KWAME NKUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI-GHANA

A TOOL SELECTION FRAMEWORK FOR CROSS PLATFORM MOBILE APP DEVELOPMENT

By:

FELIX APPIAH (B.Ed. Information Technology)

A Thesis Submitted to the Department of Computer Science, Kwame Nkrumah University of Science and Technology in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Department of Computer Science, College of Science

NOVEMBER, 2015

DECLARATION

This thesis is written towards the fulfillment of the requirements of the Master of Philosophy in Information Technology Programme and I therefore declare that it is truly my own piece of work.

I duly acknowledge that it contains no material which has been accepted for the award of any other degree of this University or any other University. However, ideas and other information used in this work, which have been previously published or submitted by me or another author has received the needed acknowledgement.

All sources of data and/or information and other important materials obtained from institutions, departments, agencies, and individuals to make this work a reality have been equally given the necessary acknowledgement. However, it is likely for readers of this work to identify some errors or omissions. In view of this, I duly accept being responsible in that regard.

Felix Appiah (PG8303812) …….. ……

(Student Name & ID) Signature Date

Certified By:

Dr. Hayfron-Acquah J.B …….. ……

(Supervisor) Signature Date

Certified By:

Dr. M. Asante …….. ……

(Head of Department) Signature Date

ABSTRACT

The mobile application development landscape is continuously getting more fragmented with the emergence of an array of platforms with disparate operating systems and development workflows. Mobile Application developers are compelled by virtue of this platform fragmentation to design applications that targets more than one platform to ameliorate the market reachability of their products. To help developers in the pursuit of this cross platform agenda, diverse tools have been introduced by different vendors to provide support for cross platform development. However, there is no acceptable metric to serve as a basis for evaluating these cross platform tools. This thesis introduces a framework aimed at assisting application developers in the selection of the requisite Cross Platform development tool which guarantees the achievement of project requirement and specifications. . The framework design was guided by the Design Science research methodology. Quantitative methods including descriptive statistics, experiments and tests were used to provide data for the development and evaluation of the framework. The framework was implemented on three (3) Cross Platform Tools; PhoneGap, Titanium and Xamrin. Pairwise comparisons were made among these tools with capabilities, Performance, Development Speed, Learning Curve, Native UI look, and Device Access as the considered criteria. The Capability criteria emerged as the most important Criteria. PhoneGap developed by Adobe also emerged as the preferred tool for Mobile Cross Platform Development. Based on the framework, PhoneGap appears to be the Platform of choice in terms of Mobile Cross Platform development.

Keywords: Cross Platform Development, Tool selection Framework, Mobile Application Development, Fragmentation, Mobile Application Development Tools.

DEDICATION

This thesis is dedicated to my dear mother Maame Afua Boatemaa, who never stepped foot in any classroom but made an avowed commitment to ensure that, all her children enjoyed the best of formal education no matter the cost. To her I say: "This is just the beginning".

ACKNOWLEDGEMENT

"To know is to know that you know nothing. That is the meaning of true knowledge"

(Socrates, 399BC)

It is with a heartfelt gratitude that I acknowledge the effort and support of my Supervisor, Dr J.B Hayfron-Acquah whose patience and calmness made this thesis a reality. His timely advice and recommendations was really on point.

This acknowledgement wouldn't be fulfilling without commenting on the immense inspiration and support from my family especially my father, Mr Ernest Appiah Kubi and my Siblings. To my entire family, I say I am most grateful.

I would also wish to congratulate all research participants especially, Mr Elvis Agah and Mr Solomon Osei-Acheampong for their assistance.

Finally, I express my sincere gratitude to my "special friend ", Miss Juliana Opoku Mensah for seeing the vision much earlier and communicating, supporting and ensuring its realization. Thank you for believing in me".

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADT	-	Android Development Tool
API	-	Application Program Interface
APK	-	Android Application Package
APP	-	Application
BB	-	Black Berry
CPDT	-	Cross Platform Development Tools
CPU	-	Central Processing Unit
CSS	-	Cascading Style Sheet
HTML	-	Hypertext Mark-up Language
IBM	-	International Business Machines
IDC	-	International Data Corporation
IDE	-	Integrated Development Environment
OTA	-	Over-the-Air
OS	-	Operating System
SDK	-	Standard Development Kit
RIM	-	Research in Motion
RISC	-	Reduced Instruction Set Computing
UI	-	User Interface
W3C	-	World Wide Web Consortium

CHAPTER ONE

INTRODUCTION

1.1 Background

The Mobile Computing community has seen major changes with the introduction of powerful devices and operating systems with enhanced capabilities and performance (Devitt et.al, 2010). According to a report by International Data Corporation in 2014, Android and iOS led the mobile operating system market share with sales of approximately 70% and 20% respectively. The worldwide sale for 2014 was estimated to be a little over 1.3 billion mobile phones which represented a 27.7% improvement on the 1 billion smartphone sales recorded in the year 2013 (IDC,2014). With these emerging trends, mobile devices have become more pervasive than ever in our daily life and activities. Currently, mobile phones that utilize video or audio accelerometer and Global Positioning Services have displaced others that only perform the calling and text messaging functionalities. The existence of these enhanced embedded features coupled with innovative applications; make mobile devices, such as smartphones very desirable and popular among a large section of people across the world.

The popularity of mobile devices has motivated majority of vendors to get on board and contribute to the provision of mobile operating systems as well as software that run on mobile devices (Pastore, 2013). Different vendors have developed their own proprietary method of developing applications for their devices using a variety of programming languages and Development Kits. This means that an application developed for Google's Android operating system will not run on the RIMs blackberry platform. This had led to a challenge in the mobile computing industry known as platform fragmentation. The issue of fragmentation becomes more

exacting for developers when applications built for a targeted platform are not able to run on different versions of hardware devices (Dhillon, 2012). Apart from dissimilarities in software platforms, variations in hardware devices also provide some level of difficulty in guarantee commensurable user experience from device to device (Agarwal et.al, 2009).

Mobile application developers are narrowed in their choice of platforms due to factors such as cost and knowledge of several coding techniques that transcends platforms. Marketing products designed for platforms with huge developer acceptability is also relatively less difficult. Research in Motion (RIM) was confronted with tremendous difficulty when migrating to BB10 platforms after evolving from applications developed natively for Blackberry Operating System. (RIM Inc. 2011).Considering the number of languages used in Mobile Application development, developers must be very versatile and businesses need to invest significant resources to have their software available on more than one platform.

This issue of fragmentation makes cross platform development very relevant and inevitable. A survey by Appcelerator and IDC in August 2012 showed that companies continue to be very interested in cross platforms regardless of the challenges and difficulties (Appcelerator Inc, 2012). In 2011, developers had shown interest in running on twice as many platforms as a similar survey in the previous year had indicated a multi-platform patronage by developers averaging an incredible four operating systems (Allen et.al, 2010). This trend continued to increase in 2012 (Appcelerator Inc. 2012).

Adjudging the developer appetite for the availability of their applications on many platforms, developers face an obvious problem. Developers lack the resources to develop natively for each target platform. They are also confronted with the challenge of learning new lines of code

specific to selected platforms. The CEO of InRuntime made a comment that, the introduction of CPDTs has shortened the time to market of mobile applications by 70% (Vision Mobile, 2012). This emphasizes that; using cross-platform tools can be very helpful in many ways: firstly Cross Platform development has the possibility of reducing the number of resources that would be expended during software development and developers wouldn't need to program in different platform environments.

Despite the popularity and potential for developing for multiple platforms, there is no adequate metric to measure the capability and performance of Cross Platform Development Tools (herein after referred to as CPDTs). This makes it difficult for developers to identify tools that could be used to create applications to match rival applications built for native platforms. Developers are also uncertain which tool guarantees the greatest value and benefit for the time spent working on their projects. This thesis therefore aims to create a framework to provide thorough comparison of CPDTs to one another and determine how best they can be used in developing applications for mobile devices. All this should be done conveniently and effectively without compromising on quality, performance and user experience. Through the development an evaluation framework based on acceptable metrics for CPDTs, developers will gain the knowledge needed to determine which tool to use for their application.

1.2 Motivation

Mobile application developers are confronted with challenges as well as opportunities looking at current trends in the market. An array of new devices ranging from smartphone to tablets is redefining what users can do. On the other hand, application developers are saddled with the question of what devices to develop for, how to create effective but simple apps and the number of languages developers should learn to become robust developers. The potential reduction in

effort of developers during the application development process in terms of cost provides the needed motivation to embark on this thesis.

The possibility that, application developers will comparatively spend less time in the selection of the requisite tools for their projects or better-still eliminate the trial and error approach in tool selection provides enough excitement to undertake this thesis.

1.3 Objective of the Study

This thesis aims to provide a tool selection framework to assist mobile application developers in choosing the preferred CPDT for their projects with consideration on project outcomes. The development of a selection framework for CPDTs would assist developers to gain the knowledge needed to determine which tool to use for their application. The specific objectives of the study are to:

i. Identify challenges and opportunities of existing tools used in designing cross platform mobile applications.
ii. Identify priorities developers assign to specific criteria in tool selection.
iii. Access the feasibility of a Tool Selection Framework for Cross Platform Development.
iv. Provide an overall ranking of CPDTs based on pre-determined set of criteria.

1.4 Research Questions

The following relevant questions are needed to be answered in order to attain the objectives outlined above:

i. What are the challenges and opportunities of existing cross platform development tools?
ii. Which criteria are mostly considered by developers during CPDT selection?
iii. How feasible or otherwise is a Tool Selection Framework for Cross Platform Development.
iv. How feasible or otherwise is a ranking for CPDTs based on pre-determined set of criteria?

1.5 Thesis Contribution

This thesis provides a tried and tested framework for evaluating CPDTs used in mobile application development projects. This has been occasioned by the absence of a pragmatic evaluation metrics specifically for CPDTs in Information Technology literature. Developers are often compelled to rely on subjective product documentations of CPDTs vendors and make decisions out of the information available to them.

The major contributions of this thesis are as follows:

i. Development of a merit-based tool selection framework to assist mobile application developers in selecting the best CPDT for their projects.
ii. The results of the thesis will provide a ranking for the various criteria used in evaluating CPDTs.
iii. The results of the thesis will also provide a ranking of CPDTs and reduce the burden of developers in tool selection

iv. The work will serve as a body of knowledge on the subject and provide a sound basis for future research in the area of Cross platform development.

1.6 Scope of the study

Since a complete analysis of several software platforms is not feasible under the timing constraints that apply to this thesis, it is crucial to define the scope of the evaluation. The target mobile operating systems were limited to Windows and Android OS. The tools and methods that were evaluated were therefore limited to three approaches: PhoneGap, Titanium, and Xamarin. The decision to choose Android and windows was based on the popularity of the two platforms in the mobile operating system market worldwide. This thesis does not include the compatibilities on any tablet of the platforms mentioned above.

Also the framework designed in this thesis would not serve as a basis for evaluating graphically intensive applications. Game applications designed for mobile phones therefore falls outside the scope of this thesis.

1.7 Limitations of the Study

Among the numerous CPDTs available, only the three mentioned above could be evaluated due to time constraints. The proof of concept application designed to test the evaluation framework could have been developed with several frameworks to compare with PhoneGap but the writer did not have enough knowledge to develop applications for those platforms.

1.8 Methodological Approach

A quantitative research design which emanates from the positivist research paradigm was adopted for this study. Quantitative research provides the methods and techniques that allow information system researchers to answer questions related to the interaction of humans and

computers (Straub et.al, 2004). Group decision making techniques which involved six software engineers was used to weight the criteria and the alternative tools considered in the framework. The Six software engineers were briefed on the use of the framework and asked to make pairwise comparison of the criteria and the tools to be compared. The Design Science research method was adopted for this study. Microsoft Excel Software was used to facilitate the ease of computations. A proof of concept application was developed to test the hypothesis which emanated from the framework implementation: PhoneGap can be used to design Cross Platform Applications comparable to native applications. Tests were run using the W3C mobile OKchecker, Platform Emulators and live devices to prove or disprove the hypothesis.

1.9 Organization of the Study

The research work is divided into five 5 chapters. Chapter 1 covers the general introduction to the study, and captures the background of the study, the statement of the problem, the objectives of study, the research questions, significance of the study, the scope of study, limitations of the study, and the organization of the study.

Chapter 2: Discusses the topic in more detail to provide necessary background information and details on some selected CPDTs. It then discusses the limited prior work in this research area.

Chapter 3: Describes the proposed solution, a tool selection framework that can be used in choosing a desirable CPDT for a project.

Chapter 4: Introduces an implementation of the framework in more detail and the procedures used to test its viability.

Chapter 5: Presents the results of the implementation of various CPDTs using the evaluation framework developed in this thesis. This chapter also describes the proof of concept application

designed with the CPDT that would be eventually selected. Concluding remarks and future work which can be used to extend this framework are also discussed at this section.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter focuses on providing relevant literature related to cross platform mobile application development. It is important to understand each of the mobile platforms and development languages before being able to understand the scope of the problem solved by CPDTs. This chapter provides background information on the mobile landscape and an overview of various CPDTs. This overview is necessary to understand which aspects are most important and should be addressed in any evaluation framework. This will be followed by discussion of related work for evaluating CPDTs.

2.2 Concept of Mobile Application Development and Relevance

Anderson and Gestwiki (2011) define mobile application development as the process by which application software is developed for low-power device such as Personal Digital Assistant, Enterprise Digital Assistant or Mobile Phones. Setting the distinction between mobile applications and desktop applications, Pcgmedia (2004) argued that, mobile applications developments are different from desktop applications development and it is worth noting what makes mobile applications special. Pcgmedia went ahead to state that, a keen understanding of what mobile devices actually are,32 and the variations between it and laptops as well as desktop computers provides the pre requisites that determine the success in mobile application development (Pcgmedia, 2004).

Other group of researchers also looked at mobile application development from the developer's perspective. They defined mobile application development as a process by which application is developed, sent to the market, procured by customers and used on mobile devices (Ballon, 2009).

Looking at the definition above, constructs such as development, marketing, sale and usage were incorporated into the application development process. The model in Figure 2.1 illustrates components in the mobile development process.

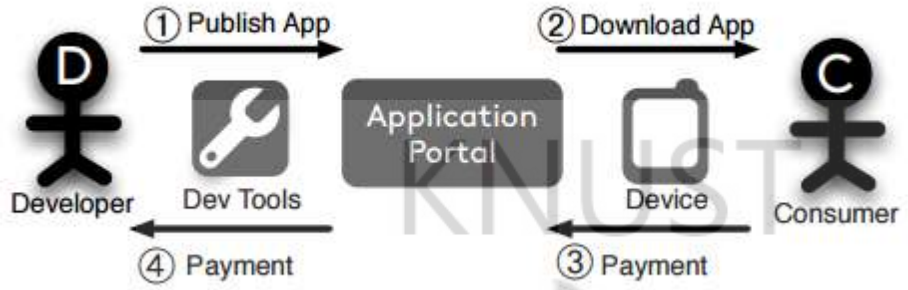


Figure 2. 1: Mobile Application Mediated Model (Ballon, 2009)

Significant attention has been devoted recently to the mobile computing by researchers and various stakeholders including mobile application developers and consumers who benefit from these applications. A report by the local and e-tailing group in the year 2012 made emphasis on the fact that, 47% of consumers suggested they used their smartphones to perform several basic operations such as locating a local sales outlet and finding relevant information on the internet. Logically as usage of smartphones shoots up, the demand for mobile applications would potentially also increase. Byrnes (2014) in his view points to the relevance of mobile application development looking at the number of mobile application downloads between December 25th and December 31st which was a holiday. The number of mobile apps downloaded within the period was estimated to be 1.2 billion in the United States. This and many other achievements in the industry open new opportunities for future mobile applications and development of mobile related services. Adrian and Ondrus (2011) predicted the possibility of the mobile application market reaching a staggering $9 million by 2011 according to compass intelligence. Another research group Gartner Inc., agreed with Adrian and Ondrus and affirmed that, global mobile app

store downloads exceeded 45.6 billion in 2012, closely twice the 25 billion downloads in 2011 which by 2016 will likely reach 310 billion downloads and $74 billion in terms of revenue (Gartner Inc., 2012). The vice president of Mobile and Wireless research at IDC Scott Ellison on his part concentrated on developers and suggested that:

> *"Mobile app developers will 'appify' just about every interaction you can think of in your physical and digital worlds. The extension of mobile apps to every aspect of our personal and business lives will be one of the hallmarks of the new decade with enormous opportunities for virtually every business."* (IDC, 2010)

Similarly, *Forrester Research* in a survey conducted in the year 2010 came out with the report that, 75% of companies attribute their increase in productivity to their workers through the usage of mobile applications, aiding better consumer experience, quicker problem resolution and faster decision making (Burns et.al, 2012).

This means that, mobile application development is assuming significance which was non-existent in times past. Comparing the current trends to previous trends in terms of mobile application development, Ballon (2009) stated that, traditionally in mobile application industry, there were several factors providing different benefits along the entire value chain. The current trend indicates that the structure of the mobile market and the value chain are really evolving. According to Holzer (2011), the roles of several stakeholders had changed, exchanged or combined. All these trends and facts lead to the conclusion that the mobile applications market is a market of the future, and is already promising great opportunities for businesses and mobile application developers and researchers.

2.3 Current State of Mobile Platform Landscape

Several mobile platforms and operating systems exist in the market but only three were considered in this thesis. The three include: Android (by Google), iOS (by apple) and Windows Phone (by Microsoft). Apart from these three notable platforms, other platforms on the market that are worth mentioning include Symbian (by Nokia) and Blackberry (by RIM), as shown in Figure 2.2. The strength of Symbian and Nokia had been dropping in terms of market share due to strong sales and penetration for Android and iOS (Gartner, 2012).

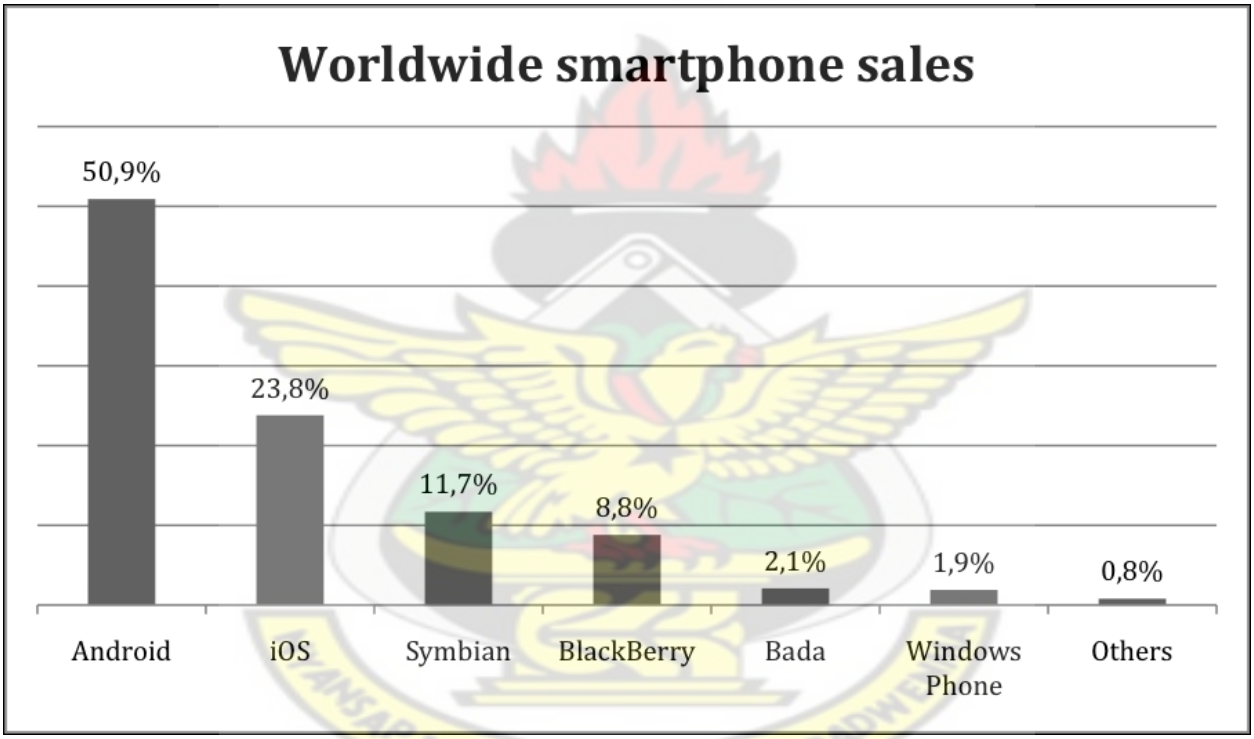


Figure 2. 2: Global Smartphone Sales Ranking of OS, Fourth Quarter 2011 (Gartner, 2011)

Similarly, a report by Drake et al (2011), released in December 2011, indicated that the competition in the mobile market was between five major companies rivaling each other with the power of their mobile operating – Google, Apple, Nokia, Blackberry and Microsoft. Table 2.1 shows the share of shipment made by these companies in year 2011, as well as an estimate for the year 2015.

