree an unforgettable experience. I thank Mr. Bharatram Raghuraman for his concern and advices to balance research interests and personal pursuits.

Mrs. Shanthi Rajendran, my beloved mother, who has been my everything and she has given me an infinite support. I dedicate my work to her. Thank you with all my heart.

CHAPTER 1: INTRODUCTION

Smartphones are considered as mini personal computers and they are capable of performing multiple operations simultaneously. A very high demand for mobile applications by businesses and consumers has significantly increased the technical requirements of smartphones. Business is gradually transferring to a higher level in smartphones. Smartphone platforms are hugely market-driven and based upon the characteristics and requirements of mobile applications.

Mobile Application development has a broad prospect due to the rapid development of smartphone market. There are various mobile platforms existing today and each platform has its own uniqueness in delivering excellence to users. There are two types of mobile application development: Native Mobile Application Development and Mobile Web Application Development. Native applications are developed using the software development kits, development languages and Application Programming Interfaces (API's) provided by mobile Operating System manufacturers such as Apple, Android, BlackBerry, Windows Phone, etc. [1]. Native applications are the applications which are created by native application development method that will be installed and operated on the mobile device. Mobile web applications are developed using web development languages (HTML 5, CSS and JavaScript) that operate in the browser of the mobile device. More specifically, Web application is a collection of web pages distributed over HTTP (Hypertext Transfer Protocol) which use server-side processing. Web app development is distinct as the apps encompass locally executable components of interactivity and determined state.

Application development in Smartphone platforms brings both opportunities and challenges to software developers. Many apps are distributed free by businesses as a channel for them to interact with their customers, whatever smartphone platform their customers choose to use (e.g. CIBC, NY Times). Moreover, since that platform may change over time, cross-platform capability provides the best insulation against their customers being disrupted by platform change. This has impacted a high increase for interest in the mobile application development services. With the possibility of improving

proficiency in every smartphone platform, many developers are increasingly considering cross-platform mobile application development.

1.1 Cross-Platform Mobile Application Development

According to Linux Information Project [2], the term 'Cross-platform' can be defined as "the ability of software to operate on more than one platform with identical (or almost identical) functionality". This approach is purely based on "Write once and run anywhere" slogan created by Sun Microsystems to illustrate the cross-platform benefits. A cross-platform application should run in more than one or many platforms and it is essential that developers can reuse as much of the same code on as many devices as possible. This would immensely reduce the extent of work required to make an application run on different platforms, which provides brand recognition across platforms and reduces cost. Cross-platform mobile application development has become a quintessential part of mobile application development in recent times, taking into consideration the time, money and resources needed to be allotted for developing an application.

1.2 Research Problem

In order to deploy the same application in more than one platform, Native mobile application development demands the knowledge of different programming languages and Software Development Kits of respective platforms. For example, if an application needs to be deployed in Android, Windows Phone 7 and iOS platform, then the developers should know the programming languages (Java for Android, Objective C for iOS and C# for Windows Phone 7) and platform knowledge of each mobile platform.

Mainly, the application should be developed separately in both the platforms. Portability is the usage of the same software in different environments and it is the key for cost reduction in development. From a mobile application developer's perspective, difference in programming languages used for developing applications across different platforms and consequently the differences in syntax of these programming languages have led to look for ways to port applications from one platform to another rather than develop applications individually for different platforms. Standard portability technology of the 1970s and 1980s found that a program is portable to the extent that it can be moved to a

new computing environment with much less effort than would be required to write it from scratch [3].

Due to the increased competition among various popular smartphone companies, alternative forms of mobile application development have been introduced. Web-based and Hybrid mobile application development approaches are the two most popular methods in the current trend. Web-based application is developed through Web languages such as HTML 5 (Hyper Text Markup language), CSS (Cascading Style Sheet) and JavaScript. It is released through internet rather than releasing in app store. It is constrained to the mobile browser and has partial access to device features. Moreover, Web applications fail to perform offline as internet access is required. Considering the drawbacks in both Native and Web-based application development, the next alternative to address the issues is Hybrid Mobile Application Development. As the name mentions, it is the combination of Native and Web-based Mobile app development approach. Applications developed with a hybrid concept are developed using Web languages and able to access native device features. Applications can be ported to different platforms and also be released in app stores. Developers have been trained to think that more features equate to better applications, but on mobile devices, that is simply not true. Even from the user's point of view, more features mean a higher learning cost. In the aspect of native platforms, developers have to undergo an in-depth study of each platform's components. Even with Hybrid app development, a user interface (UI) component in one platform may work differently in another platform. For example, a button interface in Android platform may be misplaced in BlackBerry platform. Although the Hybrid concept has many advantages, the user interfaces are not unified when it is deployed across platforms.

1.3 Objective

To provide a unified user interface for applications across platforms developed using hybrid mobile application development method and consequently ported to other platforms. However, porting an application in various platforms can be achieved by hybrid mobile development technique. Whereas, UIs need consistency for multi-

platforms. Therefore, the solution can be derived by combining two hybrid mobile app techniques (Cross-platform application framework and UI framework).

1.4 Overview

Chapter 2 explains the background concepts and facts of mobile platforms and mobile application development. Required facts and specifics that are required for the thesis work are clearly described in this chapter. Cross-compiling two different mobile platforms is one of the emerging approaches of mobile app developments. Chapter 3 (Literature survey) exemplifies the critical analysis of cross-compilation approaches and also explained the importance of UIs across platforms. Methodology chapter speaks about the techniques involved in this thesis work. Methods which are used to attain uniform user interfaces across mobile platforms are discussed in detail. Chapter 5 is the implementation part that delineates the solution by developing a sample mobile application (Currency conversion) in Android and Windows Phone platforms. Firstly, the sample application is developed natively in Android and then in Windows Phone platform. This method helped to identify the platform differences and to understand how UIs vary across platforms. Secondly, the sample application is developed using hybrid mobile application development techniques. Chapter 6 describes the evaluation part of the thesis work. Characteristics of development effort in both native and hybrid approaches are evaluated in developer's perspective. Conclusion chapter describes the future aspects of the research work.

CHAPTER 2 : BACKGROUND

2.1 Evolution of Mobile Platforms

Mobile platforms were highly influenced after the invention of Apple's first Personal Digital Assistant (Newton Platform) in 1987. It did not behave like a smartphone, and some of its functionality, such as handwriting recognition, has not been copied. Other PDAs preceded the Newton, such as the Psion, first Palm Pilot, the Craig translators, or the HP OmniGo device. Due to the impact of Newton, its models of Newton Script OO programming have not been taken up. Then, the first generation mobile platforms entered the market place by late 1990's. Palm OS, Symbian, Windows Mobile (ancestor of Windows Phone) and BlackBerry OS were the leading platforms at that time [3]. Qualcomm was the dominant OS for mobile phones prior to smartphones. Symbian, as a first-generation smartphone OS, accomplished huge success and consisted of three frameworks such as S60, UIQ and MOAPS. These frameworks were used in Nokia and Ericsson and they played a major role in software market.

The second generation (2G) mobile phones are mainly used for making voice calls and using message services. Individual phone users did not install apps, but Nokia and independent software houses sold apps to carriers. In early 2000s, multimedia features such as music, pictures and video were introduced, as well as the usage of internet. GPRS (General Packet Radio Service) offered by GSM (Global System for Mobile Communication) network providers delivered packet oriented internet service [4].

There were many limitations in the usage of internet at that time. Screen resolution and screen size (initial 40 by 16 char) were the biggest issues. High cost for services was one among the limitations. Another vital issue was limited RAM capability. A typical Qualcomm or Symbian phone had 1 Mbyte RAM, and early Windows Mobile phones 64 Mbytes RAM. Currently, the minimum requirement of Windows Phone 7 device is 256 MB of RAM with at least 4GB of flash memory. Battery lifetime was also an issue. Smartphone foundation was laid after 2002 under the third generation (3G) mobile networking. 3G phones made high speed internet possible and it offered new innovative services like streaming video or VOIP-calls (Voice over Internet Protocol – calls) [4].

Momentarily, Smartphones became personal computers for users that included a wide range of attractive software applications. In fourth generation (4G) mobile phones (successor of third generation mobile phones) ultra-broadband internet access and high mobility communication were made available.

2.2 Smartphone Platforms Overview

Mobile Devices are available abundantly and many new options are being introduced in the tremendously growing market. During the early days of mobile phones, mobile devices were mostly hardware driven but now it is entering the hugely dominant software age. Major smartphones operating system which are popular among users are discussed below.

2.2.1 Android

The Android operating system is an open source platform and released under the open source Apache license built on Linux kernel. Android applications are written in Java. Other development tools are available encompassing a Native Development Kit for applications or extensions in C or C++. Java classes are recompiled into dalvik byte code and are operated on dalvik virtual machine. The most generally used and recommended IDE (Integrated Development Environment) is Eclipse with Android Development Tools plug-in. The plug-in offers complete featured development environment that is integrated with the emulator. Emulator allows the developers to connect with the different versions of Android platform with ease and also provides debugging capabilities. There are other command-line tools, if a developer does not prefer Eclipse [5].

2.2.2 BlackBerry OS

The BlackBerry OS is a product of Research in Motion (RIM) and it became the standard phone for business professionals and executives in Europe and US. BlackBerry platform supports two different ways of developing applications. Applications can be developed by BlackBerry web development and Java application development. The first one is the newest one in which the applications are developed using the widgets. Widgets are small, distinct, standalone web applications that use HTML (Hyper Text Markup Language), CSS (Cascading Style Sheet) and JavaScript. Java application development is the standard way in which BlackBerry apps are developed in Java using MIDP 2.0 (Mobile

Information Device Profile) and RIM's proprietary APIs [5]. Developers need to have experience with Java programming to use this development method. Similar to Android platform, BlackBerry website offers extensive documentation and training videos.

2.2.3 Windows Phone

Windows Phone 7 OS is based on a variant of Microsoft Embedded OS and Windows CE 6 (Windows Embedded Compact) which provides 32-bit kernel. Windows Phone 7 OS is built on top of the CE kernel with additional specific system services and application framework for mobile phones [6]. The runtime for Windows Phone 7 is .NET Framework CLR (Common Language Runtime). Windows phone platform uses XAML (Extensible Application Markup Language) for user interface description and .NET languages (C# and VB .NET plus 90 others) for application logic. Currently, development pattern looks quite similar to Android as the concentration is on XAML for user interfaces and .NET languages for application logic. The IDE for Windows Phone is Microsoft Visual Studio IDE. In Windows Phone 7, developers are not allowed to write native code; only managed code is allowed. Managed code is every bit as much "native" for Windows CE as Java byte code is for a JVM. It is not "native" only in the sense that arbitrary machine language constructs for the physical processor might not satisfy the runtime style conventions required for the code to be "managed". It is straightforward to write machine language that satisfies the CLR conventions. Only subcategories of the general Silverlight APIs and XNA (set of tools with runtime environment) APIs are supported on Windows Phone 7 [6]. Phone makers can write native code into the system and use it in their applications by using COM (Component Object Model). Microsoft offers a special SDK (Software Development Kit) so that the phone makers can develop native COM DLLs with a partial set of native Windows CE (Windows Embedded Compact) APIs and use them in their applications. But developers do not have access to special SDKs. The operating system of Windows Phone 7 kernel is Windows CE 6.

2.2.4 iOS (iPhone)

The iPhone is a line of smartphones designed and promoted by Apple Inc. iOS is a mobile operating system for Apple products (iPhone, iPod Touch and iPad). To develop for iPhone or its products, developers need to have an Intel-based Macintosh computer

running OS X v10.5.7 or other supportive versions and need to install latest version of the iPhone SDK. They also have to validate the latest version of device's operating system. For mobile application development, iPhone SDK has certain tools such as Xcode IDE (Integrated Development Environment), iPhone simulator, and a collection of additional tools for developing applications for iPhone and Mac OS X [5]. The preferred language in Xcode is Objective-C. The Xcode IDE, for its backend, has an improved GNU compiler and debugger. The Xcode set consists of two important components. They are Interface Builder and Instruments. Using Interface Builder developers can create user interfaces for Mac and iPhone applications. Instruments offer a comprehensive analysis of developer's application runtime performance and memory usage. This feature is efficiently helpful in finding memory leaks and blockages to enrich the user experience [5].

2.3 Mobile Application Development Techniques

Mobile phones have become a crucial technology for communication and interaction among customers, associates, employees and communities. Being targeted on a native device alone, in early mobile development days, developers had to choose the right platforms to support their application [1]. Choices were available in platform selection and the development method was native. Due to the expansion of mobile usage and extensive technology, advanced development methods have been introduced. There are three popular mobile application development methods.

1) Native Mobile Application Development
2) Mobile-Web Application Development
3) Hybrid Mobile Application Development

In brief, Native applications are the applications which run locally on the mobile device with the respective platform's programming language [1] and access to the local OS and support framework. Mobile-Web apps are written with web languages (HTML, CSS, JavaScript and other scripting languages) which run within the specific mobile device's browser. Mobile-Web apps may or may not actually make use of servers running elsewhere across the Web. Hybrid Mobile Application method is the combination of

Native and Web based Mobile application techniques and runs locally on the mobile device's browsers.

2.3.1 Native Mobile Application Development

Native application development has traditionally been the most popular choice for developers. Major Smartphone platforms which are all explained above have their own uniqueness in delivering applications. Mobile applications can be classified into two types. They are native application and web application. Web applications are developed using HTML, JavaScript and CSS and they contain web pages enhanced for mobile devices. They cannot be used in off-line mode. Native applications are developed specifically for the respective mobile devices [6] and they operate on the device itself. They can be used during "airplane mode" when the radio transmitting and receiving facilities are shut off. Any application which resides within the phone is able to access phone's features such as camera, accelerometer, compass etc. Following is the pictorial depiction of native application development.

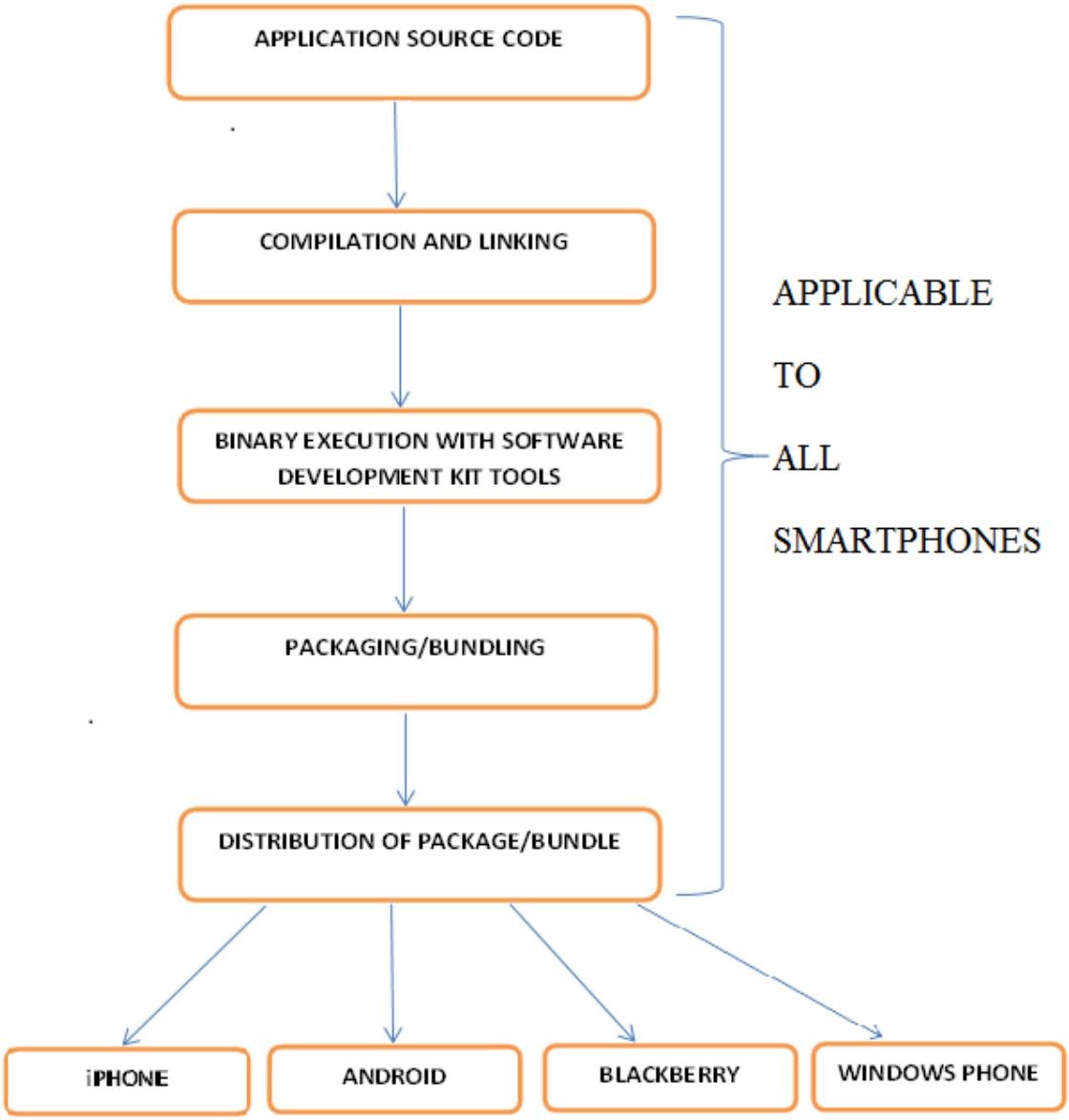

Figure 2-1 : Pictorial Illustration of Native Application Development

Application source code is a sequence of instructions written using any programming language. For iOS, Objective C is used as a programming language and for BlackBerryOS, Java is used as a programming language. A compiler is a set of programs that converts the source code from a high-level programming language to a lower level language (machine code or Virtual Machine instruction language). Each smartphone has Software Development Kit which is typically a set of software development tools that

involves for the creation and execution of application. Even more important, it comprises a library or base framework. After execution, the application is packaged and distributed. Application packaging is the process of systematizing the deployment of software (application) with a predefined set of properties. This operation is not applicable to apps that accept third-party plug-ins. Packaged application is then distributed to the application store of the respective platform. Application store is a digital application distribution platform for smartphones. The service allows users to surf and download the deployed applications.