Table 2. 1: Global shipment of smartphone by operating system, 2011 and 2015(Drake et.al 2011)

Operating System Mix	2011 Market Share	2015 Market Share
Android	49.0	46.5
Blackberry Os	11.1	10.0
iPhone Os	18.2	19.3
Symbian	16.4	0.1
Windows Phone	1.9	20.6
Others	3.3	3.5

A report by comScore on the major trends in the smartphone industry in U.S.A published in July 2014 had Google's Android operating system securing the slot as the No. 1 smartphone platform recording 51.5 percent platform market share. A report by the International Data Corporation (IDC) from the year 2011 to the first quarter of 2014 also demonstrated an improvement of Android's dominance with a market share of 84.7% as at the second quarter of 2014. IOS, Windows, Blackberry OS and others had 11.7%, 2.5%, 0.5% and 0.7% respectively. The data in Figure 2.4 shows the reports by the International Data Corporation (IDC).

Table **2. 2**: Smartphone OS Market Share (Source IDC 2014 Quarter 2)

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q2 2014	84.7%	11.7%	2.5%	0.5%	0.7%
Q2 2013	79.6%	13.0%	3.4%	2.8%	1.2%
Q2 2012	69.3%	16.6%	3.1%	4.9%	6.1%
Q2 2011	36.1%	18.3%	1.2%	13.6%	30.8%

In summary, the market of mobile operating systems globally is dominated by Google's Android operating system followed by Apple iOS. Windows is occupying the third slot now but is predicted to overtake iOS by the end of 2015 as predicted by Drake et.al, (2011). However, the other players are also doing very well and controlling a fraction of the market. Since mobile application are designed to reach a wider audience for access and profitability, any approach that seeks to include most of the platforms in terms of coverage will be a step in the right direction.

2.4 Comparison of Mobile Platforms

A direct comparison was made by Dhillon (2011) between Android, Apple iOS and Windows Phone, to identify similarities and differences. By comparing the characteristics of each of these platforms, the potential challenges in cross platform development for these platforms were identified. The foundation of each of the three platforms was identified to be fundamentally different. Whiles Android was based primarily on Java, Windows Phone was built with C# and Apple iOS was developed with Objective-C (Dhillon, 2011). Both Android and Apple iOS supports the same CPU-instruction set. They are both compactible with the ARM CPU instructions. Windows Phone currently only supports 32bit CPU instructions but Microsoft recently announced that Windows Phone 8 will provide support 64bit instructions as well. A

difference in application development on the platforms Android, Windows Phone and iPhone includes learning curve of which the iOS platform has proved to be the most difficult to learn. This is related to the fact that Objective-C uses a syntax that is not like any other syntax and that one will have to implement his own memory management. The development environments of the three compared platforms include debuggers and emulators. Development for Android is mostly done using Eclipse, which can be used to develop Android applications using the Android Development Tools (ADT) plugin. Developing applications for Windows Phone can be done using Microsoft Visual Studio, which is one of the best IDEs available in industry containing an extensive debugger and an emulator for Windows Phone applications. XCode is used to develop iOS applications, which is bundled with an iPhone emulator and has a debugger integrated.

In terms of cross-platform development and application deployment, all of the compared platforms can only be used to develop applications for that specific platform. Deploying applications on the Android platform is achieved using the Android Market or distributed via APK files. Apple iOS applications are deployed using the Apple AppStore. Windows Phone applications can be deployed using XAP file deployment procedure, Over-The-Air deployment technique, or the Windows Phone Market Place. Unlike Apple iOS, Windows and Android offer the possibility to deploy applications using installation files, whereas iOS applications can only be deployed using the AppStore. Development tools used when developing for Android are free of charge, as Eclipse and the ADT plugin both do not attract any payment. Likewise, developing applications for Windows Phone can be performed at no cost using the free Express edition of Visual Studio 2010. However, more extended versions of Visual Studio are not free. XCode, the IDE to develop iOS applications is also free of charge.

A comparison of the application development characteristics for the platforms Android, iOS and Windows Phone is presented in Table 2.3. Considering the graphical interfaces among them, all platforms support 2D and 3D graphics. When developing applications for Android or Windows phone, the developer has full support for the device in terms of functionality. However, access permissions have to be allowed in the application to permit the application to utilize those functionalities (Dhillon, 2011). For Apple iOS, the functionalities they support are limited and can only be available through APIs. The case with regards to functionality access is however different for Android and Windows. The data access for the phone is not restricted for Android and Windows Phone, meaning that data available on the device can be accessed programmatically. This is different for Apple iOS, which offers limited access to the data on the device. Bruning (2011) puts it that the runtime speed for Apple iOS can be labeled as best, as the Objective-C code is compiled into CPU-instructions which can be executed on the CPU directly.

Table 2. 3**:** Mobile Platform Application Development comparison Table Bruning (2011)

Feature/Platform	**Android**	**Windows Phone**	**iOS**
CPU instruction Set	ARM	X86	ARM
Foundation	Java	C#	Objective C
Learning Curve	Excellent	Excellent	Good
Deguggers available	Good	Excellent	Good
Emulator Available	Free Emulator	Bundled with IDE	Bundled with IDE
IDE Available	Eclipse	Visual Studio	Xcode
Crossplatform	Android	Windows Phone	Only iPhone iPod Touchpad and iPad
Deployment	Apk files or Android Market	OTA deployment, XAP files, market place free	Appstore or Jailbreak
Development tool cost	Free	Free	Free

The android platform upon its release targeted at making its architecture open to allow for component reuse by applications. The reuse was aimed at services as well as data and UI (Anvaari and Jansen, 2012). The Dalvik virtual machine used by Android, allows for application to be initially interpreted into CPU-instructions before it could be executed. The runtime speed of Android as a result is labeled as average. Windows Phone uses the Microsoft .NET Compact Framework, in which applications are converted to an intermediate language, before it can be executed on the underlying hardware. Therefore, the runtime speed of Windows Phone is also labeled as average.

The developer community can be labeled as extensive for all of the compared platforms. The comparison of the market penetration of the compared mobile platforms is convincing, Android is the big market leader holding a share of almost half of the market sales (Gartner, 2012). Apple iOS currently holds the third place in market sales with a percentage of 16.8%. Market penetration for Windows Phone can be labeled as low with a market sales percentage of 3.6%. Distribution and licensing for Apple iOS requires an Apple-issued certificate which costs $99 for a Standard license and $299 for an Enterprise license. To distribute an application using the Android Market, a fee of $25 is required. Distributing Windows Phone applications using the Market Place is free of costs for paid applications. For non-paid applications, the first 100 submissions to the Market Place are free of costs each year.

Each of the major operating system platforms comes with its own challenges and prospects, and the choice of which device and market to target can have significant impact for companies wishing to deliver their content digitally.

A comparison of the breadth for the platforms Android, Apple iOS and Windows Phone is presented in Table 2.4 as described by (Bruning, 2011)

Table 2. 4: Mobile Platform Breadth Comparison Table (Bruning, 2011)

Feature/Platform	**Android**	**Windows Phone**	**iOS**
Developer Community and Support	Extensive	Extensive	Extensive
Market Penetration	Market Leader	Low	Average
Distribution and Licensing	Unknown	Unknown	Apple issued Ceriticate

Summary

The major platforms in the mobile application development market have been compared. Google Android emerged as the leader in terms of market share and developer community followed by Apples iOS. Windows has assumed the position of a third force in the market. Both Android and iOS use the ARM architecture which is based on the RISC instruction set. Considering all the three compared platforms, only windows is using the 32bit architecture but windows Phone 8 promises to also migrate into the ARM architecture. All the three platforms have an extensive developer community with android emerging as the leader in terms of market penetration. iOS applications can only be deployed using the Apple app store whereas windows and Android can be installed by deploying the xap and apk files respectively on targeted devices. Android can as well be deployed using the Google play store. This makes the deployment of Android Apps very convenient as compared to the other windows and iOS.

2.5 Approaches to Mobile Application Development

Behrens (2010) defines mobile applications as an application developed to run on mobile devices mostly a web-based application and a native. This thesis defines a mobile application as an application developed to run on smartphones and is available via mobile stores or installed directly on target devices.

Charland and LeRoux (2011) posited that, mobile applications can be developed using one of the three main coding strategies. They are Native code, HTML5 code and hybrid code. Developers who design apps for mobile phones confronted with the decision of choosing which approach to implement in their projects. Developers may choose to go native; web or hybrid and each of the approaches have their strengths and weaknesses.

2.5.1. Native Code

A native code mobile application uses the native code of the targeted mobile device operating system platform for development. Table 2.5 shows a list of the native code languages for each mobile operating system. Applications developed with native codes have the best look and performance (Kim et.al ,2015). Native applications have the fastest graphics which is very important when playing graphic intensive games. Hartman et.al, (2011) also highlighted some benefits of developing an application with native code and posited that, native code provides a developer with access to all the built-in features required to provide a user experience that meets the expectation of users of that device. Additionally, native code-based mobile applications require an IDE environment which provides the tools for debugging, managing the project etc. A comprehensive knowledge of the native code language is very important when developing an application. Hence, it is not uncommon for a developer to be limited to just one mobile device platform (Korf and Oksman, 2013). An overview of the native language code fragmentation of the various platforms is illustrated in table 2.5.

Table 2. 5: Native Code Language Fragmentation (Korf and Oksman, 2013).

Mobile OS Type	Native Code Language
Apple IOS	C, Objective C
Google Android	Java
Rim Blackberry	Java(J2ME flavoured)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.Net
Windows 7 Phone	.Net
Hp Palm Web OS	HTML/CSS/JS
Meego	C,C++,HTML/CSS/JS
Samsung Bada	C++

2.5.2. HTML5 Code Mobile Application

Korfs and Oksman (2011) define HTML5 applications as programs that use standard web technologies- HTML, CSS and JavaScript. HTML5 uses the write-once-run-anywhere approach to create mobile applications on multiple mobile platforms. Every web page that works on a tiny screen is considered an HTML5 application.

The greatest advantage of an HTML5 based application is that it can easily be modified to fit several mobile operating system platforms. Most developers are already familiar with using JavaScript, HTML5, and CSS, thus, it is comparatively easier to use it to develop a mobile application. The drawback of developing an application with pure HTML5 coding is that it cannot access all the built in features that are available on a device which is one of key strength of native applications developed with languages such as C. HTML5-coded mobile apps can be developed on any of the cross-platform software development platform such as PhoneGap, MoSync, Apcellerator etc.

One of the most important benefits that are derived from mobile applications having features in common with web applications is that, the developers only need to develop one web page in order to reach out to all of the major smartphone platforms. This is made possible because of the platform independent nature of web pages. In the case where a company wishes to create an app that runs on multiple platforms, creating a mobile web Application greatly helps to reduce the time spent on the development of the app, thereby reducing the overall cost of creating the app. Cantrell (2011) believes developing mobile applications with HTML5 helps the organizations in getting their product ready for market in a relatively shorter time. This aspect can be very important, if the company fancies its chances of gaining market shares in a competitive market.

Reducing time to market of applications helps companies to get advantage over their competitors.

Kao et.al (2011) also analyzed the benefits of HTML5 applications by concentrating the speed of the deployment process of mobile devices and testing on emulators. As mobile web applications are normally not able to enter the native app stores, there exist special app stores where mobile web applications are allowed to enter. These app stores have a fast deployment process since there is no review and approval process whenever a new deployment needs to take place. Compared to the "review and approval" process that Apple has on their App Store, this is a thing to take into consideration if the app is required to be supported on Apples products. The process on the Android Market is not as strict as the Apple App Store, but some work is still required in order to make the app ready to enter the Android Market.

Regardless of the strengths of HTML5 applications, it has some draw backs as well compared to native and hybrid applications. Heitkoter et.al (2012) asserts that, despite the fast deployment that mobile web applications have when eluding the native app stores, it is still a big drawback that they are not guaranteed to enter the native app stores, should they wish to. There are examples of HTML5 apps being rejected from the entering the Apple App Store, but also there are examples where they are allowed to enter. The most significant determining factor is the individual evaluation of the app, and therefore the entrance of an app to an app store cannot be guaranteed.

The approving policy of Apple is stricter than Google but even entrance to Goolge's play store is not automatic for every other application.

Dhillon (2011) on his part gave several reasons why developers are at a disadvantage not to enter the official app stores. First of all, the number of users using these app stores is very high. Both Android apps from Google Play, and apps from Apples App Store surpassed 10 billion downloads going into 2012 (Steve, 2015). This means that developers would be missing a huge prospect to reach customers who only uses native app stores to secure new apps.

Secondly, from a business point of view, Ross et.al (2014) made a comparison of cross platform approaches and asserted that native stores has better business model, than the app stores for mobile web applications. This means that, in native stores it is possible, and very normal, to charge a fee each time a customer downloads the app. Steve (2014) also wrote that native app stores have an advantage in that, customers consider it natural to pay for an application downloaded from this store, where as they are much less likely to pay for a link to a mobile web application. So in order to make profit of a mobile web application, the company who owns it, will mostly have to rely on advertisement in the application, like the ones known from normal websites.

Performance issues also arise when coding in HTML5. Mobile web applications as mentioned earlier are created with the use of HTML5, CSS and JavaScript. All the logic is placed in the JavaScript part where the background work is being processed. Therefore it is in this area, that performance issues can arise. When a company decides to release a product that supports different platforms, it is imperative to ensure a similar customer experience irrespective of device choice. Ideally the performance should be uniform on all platforms. This can be difficult to realize since different browsers are using different JavaScript engines. There exist a lot of different JavaScript engines today. Due to differences in JavaScript engines, some browsers run JavaScript code faster than others (Muchmore, 2013). Companies are therefore unable to provide

guarantees on how their programs are likely to perform on different browsers when the code is written in HTML5.

Full accessibility of device features is another drawback of HTML5 developed applications. Muchmore (2013) stated clearly that there are a lot of features that can only be used through native API calls. Muchmore (2013) again posited that one of the areas where a mobile web application meets its limits is when CPU intensive processes are needed and such areas include hardware acceleration and multithreading. These processes do not run as good as on a native application. Another area that HTML5 shows significant defect is the lack of native look and feel. Users expect more from a smartphone app. Some native UI features are not accessible via HTML5 apps. Mobile application developers are often compelled to mimic the native UI features or totally ignore these features in their apps completely. The UI look and feel challenge has been an issue for many users and developers of mobile web application ever since the release of HTML5 (Charland and LeRoux 2011). New features keep getting supported by HTML5 based apps, but naturally there will always be a certain timeframe, before brand new native features get getting supported by HTML5 based apps. Charland and LeRoux (2011) made the conclusion that native applications will always be one step ahead of their HTML5 competitors in terms of user interface appeal.

2.5.3 Hybrid Code Mobile Application

A hybrid code mobile application is an application that is developed by combining native code and HTML5 code. The purpose of integrating native codes with HTML codes is to leverage on the best of both strategies of app development. Integrating native codes can be used to address the limitations and defects of HTML5 applications. However, a developer would still require native code language skills to successfully implement a hybrid code approach. Just like HTML5

code mobile apps, hybrid apps can be developed on any of the cross-platform SDK, such as PhoneGap, MoSync and Apcellerator.

2.5.4 Summary

Native, HTML5 and Hybrid are approaches to Mobile Application development. Both native and HTML5 approaches have their strength and challenges. Native code applications have access to native API's and provide native smartphone UI users. Likewise, HTML5 applications have their strengths which include one codebase and fast deployment process. The Hybrid code application combines native code and HTML5 code to leverage on both strategies of app development.

2.6 Cross Platform Development

Hietkotter et.al (2012) defined Cross-Platform Development as the process of writing applications for multiple computing platforms. Designing applications for multiple platforms is not a trivial task. Hietkotter et.al (2012) stated that, the earlier mobile application developers adopted cross platform solutions, the more relevant they will be in the mobile application spectrum. Charland and LeRoux (2011) agreed with Hietkotter et.al (2012) and also opined that mobile developers must gravitate towards the development of applications with multi-platform support to guarantee market reachability and organizational profitability (Charland and LeRoux, 2011). To that end it becomes necessary for any application developed for one platform to also be made available to other existing platforms and also afford applications the options to be ported to future platforms (Heitkotter et.al, 2012).

2.6.1 History of Cross platform Development

Java was believed to be marvelous for PC development for allowing a program to be written once and translated into an intermediary language before being run on a Java Virtual Machine

(JVM) on a user's system. This allowed a single program to be run on Windows, Mac and Linux with no platform dependency. Java Micro Edition (Java ME) came to mobile device industry but never went mainstream largely due to its limited capabilities, perceived performance issues, and device fragmentation (Cornelius, 2001). Java was soon abandoned by some handset makers or changed into customized versions with added functionality. Cross-platform tools that extended the Java Mobile architecture had gained some ground with Bedrock, Celsius, NeoMAD and alcheMo entering the market (Rajapakse, 2008). These tools allowed applications to run on a significant number of devices that supported Java, but the tools were outpaced by significant leaps in technology for native platform SDKs. These tools seemed to lose relevance with the advent of the new era of highly versatile smartphones. Since 2011, a new set of CPDTs has been coming to market with new features being added rapidly over time. This latest generation of tools, will be discussed in the next section.

2.7 Cross Platform Mobile Development Tools

There are several tools for cross platform mobile application development available on the market today with various levels of functionality and compatibility. This generation of CPDTs allow more control, functionality and diversity than the previous generation of Java based tools (Rajapakse, 2008). Five tools had been researched as shown in Table 2.5. These tools were chosen based on their flexibility, feature support and popularity among developers.

Table 2. 6: Tools for Cross platform Application Development

CPDT	Blackberry	iOS	Android	Windows Phone 7	Bada
Mobile Web	✓	✓	✓	✓	✓
PhoneGap	✓	✓	✓	✓	✓
Titanium	✓	✓	✓		
Adobe Air		✓	✓		
MoSync		✓	✓	✓	

The CPDTs described in Table 2.6 vary greatly in capabilities, language and platform support. The categories were based on the method used to compile and run applications built with the tools. Some CPDTs utilize a web wrapper to aid the running of application in HTML5 compactible browser objects whereas others such as MoSync can cross compile the application into native code. Traditional runtime and extensions such as those in Adobe Flash are widespread on desktops and notebooks but have limited support on mobile platforms and are becoming obsolete. The alternative is using similar approach to bridge the gap which is found in Adobe Air (Winokur, 2011). Each of these methods have different advantage over purely web-based applications adding possible performance, user interface and notification enhancement not available to web developers. These added capabilities can quickly become an interesting proposition for a developer wishing to have more fully featured applications.

NBC Universal, New York Times and eBay are already developing applications with varied cross platform tools and the development is aimed at reaching a wider market at a relatively reduced cost (Appcelerator Inc., 2012).

Table 2. 7: Features of Cross Platform Development Tools

CPDT	Development Language	Compilation Type	Native UI Elements	Assess to Sensors*	App store support
Mobile Web	HTML5 + Javasript	Runs in browser		Limited	
PhoneGap	HTML5+ Javascript	Native Webapp		✓	✓
Titanium	HTML5+Javascript+ Python/Ruby/PHP	Native code and Runtime	✓	✓	
Adobe Air	ActionScript/HTML/ Ajax/C/C++/ HTML5+Javascript	Runtime	✓	✓	
MoSync	C/C++/ HTML5+Javascript	Native Code	✓	✓	✓

The tools presented in table 2.7 uses different scripting language and provide various features that are not only compactible across all mobile platforms. Table 2.7 again shows many features provided by each tool, including the development language of choice.

2.7.1 Mobile Web

The promise of the mobile web as a standard based architecture is that it displays applications similarly despite being on different platforms and devices. Rich web applications are written using HTML5 and JavaScript. These provide a high degree of functionality like in Google's Gmail web application. This idea is not a new approach and although it does solve many issues

for developers, it does add a few new concerns. Mobile web applications that run through a phone's browser often do not match the phones native UI which can be quite confusing. However, these applications can leverage JavaScript and CSS (Cascading Style Sheets) frameworks such as JQuery Mobile (jQuery Foundation, 2012), Zepto (Fuchs, 2012), and Sencha Touch (Sencha Inc,2012) to provide much needed UI enhancement with little work on the developer's part. Many such frameworks exist to enhance the web development experience and these are only three of them that have been optimized for mobile handsets.

Browser security historically has not allowed storage of local data or access to device sensors such as an accelerometer. The Webinos framework proposed by webinos.org provides an alternative to allow more functionality but does not have widespread adoption. Additionally, the browsers themselves have been fragmented with different implementations of JavaScript and rendering engines which can make functionality and interface behave inconsistently on different devices (Baxter, 2014). The largest concern is often offline access where there cannot be any access to the application when there is no connectivity.

Are these problems a deal breaker in using this as a development environment? The answer is quite subjective but the downsides are growing to be less and less as the technology progresses. HTML5 has allowed for deeper integration with the handset and with WebKit being nearly ubiquitous across handsets, the rendering issues of the past are becoming less of a major consideration (Charland & LeRoux, 2011). Unfortunately, living in the browser still blocks these programs from the application markets which are useful in helping end users find the developers work.

While mobile widgets add some features, it is still largely limited to the browser (Paananen, 2011). An approach taken by many CPDT developers is taking the concept and ideals of widgets and the mobile web and enhancing it with the features that were missing. The next sections will discuss tools that take an integrated approach to cross-platform development.

2.7.2 Adobe PhoneGap

Adobe PhoneGap could be described as an open source framework used in developing native applications with tools such as JavaScript HTML and CSS. PhoneGap was introduced by Nitobi Software Inc. in the year 2008, and its freely accessible under an MIT license (Allen et.al, 2010). PhoneGap makes it possible to develop applications for several platforms including iPhone, Android, WebOS, Symbian WRT and Blackberry (Allen et al, 2010). Support for Windows Phone 7 is planned (Get Started, 2011). PhoneGap is at its best for transforming a mobile web application into native application. It is an advantage to web developers since all the application code can be HTML, CSS and JavaScript. (Allen et al, 2010).

PhoneGap operates with a principle which allows client-side JavaScript APIs to have a method for hosting web applications. Basically a PhoneGap application is native application with a fullscreen browser. PhoneGap cannot entirely match the capabilities of native applications in the sense that, there are some capabilities such as geolocation, camera, accelerometer etc. which cannot be accessed using PhoneGap's JavaScript API.

Developing with PhoneGap starts by writing a mobile web application using HTML, CSS and JavaScript. The content of the application does not have to be in any particular structure. Developers have much choice how to form the mobile web application layout and structure. PhoneGap is at its best on platforms that include the WebKit browser with the advanced

JavaScript and CSS of HTML5, such as iPhone and Android. PhoneGap's goal is to use advanced features of HTML 5 and to implement standards such as W3C Device API Group that defines standards for JavaScript APIs for mobile phone features, such as contacts and camera. Since the W3C Device API standards are not fully developed, PhoneGap contains APIs that diverge from the standard in order to build real, native applications.

Like any other cross-platform frameworks that use the browser for UI, PhoneGap is not well suited for applications that require intense math calculations, 3D animations or data-driven applications, like most enterprise applications that must work offline and synchronize local data. There is no support for database on PhoneGap, instead it relies on HTML5 database APIs that is unavailable for some devices.

In Table 2.6 and Table 2.7 there were illustrations for the wide support for devices and features with PhoneGap. Through an API, developers make use of various JavaScript function underlying native device capabilities. Most notably missing is native UI elements; however this trade-off was made in order for the ability of having it available on so many platforms.

Many 3rd party tools are available to be used with PhoneGap, along with the already mentioned JavaScript libraries; the AppMobi XDK for PhoneGap (appMobi, 2012) provides an IDE and testing environment to aid developers significantly. This comes with tools to emulate the look of the mobile application on a variety of devices. AppMobi additionally provides an analytics platform and its' own build service similar to that provided by PhoneGap directly (AppMobi, 2012). This service allows compilation of applications without the need to install an SDK for each mobile platform. With the addition of this SDK, the PhoneGap development lifecycle begins to resemble that of native applications much more closely.

Another tool that enhances PhoneGap is IBM Worklight (IBM, 2012). It provides additional APIs and server side integration frameworks for enterprise level applications. The architecture of Worklight, shown in Figure 2.3, makes heavy use of PhoneGap to bridge the JavaScript code to the native device. It adds the Worklight API to enhance security, add analytics and access to their middleware, Worklight Server. This server provides the connection to the enterprise backend systems for the mobile application (IBM, 2012).

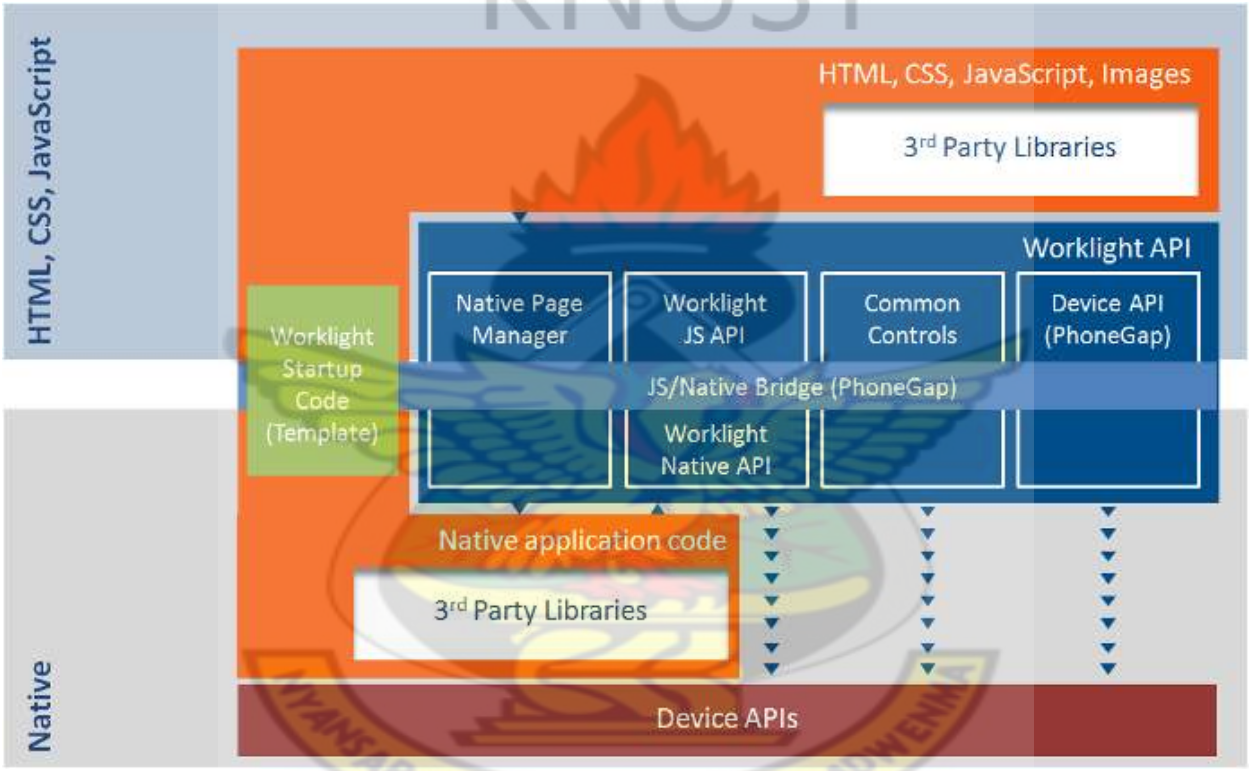


Figure 2. 3: IBM Worklight Architecture (IBM, 2012)

Rather than using the Chrome browser frame based approach as in the appMobi XDK, Worklight has an IDE called Worklight Studio that is based on Eclipse. Once applications are deployed, all analytics and push notifications are handled through the Worklight console (IBM, 2012).

Whether using the PhoneGap directly, appMobi XDK or the suite of Worklight tools, PhoneGap provides an incredibly flexible approach for mobile application development. For small scale applications that need to be completed quickly to large enterprise level applications, there is a variant of this tool suitable for many use cases.

2.7.3 Appcelerator Titanium

Appcelerator Titanium according to Appcelerator Inc, (2012) shares many traits with PhoneGap and the mobile web. Development for all three use standard web development languages, however Titanium differs in some important areas.

Running on an open source core, Appcelerator products allow for applications to run natively on devices without embedding a browser object as can be seen in PhoneGap. Instead, their approach uses a runtime object and compilation method to optimize and compile code. Options are provided to compile and run using a runtime; build into a web application, or a hybrid application similar to PhoneGap. The claim is that performance levels are increased to match those of native applications when using the runtime based approach (Appcelerator Inc, 2012). This assertion by the vendor requires independent analysis through a standard benchmarking architecture to determine accuracy. The UI approach also differs greatly with the use of native UI elements rather than focusing on a purely web-like interface. This is seen as a major advantage by many people as the application will not cause confusion by having a different look and feel than those built using native tools. Others, especially those transitioning mobile web applications, may find this as a negative and find these as constraints to the customizability of the interface (Appcelerator Inc, 2012).

Like the IBM Worklight extension to the PhoneGap core, Titanium provides a host of services for analytics and server side hosting. Similar to Worklight, server side features such as data storage and push notifications are available. The Appcelerator team does not provide the application the same level of end to end security guarantees found with Worklight and is more of a consumer, rather than enterprise focused architecture (IBM Inc, 2012). Titanium includes Titanium Studio, an Eclipse based IDE and a set of testing tools and emulators.

Appcelerator Titanium lacks in platform support with only iOS and Android being fully supported with BlackBerry OS support being put into beta release a year before 2010 (Appcelerator Inc, 2010). This beta was discontinued and replaced with an upcoming BlackBerry 10 OS beta. Supporting only two of the major mobile platforms provides only the minimum required for being called cross-platform and large segments of the market are left out when developing applications using this tool.

2.7.4 Adobe Air

Adobe Air is aimed at building rich internet applications that can be run on many platforms. It uses Adobe Flash, Adobe Flex, HTML, ActionScript and Ajax for development scripting. Flash skills are easily found in the industry and that is perceived as a major advantage for Adobe Air. Existing Flash applications can often be translated to run as native applications using Air instead of being run through a browser extension. Air applications, unlike Flash, require installation of the runtime as well as each application. Mobile support on Android and iOS is present with an independent runtime available on Android and necessary elements being packaged with the application in iOS. As of version 2.6, most features are at par with the Android and iOS versions of the platform. It is suggested that runtimes like these can be an effective way of battling the

cross-platform question however; there has been a mixed reception from device manufacturers (Cantrell, 2011)

Air developer tools are some of the most robust and supported in the industry. Since the tool stems from the popular Flash tool, the same tools used like Flash Builder and Dreamweaver can be used for development. When combined with Adobe Flex technologies, enterprise applications are possible with rich UI capabilities that may surpass those of other platforms. Adobe does not provide the integrated experience and the end to end functionality, tools and services provided by some CPDTs previously discussed.

These runtime based approaches found in Appcelerator Titanium, Rhomobile Rhodes and Adobe Air do have significant drawbacks in some areas. The main drawback is the dependence on the runtime environment itself. The mobile OS makers may at some point block these from their stores and the included runtime can cause larger file size. The features may also lag behind what is released natively as it is integrated into the runtime. As execution often uses just in time compilation, performance may be impacted when using any of these runtimes (Macadamian, 2011).

2.7.5 MoSync

MoSync is much like Air, however, the major differentiator with MoSync to other CPDTs is that it uses cross-compilation where the code is translated into native Objective C and Java code depending on the platform. The C++ compiler used in MoSync outputs an intermediary language that is then optimized and outputted as a binary for the desired platform. This is a different approach that may lead to increased performance of applications. MoSync also offers MoSync Wormhole a PhoneGap like CPDT that uses a browser object to display the application. C and

C++ code can be translated using their wormhole technology to be used in these web based applications.

MoSync offers compatibility with OpenGL ES, which allows 3D graphics for game development therefore unlike many other CPDTs, this is suitable for cross-platform gaming (Vision Mobile, 2012). It is additionally quite extensible allowing the addition of C and C++ libraries to your applications. Cross-compilation can be a very useful technique allowing the ability to deal with compiled native code in the end for increased optimization.

2.7.6 Tool Analysis and Comparison

Choosing one CPDT over another can be a daunting task. Available skills and project requirements will often dictate which tool to use in project execution. The choices are significantly narrowed when special features such as native UI's and the ability for 3D graphics are needed in a CPDT.

Dhillon (2011) came out with the following analysis. According to Dhillon (2011), the results of the evaluation of 4 CPDTs and corresponding native tools provided an assortment of winners and losers. Cross-platform tools performed much better in some situations in the controlled experiments than the native. In other instances, the performance was worse compared to native. According to Dhillon (2011), PhoneGap performed the worst on average and iOS Native was the best on its platform.

The runtime based CPDTs, must include the runtime with the application and in many instances create larger file sizes (Hu & Gadapa, 2005). Cross compilation may be an effective way of completing the task; however, these approaches discussed still have limited APIs. Web based CPDTs like Appcelerator Titanium and PhoneGap offer many features however have limited

platform support in some cases. The lack of native UI elements can be problematic but also provides freedom for custom designed layouts. With all of the tools studied, compatibility to use device functions such as the accelerometer, camera and notifications were present. These provided a much more integrated experience as opposed to mobile web apps where there is still little in the way of integration, particularly with notifications to users.

These CPDTs come with a variety of costs and licenses involved that are rapidly being adjusted. Some such as PhoneGap are free and open source and others like Appcelerator have free and paid tiers. The tiered model with support options in higher tiers is typical and also used by PhoneGap. As the tools have evolved, so have these pricing models which are a moving target.

With each of the cross-platform methods, adequate debugging functionality and documentation were still seen to be lacking or outdated (Paananen, 2011). Currently, many resources are restricted to documentation provided by the tool makers themselves. As result, a full objective comparison is difficult to establish in these early days and requires personal experience with each CPDT.

2.8 Evaluation Criteria for Cross Platform Development Tools

Research available on the comparison of CPDTs is very scanty. Hartmann et.al (2011) and Vision Mobile (2012) performed some comparison on the features of CPDTs. although they lacked greater depth. A 13 item chart was used in Hartmann et.al (2011) to allow comparison of tool features. Storage and camera access among other important features were covered in the survey. However, neither of the reports included performance evaluation or discussion of development practices and detailed costs.

Many CPDTs were discussed in Ohrt and Tarau (2012) but just partial comparison was provided. Their work compared native and web-based user interface elements as well as the importance of well performing applications. However, the authors stated that they were not concerned with the internal workings of the tools and were only concerned with the approval of the application for mobile stores. Ohrt and Tarau (2012) further discussed the lack of debugging tools in many CPDTs in the system currently and provided an 8 point scale to compare features. The authors developed a simple application that provides a screen with a text label and measured the start time and the RAM usage for nine CPDTs. The results were provided as illustrated in Figure 2.4.

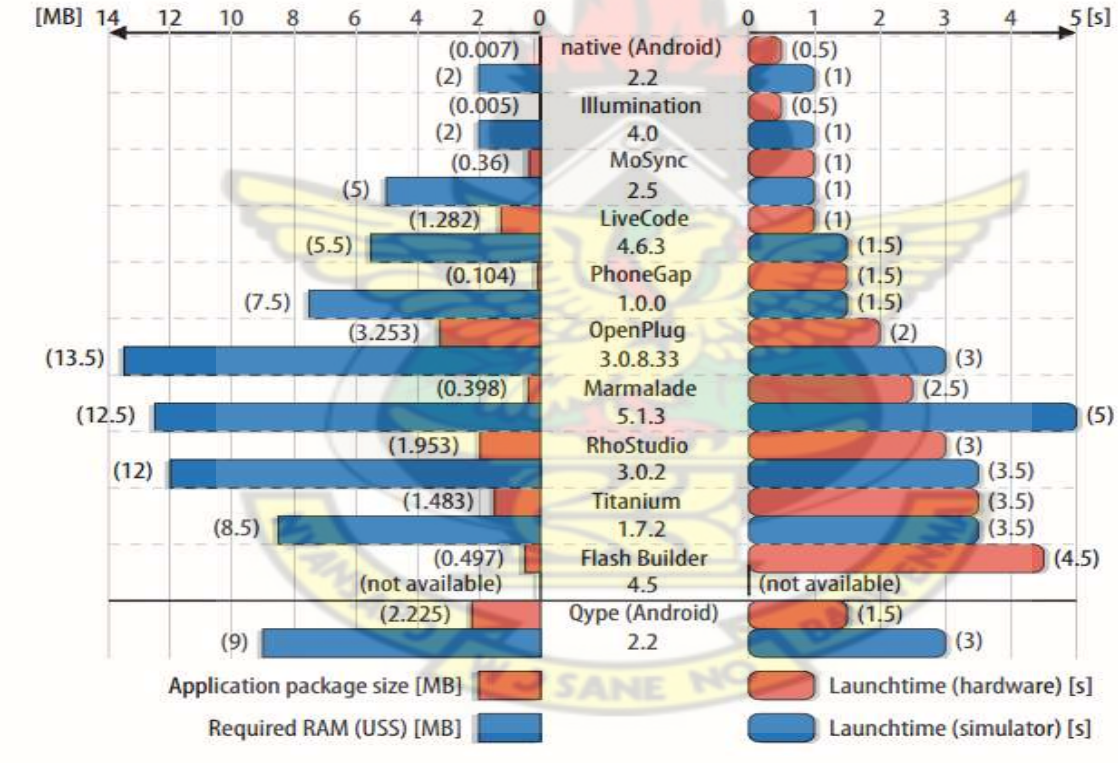


Figure 2. 4: Performance Characteristics of Various CPDTs (Ohrt and Tarau, 2012)

Different approaches have lately been used in the benchmarking of mobile applications. Benchmarking has been in existence for some time now and can be applied to many forms of

computational tasks where the overall capability of a processor or system is calculated and compared using complex benchmarking tools. These consist of a series of intensive tasks that measure the completion time. Currently, the PCMark suite is the most prominent desktop PC benchmarking software and uses several open source and commercial applications (Sibai, 2008). Using the contained test suites allow CPU, memory, graphics and hard disk performance analysis. There had not been any equivalent gold standard among these test suites in the mobile community (Uti and Fox, 2010), although work is in progress on native benchmarks for different platforms. Some currently available benchmarks are Quadrant proposed in Aurora Softworks (2012) and (Antutu, 2012). Published work on mobile benchmarking is very limited. Performing a simple test on Android has been the basis for some time now in terms comparing CPDTs to native tools.