Due to the various programming languages and devices, it is almost impossible to write a single version of portable mobile application code that runs on different mobile devices. This issue increases the production effort in almost the entire software life cycle – driving up the cost, lengthening the time to market, and narrowing the target market [7]. User interfaces plays an essential and significant role in mobile applications. Developers take special care in designing the user interfaces. Most native platforms have wonderful abstractions in common-user interface controls and experiences. No two platforms have the same user-interface patterns, let alone APIs to represent and access them [8].

When the native code is compiled, it is faster typically 1 to 2 orders of magnitude than interpreted languages such as JavaScript and JRuby (Java implementation of Ruby) [8]. Following are the main advantages of native application development [1].

Device integration: Mobile device capabilities like camera, accelerometer and network communications can be fully exploited and developers have complete authorization in controlling these services.

Performance: There is one less layer between the code and its kernel. As a result, the load times and execution speed of native mobile applications are fast.

Offline capability: Native development permits access to local storage device for offline storage capability and allows developers greater comfort in developing modified storage synchronization.

Application market integration: Developers can submit the binary distribution file to the application market. Mobile app market provides distribution and monetization of mobile application.

Native application development is still best in its own way. But in the aspect of cross-platform development, it has several drawbacks.

Profound platform knowledge: If any application is developed in two or more platforms, developers need to have knowledge of each platform's APIs and programming languages. Developers may not be familiar with two or more programming languages (for example, Objective C for Apple mobile apps, Java for Android and C# for Windows Phone 7). These factors lead to increase in development cost, time and effort. Ultimately, these combined issues become barriers for developers and organizations.

Limitations in portability: A code developed for one platform could not be easily ported to another platform. This is the fact that the existing code influences any platform specific capabilities. User interfaces vary among platforms. For example, push notification used by Android is not the same as Windows Phone 7. Developers have to write separate code for each platform to support necessary features.

2.3.2 Web-based Mobile Application

Mobile web access is the web-browser based access to an application using a mobile device connected to a wireless network. In early days, users accessed web through fixed landline connection with desktop. Web has emerged as a next generation of Internet-based services with intent to make the web a platform. The main reason is that the web has become a significant medium for users to collaborate and share information online by binding collective intelligence [9]. Due to the massive availability of information on web and rapid growth of mobile devices, the drift of accessing web-based services has been transferring from desktop computers towards wireless mobile devices. The enhancement and augmentation to mobile computing and its fundamental structure to access the advanced components of web are known as Mobile Web 2.0 or Mobile 2.0. Mobile 2.0 influences certain services greatly such as accessing web services and integrating their features on mobile platform. It provides rich user experience and it concentrates on

connecting the strength and competencies of the application supported by web 2.0 or Mobile 2.0 and expand them to the mobile platform [10].

Despite the technical specification differences present between iOS, Android, BlackBerry and Windows Phone 7, one great commonality is the standards-compliment web browsers which have been inclusively included in the mobile devices by default [12]. It is highly possible to access the browser from native code. Mobile web apps are simply web pages and users can access them on their mobile device, using the device's standard web browser. Web applications are mostly comprised of HTML, CSS and JavaScript. To create a successful user interface, there is complete utilization of HTML and CSS by WebView and browsers with different levels of proficiency. Following figure interprets diagrammatically the operation of Mobile Web App.

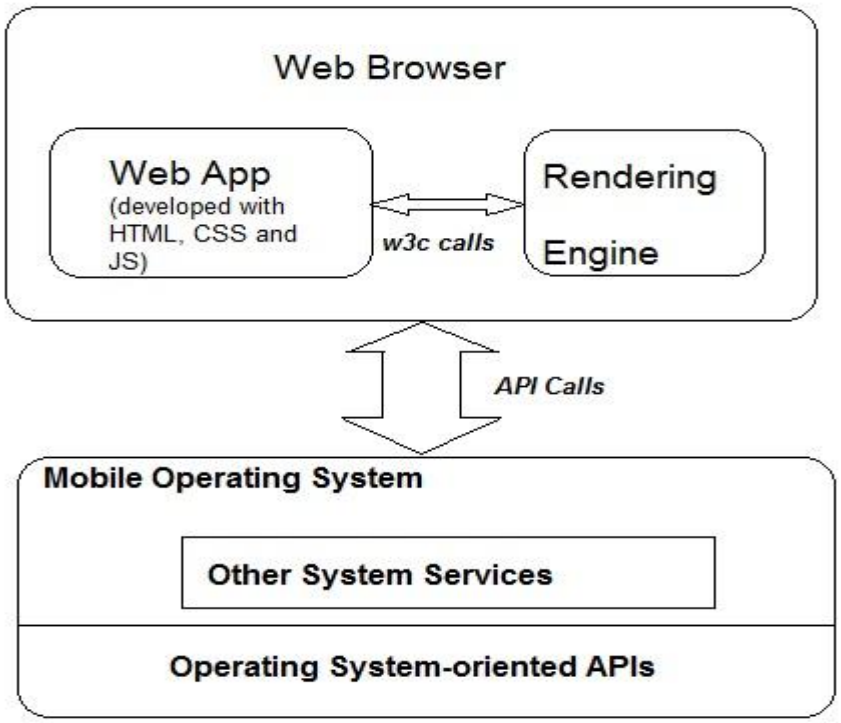

Figure 2-2 : Interaction of Mobile Web App

Web application is developed in the web browser and it interacts with the rendering engine. WebKit is a layout engine (also known as rendering engine) that provides a set of classes to display visual web information and other information like audio or music and video. It implements certain functions such as directing users through links, handling a

back-forward list, history management, etc. WebKit is common for Android, iOS, BlackBerry Tablet OS and WebOs operating System. Windows Phone 7 uses Internet Explorer. Rendering engine renders the web pages to the application and display the content. There are specific sets of APIs to interact with mobile platform system services. The highlight of web based applications is that it can operate across multiple platforms and it can influence web tools and techniques to a good level. In terms of security, the facts are saved in the servers which are in a different (separate) location. Cloud security is a big problem and losing information is not the concern. Web application does not need to have application distribution like application store as it is available through browser and most application stores deliver that way too. The issue is that not all vendors are prepared to give their products away free, and if those products have to be purchased from a special website, that website is in effect an app store.

2.3.2.1 Pros and Cons of Web-based Application

Due to the innovative advent of HTML5 combined with enhancements in JavaScript, mobile web development got huge reception. This has brought many advantages to the growing development of mobile web practice and mobile web development. W3C is creating the best practices and technologies with MWI (Mobile Web Initiative) and HTML5. Following are the advantages of Mobile Web application development [11].

- Web development skills are enough to create a web application as the application development demands web languages. It is quicker to develop a web app than a native app as the web application demands the knowledge of web languages. No in-depth study on platform's SDK and APIs are required.
- Web standards, specifically HTML5 and JavaScript bring the benefit of the slogan "Write once run anywhere" and the application can operate on native mobile platforms through device's browser.
- Mobile web development does not depend on any proprietary SDK licence agreements or any other resources. Web applications can be created by using any text editors.

Following are the limitations of Mobile web development.

- Due to the limited competence of HTML5, applications cannot access the native device features completely. The reason is that the application runs within the web browser and has restrictions in accessing the device APIs [11].
- Applications cannot provide a complete support for data-intensive calculations. With an adequately fast Internet connection, data-intensive processing can be moved from client-to server-side devices [12]. Therefore, web apps demand a continuous Internet connection and there are applications which operate in offline mode but with certain restrictions.
- Response time is one of the essential parts of the UI interaction: Slow response from the user can be evidence of user confusion, prompting for extra wizards and help. Not only do the web protocols ignore time, transmission times across the web are unpredictable. Compared to native applications, mobile web applications are likely to have web-security threats.
- Web app development cannot provide support for applications which have 3D features, intense graphics, complex UIs and advanced animated games. It is hard to expect the performance to be similar to native application.

2.3.3 Hybrid Mobile Application

Hybrid mobile application development approach is a mixture of Native and Web-based approach. Native applications are specifically and technically designed to run on a device's operating system. They are coded with a specific programming language and they are fast, reliable, and robust but are attached to a mobile platform. Developers have to replicate them using the appropriate programming languages in order to target various mobile platforms. Portability across time, i.e. language and OS revisions, has always been a big aspect of portability. HTML, XML and JavaScript versions favors portability as these languages are common in web browsers. At this juncture, alternative development of mobile applications influences the developers. One of the alternative approaches is hybrid application development. The native part of the application uses the operating system APIs to produce an embedded HTML rendering engine that assists as a bridge

between the browser and the device APIs. This bridge authorizes the hybrid app to manage the features of modern devices [13].

Once a web app is created, developer wraps that application in native package using hybrid mobile app tools. A hybrid tool performs this operation by running an embedded version of the WebKit browser inside the native package. This is a standard method for hybrid approach [14, 15]. Both hybrid apps and web apps depend on the web browser layout engine and it helps to run the application. HTML, CSS and JavaScript are the dominant languages used in web development. They are considered as powerful and significant web technologies. Following is the diagrammatic illustration of working of hybrid application structure. A hybrid application starts the development with web languages and the application ends up as a native application.

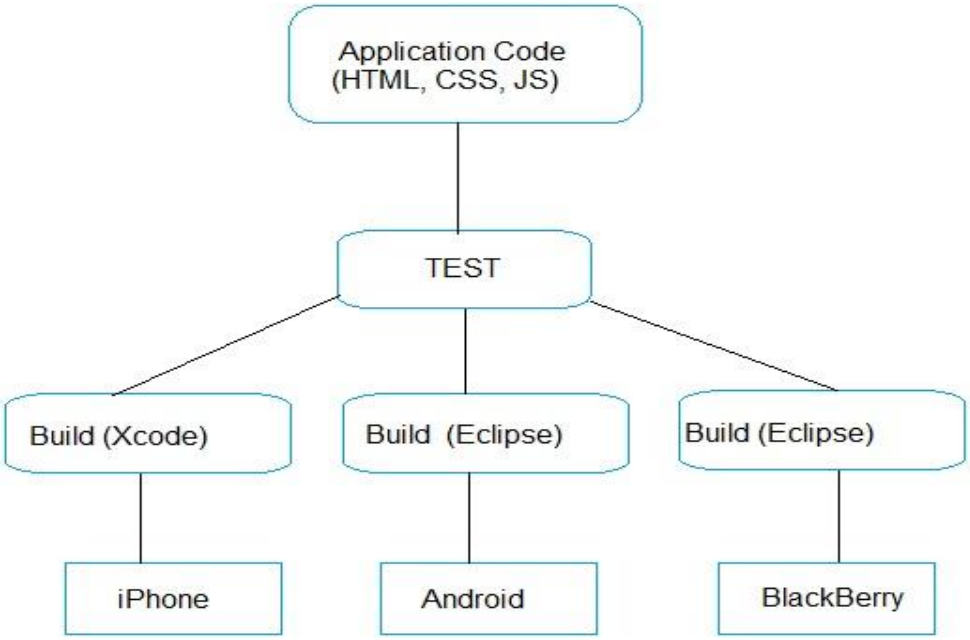

Figure 2-3 : Hybrid Mobile Application Development Mechanism

Applications are written using the web languages (HTML, JavaScript and CSS). Developers can use the development environments that native application developers use. For iPhone, developers use Xcode environment and for Android, developers use Eclipse environment. Any hybrid framework embeds the WebKit browser engine (a native

application) and it exploits this embedded browser to operate the application code written in web languages. The embedded browser engine is identical as the one which is used for mobile Safari and Android browser. Therefore, the code environment is same as for mobile web applications but differs in the final stage of implementation. Hybrid applications are deployed natively [14].

Being an essential component of native platform, WebView has the ability to deliver web pages and it provides the connection (communication) from JavaScript to native code and native code to JavaScript. WebView component cannot directly provide an interface to access the device. For example, if a hybrid application prefers to access the device features such as camera, accelerometer, contacts, etc. then an additional code is absolutely required to support the application. By implementing this functionality, any of the mobile SDK's function can be bridged to JavaScript world. It is evident that both hybrid and native approaches result in native applications with comparable performance and capabilities. The hybrid approach combines (strengthens) development and testing into one modernized project. With the recent introduction of rapid mobile processors and enhanced mobile browser HTML5 support, hybrid mobile development (part web-based and part native) is influencing many developers with increasing adoption and acceptance [1].

2.3.3.1 Architecture of Hybrid Mobile Application Development Approach

In today's web development, most of the cross-platform development frameworks are influenced by three techniques of rapid application development [11]. Firstly, layout with web standards (mark-up) that is using HTML, CSS, secondly, using identifying the screen layouts of the visual condition (simulator, device API wrapper). The third one is integrating dynamic languages. Diagram shown below is the pictorial delineation of Hybrid mobile application development architecture.

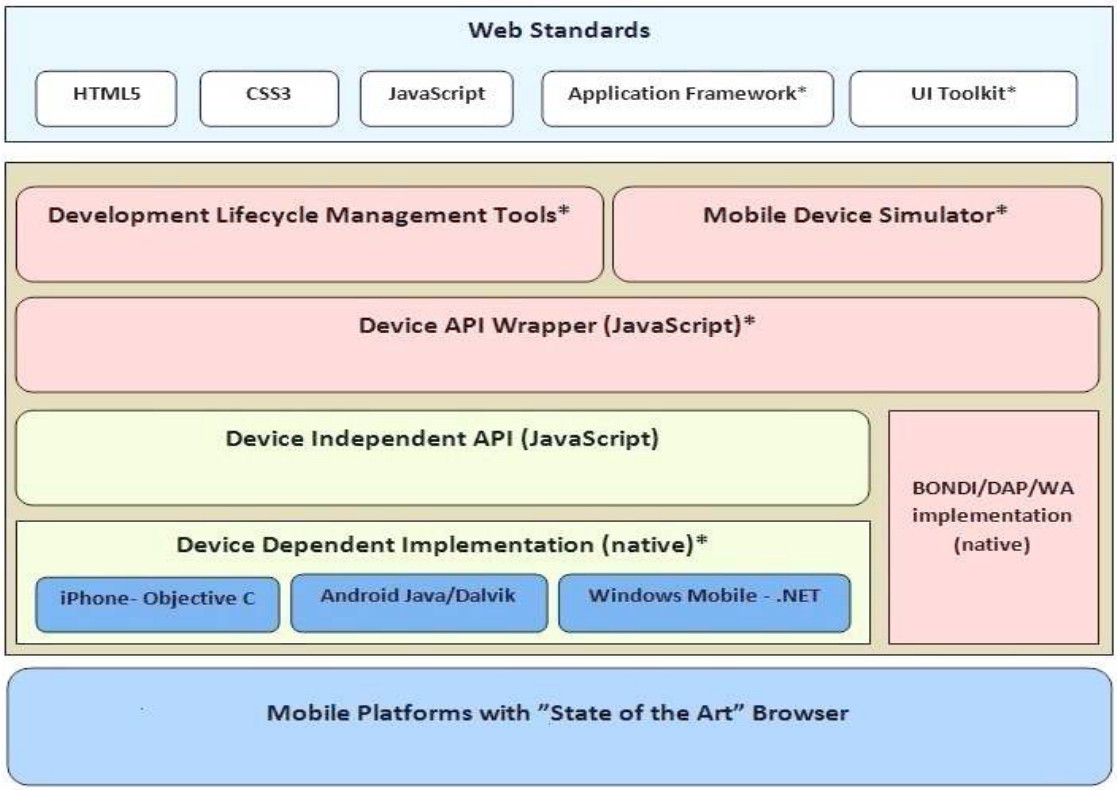

Figure 2-4 : Architecture of Hybrid Mobile Application Approach [11]

WebView is the native platform component having the ability to render web pages which connects JavaScript to native code and vice a versa. Device API wrapper wraps the WebView with a container. This container provides the access to APIs available on a device. In detail, any of the SDK's function can be bridged to JavaScript domain. Device independent APIs written in JavaScript are able to access native device components. A hybrid tool requires the SDK of that platform to be installed.

Apparently, hybrid approach enables the developer to utilize popular web development standards and allows extending the native device capability into the mobile web browser. Following are the benefits of Hybrid development approach [1].

1) Attaining benefits as native development - It is highly possible to achieve the benefits that native features have such as device integration, push notification, application market incorporation, offline access and synchronization. Better user experience is also a

possibility. Developers are not expected to have in-depth knowledge of any native platforms.

2) Combining web and native development - HTML, CSS and JavaScript, the dominant languages of web, offer extensive adoption and easy portability. These features greatly help in achieving cross-platform mobile applications. These languages brought both the web and native development techniques together.

3) Influence of HTML 5 in Hybrid Approach – Currently, due to the demand for cross-platform applications, HTML 5 is becoming a standard as it has more promising features for cross-platform development. It delivers better user interaction and capabilities with the web browser, rich support, geolocation, media playback, web application cache and web connected interactions.

The commitment to HTML 5 by popular smartphone companies is high. The strategic considerations of HTML 5 in the enterprise have been increasing rapidly. Active HTML 5 standardization efforts are storage, user interfaces, data semantics and media. Libraries (Cakejs, Highchart JS), frameworks (PhoneGap, Titanium) and tools (Eclipse, Mobl, Google Chrome) are highly exploiting HTML 5 to achieve cross-platform technique.

The main disadvantage of hybrid development could be slower browser performance. Due to an addition of another rendering layer by the browser, the performance may be slower than native applications. Obviously, it is difficult to implement apps with high definition graphics, 2D (two-Dimensional) animations and more specifically heavy weight applications. Conversely, these performance issues are being addressed through improvements in mobile device processors and mainly JavaScript engines [16]. There are many successful applications developed with the hybrid approach. Some of the successful applications in different mobile platform editions are ‘Facebook’, ‘Bank of America’, ‘Lotte card’ (South Korea), etc. Lotte card, one of the South Korea’s major credit card companies, developed a complex app that runs on different major mobile platforms through their browsers. Hybrid apps cannot guarantee unified user interfaces across platforms. Web layouts and designs do not operate well on mobile devices. Navigation is not easy and varies from difficult to unfeasible operation. User interfaces and application flow do not completely perform well on a range of device sophistication [10].

CHAPTER 3 : LITERATURE REVIEW

3.1 Porting The Application

When developing a mobile application across platforms, one essential element that by today's standard is hard to avoid is portability. In 1965, Dr. Morven Gentleman coined the term portability. According to Dr. Morven Gentleman, a program is defined to be portable if the cost of moving that program to a new environment is significantly less than would be the cost of implementing it afresh for that environment [17]. When an attempt is made to move an existing implementation on one platform to another platform, developers have to meet certain restrictions based on how the application behaves on the original platform. In mobile platforms, the restrictions impact on APIs, platform-specific components, programming languages, SDKs, etc. Interestingly, there are many tools available which may make the porting process easier but may affect the choice of the appropriate solution.

APIs are the set of methods that the operating system exposes to applications. APIs are one of the reasons for portability issues. The most general approach for cross-platform development is identifying a common denominator and then implementing an API that uses this common denominator and attaches to it. Each mobile platform has a different API structure and so it is hard to write a portable code. Creation of wrapper classes would make a difference. There are several techniques for developing cross-platform applications. Cross-compilation, interpretation and web technologies are some of the choices [18]. Portability has an important role in these techniques.

One of the suggested practices for cross-platform development is to develop templates with abstract classes, that various projects written in the same programming language can reuse [19]. Product-line architecture helps developers in mobile app development by facilitating software reuse. Commonalities and Variabilities are the important analysis in product-line architecture. Commonalities explain the attributes that recur across all members of the product family. Variabilities explain the attributes specific to some but not all members of the product family. It is hard to manually retarget mobile applications using product-line components, when accumulating reusable software components into

an application for a mobile platform. This is because of large number of mobile platforms, various SDKs and APIs, limited device capabilities, complex-product line constraints and the rapid development rate of new devices. Question that arise with respect to reuse is:

- How can programming language dependent code be reused between different mobile development platforms?

In Chapter 5, the above question is answered by developing a sample application in Android and BlackBerry to delineate the application logic reuse. Mixed language APIs is how most OS have handled this for many decades. Each language system has its own names, procedures, function calls, etc. Developers cannot connect the different program parts together, if the translation between different naming conventions is not properly implemented. Linker errors with unresolved symbols may occur. It would be difficult to reuse Android's code for iPhone application as they have different programming languages. If the programming languages are same, comparisons of programming code between different platforms could have been used to a larger extent.

Web languages help the code reuse in general across various platforms. Native wrapper classes can be created by writing a web application that can be converted into a stand-alone application for each platform.

3.2 Unified Design process by analysing iOS and Android Platforms

In order to achieve the cross-platform solution, an interpretive approach was carried out by Christian G. Acord and Corey C. Purphy [19]. They developed the approach based upon Model, View and Controller architecture and they have created a design process with a sample application that the developers can create a single set of design forms which may be implemented on either the Android or the iOS platform. In that journal [20], common design concepts such as screens and stacks, horizontal and composite navigation have been described based on MVC design pattern. To produce platform specific look-and-feel, the authors isolated the UI design phase by reducing the interaction between Models and View. On the one side, this idea acts as an advantage. Developers can see the similarity between the Android Activity class / XML layout and

iOS ViewController class / nib file combo. But on the other side, this may affect the controller's operation. Controllers have a major role in connecting the MVC pattern together and communicating between Model and View objects. For example, Buttons interface are associated to specific methods in the controllers and when a user clicks a button in an application, the equivalent method in controller reacts respectively. Therefore, the isolation of model and view may affect the design pattern in both platforms. In addition to these design issues, developers have to perform platform oriented language translations for each platform. Due to the language and implementation differences in platforms, there are alterations in the application at the screen level (View design pattern). Christian G. Acord and Corey C.Purphy [19] suggested that the design process will help the developers in reducing the design procedure when developing applications across platforms. Of course their design process explained identification of tasks, requirements, navigation structure and classes which are needed for the unified design. Also, a reasonable analysis of deploying MVC architecture pattern in both iOS and Android has been provided but the achievement of cross-platform compatibility is questionable. Few questions have not been answered or described in the research paper [19]:

1) How will the programming language differences affect the design process?
2) What are all the implementation differences?
3) How did the APIs (for accessing system functionality) differ considerably?

The authors could have used web application technique or hybrid approach for their analysis. Web and Hybrid approach seems better for unified design because of the JavaScript abstraction layer over native APIs. When the native wrapper is compiled through web languages and their resources, the interop layer connects the JavaScript APIs with the platform specific APIs. Titanium tool is capable of providing platform specific UIs as they provide huge native UI support for Android and iPhone platforms.

Christian G. Acord and Corey C. Purphy [19] described that their design process greatly reduces the inherent differences between the platforms. The suggestion of design patterns are independent but leave exclusions. While the iOS and Android platforms overlay in some UI requirements, there are some exceptions. One of the major exceptions is

application's navigation. Various navigation features such as horizontal, stack and composite navigation have been addressed in the research. Unified design with the navigation may not work with applications demanding intense UI controls. Android has a physical back button and the bottom of the screen is filled with the actual content and overlaid with the hardware menu. In iOS, menus and navigation are separated in two parts. Navigation components (back, ok, cancel and title) are at the top of the screen. To switch between views, tab bar is at the bottom. This is just one of the differences in the UI design level. This research paper [20] has not scrutinized how the design process will be applicable to other mobile platforms and what are the other design implications and issues may affect the portability.

3.3 Cross-compiling Android Application to iPhone

Cross-compilation helps the developers to deploy their application on multiple platforms. A cross compiler is a compiler that operates on one platform to generate code for a second platform. Since compilation is never done on the mobile device itself, all mobile development uses cross-compilation. Cross compiler tools are used to generate executable tasks for multiple platforms. But porting the applications to disparate platforms incurs high overhead due to their difference in programming models.

XMLVM, a byte code level cross-compiler has been introduced to overcome the complexities of different programming languages used by various smartphones [21]. In order to reduce the porting effort, XMLVM tool cross-compiles Android applications to iPhone and Palm Pre. This tool demands complex porting efforts. It is mainly because the technique involves certain intricate steps. First, the cross-compiler does Java to Objective C conversion and converts from Java byte code to an intermediate XML form. Then the XML form is passed through an XSL sheet to produce Objective C. Of course issues occur in a way of providing access. Objective C library has to deliver access to every iPhone SDK function and components that the developers want and so far, many components were not mapped or implemented. In technical terms, 'System.arraycopy()' is not implemented yet. This function is under 'java.lang.System' library used to copy one array to another. The other issue could be array references which are never null (developers cannot use a null check on 'int []').

The byte code instructions which are generated by the Java compiler are represented through suitable XML (Extensible Markup Language) tags [21]. A. Puder and I. Yoon [21] suggest that XML favors developers by allowing them to create their own set of tags at their own pace. This fact is agreeable. There are also other factors to be considered when developing applications using XML [22]. Technically, XML needs an application processing system and this could be a drawback. There are no browsers which can read XML but not HTML. API bridging is implemented by mapping compatibility libraries of respective smartphones. XMLVM tool does not offer complete operation for the API mapping. The tool nearly bridges 5% of the API.

3.4 Methods in Cross-compilation

Cross-compilation allows developers to use two different approaches. One is statistically linking wrapper libraries, which may result in increase of application size. The other is to use Application Programming Interface mapping or API bridging. The cross-compilation technique has the advantage of running an application in native device itself. For rapid prototyping change across the business logic and the user interface and for the development of cross-platform applications, a code generation tool 'Kamili' was implemented [23]. The platform choices for this research are iOS and Windows Phone 7. This research analysis presents an interesting concept of an automatic user interface generation which is carried out through an architectural pattern called Naked objects. 'Naked objects' is equivalent to the Model-View-Controller pattern and the main goal is that it separates the module of the program. Unified design process [20] which is discussed above has an idea of separating model and view layer but the authors Christian G. Acord and Corey C. Purphy have only used MVC pattern in a design perspective for their analysis. Unlike the unified design process, the authors Lars Maaloe and Martin Wiboe [23] compared the data management of MVC pattern and the naked object pattern. By using Kamili tool, developers can develop applications in an object oriented style and deploy the application objects by having a code-generator output platform-specific code. This tool benefits developers as it is a rapid development process but certain UI objects need to be supported. Components such as windows, tabs, date-pickers etc. could not be supported with this tool. Moreover, the authors Lars Maaloe and Martin Wiboe described their tool with simple prototypes. It would have been interesting if they have explained

UI considerations and device specific functionalities in detail. The unsupported features can be strengthened by extending the naked objects implementation. One of the better ways to enhance this tool is by providing a comfortable abstraction across platforms. While compiling for a specific platform, UI markup would be translated to platform-specific UI code. The tool 'Kamili' requires significant effort because every single-functionality should be recognized and wrapped into a native abstraction.

3.5 Complexities in Cross-compilation

Cross-compilation is complex because it affects compiler designs and low-level optimization [18]. Low-level optimizations involve in building objects that are not visible at the source level. Low-level optimization is important because they are away from the reach of the user. It also helps in producing substantial performance improvements and supports platform specific elements. MonoTouch and Mono are the good examples of cross-compilation tools. But for Android, compilation is not supported fully due to platform specific UI. In MonoTouch, the UI is developed in Apple's own software which in turn creates a XML file (.xib format). In Mono for Android, the UI is created in a XML file (.axml format) which prevents the same Mono application from running on both platforms. The cross-compilation method provides best performance, if accomplished in a correct way. The outcome of cross-compilation is a real native application. Native applications are faster and they are defined from their own proprietary APIs. API is the set of techniques which the operating system represents to applications. APIs hugely differ across platforms and for Android and iPhone, each platform has a complete set of API levels. Android and iPhone's Cocoa Touch contains more than thousands of methods and demands immense effort in research and implementing complete API bridging. A. Puder and I. Yoon [24] stated that their tool did not provide completeness for API mapping. This may be because of isolating dependencies in APIs. Components in a mobile platform require lots of conversion to work on other platforms. For example, functionalities that are usually varied between platforms could be file handling, media management, etc. XLMVM needs more enhancements (API bridging, strong adaptation layer) and it needs a larger community to start contributing to the framework. Therefore, like developing native application and replicating the same in different platforms, cross-

compilation consumes time. Cross-compilation technique also requires the developers to put in significant amount of work to deploy an application across platforms.

With the introduction of new versions of operating systems once every 4 months, the design patterns may not remain unchanged. If developers use the above approaches (unified design process and Kamili tool) [20, 23], they have to concentrate on certain patterns that consume time. As the unified design process [20] emphasizes the benefits of consistent design across platforms, it may not be beneficial to a larger extent. As the approach developed for mobile applications, the MVC pattern has to be applied to each screen in an application. Obviously, developers need to recognize and design specific components required for the application. Then the developers have to concentrate on the particulars of each individual class or component. Moreover, programming language differences may not help to attain similar functionality between platforms.

3.6 Summary

Portability has a variety of solutions and when moving a program to a new environment it is important to choose a method or technique that fits the solution and requirement. Cross-compilation of Android applications to iPhone applications were carried out by two methods [23, 24]. From these interesting techniques, cross-compiled applications seem as the finest solution in some cases, but not suitable for applications which demand extensive device-based functionality and consistent UIs. Specific functions that are provided by Android, may not be available in iOS. This could be the one of the reasons for features missing in the cross compilers today, especially the tool XMLVM (only 5% of APIs has been integrated). The range of technical approaches which are discussed above suggests that these techniques fundamentally adding innovative ways to improve cross-platform application development. It was found that these native cross-compilation techniques [20, 23, 24] help to deploy portable application across platforms but not to a greater extent.

One of the possibilities of creating applications across platforms is the hybrid application development approach. Most hybrid tools use web languages (HTML and JavaScript) to develop multi-platform applications. The reason could be the tools are having their own modules/libraries which try to bridge native platform libraries with one to one operation.

Hybrid tools have the facility to wrap web applications into a native application, meaning developers can develop an app with web functionality that can be converted to and delivered as a native application in addition to the web form. The cross-compilation techniques which are discussed above have not discussed the user interface functions in detail. UIs are the one of the vital factors in a mobile application and it is seriously considered in case of multi-platform application. This is one of the reasons influenced to investigate on providing consistent user interfaces on different mobile platforms. Following chapters will clearly present the idea on attaining unified user interfaces.

CHAPTER 4 : METHODOLOGY

There are two approaches carried out in this thesis work. They are cross-compilation of two native mobile platforms and hybrid mobile app development method. By understanding the significance of native mobile application development, achieving cross-platform technique is approached natively i.e. by trying to cross-compile two different native platforms.

Android and BlackBerry platforms have been selected for cross-compilation. The reason for choosing these two platforms is the set of common factors in both the platforms. Already, Android applications are cross-compiled to iPhone and Palm pre [21] and therefore, BlackBerry platform has been chosen. BlackBerry and Android use Java as their development language and so, one might see similarities in user interface functionalities and APIs. Due to the identical programming language (Java), developers can reduce the development effort by reusing the application logic. The differences would be in interactive UI, APIs, Operating System and application framework. Therefore, through cross-compilation technique, it is possible to cross-compile two platforms' APIs. A practical exploration would be very useful for the cross-compilation analysis.

A small word game application is developed first in Android and the same application is replicated later in BlackBerry. This practical investigation helps to identify commonalities of user interfaces in two platforms, identical APIs, chances for API mapping and linking compatibility libraries of two respective platforms. The application logic from Android application is used with minor changes in BlackBerry application. Minor changes are due to platform oriented API components. Concentration should be directed towards identifying suitable user interfaces and other functionalities in BlackBerry platform. Unlike Android, BlackBerry platform is not an open source platform and layers in BlackBerry are limited to perform cross-compilation. The idea is to implement an adaptation layer on top of BlackBerry layer and the interface between BlackBerry layer and adaptation layer seems unlikely. Due to this issue, cross-compilation is highly difficult to map all the API features. Immense determination is required for developers and looking for an alternative seemed a good idea.

Other Choices in the Mobile App Development

Mobile web application method looked viable. Even though web applications provide satisfactory portability, implementing user interfaces across platforms remain an issue. UI widgets which perfectly fit in one mobile platform should also be acceptable in other platforms. Moreover, there is no comprehensive support for accessing native device features and web applications cannot be integrated in app stores. Considering the issues in Mobile web development technique, another alternative is expected.

'Proprietary middleware and clients', a pre-built service, is another way of developing mobile apps [16]. There are some free online tools which allow developers to create mobile applications through a web interface by choosing pre-built modules. There is no need of coding and this method is simpler than other mobile app development methodologies. A specific pre-built service tool has categories that have different pre-built modules. Apparently, this methodology has limitations in providing advanced features. Although this method allows creating applications with ease, the weakness is that the functionality and design are limited.

After an analysis of cross-compilation and web-based mobile app technique, it is found that the limitations in those techniques do not help to attain consistent UIs across platforms. Therefore, practical investigation has been made under hybrid approach. Hybrid tools such as PhoneGap and Titanium aid the developers in cross-platform development. However there remain differences in the functionality, appearance and operation of applications. Hence, there is a need to look at mobile application development at its grass-root level. PhoneGap and Titanium fall under Hybrid application development techniques. Hybrid application technique is chosen as an alternative solution and PhoneGap (cross-platform application framework) is used for practical analysis. PhoneGap is now called as Apache Cordova, an open-source mobile development framework. It enables developers to develop mobile applications using HTML5, JavaScript and CSS3. The resulting application is hybrid (neither truly native nor purely web based). To demonstrate the issues more technically, a native application (Currency Conversion app) is developed in Android platform. The application is later replicated in Windows Phone 7 platform. This comparison is implemented to recognize the variances of UIs and its functionalities in two different platforms. Moreover, developer's

perspective in mobile app development is showcased. This research's main aim is to facilitate the application development across platforms for developers by attaining unified user interfaces across platforms, thereby achieving cross-platform app development features. Detailed practical implementation is discussed in the next chapter.