Unlike Quadrant Proposed by Softworks (2012) which works by running through pre-selected algorithms, the TMAPP project running on a netbook based on Meebo OS used Firefox and Open Office to run through scripts. The authors suggested that this method is superior to preselected algorithms as these are real world used cases (Issa, 2011). However, this may be difficult to replicate with the security models on some mobile platforms. Many benchmarks available for PCs cannot be translated over to mobile devices due to restrictions from the operating system, such as running a script of opening and closing multiple applications as done in TMAPP (Issa et.al, 2011).

Bull et.al (2001) also used a benchmarking approach similar to Issa et.al (2011) to compare algorithms written. They used various programming languages. Each language builds a benchmark using standard algorithms that were implemented independently in each language. The resulting tests showed language and compiler efficiencies similar to the way one seeks to

verify CPDT application performance. In Bull et.al (2001) the author used the Linpack benchmark suite as a guideline for the algorithms due to their long study over decades. Various sorting and complex scientific calculations were used in each of these papers to test each language and provide a common line of comparison.

A positive UX is important for all applications. Creating a consistent UX when using cross-platform development can be difficult. In Waljas and Mattila (2009), metrics for strong cross-platform UX were discussed. The authors found that the common themes of usability, performance, social integration and context-aware services were important for users when they are using a similar service on both mobile and desktop terminals. A CPDT that wishes to work across device categories should ensure they have this functionality built in.

The Swerve Studio and X-forge CPDTs were discussed in (Xin, 2009). These tools were focused on cross-platform game development which was shown to have its own set of requirements that differ from those discussed in previous papers.

Finally Dhillon (2011) proposed a cross platform evaluation framework which was based on high level capabilities using criteria such as capabilities, performance and development experience. Based on the framework proposed in Dhillon (2011), the results of the evaluation of 4 CPDTs and corresponding native tools provided winners and losers in separate categories. What this thesis proposes which is different from that of Dhillon (2011) and other researchers is that, mobile software developers will be able to make a situational comparison of platforms based on requirements of specific projects.

Generally, current researches in benchmarking CPDTs have particular limitations. When comparing CPDTs, research has narrowly focused on scales that only incorporate very few

features and does not provide the required scope necessary for decision making. Furthermore, current performance benchmarks are limited to opening an application and do not provide further processing. These evaluation benchmarks are inappropriate in arriving at conclusions on whether there are differences in performance depending on which tool is used by mobile application developers. A more advanced testing and benchmarking framework is necessary to answer the greater questions of which tool could be used to achieve project specifications before the commencement of mobile application development projects.

2.9 Summary

In this chapter many mobile platforms that currently exist in the marketplace and the limitations and challenges over their implementation have been discussed. A new generation of CPDTs has been outlined with some comparison provided.

Benchmarking procedures for mobile devices and applications are still in their infancy; however, some work has been done in Issa et.al (2011) as well as some generally available applications. Investigation of several programming languages and how to adequately compare them are found in literature in Bull et.al (2001) and (Hoste et.al, 2006).

It can be seen from an overview of previous work in this area that only surface evaluation of comparing CPDTs is available. As these tools are fairly new, benchmarks comparing them are not available at this time; however, comparable benchmarking procedures used for other purposes have similarity and can be adapted for this purpose.

In the next chapter an evaluation framework for CPDTs would be presented. The phases of evaluation and components of the framework will be outlined at a high level.

CHAPTER THREE

METHODOLOGY AND FRAMEWORK DESIGN

3.1 Introduction

This chapter considers the methodology employed in this study. Strauss and Corbin (2008) describe a research methodology as a way of studying and pondering over a social reality. It describes the systematic way to solve the research problem (Creswell, 2009). Research Methodology should not be mistaken for research methods since there is a striking difference. According to Creswell (2009), research methods encompass all the techniques used by the researcher in executing research operations including methods of data collection, statistical techniques for analysis as well as the methods used to assess the validity of results obtained. Quantitative and Qualitative approaches are the two main methodologies. Both approaches should not be viewed as opposites; instead they represent different ends on a continuum (Newman and Benz, 1998).

3.2 Research Design

This study employed the descriptive quantitative research design. Quantitative research provides a means of explaining phenomenon by collecting numerical data that are analyzed using methods that are mathematically based (statistics) (Aliaga and Gunderson, 2000). Research is influenced by philosophical worldviews though these worldviews are not explicitly seen in most research works. Underlying paradigms or philosophical assumptions of researchers are necessary and need to be identified in any research work (Slife and Williams, 1995). The underlying epistemology adopted in this study is the positivist pragmatic worldview. The pragmatist approach is flexible and adopts different methods at different stages depending on the research

question under consideration and research objectives (Creswell, 2009). Quantitative research provides a means of testing objective theories by examining causal relationship among variables. These variables can be measured typically on instruments so that numbered data can be analyzed using statistical procedures (Creswell, 2009). The quantitative research methodology was appropriate for this study because the types of data generated from this research were predominantly numbers emanating from experiments which were quantitative in nature. According Sjoberg et.al, (2007) Quantitative methods encompass techniques and designs that produce distinct values or discrete data. Methods that fall into this category include surveys, experiments, case study, grounded theory, action research and design science. Among the various quantitative methods, the Design Science method was adopted due to its appropriateness for the achievement of the research objectives. The processes involved in a Design Science research are illustrated in Figure 3.1.

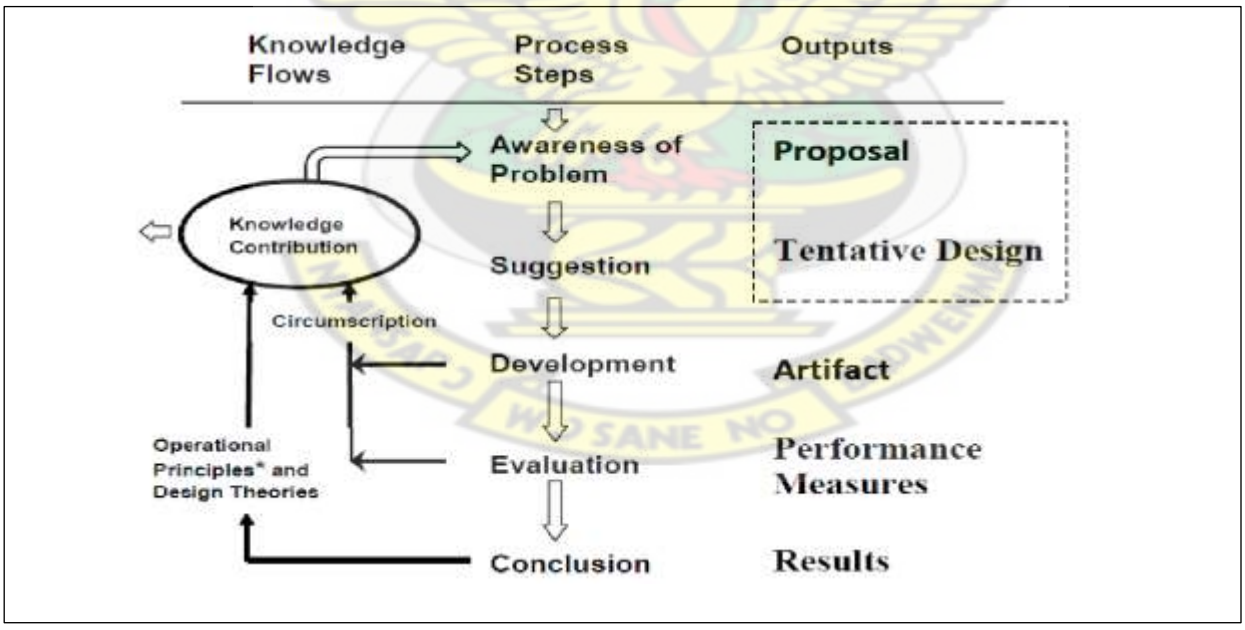


Figure 3. 1: Design Science Research Process Model (Vaishnavi and Kuechler, 2004)

3.2.1: Rationale for Design Science Research Methodology

The original research method considered for this research was case study. However, due to the limitations concerning the generalizations of case study research, the Design Science research methodology which allows for communication of knowledge to both technical and non-technical audience was adopted (Vaishnavi and Kuechler, 2004). The Design Science method fitted into the pragmatic research epistemology adopted for this study (Henver et.al, 2004). This means that the Design Science researcher is considered as a pragmatist. The flexibility of the design science research method, allowing the utilization of diverse methods at certain stages of the research process afforded the inclusion of other relevant methods as and when needed. For Example, the experimental method was used at the evaluation stage to test the hypothesis that emanated from the implementation of the development phase.

3.3 An evaluation Framework for CPDTs

Evaluating CPDTs must be done using a variety of methods. Since there is currently little research available for comparing the capabilities of CPDTs to one another and against native platforms, this framework uses standard practices adapted from other frameworks in similar domains to solve this problem.

This framework aims to ensure a scientific basis for cross platform tool selection. The scope of the framework does not extend to mobile applications that are graphically intensive. The framework will therefore not be ideal for evaluating tools used for game development which would require different evaluation methods. Gaming uses different engines and 3D graphics where frame rates are an important factor rather than the items discussed in this framework. Similarly, this research focuses on the smartphone but is extensible to tablets in future work.

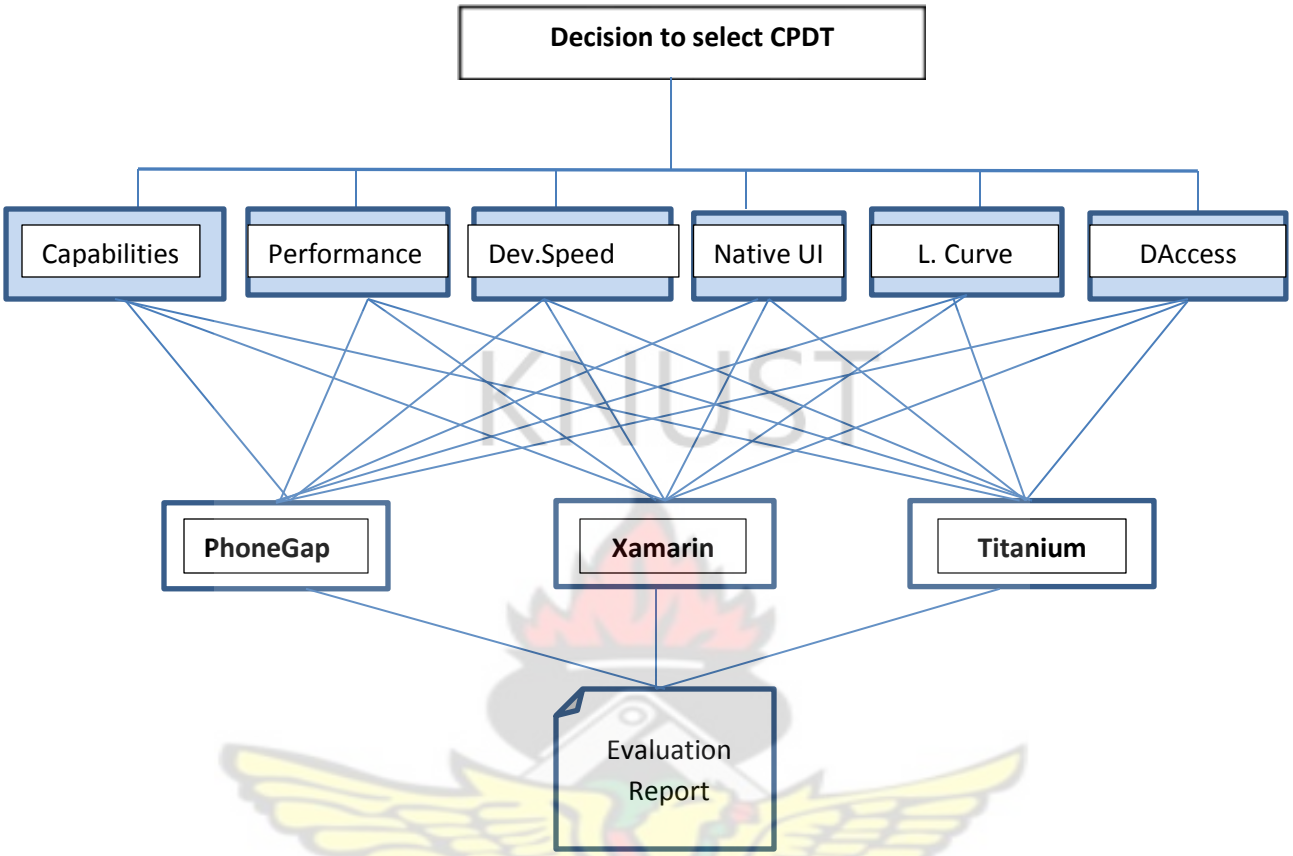


Figure 3. 2: CPDT Evaluation Model

1. L. Curve = Learning Curve
2. Dev.speed = Development speed
3. Native UI = Native User Interface
4. DAccess = Device Access

3.4 Analysis of Framework Design

The evaluation framework consists of four phases as shown in Figure 3.1. The first phase focuses on the problem which is the decision to choose a cross platform tool to implement in a given project. The second phase provides a set of criteria that must be considered in selecting a particular CPDT adapted from the selection criteria suggested by Vision Mobile (2013) in their second quarter report. This is followed by the third phase where various CPDTs are benchmarked in terms of how best they satisfy the criteria outlined in phase 2. In the final phase, more subjective development experience items are discussed based on the criteria in phase 2 in order to provide context and additional information to understand the result. Completing each of these phases provides a firm idea of the capabilities of the studied tool and the outcome summarized in an evaluation report. The following list provides a brief summary of processes involved in the framework:

i. Formulate a decision hierarchy by specifying a hierarchy of interrelated decision elements.
ii. Collect input data by performing a pairwise comparison of each decision element.
iii. Estimate the relative weight of decision elements by using the eigenvalue method.
iv. Aggregate the relative weights up the hierarchy to obtain a composite weight which represents the decision maker's opinion of the relative importance of each decision alternative.

3.4.1 Level 1: Decision to select CPDT

A decision is a result of a comparison of one or more alternatives with respect to one or more criteria that is considered relevant for the decision. Among these relevant criteria, some are considered more and some as less important, a process that involves assigning weights to the

criteria according to their relative importance. For the majority of our everyday decisions which usually have an impact only on us and our immediate future, weights are assigned intuitively based on relevant decision criteria.

Mobile application developers find it difficult reaching a larger market considering the fragmentation of the mobile market. Cross platform development has become the way forward. With cross platform development, there is an array of tools available at the disposal of developers and each of these tools offers different set of functionalities. Developers are confronted with the choice and decision of which platform serves their interest better and satisfies the requirements of the application that they intend to develop.

3.4.2 Level 2: Selection of Criteria for Evaluation

In arriving at the criterion to be included in the framework, relevant data had to be collected from varied sources including primary and secondary sources. The sources including documentation from tool vendors as well as experts in the field of software development helped to gather relevant facts to arrive at the appropriate criteria to include in the framework. Procedures embarked upon to obtain the data are explained in the next section.

3.4.2.1 Sources of Primary Data

The main sources of primary data were eight (8) Software Engineers who were considered as experts in the field of software development. Instruments used in the collection and validation of primary data included focused group interviews and observation using contextual enquiry. Preece et.al (2007) described observation as an effective technique for gathering data and forming requirement definitions at any stage of a research or during a system development. Although the criterion to be determined did not necessarily form requirement, they served as a

basis for inclusion or exclusion. Dix, *et al*, (1993) argued that observation, whether formal or informal, is indispensable if a researcher is to get an understanding of the research situation.

3.4.2.2 Secondary Sources of Data

In the case of secondary data, a desk study was used to gather relevant data from both local and foreign sources such as books, journal articles and conference papers. Also, platform documentations from tool vendors such as PhoneGap, Appcelerator and Xamarin were also explored.

3.4.2.3 Comparison of Criteria

The relative preferences among the various criteria are measured by comparing individual factors against each other in a pairwise comparison matrix. Numerical values expressing a judgement of the relative importance (or preference) of one factor against another had to be assigned to each factor. A comparison scale suggested by Saaty and Vargas (2001) was used to make comparison between factors (criteria). The scale for comparison consisted of values ranging from 1 to 9 which describe the intensity of importance, whereby a value of 1 expresses ‘‘equal importance’’ and a value of 9 is given to those factors having an ‘‘extreme importance’’ over another factor as shown in table 3.1.

Table 3. 1: Scale for Comparison and Explanation (Saaty and Vargas, 2001)

Intensity	Definition	Explanation
1	Equal preferred	Two elements contribute equally to the objective
2	Between Equal and Moderate	
3	Moderate preferred	One element is slightly more relevant than another
4	Between moderate and Strong	
5	Strong	One element is strongly more relevant than another
6	Between Strong and Very Strong	
7	Very Strong	One element is very strongly more relevant than another
8	Between Very Strong and Extremely Strong	
9	Extreme	One element is extremely more relevant than another

The scale indicated how many times an alternative is more relevant than another one, with respect to a specific criterion. The relevance is established according to either subjective or objective statements.

A matrix at this stage will collect the pairwise comparison of the criteria by the decision maker as illustrated in Table 3.2.

Table 3. 2: Pairwise Comparison for the criteria and consistency matrix

	Capability	Performance	Dev.speed	Native UI	L.Curve	DAccess
Capability						
Performance						
Dev.Speed						
Native UI						
L. Curve						
DAccess						

Saaty and Vargas (2001) argue that it is easier and more accurate to express one's opinion on only two alternatives than concurrently on all likely alternatives. It also ensures consistency and cross checking among the pairwise comparisons. This model uses a ratio scale, which, contrary to methods using interval scales. Shizaka & Ashraf (2009) indicated that, no units are required in the pairwise comparison. The judgement is a quotient c / d of two quantities c and d having the same units (grams, meters, utility, etc). A relative verbal appraisal between pairs, similar to what happens in daily conversations is adequate on the part of decision makers during comparison. For the benefit of this thesis, the criteria compared were CPDTs capabilities, performance, developer experience, Native UI look and feel, learning curve and Access to native APIs.

The various criterion used in the framework which provided a basis for comparison is explained in the next section.

3.4.2.3.1 CPDTs Capabilities

Capabilities of CPDT were narrowed down to the features that particular tools can support. The evaluation required the evaluator to go through the documentation and SDK of the CPDT to verify if the features are supported, not supported or partially supported. Within an evaluation of capability, the features to be considered included Availability across platforms, Basic Elements, Security, Notifications and Monetization.

3.4.2.3.1.1 Availability Across Platforms

Several CPDTs including PhoneGap, Appcelerator, Xamarin are available in the market for the design of mobile applications. All these tools are intended at developing multi-platform apps to neutralize the effects of mobile platform fragmentation. Availability across platform considers the capacity of applications to run on different platforms using different operating systems. The wide platform support of an application makes it more marketable.