Sample Application for Practical Investigation

This research work is carried out based on developer's standpoint. To demonstrate the UI and functionality differences, a sample application (Currency Conversion) has been developed. This application covered important aspects required for this research work such as UI features, web content and native device feature access. The app uses the web for currency feeds i.e. to retrieve the currency rates from the web (Google finance service). Google Calculator API is used for the conversion and finance elements references are used to retrieve the currency rates. "Convert" function has been created and Google calculator URL is used to retrieve the currency rates. Using "Geolocation" provides the location information for the device. Most Smartphones are GPS-enabled and are able to receive signal from several satellites in space that help triangulate the exact position of an individual. The base currency is set based upon user's current location. For example, if a user is in Australia, the base currency is automatically set to Australian Dollar. This feature is added to delineate that the application can access a native device feature.

Developing the Application Natively using Android and Windows Phone 7

In the native development phase, Android and Windows Phone 7 are used as target platforms. Android development tools such as Android SDK, ADT plug-ins, Dalvik virtual machine and resource editors are used for the application development. ADT is a set of components which extend the Eclipse Integrated Development Environment and allow certain functionalities such as create, compile, debug and deploy Android applications from the Eclipse IDE. Resource editors are mainly used for creating specific UIs required in the Currency Conversion application. Developers can also manually write code for designing user interfaces. To make the development easier, resource files (e.g. layout files) are involved in the development. These files technically helped to switch between the XML representation of the file and a richer user interface through tabs on the bottom of the editor. Therefore, XML layouts are used as a front end to design the UIs.

For accessing the location of a user, “android.webkit.GeolocationPermissions” is used to manage permissions for the WebView’s geolocation JavaScript API. Callback parameter “ValueCallback” is used to get the feedback of the request. The result will be invoked asynchrounously with a set of strings containing the origins for which Geolocation permissions are stored.

After developing the application in Android platform, the application is replicated in Windows phone 7. The IDE used for this platform is Microsoft Visual Studio 2010 express. The required resources for Windows Phone app development are Windows Phone 7.1 software development kit, Silverlight tool and Microsoft Internet Information Services (IIS) 7.0. In Visual Studio Express IDE, necessary features such as unit testing and other UI tools are not available. So, another Silver tool kit version is used to support the development. Other than a usual default emulator attached with SDK toolkit, Windows Phone 7 series emulator is employed specifically. This is integrated with Visual Studio Express tightly so a developer can deploy the app in action instantly (less loading time) and debug it with any other VS project. There are two distinct approaches can be used to create any application in Windows Phone. They are Silverlight for Windows Phone and XNA framework. Additonally, Silverlight toolkit 5 version is utilized as it has extended platform technologies from the web, desktop and console to the phone giving developers a broader application development experience. Silverlight approach has chosen to develop a sample application as the development closely resemble Android’s development pattern. A Silverlight application combines XAML to design the user interface and code written in a .NET framework language (C#) to control the application’s logic. For native device access (Geolocation), “GeoCoordinateWatcher” class which reveals the location of a user has been used. This calss mainly represents a locatin expressed as a geographic coordinate.

Developing the Application with Hybrid Technologies

Problems faced in native application development have been resolved in hybrid application development by demonstrating the practical investigation.

After implementing the Currency Conversion application natively in Android and Windows Phone 7 platforms, the same application is implemented in those mobile platforms using hybrid approach. User interfaces and application logic are developed

using jQuery Mobile. PhoneGap is used for accessing native device features and mainly porting the application in different mobile platforms.

Software development kits of Android and Windows Phone are already installed when the application was first developed natively. Hybrid application framework alone is not able to achieve uniformity in user interfaces. PhoneGap tool is new and it lacks in pre-built UI widgets, transitions, standard controls, etc. Due to the shortage of interactive UI widgets, developers probably need to spend more time to focus on user interfaces. Cross-platform UI framework provides highly interactive widgets and functionalities across platforms. Tools such as Sencha Touch, IQ Touch or similar tools with pre-built UI elements provide great support for the app development. Therefore, cross-platform application framework can be combined with cross-platform UI framework (jQuery Mobile) to produce effective cross-platform applications. jQuery Mobile is the extension or additional plugin of jQuery framework. It is built on the jQuery JavaScript library. jQuery mobile supports many smartphones by providing restyled UI controls and animations and claims a small file size. It offers same user experience regardless of device. This method seems to be a possible solution and it has been implemented by developing currency conversion application in Android and Windows Phone 7 (two different platforms in hybrid approach).

PhoneGap (now called as Cordova) is downloaded and the contents of PhoneGap are extracted. A path has been set to connect Android SDK platform-tools and tools directory of PhoneGap. Eclipse has been used as an IDE for Android platform. The contents (libraries) of PhoneGap and jQuery Mobile should be added. A folder named “www” has been created and components related to html/css/js resources are included. A html file called “index.html” is created and the entire code is being written in that html file. In the IDE of Android platform the contents and libraries of PhoneGap (Cross-platform application framework) and jQuery Mobile (Cross-platform UI framework) are added. As both of the hybrid technologies share the common scripting languages, integrating the components of two frameworks are highly possible. The location tracking functionality is implemented using “W3C Geolocation API” offered by PhoneGap. The method “geolocation.getCurrentPosition” (asynchronous function) used for returning device’s current position when a change in position has been detected. The JavaScript APIs

associated with the PhoneGap framework connects to the JavaScript APIs in the native device. When the device has retrieved a new location, the “geolocationSuccess” callback is invoked. Incase of any error, “geolocationError”callback is invoked and both the success and error callback functions are invoked using “positionError” object. The reason for choosing jQuery Mobile is that it is compatible with all major mobile platforms as well as all major desktop browsers. In the body of html file (index.html), there are attributes such as header, footer and contents. The features gave a great advantage in structuring the application. No specific pattern or order is maintained during development. The code can be arranged according to the developer’s ease. More pages have been created and for each page, separate ids are used (e.g. href=”#firstpage”, href=”#secondpage”). The html file contained more than one page and it is necessary to load one file including multiple pages. Therefore, one page can connect to another page within the same file.

Following described is the pictorial representation of proposed architecture. There are two phases involved in the development.

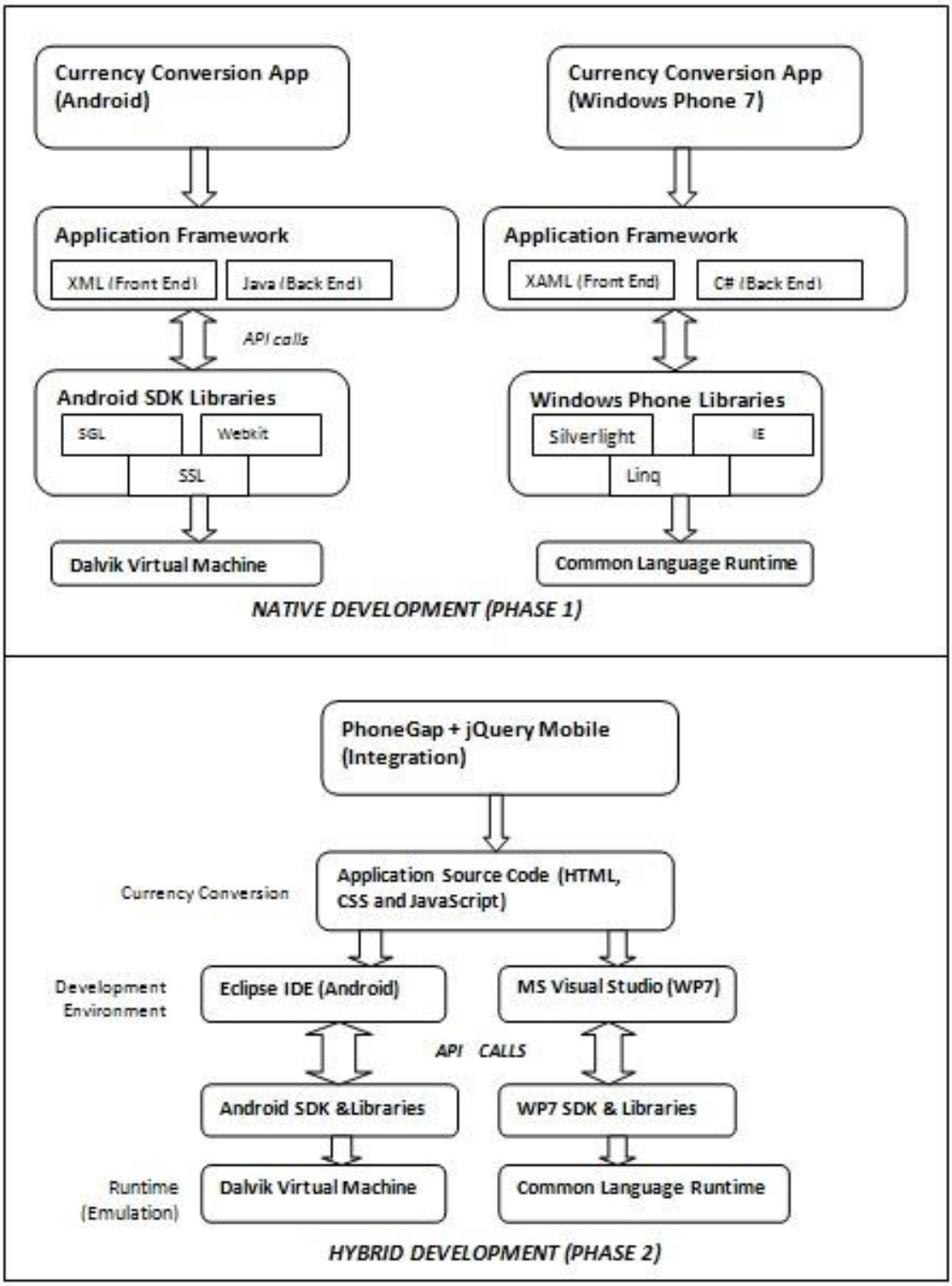

Figure 4-1 : Architectural Overview of Proposed Approach

CHAPTER 5 : IMPLEMENTATION

5.1 Cross-compiling Android to BlackBerry Applications

Native applications are faster than web based applications as the native applications are defined from their own exclusive APIs. Android application is written in Java and avails specific APIs of Android platform. A cross-compiler tool can be used to bridge two platforms' APIs. To surmount the heterogeneity of different programming models used by different smartphones, a cross-compiler could be used. Cross-compilation can be achieved by analyzing equivalent classes and methods in both APIs of Android and Blackberry and performing a workable mapping between the APIs. Compatibility libraries map the APIs between smartphones [24]. To perform this mapping, developers should perform an in-depth analysis and familiarize themselves with one platform. Once they are familiar with API mapping in Android platform, Android apps can then be cross-compiled to other smartphone platforms. Android platform is designed to adapt to multiple mobile systems and is not confined only to smartphones. Android has the ability of adapting to other devices with ease as the Android APIs allow developers to explore the device's abilities. Moreover, Android is an open source platform under permissive license and has well documented SDKs suitable for developers ranging from novice to experts.

Application developers require their applications to be available on many platforms and would also want to increase the prospective of application distribution. As Android is an open source platform, the possibilities for detailed investigation and technical modifications are much higher when compared to other platforms.

Android's architecture is comprised of five layers namely (from the bottom level) OS layer (Linux kernel), Compatible Libraries, DalVik Virtual Machine, Application Framework and Applications [25]. Cross-compiling deals with mapping Application Programming Interfaces between the two platforms. The idea here is to link together the Android compatible libraries and Blackberry compatible libraries. Developers need to be aware of the skill set required for Android platform and only one code base has to be maintained.

5.2 Native API Bridging

An application which uses API 1 (Android) should make use of API 2 (BlackBerry) and that is the aim of mapping APIs. In this case, the idea is deploying application across platforms using native application development. Therefore, an Android application that uses the Android API should use the BlackBerry API instead. There are two ways to accomplish this:

First method is the alteration of application at API level i.e. the Android API should make use of BlackBerry API. More specifically, rather than instantiating the class *android.widget.TextView* (text field widget of Android), the class *device.api.ui.component.TextField* (text field widget of BlackBerry) should be instantiated. Developers need to perform a profound analysis of the method by which the application uses the BlackBerry API, instead of using Android API. Changing the Android app to directly make use of a different API is a complex task. Another way of accomplishing Cross-compilation is by linking compatible libraries of Android and BlackBerry.

User interface is one of the most important aspects in building an application. It defines the interaction between user and application. There are some similarities between Android and BlackBerry platform as the applications are developed using Java and Eclipse plug-ins [6]. It is highly difficult to expect the same or even identical user-interface hypothesises and APIs. For example, the push notification interface in Android allows users to clearly see the event notifications and details about the notifications which are displayed on the top. In BlackBerry, number of notifications is displayed but users cannot read the details about the events. In this issue, web platform is reliable but the number of SDK controls is limited. User experience is strongly motivated by user interfaces. It impacts more when any applications are developed across platforms. There are features such as screen resolution, pixel density, color depth, touchscreen, trackball and camera that the developers should be keenly concerned about while developing for multiple platforms. These features should be deemed important by the developers as they determine fundamental operations in any device. The combined effect of these features hugely impacts the appearance of an application. Even if one particular combination fails, user interaction is badly affected.

To understand the similarities between the Application Programming Interfaces in Android and BlackBerry platform, a word game application named Flames with a simple interface is created. The 'play' button directs the user to the next page. Users can enter two names (male and female) and the two given names are compared. Identical letters are removed and non-identical letters are counted. After some internal calculations, application displays the relationship status of given two names. 'What is Flames?' is the description of the game. An Android version of the game developed using Android emulator is shown in Figure 5-1.

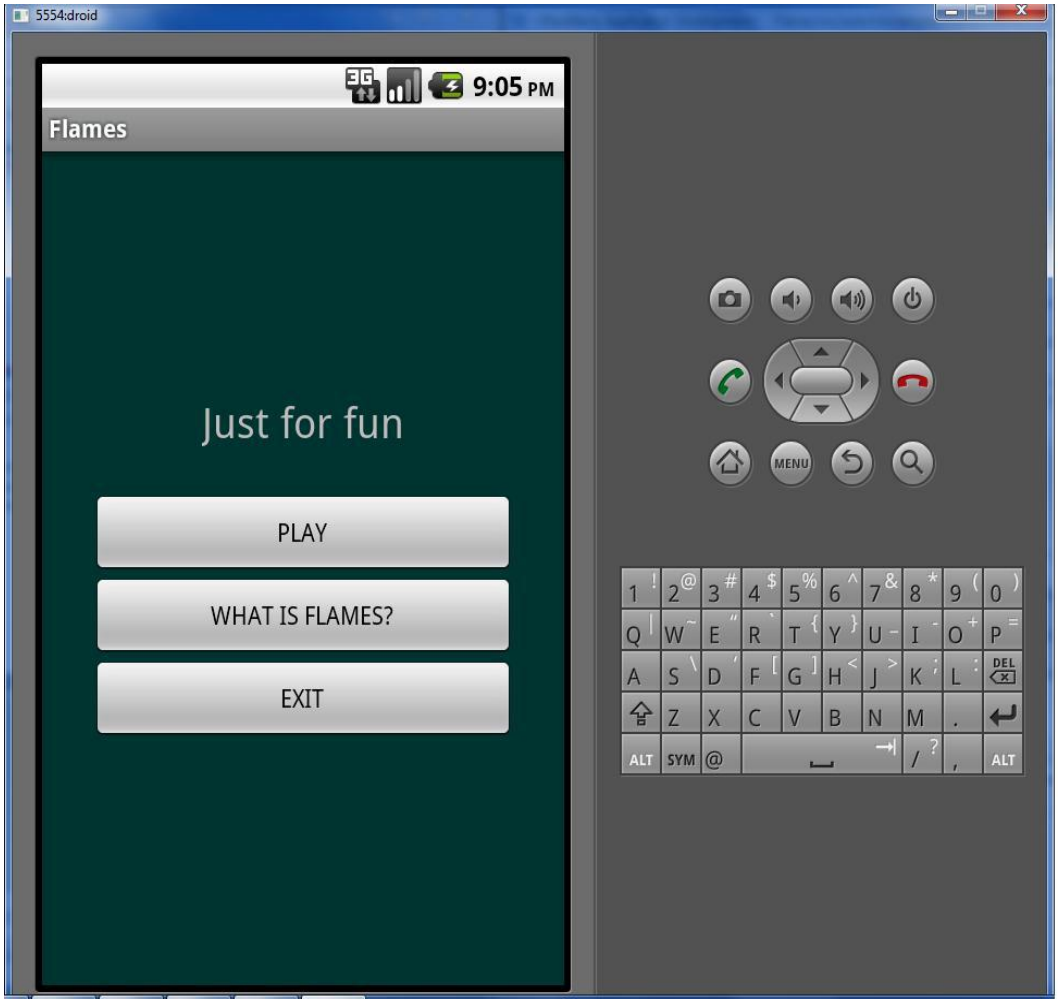

Figure 5-1 : Opening screen of Flames Game in Android

A similar application created for the Blackberry platform using Blackberry emulator with a similar interface is shown in Figure 5-2.

Figure 5-2 : Opening Screen of Flames Game in BlackBerry

5.2.1 Commonalities of the User Interfaces in Android and BlackBerry Platforms

Blackberry uses 'UI application' class and Android uses 'Activity' class. As both Android and Blackberry uses Java, classes are extended from 'Activity' and 'UI application'.

An activity is a core component of the Android Platform. Each activity represents a task that an application can accomplish [4]. Activity involves displaying a layout, playing music and launching several events. A UI application is a core component of the Blackberry Platform and it upholds a stack of Screen objects. It involves in displaying a screen (same as Android) and only the screen on the top of the stack receives input events.