3.4.2.3.1.2 Basic Elements

The utilization of major cross platform development tools brings some common basic questions such as which mobile platforms are supported and what license the particular CPDT is under. Issues such as initial cost of purchase and ongoing cost during development are considered. Time span for the development processes are also considered since a delay in development affects the time to market of an application.

3.4.2.3.1.3 Security

Security is of concern for most developers to ensure the user data is not compromised and the application is not made available to unauthorized users. Cross Platform Development Tools must

posses methods to securely and safely store information and be retrieved via network connections. Additionally; the CPDT should ensure that when deployed, users do not have the ability to read application source code.

3.4.2.3.1.4 Notification

A notification is a message that can be displayed to the user outside the application's normal UI. These messages are normally sent from a device from external services in order to notify the application or the user of some information or to perform an action. The availability of a method to perform these actions with reliability is necessary for many applications. Platform vendors may provide their own method of sending device notifications and integration with these or the ability to use a 3rd party solution should be specified.

3.4.2.3.1.5 Monetization

In order to enable monetary returns from applications developed, the CPDT must support a variety of features to give developers the flexibility to choose the pricing scheme they consider beneficial. Many free applications may want to use an advertisement platform, as a way of generating revenue either from platform vendors or 3rd party organizations. Paid developers might opt for an outright app purchase. Other monetization models may become common and could be further included. The ability to get the application in users' hands with high visibility through various outlets such as application stores should be included in evaluating platforms on grounds of monetization.

3.4.2.3.2 CPDTS Performance

In testing the ability of Cross Platform Development tools, performance metrics form a significant part in terms of task execution. For this phase the areas that were considered included benchmarks that were Processor Intensive, Data Driven benchmarks and Device Access benchmarks. Pairwise Comparison was done on the alternative CPDTs at level 3 using these benchmarks to evaluate the performance of various platforms against the others.

3.4.2.3.2.1 Processor Intensive Bench Marks

This refers to how CPDTs are able to assist in the implementation of processor intensive applications. An example of a processor intensive application is SunSpider Javascript Benchmark. The aim of using any of these well-studied tests is to stress test the CPU to see if code from cross-platform tools is as efficient to produce required outcomes.

3.4.2.3.2.2 Data Driven Benchmarks

Mobile applications make use of varied data sources, and mostly combine them into reasonable forms for users. At this stage, CPDTs were compared based on how many data sources they supported and the kind of result they produced.

3.4.2.3.2.3 Device Access Benchmarks

Mobile applications possess the ability to capture data from different sensors and store the result. The data can subsequently be used in the application. Comparisons had to be made on the number of devices that the compared platforms are able to support as against the others.

3.4.2.3.3 Development speed

With the goal to become competitive in the market, developers are harnessing efforts to deliver applications to customers within a shorter time. However, developers encounter tough times when it comes to matching with the speed of their competitors predominantly native developers. One of the major advantages CPDTs have over their native counterparts is speed of development in relation the varied platforms they support. Development speed refers to the time taken to complete a particular project with the use of a particular CPDT.

3.4.2.3.4 Native UI look and feel

Various Mobile operating systems such as Android, iOS and Windows come with their native look and feel which mobile application users are already familiar and accustomed to. The alternative platforms comprising PhoneGap, Adobe Air and Titanium are compared to evaluate the extent at which the applications developed from them are closer to native. Features that were compared with regards to native look and feel include UI styles, interactivity, smooth performance and fast response time.

3.4.2.3.5 Learning Curve

Application developers often face the challenge of learning to use new tools at various stages especially when old tools had proved not to be effective or efficient. As technology changes, learning curve has a bigger impact. Every new combination of technology creates a new learning curve and the same applies to CPDTs. Comparison was made on CPDTs to determine how fast a developer can get conversant with the techniques needed develop applications for different platforms.

3.4.2.3.6 Device Access

Mobile devices provide support for different sensors that can gather data to feed into applications as input. New sensors are quickly being added which are quickly adopted by native software development kits such as Android. However CPDTs may not have access to each of these. This section compared what sensors are available in the CPDTs listed in level 3.

3.4.3 Level 3: Alternative CPDTs to be evaluated

In phase III, the thesis compared the CPDTs using all the criteria mentioned in level II. The comparison was done using a criteria matrix with the rows and columns having the names of the CPDTs to be compared. This resulted in 6 different matrices with each matrix representing separate criteria. Table 3.1 shows an example of the matrices used at this stage using the capability criteria.

Table 3. 3: Pairwise Comparison Matrix using the Capability Criteria

Capability	**PhoneGap**	**Titanium**	**Xamarin**	**Priority**	**Geometric Mean**	**Lamdamax**	**ci**	**ri**	**cr**
PhoneGap									
Titanium									
Xamarin									

The priority for the various criteria was calculated out of each matrix. The Geometric mean, Lamdamax and Consistency ratio (cr) were all reckoned.

The **Priority** (a.k.a. normalized, principal eigenvector) column is the relative ranking of the criteria produced by finding the quotient of each element of the matrix and sum of its column.

The average across the rows was also computed. The sum of priority criteria vector was found to be 1. The largest value in the priority weight is the most desirable in terms of the criteria selected.

The **Geometric Mean** is an alternative measure of the **Priority** and was found by taking the nth root of the product matrix of row elements divided by the column sum of row geometric means. The **Geometric Mean** agrees closely with the **Priority**.

Lambdamax is an eigenvalue scalar that solved the characteristic equation of the input comparison matrix. Ideally, the **Lambdamax** value should equal the number of factors in the comparison (n=6) for criteria and (n=3) for alternatives in order ensure total consistency.

The consistency index (**ci**) measures the degree of logical consistency among pair-wise comparisons. The random index (**ri**) is the average ci value of randomly-generated comparison matrices using Saaty's preference scale (Table 3.1) sorted by the number of options being considered.

Consistency ratio (cr) according to Saaty and Vargas(2001) indicates the amount of allowed inconsistency (0.10 or 10%) . Higher numbers mean the comparisons are less consistent. Smaller numbers mean comparisons are more consistent. Consistency ratio above 0.1 means the pair-wise comparison should be revisited or revised.

3.4.4 Level 4: Evaluation Report

At level 4, the results of the individual matrices were interpreted and rankings of the compared platforms made based on the criteria selected. The platform with the highest standardized weight was chosen as the ideal platform for cross platform development.

3.5 Summary

This chapter clearly outlined the methodology and data collection methods adopted for this study. The various stages and levels involved in the development of the proposed framework were clearly spelt out in detail. This included the six (6) criteria considered as basis for the evaluation of the compared tools.

In the next chapter, the framework was implemented to create specific tests that were relevant to three CPDTs on the market today. The experiments that were conducted were outlined and discussed.

CHAPTER FOUR

IMPLEMENTATION OF THE FRAMEWORK

4.1 Introduction

This chapter goes through the detailed explanation and description of how the framework proposed in chapter 3 was implemented in selecting one CPDT among three alternatives namely; PhoneGap, Appcelerator Titanium and Xamarin. The criteria used in the selection included capabilities, performance, Development speed, Native UI look and feel, Learning Curve and Device Access to native APIs such as sensors and cameras.

4.2 Managing Stakeholders

Deciding on how to weight the criteria was important. Many stakeholders involved at various stages in a project cycle might have very different opinions about the importance of a particular criterion. A range of people involved in the project represented such stakeholders; they were mainly Software Engineers. Eight (8) senior developers played the role of stakeholders for the study. The personal experience of the author of this thesis was additionally used to assign weights that were acceptable to all parties. The various stages involved in the implementation of the Framework is illustrated in table 4.1.

Table 4. 1: Summary of the steps involved in the implementation of the framework.

<table>
<tr><td colspan="3">PROBLEM (Level 1)</td></tr>
<tr><td colspan="3">To find the most optimal alternative in the development of cross platform mobile applications</td></tr>
<tr><td colspan="3">CRITERIA (Level 2)</td></tr>
<tr><td colspan="2">Capabilities</td><td>Performance</td></tr>
<tr><td colspan="2">Native UI look and Feel</td><td>Learning curve</td></tr>
<tr><td colspan="2">Device Access</td><td>Development Speed</td></tr>
<tr><td colspan="3">ALTERNATIVES(Level 3)</td></tr>
<tr><td>PhoneGap</td><td>Titanium</td><td>Xamarin</td></tr>
<tr><td colspan="3">PROOF OF CONCEPT</td></tr>
<tr><td colspan="3">EVALUATION REPORT (Level 4)</td></tr>
</table>

4.3 Level 1: Decision of select CPDTs

The main aim of the thesis was to provide a framework to assist mobile application developers in selecting a suitable CPDT for their projects. The first stage of the model stems from the main objective of the study which is the decision to make a selection among different tools in cross platform development. One CPDT needed to be selected among PhoneGap, Appcelerator Titanium and Xamarin.

4.4 Level 2: Selection and Evaluation Criteria

The selection of a suitable CPDT was based on a set of criteria. The set of criteria used at this stage included Capabilities, performance, development speed, Native UI look and feel, Learning Curve and Device Access to native APIs.

After the problem was set up, the relative weight of each of the attributes were determined in order to make a ranking of the criterion in terms of priority.

4.4.1 Weighting the Evaluation Criteria

Table 4.2 shows the relative comparison of the criteria. The relative importance of the elements with respect to project goals was evaluated. The aim was to find the optimal alternative approach for the development of cross-platform mobile applications. The weighing was done using the scale in Table 4.2.

Table 4. 2: Scale for Comparison (Saaty and Vargas, 2001)

Intensity	Definition
1	Equal preferred
2	Between Equal and Moderate
3	Moderately preferred
4	Between moderate and Strong
5	Strong
6	Between Strong and Very Strong
7	Very Strong
8	Between Very Strong and Extremely Strong
9	Extremely preferred

Table 4. 3: Weight of Pair-Wise Comparison of Evaluation Criteria

	Capabiliti	Performance	Dev Speed	Native UI	L.curve	D.Access
Capabilities	1.00	5	3	7	5	5
Performance	0.200	1	0.5	3	2	2
Dev.Speed	0.333	2	1	4	3	2
Native UI	0.143	0.333	0.25	1	0.5	0.5
L.curve	0.2	0.5	0.333	4	1	2
D.Access	0.2	0.5	0.5	2	0.5	1

As shown in Table 4.3, all elements that formed part of the main diagonal (from the upper-left cell to finishing at the lower right cell) were assigned the value 1 since each attribute was relatively compared to itself. By definition, the values of cells above the main diagonal in the table were mathematical inverses of cells that lied below the main diagonal. As an example, because capability was strongly more important than learning curve, the value of 5 was assigned. The relative importance of learning curve got 1/5 which is the mathematical inverse of 5.

4.4.2 Calculating the Priority Vector of the Criteria

The next step was to calculate the priority vector by summing each column and then dividing the relative importance value by the sum of the column. The relative importance value was obtained by computing the nth root of the product of all numbers in a given row. In the last step the average for each row was calculated. Table 4.4 shows the first step of calculating the priority vector. Table 4.5 displays the results after the second step where each entry of the table had been divided by its column sum. This generated the priority vector that was summed up for each top-level attribute also shown in Table 4.5. In relation to the calculation, each value of the priority vector generated results between 0 and 1. Likewise, the sum of all values in any priority vector was equals to 1.

Table 4. 4: Pairwise comparison matrix for set criteria

	Capabiliti	Performance	Dev Speed	Native UI	L.curve	D.Access	product	6th root	priority
Capabilities	1.00	5	3	7	5	5	2625	3.714111	0.457376
Performance	0.200	1	0.5	3	2	2	1.2	1.030853	0.126945
Dev.Speed	0.333	2	1	4	3	2	15.984	1.587136	0.195449
Native UI	0.143	0.333	0.25	1	0.5	0.5	0.002976	0.379266	0.046705
L.curve	0.2	0.5	0.333	4	1	2	0.2664	0.80215	0.098781
D.Access	0.2	0.5	0.5	2	0.5	1	0.05	0.606962	0.074745

4.4.3 Calculating the Lamdamax

The next stage was to calculate the lamdamax. This was computed by summing all the figures in the columns and multiplying the sum by the priority for each criterion. Example, the sum of all the figures in the capability column was 2.08. The priority vector for the capability row was 0.457376. The product of the two figures gave the lamda values for that criterion. A summation of all the lamda figures gave the lamdamax. Figure 4.4 shows a table illustrating the computation of lamdamax.

Table 4. 5: Computing of Lamdamax.

	Capabiliti	Performance	Dev Speed	Native UI	L.curve	D.Access	product	6th root	priority	lamdamax
Capabilities	1.00	5	3	7	5	5	2625	3.714111	0.457376	6.325962
Performance	0.200	1	0.5	3	2	2	1.2	1.030853	0.126945	
Dev.Speed	0.333	2	1	4	3	2	15.984	1.587136	0.195449	
Native UI	0.143	0.333	0.25	1	0.5	0.5	0.002976	0.379266	0.046705	
L.curve	0.2	0.5	0.333	4	1	2	0.2664	0.80215	0.098781	
D.Access	0.2	0.5	0.5	2	0.5	1	0.05	0.606962	0.074745	
sum	2.08	9.33	5.58	21.00	12.00	12.50		8.120479	1.00	
sum*priority	0.949512	1.184776707	1.091189655	0.980801	1.185374	0.934308	6.325962			

4.4.4 Calculating the Consistency Index for the Criterion

The consistency index (ci) was computed using the formula $ci = \frac{lamdamax-n}{n-1}$ where n = number of criteria to be compared. With respect to this problem, the number of criterion is equal to 6, therefore n is equal to 6. $Consistency\ index = {}^{6.325962\ -\ 6}/_{5} = 0.0651924$.

4.4.5 Calculation of Consistency Ratio for the criterion

The consistency ratio was also arrived at using the formula $cr = \frac{ci}{ri}$ where cr = consistency ratio, ci = consistency index and ri = random index. The rational index is obtained using the rational index table as illustrated in Table 4.6.

Table 4. 6: Table of Random Index (ri) (Saaty and Vargas,2001)

n	Random Index(ri)
1	0.00
2	0.00
3	0.58
4	0.90
5	0.12
6	1.24
7	1.32
8	1.41
9	1.45

Looking at table 4.6, the corresponding value for the random index was 1.24 considering the number of criteria compared which was equal to 6.Therefore, consistency ratio was computed as

$cr = \frac{0.0651924}{1.24} = 0.052575$

4.5 Level 3: Developing Ratings of each decision alternative

At this stage, the individual criteria were used to develop matrix to determine the ranking of the three alternatives namely; PhoneGap, Titanum and Xamarin. With this, 6 different 3×3 matrix were developed which are illustrated in the Tables 4.7 to 4.12.

Table 4. 7: Comparison Matrix of CPDT within the capability criteria

Capabilities	Phonegap	Titanium	Xamarin	3rd root of product	priority	lamdamax	ci	ri	cr
PhoneGap	1.00	5.00	2.00	2.15443469	0.581554	3.00366242	0.001831	0.58	0.003157
Titanium	0.2000	1	0.3333	0.405466617	0.109449				
Xamarin	0.5	3.000	1.000	1.144714243	0.308997				

Table 4. 8: Comparison Matrix of CPDT within the performance criteria

Performance	PhoneGap	Titanium	Xamarin	Priority	Lamdamax	ci	ri	cr
PhoneGap	1.00	0.20	0.33	0.109126	3.001498763	0.000749	0.580000	0.001292
Titanium	5.0000	1.00	2.0000	0.58177				
Xamarin	3.0000	0.50	1.00	0.309109				

Table 4. 9: Comparison Matricx of CPDT within the development speed criteria

Development Speed	PhoneGap	Titanium	Xamarin	Priority	Lamdamax	ci	ri	cr
PhoneGap	1.00	3.00	5.00	0.648329	3.003694598	0.001847	0.580000	0.003185
Titanium	0.3333	1.00	2.0000	0.22965				
Xamarin	0.2000	0.50	1.00	0.122020				

Table 4. 10: Comparison Matrix of CPDT within the Native UI look and feel criteria

Natuve UI	PhoneGap	Titanium	Xamarin	Priority	Lamdamax	ci	ri	cr
PhoneGap	1.00	0.17	0.25	0.088983	3.009202713	0.004601	0.580000	0.007933
Titanium	6.0000	1.00	2.0000	0.58763				
Xamarin	4.0000	0.50	1.00	0.323386				

Table 4. 11: Comparison Matrix of CPDT within the Device Access Criteria

Device Access	PhoneGap	Titanium	Xamarin	Priority	Lamdamax	ci	ri	cr
PhoneGap	1.00	0.17	0.33	0.095338	3.018294707	0.009147	0.580000	0.015771
Titanium	6.0000	1.00	3.0000	0.65481				
Xamarin	3.0000	0.33	1.00	0.249856				

Table 4. 12: Comparison Matrix of CPDT within the learning curve criteria

Learning Curve	PhoneGap	Titanium	Xamarin	Priority	Lamdamax	ci	ri	cr
PhoneGap	1.00	5.00	7.00	0.739594	3.014151882	0.007076	0.580000	0.0122
Titanium	0.2000	1.00	2.0000	0.16659				
Xamarin	0.1429	0.50	1.00	0.093813				
sum	1.34	6.50	10.00	1.000000				

4.5.1 Weighted average rating of each decision alternative

The weighted average rating of each decision alternative was achieved by first multiplying the criteria weights from level 2 by the rating of the decision alternatives of each criteria and finally summing up the respective products as shown in Table 4.13

Table 4. 13: Weighted Average rating of each alternative.

criteria / options	Capabilities	Performance	Dev. Speed	Native UI	Learning Curve	D.Access	Score	
	0.457376	0.126945	0.195449	0.046705	0.098781	0.074745	1.00000	
PhoneGap	0.885246	0.109126	0.648329	0.088983	0.739594	0.095338	0.629798	Winner
Titanium	0.004098	0.58177	0.22965	0.58763	0.16659	0.65481	0.213457	2nd
Xamarin	0.110656	0.309109	0.12202	0.323386	0.093813	0.249856	0.156746	3rd
sum	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	

4.6 Proof of Concept Application Development

4.6.1 Introduction

Based on the outcome of the evaluation which had PhoneGap emerging as the overall winner among the CPDTs compared, the researcher decided to develop an application using PhoneGap to verify whether it provides the features as described in chapter 4 and 5. The application which is a multiple choice application deployed on mobile platforms was named “EDU GLOBAL QUIZ”.

4.6.2 Features and Functionality of the Mobile Web Application

The application designed as a proof of concept app was a multiple choice quiz application which allowed users to select answers from a list of options. The app provides scores to users after completing all questions. The features that were implemented included the following:

1. Starting the application

2. Reading instructions for the quiz

3. Selecting answers from a list of options

4. Navigating to the next question using buttons

5. Viewing scores after completing the quiz

Functionality was kept as easy as possible, since the application was also targeted for smartphones that did not have a large screen space. Only one button was implemented to allow users navigate from one question to the other.

4.6.3 Structure of the Application

The structure of the application focused on the implementation requirements that were required by the PhoneGap framework that was used to develop the application. The framework consisted of the following file structures which was customized in order to cater for the Mobile Quiz app:

- CSS – a folder used to store the framework for CSS and images
- JS - a folder used to store javascript files including jQuery libraries.
- index.Html – an index page which loads when the application runs
- logo.gif - an application logo displayed at the top of the page when the application loads

4.6.4 Technical Environment

The application developed will be deployed on Android and Windows platforms. These two platforms were chosen because of its popularity and availability. The cost of deploying to app stores for these two platforms was also free.