'Linear Layout' is an activity in Android that displays screen components in a linear direction, either horizontally or vertically. In most Android applications, UIs are defined using formatted XML (Extensible Markup Language) files called layouts [12]. There are two ways to configure layout resources. One is by using XML and next the user can even create interfaces without XML.

There are more commonalities between the interfaces which are required for creating Buttons, Dialog windows, Label fields, etc. in both the platforms. Following table illustrates the similarities between the interfaces in both the platforms.

ANDROID (Interfaces)	**BLACKBERRY (Interfaces)**
TextView flames;	*LabelField flames;*
Button play, aboutflames, exit;	*ButtonField play, aboutflames, exit;*
Dialog about;	*Dialog about;*

There are two interfaces for triggering buttons namely 'FieldChangeListener' (Blackberry) and 'OnClickListener' (Android). These interfaces are responsible for click events. Coding in Android can also be done using XML. To show the similarities between Android and Blackberry, XML for Android is not used above.

5.2.2 Reuse of Application Logic Code

Similarities in the user interfaces of both Android and BlackBerry are described in the next section. Application logic is used to describe the functional algorithm that manages

information exchange between application and a user interface. Application logic looked almost same in both the platforms and after developing in Android, the application logic is reused with minor changes in the BlackBerry platform. Specifically, 'calc ()' function used in the application receives two different names from the user and compares the given names. This is implemented by comparing each letter in the given two names and the function removes the identical letters in both the names (input). Finally, the output would be the remainder of letters which are not identical in both the names (inputs).

Since the same programming language is used for development in both the platforms, functions and procedures with minor changes have been reused in another platform. Therefore, for applications involving complex UIs and intensive operations, developers can create internal abstractions so that definite portions of their program can be reused. They can also create custom libraries for their own use.

5.3 Unified Approach for Cross-compilation

Both Android and BlackBerry environments are frameworks layered on top of Linux Kernel. Layers of Android have much richer functionality but also make it more complex to deliver the functionality. Despite the fact that both Android and BlackBerry use the same programming language (Java) for application development, their applications are difficult to port because of differences in hardware, Software Development Kits and Application Programming Interfaces. The solution is to provide an adaptation layer on top of the native framework so that portable applications can run on the other platform as well as on its native platform. The adaptation layer systematizes platform specific code which implements the abstract interface by using a specific programming technique. Adaptation layer should deliver support for proficiently mapping the entities between the abstract interface and native APIs [26]. Applications can be modified to operating conditions by the interference of an adaptation layer that is flawlessly inserted between the application and the operating system. Specifically, providing a thick layer on the BlackBerry platform to support apps designed to operate on Android.

If Android applications are cross-compiled to BlackBerry applications, the insertion between the adaptation layer and BlackBerry layer seems nearly impossible because the native framework of Blackberry is very limited. Profound analysis on mobile platforms is

required for mapping a considerable amount of APIs between the two platforms. Moreover, developers have to look up for equivalent classes and methods for cross-compilation and not all the APIs of one platform (Android) match with another platform (BlackBerry). Difference in APIs would lead to complexities when porting applications across platforms. Moreover, BlackBerry platform is not an open source platform like Android. License and other company policies act as barriers in mobile application development for developers. Therefore, an alternative solution to address cross-platform application development is certainly required.

5.4 Hybrid Application Development Approach as an alternative solution

Hybrid development technique is chosen because both the hybrid and native approaches have comparable performances and competences. The significant advantage of hybrid technique is that it is designed in a way to support the development of mobile applications that are written as embedded dynamic websites. They utilize native phone components and serve as a reliable and simple development approach for developers.

5.5 Comparison of Application developed in Android and Windows Phone

In order to identify the differences in platforms, user interfaces and functionalities, currency conversion application is developed in Android and Windows Phone 7. This practical analysis helped to recognize issues faced during development. UI is defined using XML format and during runtime, the UI is loaded by 'onCreate()' event handler in Activity class. During each compilation, every element in the XML file is compiled into its equivalent Android GUI class, with attributes represented by methods. The Android system then creates the UI of the Activity when it is loaded.

Windows Phone 7 applications are based on a Silverlight page model where the screen navigation is forwarded through different pages of content via links and backward using the Back button. The Back button goes back or exits the application (automatically remains on the back stack). In some applications in Windows Phone, if a user navigates to 5^{th} page of app and wants to close/exit the application, then he/she needs to navigate back to first page and then exit the application. This operation allows users to perform more actions. In Android, users can exit from any application, no matter which page the user is in, exiting the application from the page is possible. But in Windows Phone, the

page navigation model allows developers to explicitly add links to other pages within the application. This is similar to the operation of web browser displaying and navigating web page history.

5.5.1 Why Android and Windows Phone 7? – Platform choices

Developers have to put in substantial efforts while developing the same application in two different platforms. Android and BlackBerry could have been chosen to develop this application but both platforms use Java as their programming language even though both platforms differ in operating system and application framework. Earlier, a small word game was developed in both platforms. There were some differences in APIs and application structure but the application logic code of Android was reused (almost) in BlackBerry. Developers observed that the BlackBerry development is time consuming and developing advanced UIs are not simple [27]. This may be the reason for the less number of BlackBerry applications in the application market. Developers cannot make use of enhancements made to Java such as 'java.util.ArrayList' and 'java.util.HashMap' which are not available. BlackBerry API has less advanced components than Android or Windows Phone 7. Therefore with the available resources in BlackBerry, users have to accept an interface that looks less polished than Android or any other smartphone platforms [27]. Microsoft allows developers to explore more in Windows Phone 7 and it has well enhanced SDK documentation.

Things are not as pretty when it comes to two completely different mobile platforms. Both Android and Windows Phone 7 use different operating systems, application framework, components/objects and mainly programming languages. While developing for Android and BlackBerry, detailed analysis was performed about those platforms' design architecture, Software Development Kits and APIs. When it comes to Android and Win7 Phone, they are different in many aspects including the programming language.

5.5.2 Application Framework Differences

Application Framework plays a significant role in each mobile platform. It comprises of many essential components. The differences between the application frameworks of these two mobile platforms are critical to cross-platform application development. Table 5.1 shows the differences in application frameworks.

Table 5-1 Differences between Android and Windows Phone 7 feature

	ELEMENTS	ANDROID	WINDOWS PHONE
1	Application Frameworks	Android Framework	Microsoft Silverlight and XNA
2	Programming languages	Java	.NET languages (C# and Visual Basic .NET)[MS lists dozens, some claim 90+]
3	Software Development Kit	Android SDK available in Windows, Linux and Mac OS X	Windows Phone SDK available on Windows 7
4	Integrated Development Environment	Eclipse (with ADT plug-in)	Visual Studio 2010 Express
5	User Interface Definition	Extensible Markup Language with Widgets and Layouts	Extensible Application Markup Language
6	User Interface Event Mechanism	Event listeners in View classes	Event handlers in Common Language Runtime

Android uses an XML based layout to describe the user interface. Developer should concentrate on the order of arranging user interface elements as the UI description is hierarchically organized. In this application, every component (buttons, text boxes, text views) is arranged hierarchically in a layout type. During runtime the layout is loaded by the Android class that it was built for. Windows Phone platform uses XAML (Extensible Application Markup Language) for user interface description and .NET languages (C# and VB .NET) for application logic. Development pattern looks quite similar to Android as the app development's concentration is on XAML and .NET concurrently. Android uses an XML based layout to describe the user interface. Developer should concentrate on the order of arranging user interface elements as the UI description is hierarchically organized. In this application, every component (buttons, text boxes, text views) is arranged hierarchically in a layout type. During runtime the layout is loaded by the

Android class that it was built for. Windows phone platform uses XAML (Extensible Application Markup Language) for user interface description and .NET languages (C# and VB .NET) for application logic.

5.5.3 Operating System differences

At the core (bottom layer) of Android device software is the Linux kernel. The drivers and the hardware abstraction layer modules are all Linux-based. On top of the Linux Kernel are the Dalvik Virtual machine and its runtime environment along with native libraries. Application Framework comprises the major portion of the application and it consists of Java services, APIs and libraries. On top of these component layers is the application [1]. Two application development practices can be found on Android: Java applications that run entirely in the Dalvik virtual machine and Java applications that use Java Native Invocation to call into native libraries on the system.

The operating system of Windows Phone kernel is Windows CE 6 or 7. Microsoft offers a special SDK so that the phone makers can develop native COM DLLs with a limited set of native Windows CE (Windows Embedded Compact) APIs and use them in their applications. But developers do not have access to special SDKs. In Windows Phone, developers are not allowed to write native code; only managed code is allowed. Only subsets of the general Silverlight APIs and XNA APIs are supported on Windows Phone[1]. Phone makers can insert native code into the system and use it in their applications by using COM (Component Object Model).

5.6 Developer's role in Application Development

After identifying different mobile platforms, developer's role is very crucial in app development. Developers/designers should completely understand the constraints. Developers have been trained to think that more features equate to better applications, but on mobile devices, that is simply not true. As Android and Windows Phone are completely different in their architecture, developers familiar with Android development take more time to understand the Windows Phone 7 platform development. While developing apps for a platform, the developers become familiar with a set of functionalities and user interface objects that are native to the platform. When the developers try to develop the same app on a different platform, it is quite natural of the

developers to expect similar, if not the same set of functionalities and user interface objects on that platform. Mostly, this is not the case. For example, Android has an “Exit” function but Windows Phone uses physical “Back” button to exit any application. Developers would get stuck a little in these cases.

Ultimately, developers need to know each platform’s programming language and application interfaces in order to deliver an effective application. Incomplete applications or applications with bugs affect user’s experience greatly.

Implementing a mobile application begins with conception and design. A sample application has been developed for this research work’s practical investigation. This application performs currency conversion, a popular functionality among many mobile phone users. Every mobile platform has its own perspective of user experience. Developing the same application in two different mobile platforms allows identifying the differences in user interfaces and its functionalities. For a concrete analysis, the application has been first developed on Android and then on Windows Phone 7.

Opening Screen is an Activity and it should be included in Android Manifest XML from which a XML class is compiled into Android GUI class. An Android version of the application is shown in Figure 5-3.

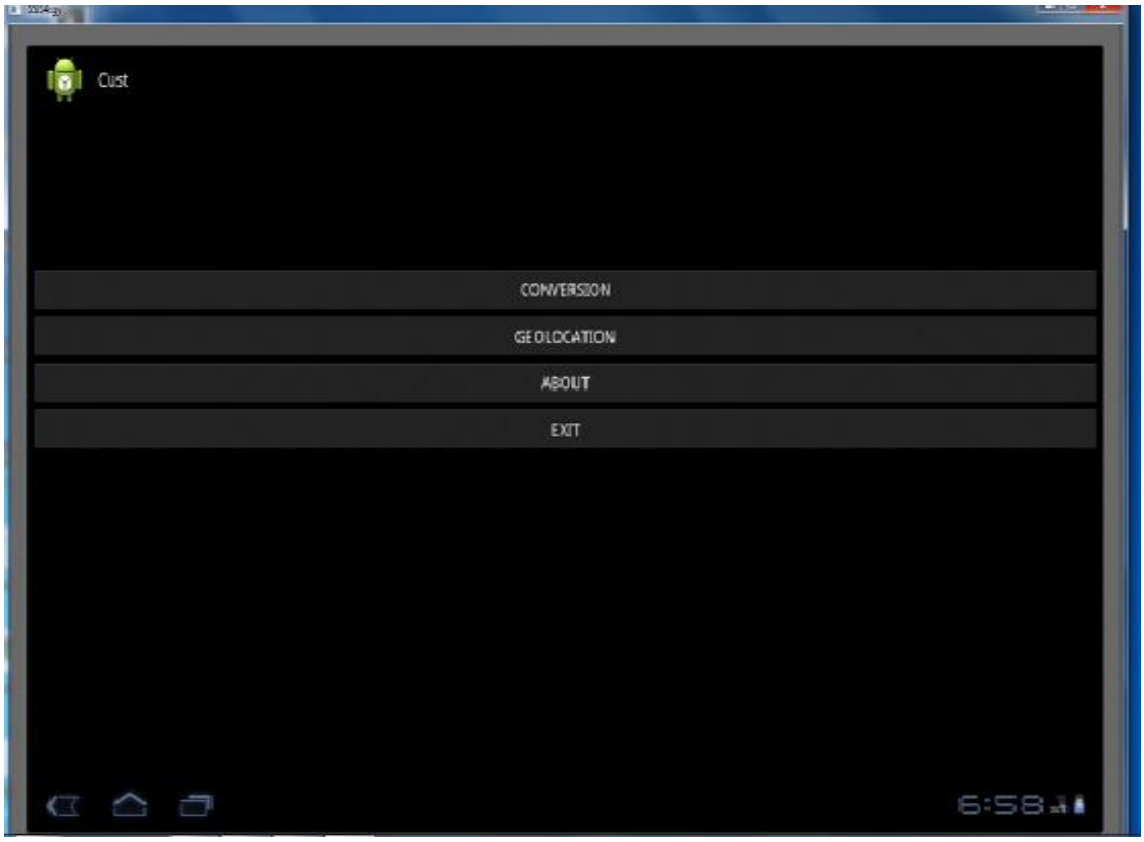

Figure 5-3 : Opening Screen (Android)

In Windows Phone, the core elements of an application include a top-level container control called a frame that displays the pages. Only one frame is allowed per application, but there is no limit to the number of pages. Windows Phone 7 version of the app is shown in Figure 5-4.

Figure 5-4 : Opening Screen (Windows Phone)

It is clear from Figure 5-4 that "Geolocation" activity and "exit" activity are not present in Windows Phone. Windows Phone has sliding motion feature and if the user slides the current screen to the left or touches on "GE" the second screen will appear.

Figure 5-5 shows the "Geolocation" Screen of Android.

Figure 5-5 : Geolocation Screen in Android

User can get back to the main screen by sliding to the right from "Geolocation" screen. Windows Phone has this design pattern to facilitate navigation. Therefore, there is no need for "Main Menu" button which is present in Android. Android has a "Geolocation" button for navigation. Figure 5-6 shows the geolocation screen of Windows Phone 7.

This sliding operation is not present in Android. Transition Element is an abstract class that has the abstract method "GetTransition" class, which takes UI elements and displays a new page. "Slide Transition" is one among the Transition Elements. It provides a mode property that lets the developers to select the transition type and in this screen, the transition type is "Slide transition" i.e. the page appears by sliding from the first page.

Figure 5-6 : Geolocation Screen in Windows Phone

Now, taking a look at the "Conversion" screen, one can notice that there is a difference between the UIs. The drop-down list box feature that is present in Android is not available on Windows Phone. Figure 5-7 shows the "Conversion" screen of Android app.

Figure 5-7 : Conversion Screen in Android

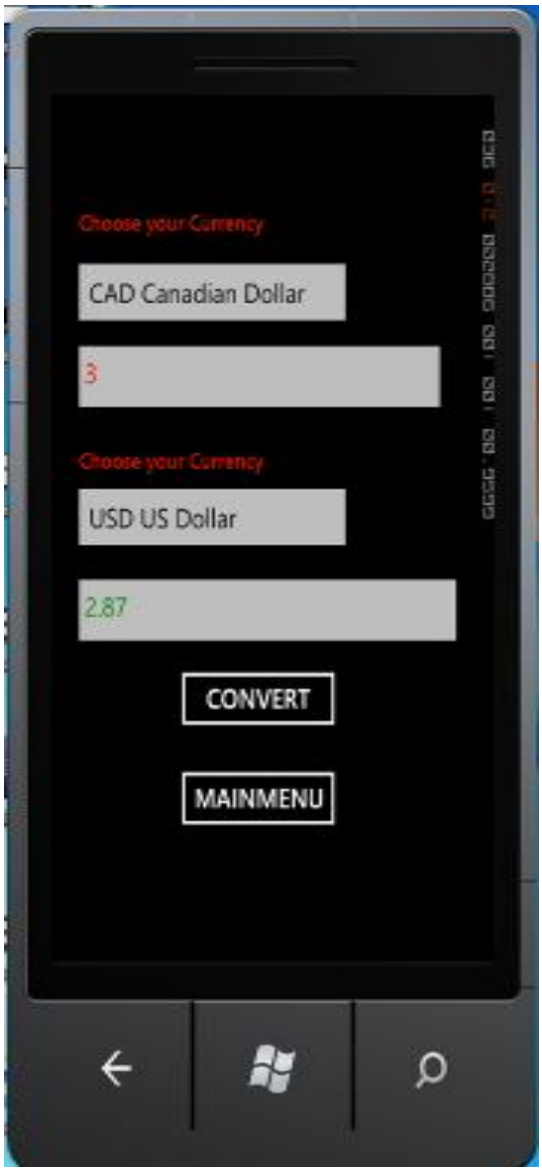

Figure 5-8 : Conversion Screen in Windows Phone

Android uses “spinner class” for drop-down list box which can be easily identified by the developers. Windows Phone does not provide drop-down list box. “List picker” tool is used to create menus/lists (currencies). “List picker” is one of the new components introduced recently which is an equivalent of the Windows Phone ComboBox control.

5.6.1 How the differences in UIs affect Developers?

From the developers' perspective, developing the same application in different platforms involves more time and learning new programming languages. Developers' concentration on user interfaces requires additional attention as the development is not limited to one platform.