4.6.5 Development Tools

No special tools were needed to develop the multi-platform mobile application. In fact the only tools that were used to develop the application were a webkit browser name Chrome and a text editor called Notepad++. The benefits of using the Chrome browser were that its default debugging tools were enough to test and analyze the application code. Also, the Chrome browser supported WebSQL that was used as a persistent storage for the mobile web application created. The notepad++ tool due to its syntax highlighting feature helped to highlight various codes such as HTML, JavaScript or CSS which facilitated the development of the Mobile Web Application.

4.6.6 Application Design

PhoneGap was used to make the transition from the native mobile languages to web-based programming languages in both Operating Systems. The backend code of every platform was written in native language. As seen in figure 2.3, the PhoneGap framework bridges the gap between the native language and HTML by working as a wrapper for the app and generating Javascripts used in the app for accessing the native API. Due to jQuery and jQuery Mobile the application is written in one HTML file thus allowing every page to be within a div tag and identifying them with different ID's when used to navigate through the app.

4.6.7 Testing the Application

During the process of developing the application, the functionality and operation of the app was tested in two ways. One way was to use simulators integrated into the IDEs of native platforms. Another way was to use real mobile devices. However both methods were usually combined at various stages of the development. The reason for using the simulators were because of the issue of cost (in most cases completely free) and they could sometimes be used to test when devices are not available. Another benefit of simulators was that it was easy to get started with the testing process, by just downloading the simulator. The main drawback of simulators was the fact that the testing was not performed on the real platform, and it did not provide a guarantee that the app would actually work on the target devices. There were also some performance differences between the simulators (considering the hardware of the computer) and the physical mobile devices. The app most likely performs differently in a simulator, on a computer with more CPU power and more memory, than on a smartphone. The advantage of testing on real mobile devices is of course that the testing is done on the devices the app is supposed to run on.

4.6.8 Test Results

The test results showed that the current state of PhoneGap compiled applications developed with HTML5, CSS3 and JavaScript on mobile devices is getting better and better. From the Checklists designed for each device that was tested, it was evident that majority of the tests were passed. For the Lenovo Phone running the android operating system, the application passed 9 out of the 10 tests representing 90% whereas the Nokia Lumia 520 phone running the windows 8 operating system also passed 8 out of the 10 tests representing 80%. The mobile OK Checker also recorded 87% which equally gave ample evidence to the seeming readiness of PhoneGap aided by

HTML5, CSS3 and JavaScript in cross platform development. The results of the various tests that were conducted and UI on the various platforms can be found in the Appendix.

4.7 Summary

This chapter described in detail the processes that were followed to implement the framework proposed in chapter 3. It outlined the ranking of the criterion used in evaluating CPDTs and also provided figures to support the ranking of the three CPDTs compared in this thesis.

While the framework allows the criterion to change over time according to requirements of specific mobile application development projects, the selected parameters in this chapter were used to conduct the testing of the outlined tools, of which the results were found in this following chapter.

CHAPTER FIVE

RESULT, DICUSSSION AND CONCLUSION

5.1 Introduction

By implementing the individual stages in the CPDT evaluation framework, each phase of evaluation provides significant information regarding the tools strengths and weaknesses. These results are presented in this chapter. The results show the relative importance of each of the criterion and their subsequent ranking derived using the framework.

The alternative tools compared; namely PhoneGap, Titanium and Xamarin had to be ranked based on the results derived from chapter 4 during the implementation of the evaluation framework.

5.2 Ranking of CPDT Evaluation Criteria

The result for the ranking of criteria used in the study was given as illustrated in Table 4.5. In the table, it is evident that, the capability criteria received the priority vector of 0.457376 which meant that, capability is the best ranked criteria with respect to the choice of CPDT. In terms of percentage, the capability criteria recorded 46% which meant it was just 4% shy of getting a 50% mark. Development speed was the next criterion in terms of ranking based on priority figures. It recorded 0.195449 with a percentage of 20%. Performance, Learning curve and Device access also recorded 0.126945, 0.098781 and 0.074745 respectively. The least recorded criteria in terms of ranking was Native UI which had a figure of 0.046705 representing 5%. The significance of this ranking is that, mobile applications developers should consider the capabilities of the application and make it a priority during development before any other consideration. If there should be some tradeoffs, the criteria to be sacrificed should be the least in the ranking which in

this case should be native UI look and feel. Figure 5.1 illustrates the priority and percentages of the various criterion extracted from Fig 4.3.

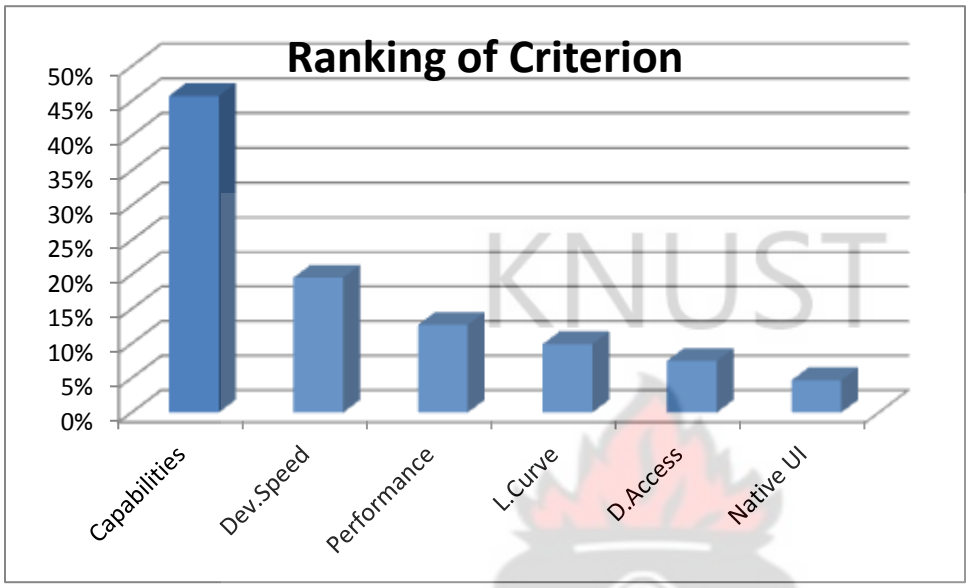


Figure 5. 1: Ranking of CPDT Evaluation Criteria

5.3 Ranking of CPDTS based on Capability

The result from the ranking based on capabilities was listed in Table 4.7. It was evident that PhoneGap received the best result with a ranking vector of 0.581554 representing fifty eight percent whiles Xamarin came second in terms of the capability criteria with a ranking vector of 0.308997 representing thirty one percent. The last CPDT in terms of capabilities was Titanium with a ranking vector of 0.109449 representing the least percentage of eleven percent. Figure 5.2 shows the ranking of the three (3) tools compare based on capability.

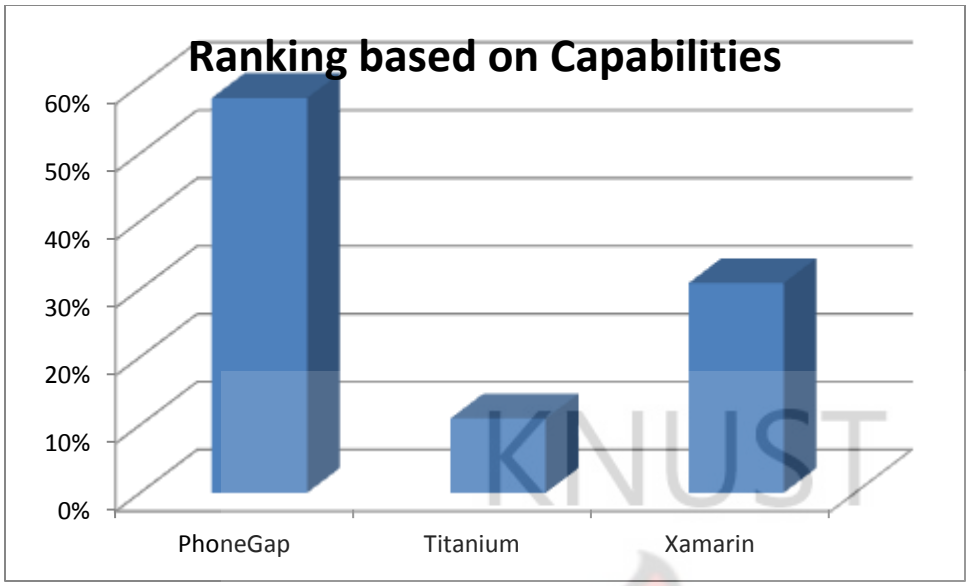


Figure 5. 2: Chart Showing Rankings of CPDT Based on Capability

5.4 Rankings of CPDT based on Development Speed

The result from the ranking based on development speed was listed in Figure 5.3. Once again, PhoneGap received the best result with a ranking vector of 0.648329 representing sixty five percent whiles Titanium came second in terms of the development speed criteria with a ranking vector of 0.22965 representing twenty three percent. The last CPDT in terms of development speed was Xamarin with a ranking vector of 0.122020 with the least percentage of twelve percent.

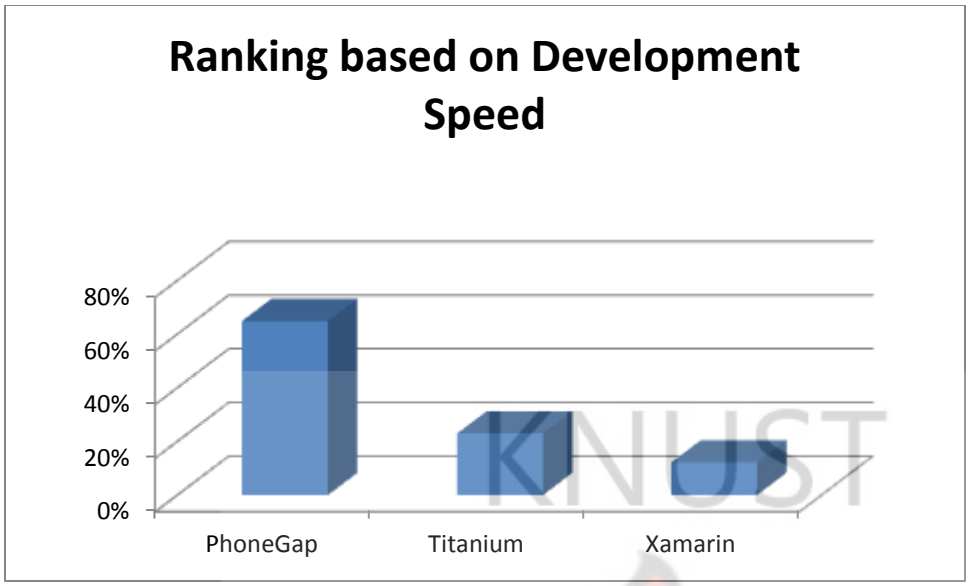


Figure 5. 3: Chart Showing Ranking of CPDTs based on Development Speed

5.5 Ranking of CPDTS based on Performance

The result from the ranking based on performance was illustrated in Figure 5.4. Titanium this time around gained a highest priority vector of 0.58177 representing fifty eight percent. Xamarin came second in terms of the performance criteria with a ranking vector of 0.309109 representing thirty one percent. The last CPDT in terms of performance was PhoneGap which ironically attracted the highest priority vector in terms of capability and development speed. PhoneGap gained the least priority vector of 0.109126 representing eleven percent. The lower priority of PhoneGap in relation to performance might be attributed to the lack of native capabilities such as UI elements as suggested by (Dhillon, 2011).

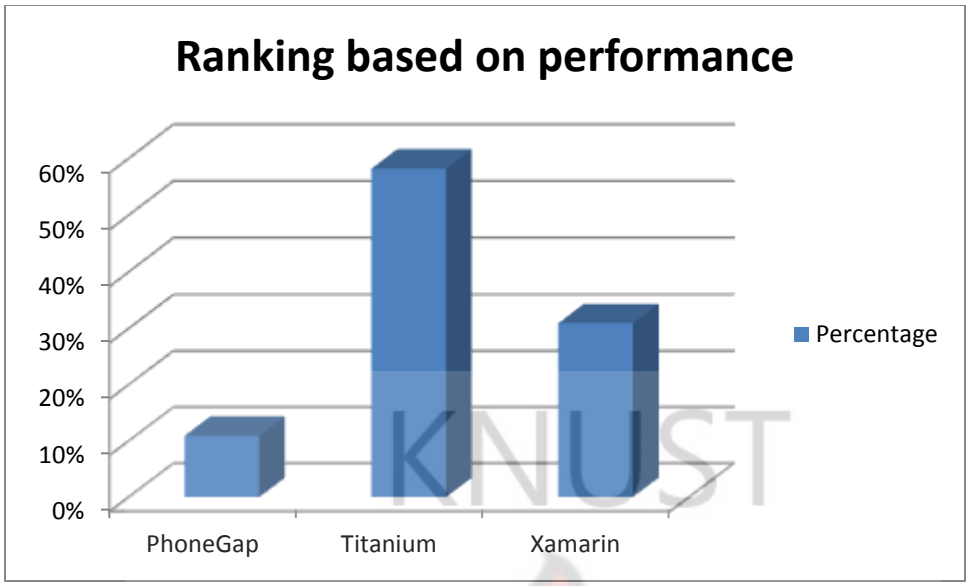


Figure 5. 4: Chart showing ranking of CPDTs Based on Performance

5.6 Ranking of CPDTS based on Learning Curve

The result from the ranking based on learning curve was illustrated in Figure 5.5. PhoneGap for the third time out of four criterions presented so far gained a highest priority vector of 0.739594 representing Seventy four percent. Titanium came second in terms of the learning curve criteria with a ranking vector of 0.16659 representing seventeen percent. The last CPDT in terms of learning curve was Xamarin which gained the least priority vector of 0.093813 representing nine percent. This results further confirm the ease of use of Javascript and HTML in development; the development platform for PhoneGap which most developers are already familiar with. This result agrees with the results of (Paananen, 2011).

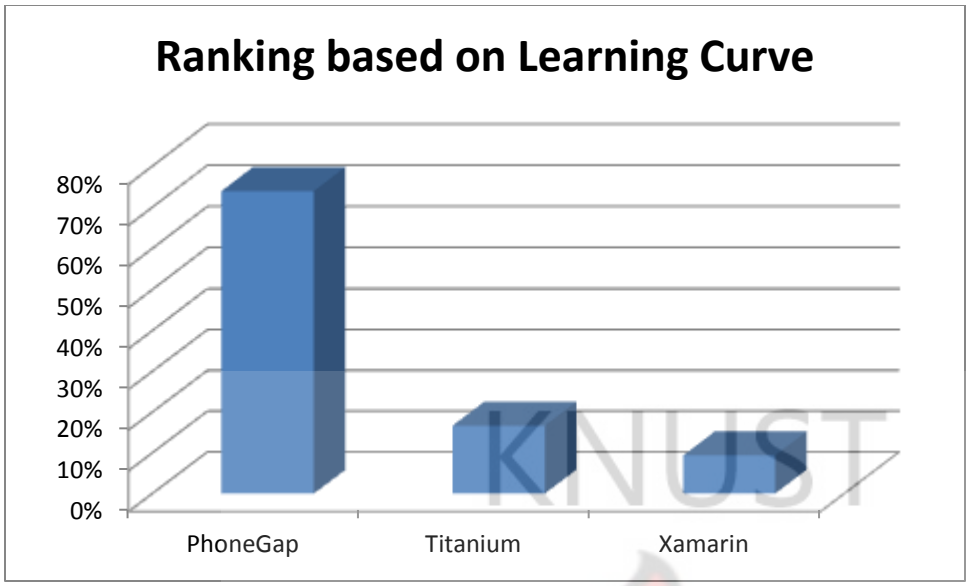


Figure 5. 5: Chart Showing Ranking of CPDTs Based on Learning Curve

5.7 Ranking of CPDTS based on Device Access

The result from the ranking based on device access was illustrated in Figure 5.6. Titanium gained a highest priority vector of 0.65481 representing Sixty five percent. Xamarin came second in terms of the learning curve criteria with a ranking vector of 0.249856 representing twenty five percent. The last CPDT in terms of device access was PhoneGap which gained the least priority vector of 0.095338 representing ten percent.

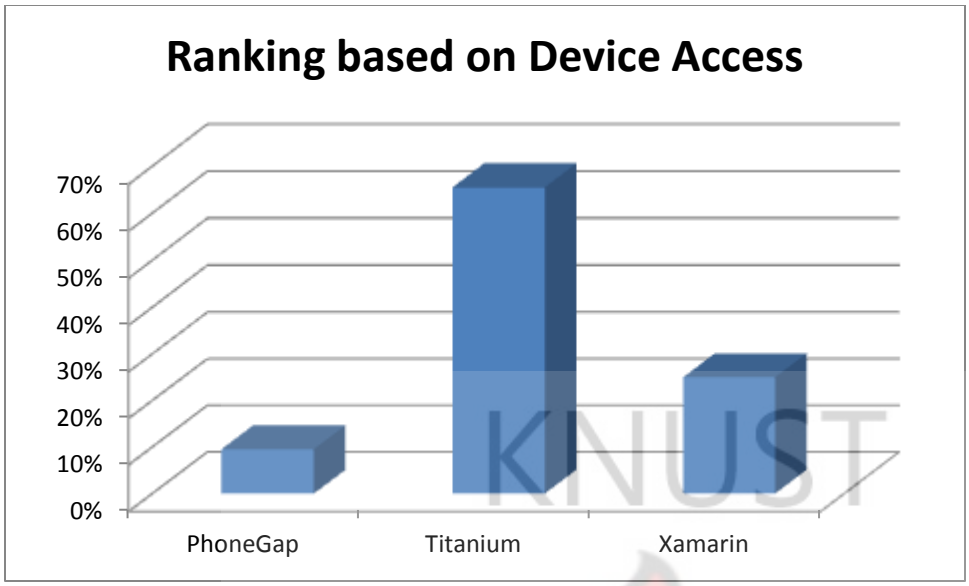


Figure 5. 6: Chart Showing Ranking of CPDTs based on Device Access

5.8 Ranking of CPDTS based on Native UI Look and Feel

The results from the ranking based on Native UI look and feel were listed in Figure 5.7. It was evident that Titanium received the best result with a ranking vector of 0.58763 representing fifty nine percent whiles Xamarin came second in terms of the native look and feel criteria with a ranking vector of 0.323386 representing thirty two percent. The last CPDT in terms of capabilities was PhoneGaP with a ranking vector of 0.088983 and the least percentage of nine percent. This means that, if look and feel is the mean reason for developing an app, then PhoneGap will not be the platform of choice but Titanium should be the most preferred among the three compared platforms.