1) As mentioned in Figure 5-3 and Figure 5-4, the slide transition feature (for Geolocation) in Windows Phone 7 is different from the feature/operation in Android. Instead of slide transition, Android has "Geolocation" button and it takes the user to the geolocation screen.
2) In Windows Phone 7 app, it is easy to go back to the main menu by simply sliding towards left. This eliminates the need of placing "Main Menu" button which is found on Android "Geolocation" screen.
3) In Windows Phone 7, "List Picker" component shows the selected item from a list and allows the user to pick from a list. This component is not present in the IDE's toolkit. It has to be added internally from another Microsoft Silverlight Toolkit. Figure 5-9 shows the list boxes and text boxes in Windows phone 7. One can easily notice that these list boxes are quite indistinguishable from a normal text box.

Figure 5-9 : Currency Conversion Screen in Windows Phone 7

As soon as the user taps on currency, the items (currencies) are displayed in a separate page. Figure 5-10 shows the list of currencies.

Figure 5-10 : Currency List in Windows Phone 7

1) During development, this functionality may consume more time in implementation than Android due to the addition of a component in the toolbar and also in the XAML code as the elements are displayed in a separate page.
2) In Android, due to the availability of "Spinner" function, it is easy to identify the list box for displaying currency. There is clear distinction between a list box and a text box, as can be seen from Figure 5-11.

Figure 5-11 : Currency Conversion Screen in Android

3) Unlike Windows Phone, the list items (currencies) are not displayed in a separate page. As soon as the user taps on currency, a pop-up window as shown in Figure 5-12 is displayed.

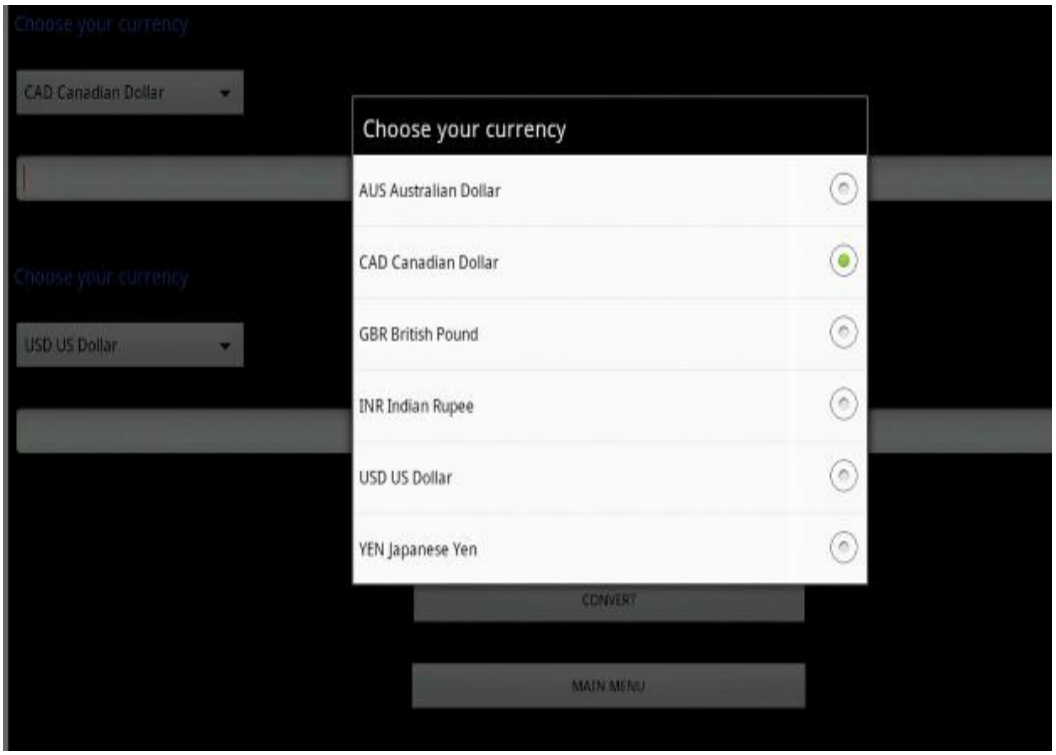

Figure 5-12 : Currency list in Android

This layout is easier than the interface developed in Windows Phone. The application has the same functionality but differs in user interfaces. As the application under consideration happened to be a light weight application, the difference in the interfaces may appear to be trivial but the same cannot be said of an application that utilises more features and is developed natively.

Therefore, from this practical analysis, one can conclude that developing same application in two different platforms consumes more time. To develop this application, two markup languages such as XML and XAML and two programming languages such as Java and .NET must be used. Moreover, there is a requirement to learn the APIs and SDK components in both Android and Windows Phone platforms. Overall, developing and distributing application across different devices using native application development technology have issues such as multiple languages support, multiple code bases, multiple maintenance (which is uneconomical for Enterprises) and user interface differences. Issues regarding user interfaces are showcased through above pictorial illustrations.

5.7 Hybrid Mobile Application Technique

Applications developed using hybrid concept are native applications that use web technologies in the place of programming languages like Java, Objective C and C#. In other words, hybrid apps are web-based apps that are built into native applications, so they can exploit the benefits of both native and web-based features.

Instead of learning multiple programming languages, SDKs, and IDEs, in hybrid approach, developers should know web technologies (HTML, JavaScript and CSS). Unlike the other alternative mobile application development techniques which are dependent on browsers (with limited functionality), hybrid approach provides developers better control over application design. Mainly, allowing access to device features. These benefits make it possible to develop multiplatform applications from a single code and still have performance and availability comparable to that of native applications [28]. PhoneGap, Rhodes and Titanium are the popular Hybrid mobile app tools in the current software market.

Application can be developed across platforms and can be ported too. Although hybrid tool helps in porting, accessing native features and developing application logic, for any application that needs appealing UIs (developed with themes), hybrid tools fall short of expectations.

Hence, user interfaces remain an essential issue in hybrid approach. UIs are vital in any application as they play a significant role in user experience.

5.7.1 Uniformity in UIs using Hybrid Approach

After developing "Currency Conversion" application in native platforms such as Android and Windows Phone 7, later the same app has been replicated in two platforms (Android and Windows Phone 7) but with Hybrid Mobile Application Approach. Reason for developing this application is that this app utilizes a certain native feature. It is the combination of both native and web application. Using PhoneGap (Hybrid tool), a native feature (GPS to get the location) is accessed through JavaScript APIs and current location is acquired. Automatically, the base currency (currency 1) is set according to the device's current location.

As discussed earlier, while developing any mobile application across platforms using cross-platform application frameworks, the user interfaces and functionalities are not unified completely. Mainly, user interfaces remain an essential issue across platforms. With Cross-platform UI application framework, it is possible to provide unified user interface across smartphone platforms.

The solution is to combine the Cross-platform Application framework and Cross-platform UI framework. By combining these two frameworks, unified user interfaces can be achieved. HTML is used for designing user interfaces, JavaScript APIs are used for accessing native device features, and CSS for styles and themes.

This approach is based on "write once run everywhere" concept. Single code has to be written and it is possible to use the same code for deploying the application in different platforms.

Here, the same code has been used for Android and Windows Phone 7. Issues which are discussed above are resolved through this approach. Figures 5-13 through 5-18 are the snapshots of the application deployed in two different platforms with same code. Also, UIs look same and the interfaces are significantly similar though the applications have been developed for different platforms.