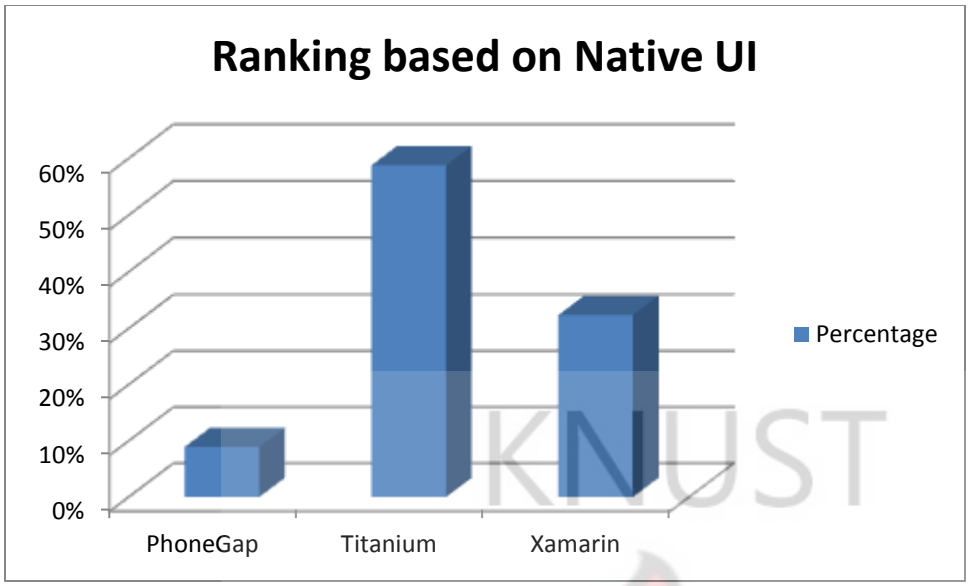


Figure 5. 7: Chart showing ranking of CPDTs based on Native UI

5.9 Ranking of CPDTs based on all criteria evaluated

Considering the entire six selected criteria for the three platforms compared, PhoneGap emerged as the overall winner with the highest weighted average rating of 0.629798 representing Sixty three percent (63%) whereas Titanium placed second with a weighted average rating of 0.213457 representing twenty one percent(21%). Xamarin recorded the least weighted average rating of 0.156746 representing sixteen percent (16%). Unlike Dhillon (2011) which could not pronounce a clear winner after the evaluation, this thesis uses the weighted average ratings of each CPDT to declare PhoneGap as the winner among the three compared platforms. This is illustrated in Figure 5.8.

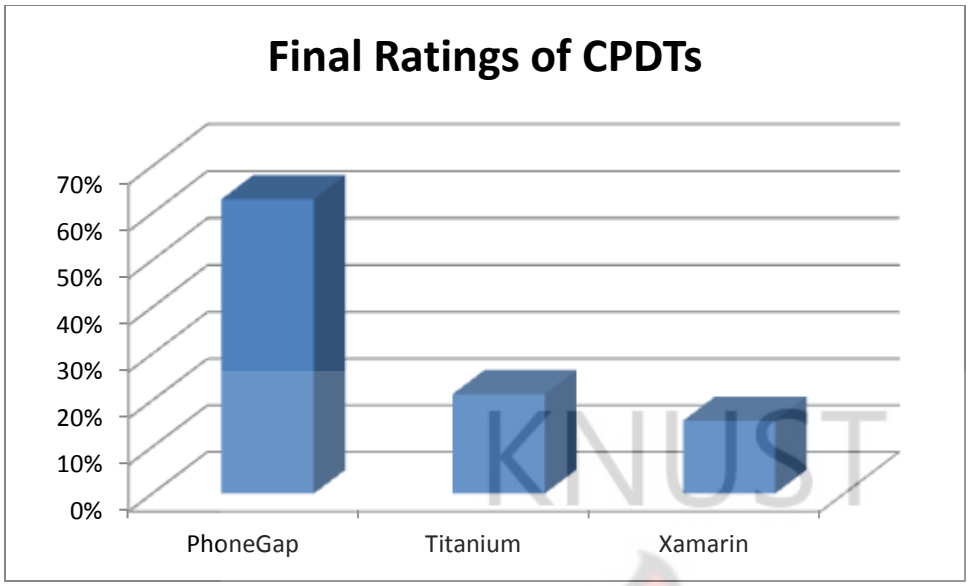


Figure 5. 8: Chart showing ranking of CPDTs based on entire evaluation criteria

5.9 Analysis of results

The results of evaluation of CPDTs provided winners and losers based on different evaluation criterion. In some instances, some CPDT performed much better relative to specific criteria than others. PhoneGap for instance recorded highest ranking vector in terms of capability, learning curve and development speed. However, it recorded the least ranking in terms of performance, device access and native UI look and feel. The high scores might be attributed to the fact that, PhoneGap uses purely HTML and Javascript code in its implementation which gives it enormous advantage over others in areas such as number of platforms it is available to, familiarity of most developers with HTML/Javascript and most browsers recognizing the HTML format. The low mark of PhoneGap in device access, performance and native UI was also due the closeness of the two other platforms namely Titanium and Xamarin to native SDKs.

The most important criterion was shown to be capability followed by development speed. Learning curve, Performance, Device Access and Native UI look and feel also followed

respectively. This means that, the most important factor to consider in selecting a CPDT for a project is capability which has many important elements such as availability across platforms which is the essence of cross platform application development. Native UI look and feel is least considered since it has no effect on portability.

PhoneGap was shown to be the most capable cross platform development tool. It was also the easiest to learn and the fastest platform that can shorten the time to market of applications. However, it has some downsides in terms of performance, device access and Native UI look and feel as discussed earlier. Titanium also required modification as the UI must be independently developed for both Android and iOS. This however, was seen to provide better performance in comparison to PhoneGap which provided the most portable code.

UI intensive tasked favours Titanium since it provided the best native look and feel. It is also good for applications with complex computation and calculation since it provided the best performance. However, the downside of Titanium is its inability to reach a wider audience as compared to PhoneGap.

Xamarin fell short in terms of most the criterion evaluated. Xamarin therefore might not be the preferred tool for developing applications to target many audience since PhoneGap does that well. It also does not match up with the robustness of Titanium in developing complex, high performing application for smaller audience although this may change as the tool matures.

Finally, the results of this evaluation have shown that, the preferred CPDT for mobile application development is PhoneGap considering the criterion indicated in this thesis.

5.11 Conclusion

Mobile application development is recently considered as fast growing field due to the pervasiveness and ubiquitous nature of mobile devices. Developers in this area are however confronted with the challenge of operating system fragmentation. A life line seems to have emerged with fingers pointing to the direction of cross platform development. Cross platform mobile application development looks promising with several prospects including the ability to write once and run anywhere, reduction in learning curve and reduction in development cost among others.

Despite the prospects of cross platform development, developers are confronted daily with the decision of which cross platform tool to choose in the implementation of their projects while keeping a certain level of performance and functionality. Through development of a framework to facilitate the selection CPDTs in this thesis, a solution to this pertinent tool selection dilemma is not farfetched. The framework was subsequently implemented on three (3) different tools. The result had shown that the framework provides the details needed to answer this question through its benchmarking and evaluation processes.

The result of the thesis has shown that tool selection can have a causal effect on development of a mobile application. Some CPDTs were shown to have performance issues while others provided too little capability. The most attractive part of this framework is its ability and flexibility to extend to include new criterion brought by future releases of CPDTs with the core concepts remaining. It has become evident that it is important for developers to choose the right tool for application development and this framework provides enough detail on the CPDTs tested. This affords developers ample resource to choose which tool best fits the purpose they intend to achieve and which criteria to use in evaluation these tools.

The contributions of the thesis are:

i. A high level extensible framework for evaluating CPDTs
ii. An implementation guideline of the framework with specific criterion that can be used for various CPDTs.
iii. A ranking of various criteria for evaluating CPDTs
iv. A ranking of CPDTs based on six major criteria
v. Provision of an independent evaluation of CPDTs with emphasis on their strengths and weaknesses

Through the implementation of the cross platform evaluation framework developed in this thesis, it was evident that some features provided by CPDTs are more significant than others. And they are the most likely to appear in requirement definitions of mobile application development projects. Among the features evaluated which were presented in this thesis as criteria, the most significant was the capability of the tool. Also it became evident that various CPDTs have their strengths and weaknesses. For example, PhoneGap is strong in terms of capability, learning curve and development speed whereas Titanium and to some extent Xamarin provides better performance, device access and native UI look and feel. Putting all the criteria together, the platform of choice with respect to the three compared alternatives was PhoneGap. The development process was considered to be very fluid for mobile application development and it seems that will continue to be true over the next few years.

Using this framework and consistently updating criteria in accordance with current trends and market demands will help developers overcome the decision problem of choosing a CPDT. With the number of platforms available, developers must be very versatile and businesses need to

commit significant resources to have their applications and make them available on more than one device. Interest in the usage of cross platform tools is only set to grow making this framework necessary and timely.

5.12 Future Work

This thesis has been successful in providing a pragmatic selection framework to evaluate cross-platform tools from many angles, but there is room for evolution. The number of criteria used in the evaluation can be broadened to involve several aspects of a CPDT such as user experience, and extent of platform documentation which in this thesis to some extent was captured under capability. The alternatives can be extended to include more than three CPDTs. Sub criteria under the main criteria can also be considered and treated as a different level. The framework can be revised to test other uses such as game development which was out of scope of thesis.

REFERENCES

Adrian, H. and Ondrus, J. (2011) "Mobile application market: A mobile network operator's perspective".

Agarwal, V., Goyal, S., mital, S. and Nukherjea, S. (2009) "A Middleware Layer to Gandle Fragmentation of Platform Interfaces for Mobile Applications," in Middleware '09: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware.

Allen, S., Vidal, G., Lee, L. (2010) "Phonegap. Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution*"* (1:Ed) Berkely, CA, USA: .(pp21-36) Apress.

Anderson, R. and Gestwicki, P. (2011) "Hello, worlds: an introduction to mobile application development for IOS and Android" *Journal of Computer Science.*

Aliaga, M. and Gunderson, B. (2000). "Interactive Statistics" Saddle River, p3-15

Antutu Hong Kong (2012): Antutu Benchmark: " know your Android better" [Online].Available from: http://www.antutu.com. [Acessed: 27th November 2014].

Anvaari, M. And Jansen, S. (2010) "Evaluating architectural openness in mobile software platforms". In *Proc. ECSA, 10* (pp. 85-92). ACM.

Appcelerator Inc. (2012) Appcelerator research [Online] Available from http://www.appcelerator.com. [Accessed: 29th June 2014]

Aurora Softworks. (2012): "Aurora Softworks Home: [Online] Available from http://www.aurorasoftworks.com [Accessed 29th July 2014]

Appmobi. (2012). HTML5 App school Webinar: *Track your app's usage using appMobi's stat Mobi analytics* [Online] Webinar Available from http:// www.youtube .com/watch?v=IU&NJY^RYQE.[Accessed: 16th September 2014]

Ballon, P (2009) "Mobile application development: a developer's perspective*"* [online] available from http://www.janondrus.com/wp-content/uploads/2008/05/telematics2011.pdf.[Accessed: 29thJune 2014].

Baxter, R (2014) "The Basics of Javascript Framework SEO in Angular JS" [Online] Available from http:// www.builtvisible.com [Accessed 10th March 2014]

Behrens, H (2010) "Crossplatform App Development for iPhone, Android and Co: A comparison in MobileTechCo", Mainz.

Bryman, A. and Bell, E. (2007) "Business Research Methods" (2nd ed), Oxford University PressInc., New York

Brunning, M (2011) " Native Cross platform Mobile Application Development using Voind". pp 15-25

Brynes, S (2011). 1.2 billion Mobile apps were downloaded over the holidays. [Online] Available from: http://techcrunch.com/2012/01/02/1-2-billion-apps-holidays/ [Accessed: 19th May 2014]

Bull, J, M, Smith L, A. Pottage, L. and Freeman, R. (2001): "Benchmarking Java against C and Fortran for scientific applications," in Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande, 2001, pp. 97-105.

Burnette, E. (2009) "Hello, Android: Introducing Google's Mobile Development Plat- form" (2nd ed) Pragmatic Bookshelf (pp 19-27)

Burns, M., Boelin, E. and Peterson, J (2012) "Customer Experience Index" [Online] Available from http:// http://www.forrester.com [Accessed 12th June 2014]

Byrnes, S (2014) "Mobile Metrics, like the web but a lot harder: Challenges of Mobile analytics" [online] Available from radar.Oreilly.com [Accessed: 10th March 2014]

Cantrell, C. (2011): "iOS features in Adobe Air" [online] Available from http://www.adobe.com/devnet/air/articles/ios_features_in_air26.html.[Accessed: 5th March 2015]

Charland, A. and LeRoux, B. (2011): "Mobile application development: Web vs Native" Queue vol 9. Pp 21-30

Chudnov, D. (2010): "A mobile strategy web developers will love. Computers in Libraries", 30(4):24–26.

Cornelius, B. (2001) "Understanding Java". Harlow, England: Addison-Wesley

Creswel, J.W.(2009). Research Design: Qualitative, quantitative and mixed method approaches. Sage publications (3rd Ed) pp 3-44

Devitt, S., Meeker, M. and Wu, L. (2010) "Internet Trends". Morgan Stanley Research. [Online] available from: http://www.morganstanley.com/institutional/techresearch/pdfs/Internet_Trends_041210.pdf. [Accessed: 18th June 2014].

Dhillon, S. (2012) "An Evaluation Framework for cross platform Development". University of Guelph, Ontario Canada.

Dix, A., Finley, G. And Beale R(1993) "Human Computer interaction" 2nd Edtion: Prentice Hall

Drake, S., Stofega, W., Llamas, R. T. and Crook, S. K., (2011). Worldwide Smartphone Mobile OS 2011-2015 Forecast and Analysis,: IDC.

E-Tailing Group (2012): *e-tailing group proprietary research findings* [Online] Available from: http://e-tailing.com/technologymarketing/index.html [Accessed: 3rd January 2014]

Fuchs, T (2014) "Zepto.JS: the aerogel-weight jQuery-compactible javascript library"[online] Available from http://Zeptojs.com [Accessed: 4th June 2014]

Gartner INC. (2012) "Information Technology predictions and trends" [Online] Available from: http://horizonwatching.typepad.com/horizonwatching/2011/12/gartner-2012-information-technology-predictions-and-trends.html [Accessed: 19th May 2014]

G. Hartmann, G. Stead, and A. DeGani (2011): "Cross-platform mobile development," Tribal, Lincoln House, The Paddocks, Tech. Rep. March 2011.[Online].http://www.mole-project.net/images/documents/deliverables/WP4_crossplatform_mobile_development_March2011.pdf. [Accessed 19th May, 2014]

Heitkotter, H., Hanschke, S. and Majchrzak, T. A (2012) "Comparing Cross Platform Development approaches for mobile applications" in WEBIST, SciTepress pp 299-311

Henver, A.R., Salvatore, T.M, Park, J. and Sudha, R(2004). Design Science in Information System research. MIS Quarterly vol 28 No 1 pp 75-105.

Holzer, A. (2011) *"*Trends in Mobile Application Development*"*. Springer Berlin Heidelberg.

Hu, G. and Gadap, A (2005) "Compiling C++ programs to java bytecode" in SNPD-SAWN iOS: proceedings of the sixth international conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and first ACIS international workshop on Self-Assembling Wireless Networks pp 56-61.

International Data Corporation: (2014), smartphone market share Q4 [online] available from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Accessed: 20th June 2014]

Ishizaka A. and Labib, A. (2009) "Analytic Hierarchy Process and Expert Choice: Benefits and Limitations" ORInsight.

Issa, J (2011) "TMAPP – Typical Mobile Applications Benchmark," in MoBS7 '11: Seventh Annual Workshop on Modeling, Benchmarking and Simulation.

International Busines Machines (2012) "IBM Worklight Station" [online] Available from http:// www.worklight.com [Accessed: 10th November 2014].

Kim, H, J., Sudara, K., Holger, R. and Warren, I (2015) "Evaluation of Cross platform Toolsfor Patient Self-Reporting on Mobile Devices" pp 55-60.

Kaltofen, S,. Milrad, M,. and Kurti, A. (2010) "A Cross Platform Software System to Create and Deploy mobile mashups" in ICWE'10: Proceedings of the 10th international conference on Web Engineering Berlin, Heidelberg p. 518-521.

Kao, Y., Lin, C,. Yang, K. and S. Yuan. (2011) "A Cross-Platform Runtime Environment for Mobile Widget-Based Application," in CYBERC '11: Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery,

Korf, M. and Oksman, E. (2011) "Understanding your mobile application development options"[online] [Availabefrom:https://developer.salesforce.com/page/Native,_HTML5,_or_Hyn rid:] [Accessed:20th March 2014]

Michaels, R. and Cole, R. (2014) "Native Mobile apps the wrong choice for business" pp 2-10

Macademian (2011). "Building Cross platform Mobile Apps- when to build a native app"[online] Available from: http://www.macadamian.com/2011/03/ [Accessed: 9th November 2014]

Muchmore, M (2013) By numbers: the fastest Browser [Online] Available from http://PCmag.com [Accessed 10th May 2014]

Newman, I and Benz, C.R (1998). Qualitative-quantitative research methodology: Exploring the interactive continuum. Carbondale and Edwardsville: Southern Illinois University Press.

Ohrt, J. and Tarau, V. (2012): "Cross Platform Development Tools for Smartphone Applications", Computer Vol.45 No. pp 72-79.

Paananen, T (2011) " Smartphone Cross Platform Framworks: Jank University of applied Sceinces Thesis" [online] Available from http:// publication.theseus.fi/bitstream [Accessed: 10th May 2014]

Pastore, S. (2013) “Mobile operating system and app development strategies”: International Conference on Systems, Control and Informatics.

PCGMEDIA (2004) “*Characteristics of mobile applications*” [online] Available from http://ptgmedia.pearsoncmg.com/images/0321269314/samplechapter/salmre_ch02.pdf [Accessed 14th July 2014].

Preece, J., Helen, S. And Yvonne, R.(2007): “Interaction Design: Beyond Human computer interaction”, Wiley publication.

Rajapakse, D. C (2008) “Techniques for De-fragmenting Applications*: A Taxonomy”*. In SEKE. Knowledge Systems Institute Graduate School.

RIM Inc. (2011) “RIM Unveils BlackBerry BBX - Combines the Best of BlackBerry and QNX to Provide a Next Generation Platform for BlackBerry Smartphones and Tablets. Press Release, [Online] available from: http://press.rim.com/release.jsp?id=5230. [Accessed: 20th June 2014]

Saaty and Vargas (2001): “Decision making with the analytic hierarchy process”, Int. J. Services Sciences, Vol. 1, No. 1, pp.83–98.

Sencha Inc. (2014) “est JS Version S” [online] Available from http://www.sencha.com/product[Accessed: 19th January 2014]

Sibai, F (2008): "Evaluating the Performance of Single and Multiple Core Processors with PCMARK 05 and Benchmark Analysis," SIGMETRICS Perform. Eval. Rev., vol. 35, pp. 62-71.

Sjøberg, D. I. K., Dybå T. and Jørgensen, M. (2007). “The future of Empirical Methods in Software Engineering Research”: Future of Software Engineering, IEEE-CS Press

Slife, B.D and Williams, R.N (1995). What's behind the research? Discovering hidden assumptions in the behavioural sciences. Thousand Oaks, CA. Sage

Steve, R. P. And Mac, F. (2001): "A Practical Guide to Feature-Driven Development" (1st Ed).

Straub, D, Marie-Claude, B and Gefen, D. (2004) "Validation Guidelines for IS Positivist Research" CAIS. [Online]: available from http://www.researchgate.net/publication. [Accessed: 19th June 2014]

Strauss, A.L and Corbin, J.M (2008). Basics of qualitative research: Techniques and procedures for developing grounded theory. Sage publication Inc.

Steve, R. (2004) "iOS versus Android" [Online] Available from http:// zdnet.com [Accessed: 10th January 2014]

Uti, N and Fox, R (2010): "Testing the Computational Capabilities of Mobile Device Processors: Some Interesting Benchmark Results," in 2010 IEEE/ACIS 9th International Conference on Computer and Information Science (ICIS), Washington, DC, USA, 2010, pp. 477-481.