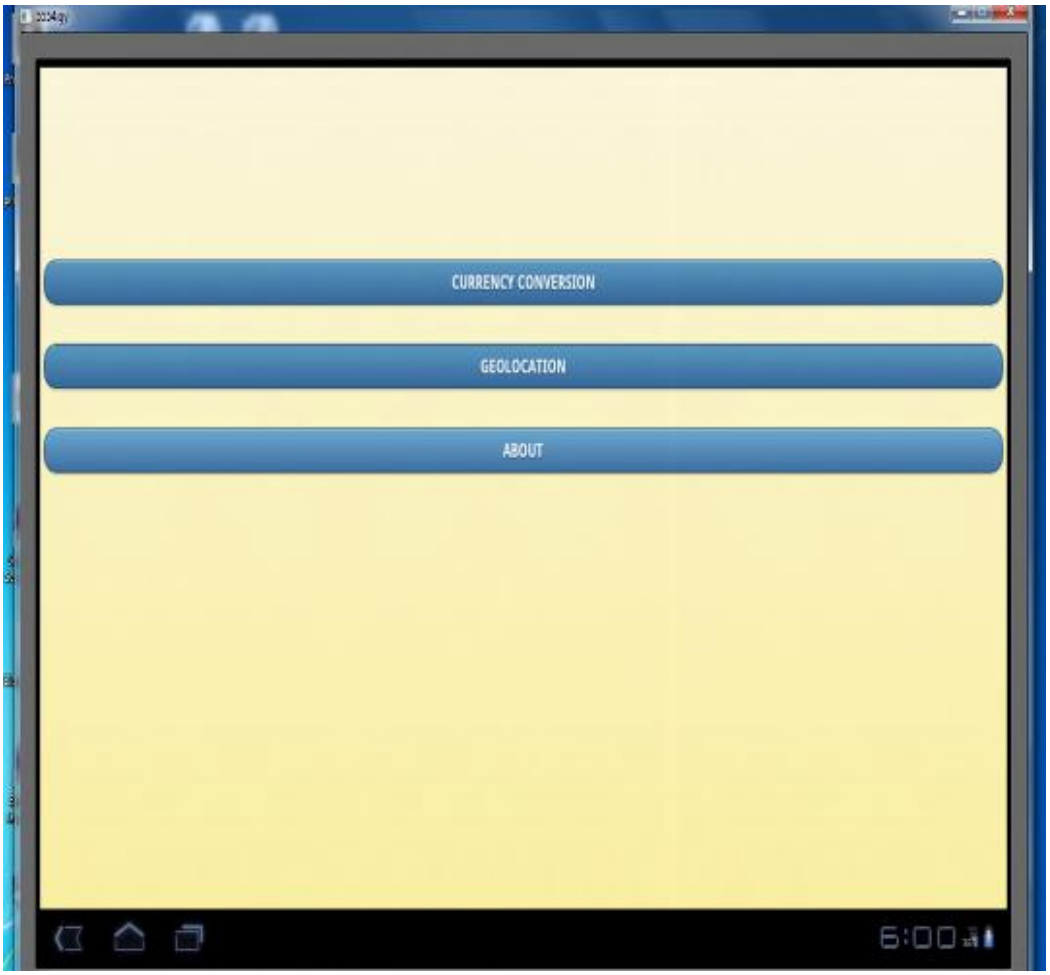

Figure 5-13 : Opening Screen in Android

Figure 5-14 : Opening Screen in Windows Phone

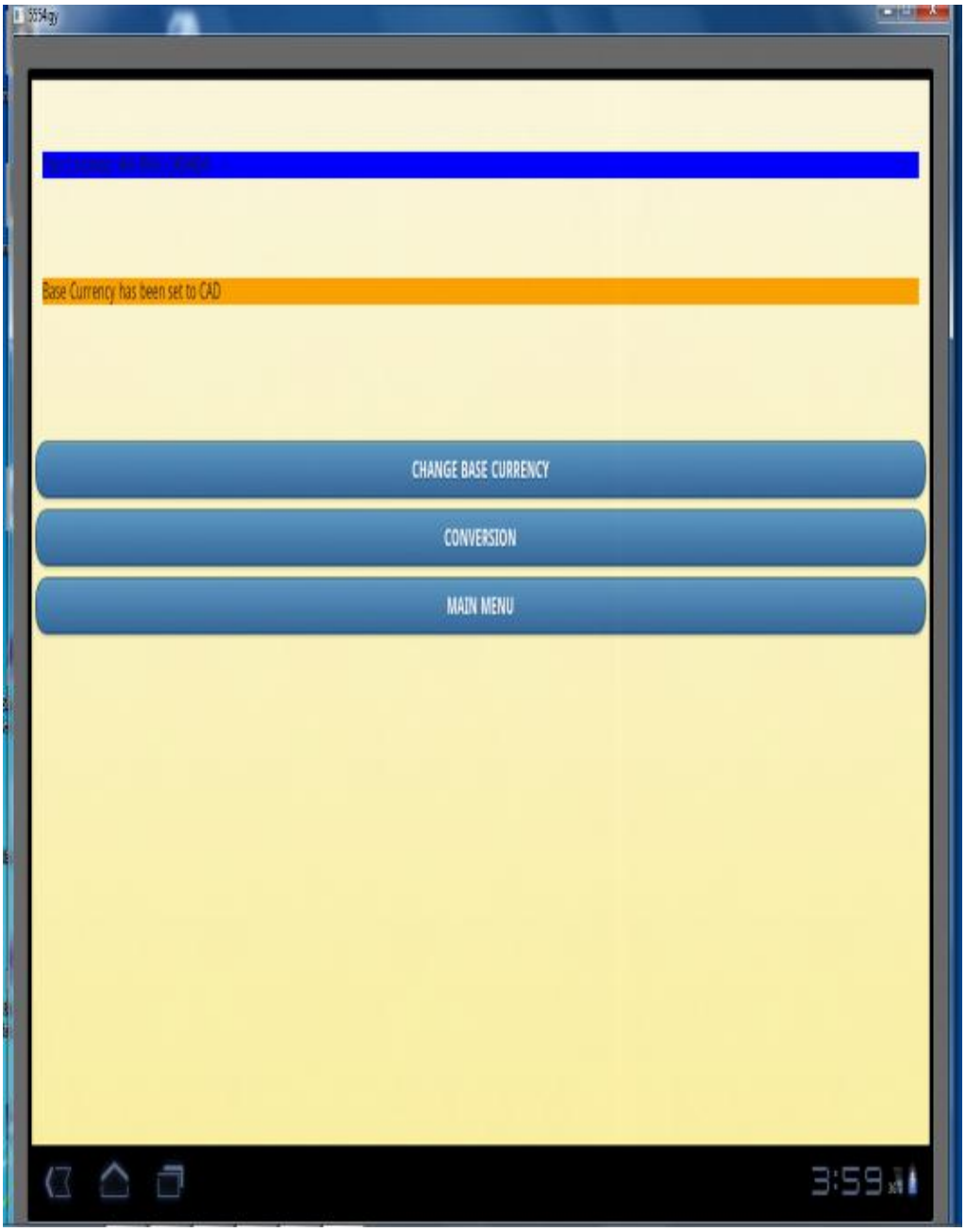

Figure 5-15 : Geolocation Screen in Android

Figure 5-16 : Geolocation Screen in Windows Phone

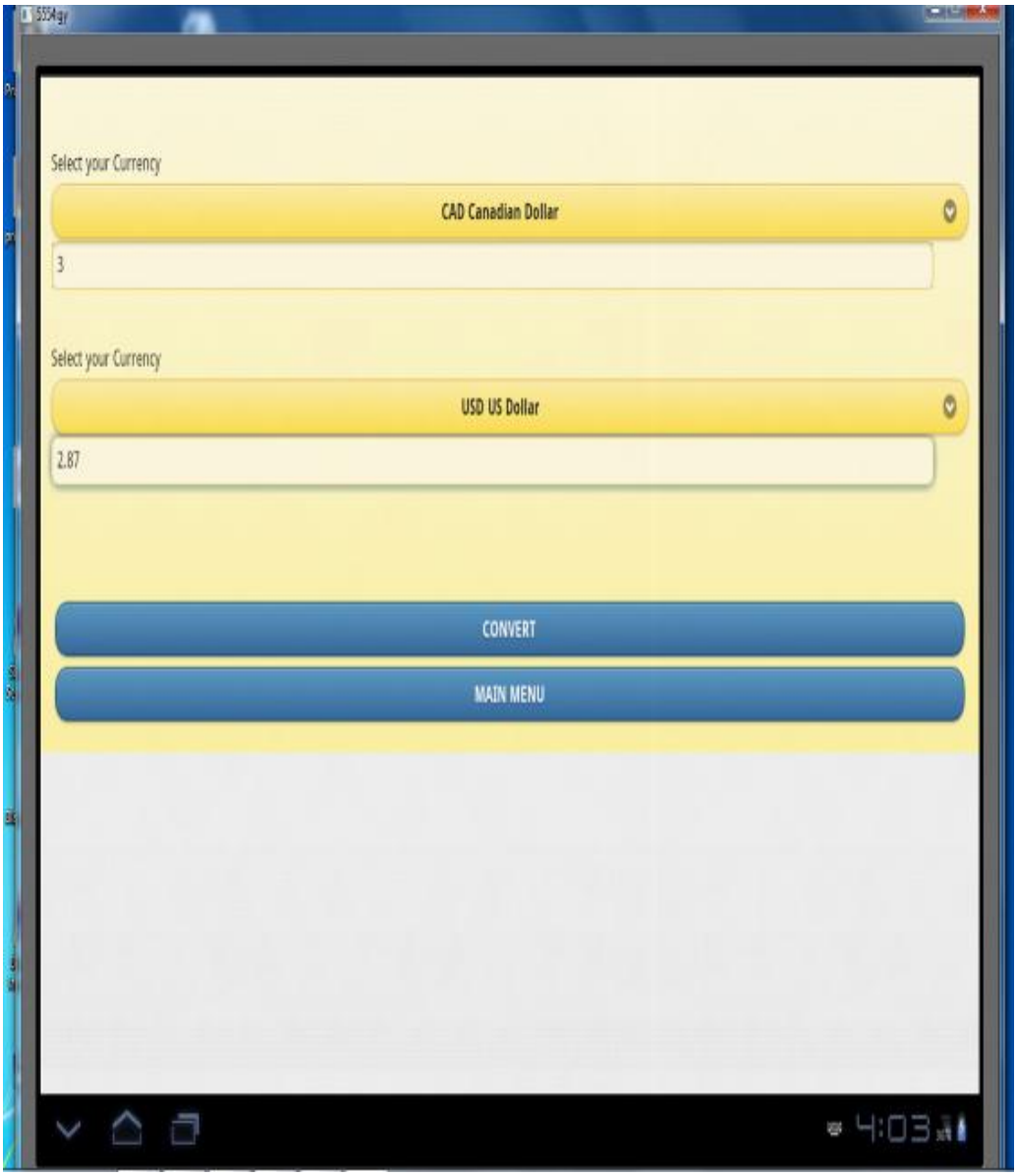

Figure 5-17 : Conversion Screen in Android

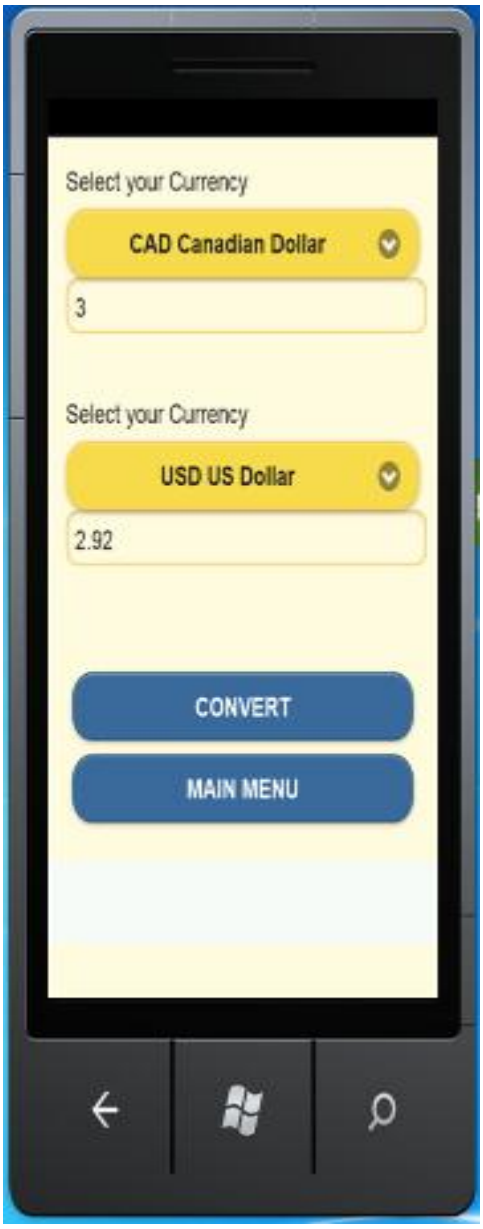

Figure 5-18 : Conversion Screen in Windows Phone

One can notice from Figure 5-18 that a drop-down list box is provided on Windows Phone 7. Earlier in the native app deployment of Windows Phone 7, "List picker" component was used for listing currencies. That component was added from Silver light toolkit. In this hybrid approach, no component has been added from an external source and yet a unified interface has been provided.

5.7.2 Unified Interfaces with Native Platform features

Although the application is developed with web technologies (HTML, CSS and JavaScript), platform's native interface features can still be accessed. This is evident from Figure 5-19.

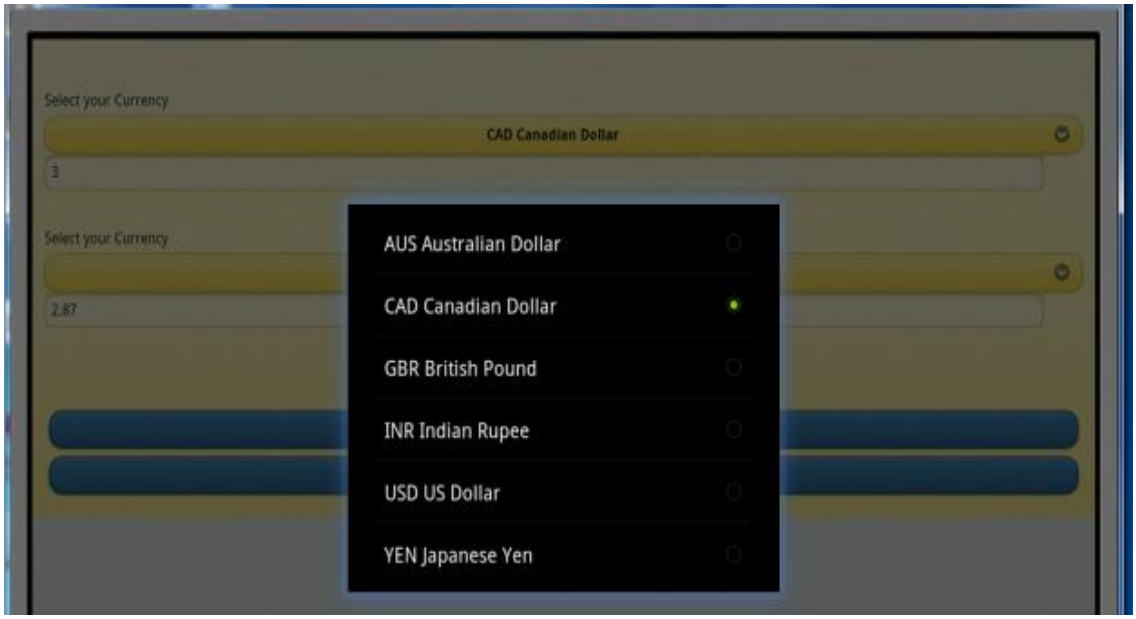

Figure 5-19 : Currency Listing in Android (Hybrid Approach)

One can notice the similarity in interface between the native app developed for Android and the hybrid app developed for Android through the pop-up window that opens up to display the list of currencies. This conforms that the native feature of Android platform can still be utilized through a hybrid app.

Figure 5-20 shows the list of currencies on Windows Phone 7. This opens up on a separate screen, as was already seen on the native app developed and ported to Windows Phone 7. The native rendering of elements are still maintained in the app.

Figure 5-20 : Currency Listing in Windows Phone (Hybrid Approach)

5.7.3 Programming constraints and Structure

In Android, XML was used as front end for user interfaces and Java along with Android APIs was used as back end for application logic. In Windows Phone XAML (front end) and .NET (back end) were used to create a native application. Markup classes were created for each screen to define the UIs. This looked a little fuzzy and it was complex to look up if an error had occurred in any of the class files. If a page (Screen) is created, markup (XML and XAML) classes were followed by Java and .NET classes in Android and Windows Phone 7 respectively. Hence, simultaneous concentration is needed on both sources. Mainly, developers should know XML, XAML, Java, .NET and Platforms' APIs to design an application. These factors increase development effort.

In the hybrid approach, combined functionality of cross-platform application framework and cross-platform UI framework provided unified interfaces. Programming constraints are limited and do not impact as much as in the native platforms. Developers should know the web technologies (HTML, CSS and JavaScript). Unlike, native platforms, in hybrid approach; pattern-specific programming is not required. Currency conversion (hybrid approach) was developed with ease as it did not involve pattern-specific programming structure. This is a great advantage of using web languages.

5.7.4 No differences in Interfaces

There are differences in UIs while developed natively. In Android, “Geolocation” button was required but in Windows Phone 7 it was not required. Slide functionality was used instead of having a button that linked to another page. But the interfaces were not identical in both platforms. With Hybrid apps, the UIs were identical in both platforms. Even though both platforms have different APIs, different programming languages and design architecture, hybrid approach correctly produces the platform oriented interface features (currency listing feature). Through geolocation feature, it is evident that hybrid approach can access native device features.

Hence, by the combined operation of cross-platform application framework and cross-platform UI framework, developers could effectively produce unified user interfaces across platforms. This approach would be greatly useful from the developers’ perspective. As hybrid applications run in native mobile browser, performance cannot be expected to be as good as that of native platform’s performance. Still, hybrid application offers commendable performance due to the advancements in mobile browsers and highly configured processors. But if it comes to heavy weight applications and 3d graphics applications, hybrid approach needs to be enhanced.

Hybrid tool like PhoneGap is pretty new and it is steadily evolving. It will have better functionality support for each individual OS and with jQuery Mobile; it will be highly feasible to develop efficient cross-platform applications in the near future.

CHAPTER 6 : EVALUATION

Hybrid technique is one of the ways to address cross-platform application development. The evaluation for this research work is based on developer's viewpoint. It is not feasible to measure the app development by usage of lines of code in the development or any mathematical analysis. There could be some differences between a novice programmer and an expert programmer in the aspect of coding knowledge, skill levels and expertise. Solutions given by an expert developer would definitely vary with a junior developer. Therefore, the evaluation can assessed based upon the development effort and mainly the features/elements which provide good support to improve user experience with hybrid approach. Characterization of development effort and supporting features for enhancing user experience have been evaluated in the following section.

A thorough analysis of hybrid technique led to a practical investigation and it greatly helped in recognizing the issues. However, there are complexities that increase the development effort. The development effort for each method (Native and Hybrid) and for the two platforms (Android and Windows Phone 7) has been characterized in Table 6.1, based on the following functions performed during development.

Table 6-1 : Characterization of development effort

Development elements	**Android**	**Windows Phone 7**	**Hybrid (PhoneGap+ jQuery) technique for Android and Windows Phone 7**
Installation	Eclipse, JDK, Android SDK and Plug-ins	Microsoft Visual Studio, Windows Phone 7 SDK and Silverlight toolkit	Android SDK, Windows Phone SDK, Eclipse, Microsoft Visual Studio, PhoneGap and jQuery Mobile
Designing User Interfaces	XML is used	XAML is used	HTML5 and CSS3 using jQuery Mobile wrapper
Ex: Drop-down list box feature	Spinner class which is already	'Listpicker' tool was added internally from	Drop-down list box is designed using HTML5 in jQuery

	present in Android SDK	silver light tool kit	mobile
Development	From scratch	From scratch	Single code base
Application Logic	Java	C# and VB.NET	HTML5. JavaScript is used for accessing native device features

In Android, Spinner class provides an easy way to select one value from a set. Spinner class adds a semi-transparent overlap over a DOM element with a spinning AJAX icon. When developing this same feature for Windows Phone, apparently, it was assumed that a class similar to 'Spinner' will be available in Windows Phone 7. But, it was found that Windows Phone 7 did not have a drop-down list box similar to Android's Spinner class. Instead, 'List picker' tool was used to create drop-down menus in WP7. 'List picker' is one of the new components introduced recently and it is a dependency property of type 'ListPickerMode'. It was internally uploaded from another Silverlight tool as it was not included in the MS Visual Studio IDE. Here, a comparative analysis cannot be performed between 'Spinner' class and 'List picker' tool since each solution is native to the platform. Developers who are familiar with Spinner class and who use it frequently to design a drop-down box in Android may find it cumbersome to do the extra work required to design a drop-down box in Windows Phone 7. It would help the developers if a pre-defined class similar to 'Spinner' class is provided in Windows Phone 7.

For the Hybrid version of the application, 'Drop-down list' interface is created in HTML5 using jQuery Mobile for both Android and Windows Phone 7. The code, once developed in Android is simply ported to Windows Phone 7.

PhoneGap lacks support for interactive UI widgets, design patterns and development tools [29]. Hence, it was understood that the hybrid tool alone would not help attain uniformity in UIs across platforms. The solution should couple hybrid development technique and uniform UI. Ultimately, cross-platform application framework and cross-platform UI framework are combined to produce unified user interfaces for hybrid applications. Cross-platform UI framework provides interactive UIs with customization

and branding across platforms. Though unified user interfaces are implemented across platforms, certain users may feel uncomfortable with the developer's unified user interface and may feel the absence of native look-and-feel local to the development platform. In specific, Android users and Windows Phone 7 users would expect the UIs to have their own native look-and-feel respectively. In the practical assessment carried out in this work, this issue has been addressed to a small extent. Although the application's interfaces were developed using jQuery Mobile (wrapper with HTML, CSS and JavaScript), some of the platform's native interface features had been utilised. Figure 6-1 and Figure 6-2 show the currency list (list of available currencies) which looks like the interfaces in native platforms (Figure 6-3 and Figure 6-4) and at the same time, the user interfaces generated by the hybrid frameworks are comparable in look and feel to native applications.

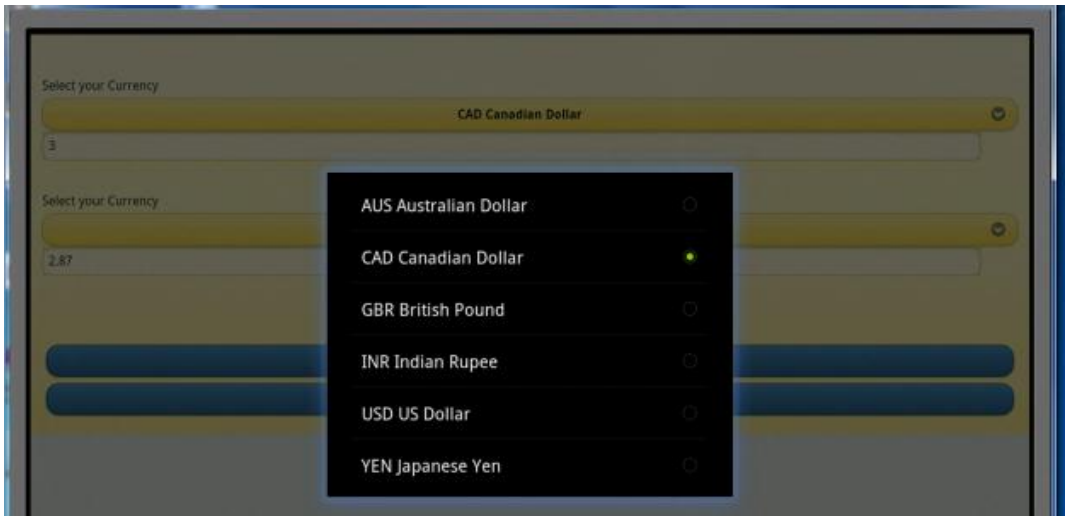

Figure 6-1 : Currency Listing in Android (Hybrid Approach)

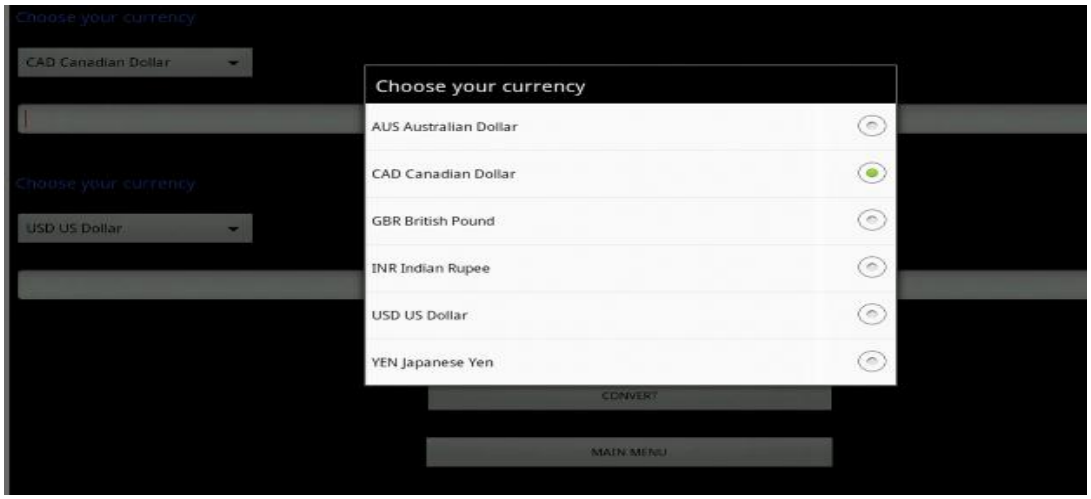

Figure 6-2 : Currency List in Android (Native Version)

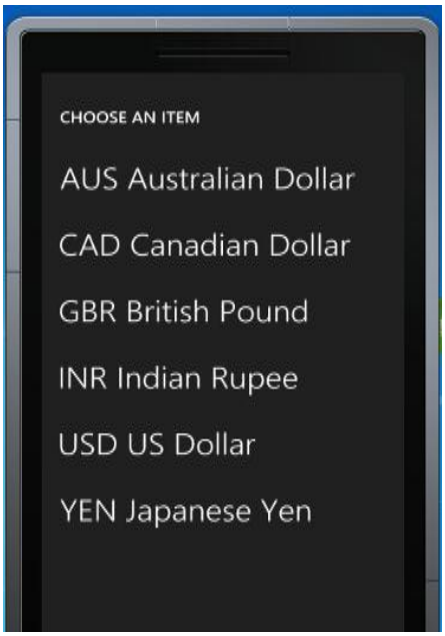

Figure 6-3 : Currency Listing in Windows Phone (Hybrid)

Figure 6-4 : Currency List in Windows Phone 7

PhoneGap and jQuery Mobile do not have a wide-range of Application Programming Interfaces but they seem more stable and support many mobile platforms with unified user interfaces. In order to produce native look-and-feel in cross-platform applications, a hybrid tool should have well-enhanced Software Development Kit and integrated native platform APIs. Table 6.2 shows important features which would help to attain native look-and-feel and thereby, enrich the user experience.

Table 6-2 : Supporting features for enhancing user experience

Features	**Description**
Native UI Support	This feature is responsible for native user-interface components and interaction inside a hybrid tool framework. It should be enhanced in a way to provide a native experience along with native performance by compiling JavaScript code into their native counterparts as part of the build process.
Native code support	It is a feature which permits extending the built-in functionality by creating extension

	libraries in platform's native programming language and using the module/library with the framework's non-native code
Device API support	Framework should have strong API support which utilize native user interfaces and platform APIs to access native UI components such as navigation bars, menus, dialog boxes, etc.

Although native apps gain benefits from an optimal integration into the respective mobile operating system and good developer support, the practical analysis showed that the hybrid approaches are workable alternative. Hybrid approach is fully suitable when mobile apps have to be developed for multiple platforms under tight budgets with small developer teams. Following table is the outcome of an evaluation of some significant aspects in the development.

Table 6-3 : Evaluation of essential aspects in app development

	Native Development (Android, Windows Phone and BlackBerry)	**Hybrid Technique (PhoneGap + jQuery Mobile)**
Ease of Development	Getting-started guidelines support beginners, Google regularly circulates blog spots and developers can resort to the active community. Programmers need to acquire additional knowledge about the mobile operating systems.	PhoneGap's documentation is clearly structured and provides more examples. But in jQuery Mobile, examples are not adequate although it presents all elements and design options.
Maintainability	As they use object oriented constructs and separate the code into classes, native apps are easy to maintain though they appear to be more heavyweight than apps developed in scripting languages. As these operating systems use different APIs and	Hybrid apps do not need more lines of code than native apps. The source code is short and clearly structured. This guarantees the ease in maintainence.

	components, additional concentration is required for each platforms.	
Long-term feasibility	The aspect of future of the smartphone market seems that these operating systems will remain to be popular. Technical communities, regular bug-fixes and updates aid developers in a huge way. Knowledge of different programming languages could be a technical burden.	As PhoneGap and jQuery Mobile are comparatively young projects, it is hard to estimate. But the frameworks are steadily enhancing and evolving as there are increasing communities for bug fixing and technical updates.
Scalability	Programming logic and GUI can be easily isolated from each other. Each view of an app can be developed on its own. Object-oriented concepts make the development teams to scale even better than other frameworks.	Apps can be eaily divided into a large number of small files that fit into the overall design. These frameworks support modularization well.
Speed and cost of development	An application need to be repeatedly developed for every platform, costs of development (obviously) are much higher than cross-platform approaches. Native development requires huge degree of specific knowledge and experience.	Comparing to other native and web frameworks, these tools take the shortest amount of time and development features are technically mature and the design of UI can be carried out fast and cost-efficient.

ROI (Return on Investment) is one of the prevalent evaluation metrics used in business analysis. Code-building-effort impacts the ROI. ROI is typically discussed by developers responsible for developing and delivering applications extensively. The elements which help to achieve efficient solutions are fast development cycles with minimal resources, minimizing code duplication and maximizing code reusability, better and cheaper ways for distribution and deployment. These factors support the areas such as fragmentation, browser capabilities and unified distribution. jQuery Mobile helps in providing consistent hybrid application development and user experience across platforms [30].

A survey was conducted on a developer intent index for 2011 and 2012. The survey represented the percentage of developers aiming to use major mobile platforms.

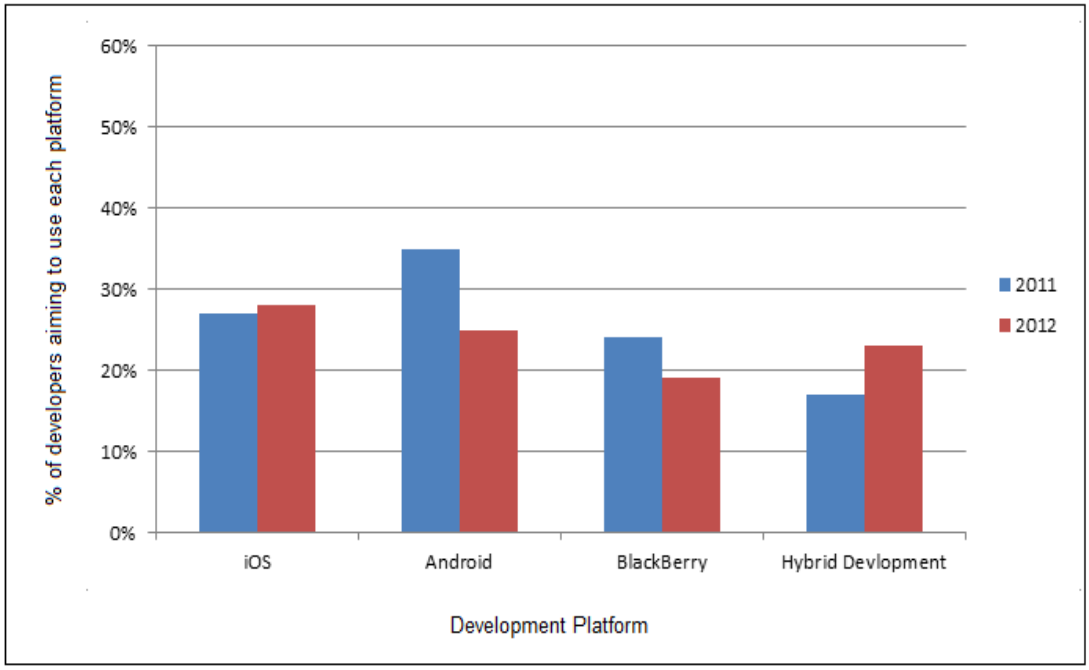

Figure 6-5 : Developer Intent Index for 2011 and 2012 [30]

It is interesting to note that hybrid development is gaining the attention of developers. The decisive factors that aided the developers in choosing a platform were 'Largely installed base of devices – 54%', Low cost development' – 43 %, 'Familiar development environment' – 31% and 'proper documentation and technical support' – 30% [31]. From Figure 6-5, iOS topped the chart by 28 % and faced a slight increase in 2012. Hybrid app development has a substantial increase in 2012 and it seems that the developers are paying attention towards hybrid technique.

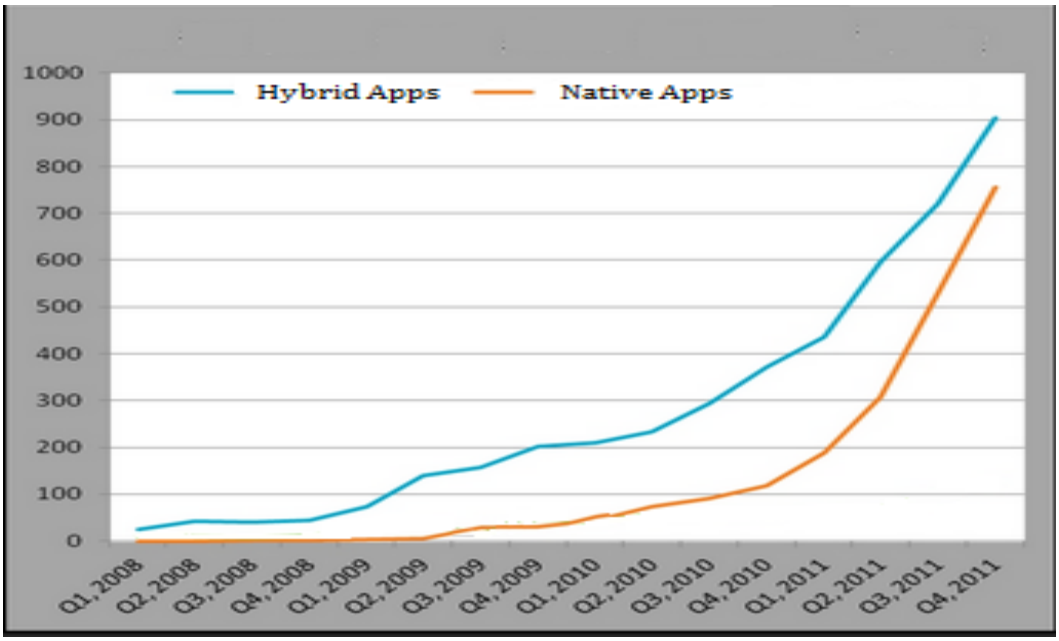

Figure 6-6 : Number of apps released per quarter by app type [31]

Figure 6-6 shows the number of applications released per quarter and reflects the growing trend of applications. Increasing number of users require mobile web versions of their applications than they require native apps [30]. After 4th quarter of 2010, the curve grows steadily for hybrid applications. Table 4 shows the developer ratings of their primary platform. Mainly the survey shows how developers identify major platforms in terms of user reach, revenue generation, cost of development and few other factors [31, 33].

Figure 6-7 : Developer barometer by platform [30]

Mobile web is perceived positively when it comes to ease of coding and ability to reach the users. In the perspective capabilities such as graphics and transformation, performance, connectivity and storage, the mobile web browser is facing rapid advancements [30]. This research's main objective is based on the developer's perspective of attaining uniformity in UIs and revealing the difficulties experienced during development.

CHAPTER 7 : DISCUSSION

Moreover, the solution of attaining UIs has been achieved using a single sample web-based application. Questions have been raised on certain aspects after developing the application.

- How the UIs differ among platforms when an application demands highly interactive graphic features?
- How will the programming language differences impact the design and implementation process for various levels of mobile applications?
- How did the APIs (for accessing system functionality) and performance differ considerably across platforms?

There are at least two other important objectives that should be considered. They are: enhancement of user experience and performance. User experience emphasises experiential, meaningful and valuable aspects of user interaction with the application. In specific, it deals with the user's perceptions. However, there is a consequential necessity to look at user interface design from the user's point of view (e.g. error proneness). Therefore, the future work would be carried out broadly on performance and integration of native APIs with hybrid tool's SDK in order to enrich user experience. Furthermore, substantial improvements can be made by continuing this research work. Future research topics include

- Following the progress in mobile development frameworks and re-examining/reassessing existing technology as the platforms evolve
- Scrutinizing how essential device-specific functions might become available through standardized APIs
- Organizing to provide decision opinions based on companies' requirements for app developers

CHAPTER 8 : CONCLUSION

The mobile application development techniques delivered immense knowledge after an in-depth analysis in mobile application development area. Major smartphone platforms such as Android, BlackBerry and Windows Phone 7 are studied and used for this research's practical examination.

This research's primary goal is to merge cross-platform application framework and cross-platform UI framework to produce consistent or unified user interfaces across platforms. The objective has been achieved by demonstrating a sample application in two different platforms. The sample application could have been deployed and examined with Apple's iOS. One of the leading smartphone operating systems in the current market is iOS from Apple (previously iPhone OS). Practical investigation on iOS has not included in this research work even though it has numerous interesting functionalities. One of the main reasons is hardware selection. iOS's development tools such as Xcode, iOS simulator, instruments and interface builder demand Mac operating system. The practical assessment for this research work has been carried out in Windows 7 operating system. Apple has certain restrictions and policy guidelines in the development process where developers are charged and enrolled for application development. The other objectives to be considered are performance and user experience.

The web browser has been rapidly progressing from a renderer of simple html into a runtime environment capable of delivering rich interactive applications across platforms. HTML 5 has been paying a huge contribution towards cross-platform application development and it has strong future perspectives. Browser development is mainly focused around the advent of HTML5 standard [34, 35]. There are many new features of HTML 5 that have been enhanced for improving the user interaction. Some of the improved design specifications include offline storage capability of web content, canvas element, server interaction and others [34, 36]. Inclusion of canvas element makes drawing and animating graphics considerably quicker. Innovative and advanced developments are expected in HTML in near future.

REFERENCES

[1] C. McGuirk, T. Pekala, J. Petrin and E. Renardi, "Choosing the Right Mobile Development Method," Available: http://www.rdacorp.com/thought-leadership/custom-application-development/choosing-the-right-mobile-development-method/, [Accessed: 23 February 2012]

[2] "The Linux Information Project (LINFO)", Available: http://www.linfo.org/cross-platform.html, December 2005, [Accessed: 24 April 2012]

[3] S.C. Johnson, D.M. Ritchie. (1978, Aug) "Portability of C Programs and the UNIX System," The Bell System Technical Journal, Vol. 57, No. 6, Part 2

[4] S. Helal, R. Bose and W. Li, "Mobile Platforms and Development Environments," *Morgan and Claypool Publishers*, February 2012

[5] T. Paananen, "Smartphone Cross-platform Frameworks," Jamk University of Applied Sciences, April 2011

[6] S. Allen, V. Graupera, and L. Lundrigan (2010, Sep) "Pro Smartphone Cross-platform Development – iPhone, BlackBerry, Windows Mobile, and Android Development and Distribution," Apress Berkely, CA, USA, 2010.

[7] D. Sambasivan, J. N. Udayakumar. S, Gupta. R , "Generic Framework for Mobile Application Development," Internet (AH-ICI), 2011 Second Asian Himalayas International Conference), November 2011

[8] D. Gavalas and D. Economou, "Development Platforms for Mobile Applications: Status and Trends," University of the Aegean, Volume: 28, January 2009, pp. 77-86

[9] A.Charland and B. Leroux, "Mobile Application Development: Web vs. Native,", *Mobile Computing, Communications of the ACM*, Volume: 54, No.5, May 2011

[10] H. Nagar, B.L. Lim, "Mobile Computing With Web 2.0: Current State-Of-The-Art, Issues and Challenges," Illinois State University, Volume: IX, No.2, 2008

[11] Z. Hussain and S. Torma, "Loose Coupling Between Services – Mobile Web and Event-based Interaction," Aalto University, School of Science, Version 1.0, December 2011

[12] D. Na, "The What, Why, and How of Mobile Applications," Sigma, Noblis, Volume 11 Number 1, October 2011, pp. 20 – 26.

[13] Worklight, “HTML5, Hybrid or Native Mobile App Development,” White paper, Available: http://123seminarsonly.com/Seminar-Reports/024/65749185-HTML5-Hybrid-or-Native-Mobile-App-Development.pdf, [Accessed: 15 February 2012]

[14] R. Rodger, “Beginning Mobile Application Development in the Cloud,” Published by *John Wiley & Sons, Inc*, 2012

[15] C. Kaiser, “How to Develop Mobile Applications with Web-technologies,” University of Fribourg Suisse, May 2011

[16] A. M. Christ, “Bridging the Mobile App Gap,” Sigma, Noblis, Volume 11 Number 1, October 2011, pp. 27 – 32.

[17] W. M. Gentleman, “Portability and other source management problems,” Problems and Methodologies in Mathematical Software Production, University of Waterloo, 1982, pp. 152 – 185

[18] S. Debray, “Abstract Interpretation and Low-Level Code Optimization,” University of Arizona, 1995

[19] R. E. Johnson, V. F. Russo, “Reusing Object-Oriented Designs,” Purdue University, May 1991

[20] C. G. Acord and C. C. Murphy, “Cross-platform Mobile Application Development: A Pattern-based Approach,” Naval Postgraduate School, March 2012

[21] A. Puder, “XMLVM: A Smartphone Cross-Compilation Framework”, SanFransico University, 2010

[22] A. Toth, G. Nemeth, “Creating XML Based Scalable Multimodal Interfaces for Mobile Devices,” Mobile and Wireless Communications Summit, July 2007

[23] L. Maaloe, M. Wiboe, “Kamili – A Platform Independent Framework for Application Development for Smart Phones,” Kongens Lyngby, 2011

[24] A. Puder and I. Yoon, “Smartphone Cross-Compilation Framework for Multiplayer Online Games,” San Francisco State University, 2010

[25] Z. Zhou, R. Zhu, “Windows Phone 7 Programming for Android and iOS Developers,” 2011

[26] W. Blochinger and W. Kuchlin, “Cross-Platform Development of High Performance Applications Using Generic Programming,” University of Tubingen

[27] Z. Kurmas, “Atomic Spin – Atomic Object’s Blog on Software Design and Development”, Available: http://spin.atomicobject.com/2010/11/22/the-cost-of-building-blackberry-apps/, November 2010, [Accessed: 11 July 2012]

[28] R. Godwin-Jones, “Emerging Technologies: Mobile Apps for Language Learning,” Language Learning & Technology, Volume 15, Number 2, June 2011, pp. 2-11.

[29] G. Hartmann, G. Stead, A. DeGani, “Cross-platform Mobile Development”, Mobile Learning Environment, March 2011

[30] C. Enrique Ortiz, “Mobility in 2011: Mobile Apps, Webapps and Tipping Points,” March 2011. Available : http://weblog.cenriqueortiz.com/mobility/2011/03/08/mobility-in-2011-mobile-app-vs-webapps-and-tipping-points/, [Accessed: 12 July 2012]

[31] A. Avram, “Mobile Platforms: What is the Developer Mindshare, Intentshare, App-building Costs and Revenue?” Available: http://www.infoq.com/news/2012/06/Developer-Economics-2012, June 2012, [Accessed: 12 July 2012]

[32] M. Brown, “Where Does your App Rank? U Test AppGrader,” Available: http://www.mobileapptesting.com/tag/utest/, March 2012, [Accessed: 13 July 2012]

[33] O.J, Turau. (2012, Sept) ”Cross-Platform Development Tools for Smartphone Applications,” Hamburg University of Technology, Computer – Journals and Magazines, Volume 45.

[34] T. Melamed and B. Clayton, “A comparative Evaluation of HTML5 as a Pervasive Media Platform,” Hewlett-Packard Labs Europe, Bristol, UK, 2010, pp. 307-325

[35] F. Jiang, Z. Feng, L. Luo, “xFace – A Lightweight Web Application Engine on Multiple Mobile Platforms,” 2010 10th IEEE International Conference on Computer and Information Technology, 2010

[36] P. Smutny, “Mobile development tools and cross-platform solutions,” Carpathian Control Conference (ICCC), 2012 13th International, Technical university of Ostrava, 2012