Vaishnavi, V. and Kuechler, B (2004). "Design Science Research in Information Systems" pp 7-25

Vision Mobile (2012): "Cross-Platform Developer Tools: Bridging the worlds of mobile apps and the web". [Online] available from http://www.slideshare.net/andreasc/vision-mobile-crossplatformdevelopertools2012 [Accessed: 21st June 2014].

Vision Mobile (2013): “Developer Economics 2013- Developer Tools: the foundations of app economy” pp 1-61

Winokur, D. (2011): “Flasht to Focus on PC Browsing and Mobile Apps: Adobe to more Aggressively Contribute to HTML5” [online] Available from: http:// blogs.adobe.com/conversation/2011/11/flash-focus.html [Accessed: 25th September 2014]

Wotaka, F, Bloice, M.D and Holzinger, A (2009) “Java’s alternatives and the limitations of java when writing cross-platform applications for mobile devices in the medical domain” in ITI 09. Proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande.

Xin, C. (2009). "Cross-Platform Mobile Phone Game Development Environment," in *IIS '09: Proceedings of the 2009 International Conference on Industrial and Information Systems*.

APPENDIX A: Source Code for Proof of Concept Application

The following is a sample of the programming source code of multi-platform mobile application designed.

HTML Code:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<div id="logo"></div>

<h2 id="test_status"></h2>

<div id="test"></div>

</body>

</html>
```

CSS Code:

```
div#test{ border:#000 1px solid; padding:10px 40px 40px 40px; }

#test {
        left: 0px;
        right: 0px;
}

#logo {
        background-image:url(applogo.png);
        background-repeat:no-repeat;
        width:100%;
        height:70px;
        max-width:450px;
        max-height:70px;
}

body {
   background-color: #002974;
   color: #FFFFFF;
```

```
    font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;

}
```

JavaScript Code:

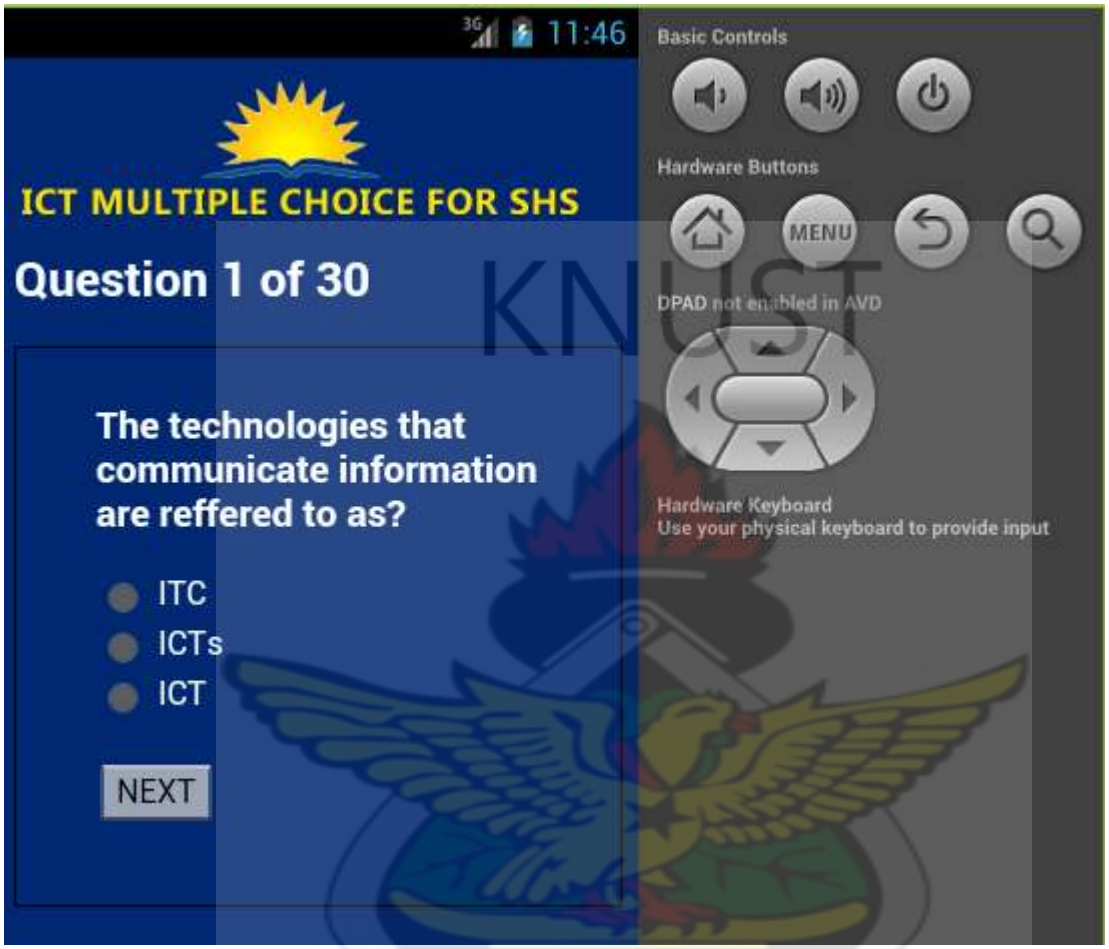
```
<script>

var pos = 0, test, test_status, question, choice, choices, chA, chB, chC, correct = 0;

var questions = [

   [ "The technologies that communicate information are reffered to as?", "ITC", "ICTs", "ICT", "B" ],

        [ "Data that is obtained from outside an organisation is termed as", "Organised", "Internal", "External", "C" ],

        [ "The raw and unorganised facts and figures are known as?", "Data", "Information", "Knowledge", "A" ],

        [ "A term used for the face to face teaching and enhanced learning is termed as?", "E-learning", "E-commerce", "blended learning", "A" ],

        [ "In a computer system, what is the screen used for?", "Input", "Output", "Storage", "B" ],

        [ "Which of the following is not an output device?", "Microphone", "Speaker", "Plotter", "A" ],

        [ "A device that accepts data, process it and gives information is called....?", "analyzer", "Computer", "Processor", "B" ],
```

```
[ "Which of the following is the odd one out?", "printer", "Microphone", "Keyboard",
"A" ],

[ "A physical component of a computer that can be seen and touched is termed as ...?",
"Hardware", "Software", "Hardcopy", "A" ],

[ "The network within an organization is termed as?", "Extranet", "Interent", "Intranet",
"C" ],

[ "Which of the following is a type of monitor?", "Crystal liquid Display", "Liquid
Cathod Display","Light Display", "B" ],

[ "Which of the following is not a basic part of a desktop computer?", "Printer", "Mouse",
"Keyboard", "A" ],

[ "The ability to manipulate both hardware and software is referred to as?",
"Technology", "Computer competency", "Computer literacy", "C" ]

];

function _(x){

    return document.getElementById(x);

}

function renderQuestion(){

    test = _("test");
```

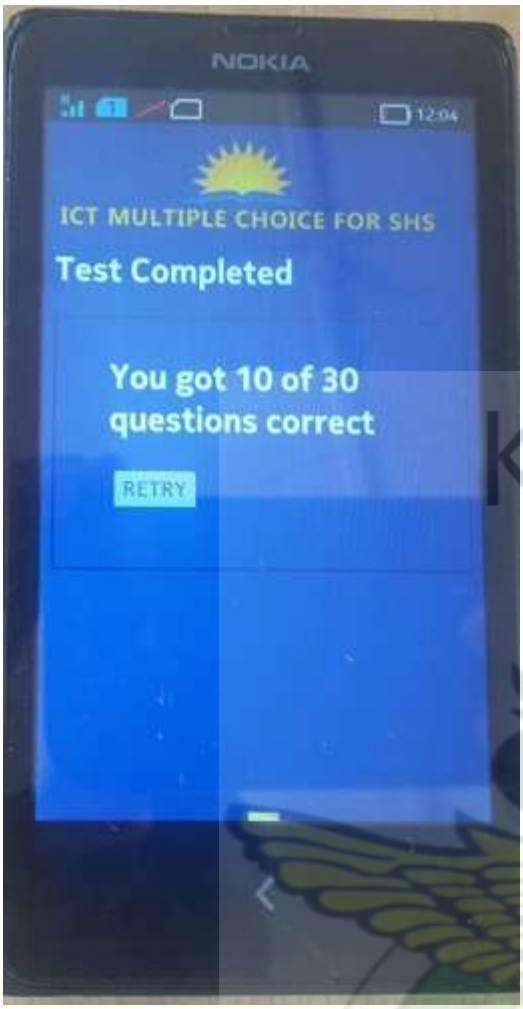
```
if(pos >= questions.length){
        test.innerHTML = "<h2>You got "+correct+" of "+questions.length+" questions correct</h2>";
        _("test_status").innerHTML = "Test Completed";
        pos = 0;
        correct = 0;
        test.innerHTML += "<button onclick='retry()'>   RETRY   </button>";
        return false;
}
_("test_status").innerHTML = "Question "+(pos+1)+" of "+questions.length;
question = questions[pos][0];
chA = questions[pos][1];
chB = questions[pos][2];
chC = questions[pos][3];
test.innerHTML = "<h3>"+question+"</h3>";
test.innerHTML += "<input type='radio' name='choices' value='A'> "+chA+"<br>";
test.innerHTML += "<input type='radio' name='choices' value='B'> "+chB+"<br>";
```

```
        test.innerHTML += "<input type='radio' name='choices' value='C'> "+chC+"<br><br>";

        test.innerHTML += "<button onclick='checkAnswer()'>   NEXT   </button>";

}

function checkAnswer(){

        choices = document.getElementsByName("choices")

        for(var i=0; i<choices.length; i++){

                if(choices[i].checked){

                        choice = choices[i].value;

                }

        }

        if(choice == questions[pos][4]){

                correct++;

        }

        pos++;

        renderQuestion();

}window.addEventListener("load", renderQuestion, false);

</script>
```

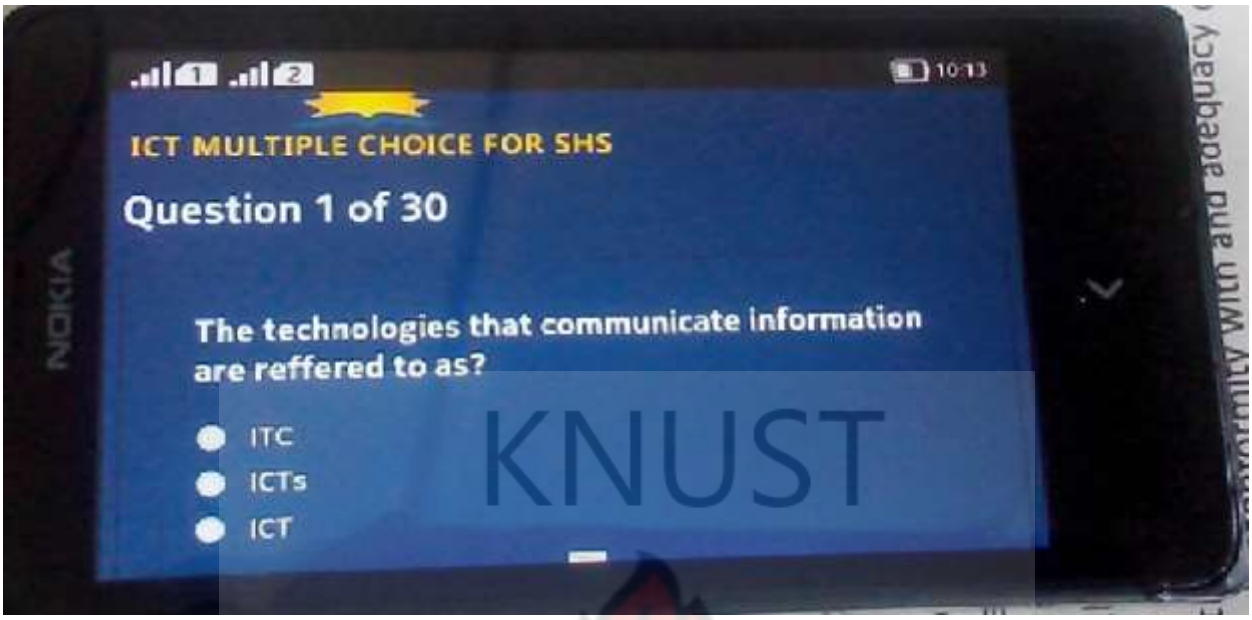
APPENDIX B: Screen shot of sample Application in emulators and devices



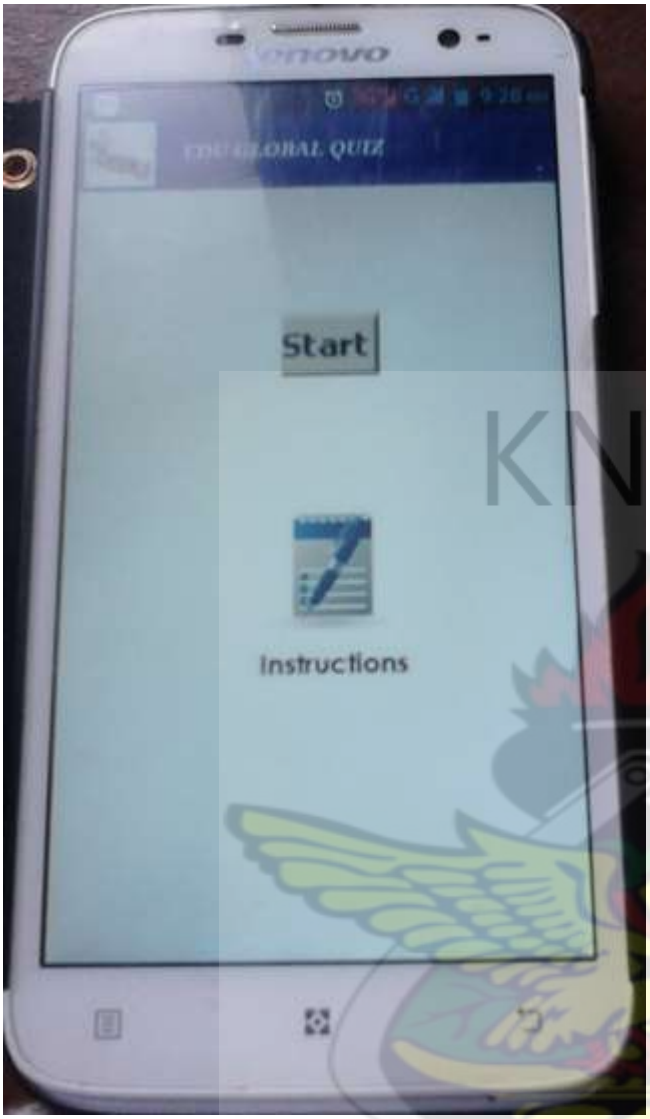
Screen shot in Android Virtual Device (AVD)



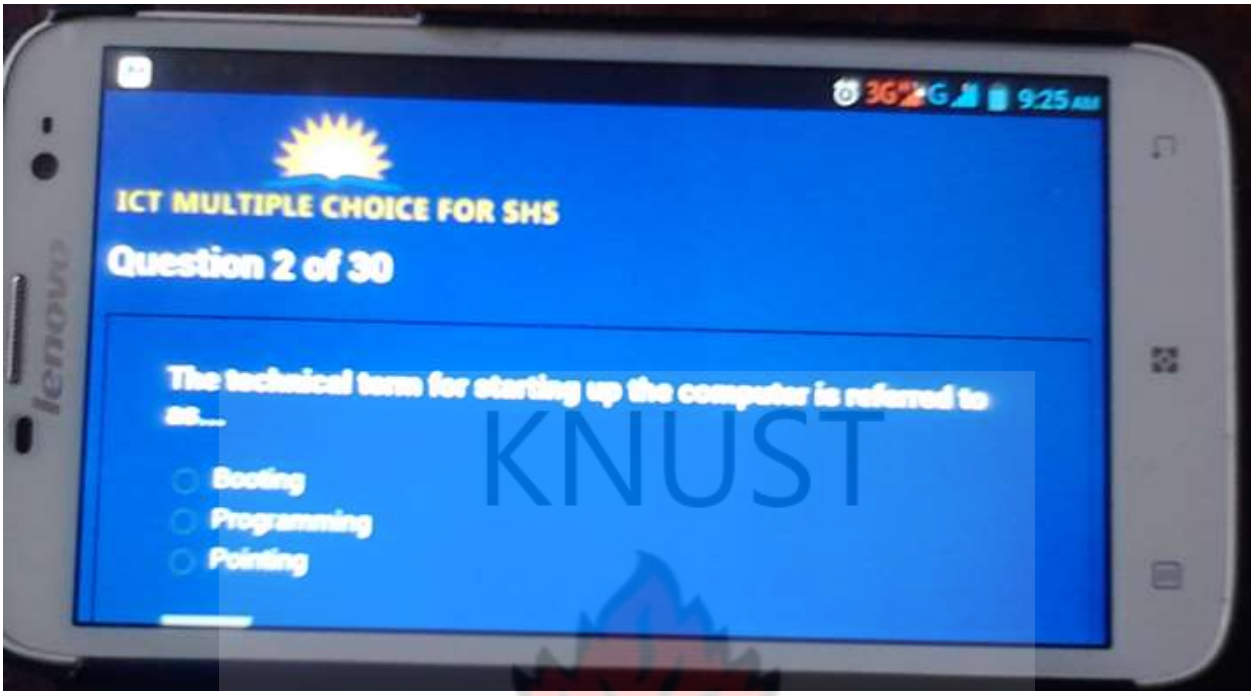
Portrait view of Test Scores screen on Nokia Lumia phone running windows operating system



Landscape view of question screen on Nokia Lumia phone running windows operating system



Portrait view of Start Screen on Lenovo phone running Android operating system



Landscape view of Question Screen on Lenovo Phone running Android operating system

APPENDIX C: Application Checklist for Evaluating Features of the Application

MULTIPLATFORM MOBILE APPLICATION CHECKLIST		
NO	**VARIABLE**	**ANSWER**
1	Mobile application loads without errors	☐
2	Mobile application fits within the device screen resolutions	☐
3	Designs displays correctly in the main device browser	☐
4	Application displays correctly in both orientations	☐
5	Buttons are clickable and redirects correctly	☐
6	Users are able to choose answers successfully	☐
7	Next question loads successfully	☐
8	Mobile application calculates correct answers successfully	☐
9	Mobile application looks and feel like other native applications	☐
10	Passes usability test?	☐

APPENDIX D: Test Results on Lenovo Phone Running Android 4.0

MULTIPLATFORM MOBILE APPLICATION CHECKLIST		
NO	**VARIABLE**	**ANSWER**
1	Mobile application loads without errors	☑
2	Mobile application fits within the device screen resolutions	☑
3	Designs displays correctly in the main device browser	☑
4	Application displays correctly in both orientations	☑
5	Buttons are clickable and redirects correctly	☑
6	Users are able to choose answers successfully	☑
7	Next question loads successfully	☑
8	Mobile application works in both online and offline mode	☑
9	Mobile application looks and feel like other native applications	☒
10	Passes usability test?	☑

APPENDIX E: Test Results on Nokia Lumia 520 Running Windows 8.0

MULTIPLATFORM MOBILE APPLICATION CHECKLIST		
NO	**VARIABLE**	**ANSWER**
1	Mobile application loads without errors	☑
2	Mobile application fits within the device screen resolutions	☒
3	Designs displays correctly in the main device browser	☑
4	Application displays correctly in both orientations	☑
5	Buttons are clickable and redirects correctly	☑
6	Users are able to choose answers successfully	☑
7	Next question loads successfully	☑
8	Mobile application works in both online and offline mode	☑
9	Mobile application looks and feel like other native applications	☒
10	Passes usability test?	☑

APPENDIX F: Test Results using W3C mobileOK Checker