ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Deployment of Mobile Application for Multiplatform Operating Systems

Sayali Kamble[1], Mitali Joshi[1], Shreya Kale[1], S. Mahajan[2]

BE Students, Department of Information Technology, PVG'COET, Pune, India[1]

Assistant Professor, Department of Information Technology, PVG'COET, Pune, India[2]

Abstract: Due to the current restrictions and differences between iOS and Android platforms, to have an application deployed on both platforms requires it to be developed twice. Hence this doubles the effort and time needed for development of the application. The mobile applications are used more than the calls. People use the applications as per their need. There are various mobile operating systems like Android, iOS, Blackberry, Windows, etc. Each operating system has its own specialization. The applications defer from one operating system to the other. So the customer needs to use more than one mobile operating system to fulfill their requirements. We'll be developing a mobile application for Multiplatform Operating Systems which will reduce the code development time and energy to half.

Keywords: Mobile Appllication, Android, iOS, Multiplatform.

I. INTRODUCTION

The use of mobile applications is increasing steadily and continuously. The mobile devices being use are getting more and more powerful due to its wide search and easy accessibility. Owing to this many businesses are adapting mobile technology to tap their targeted commercial and industrial markets. The two most powerful operating systems where mobile app developers are focusing these days are iOS and Android. However, earlier due to configuration differences between these platforms, apps that were supported by one, was not always supported by the other. This was the headache for developer as he needed to develop different code for each platform . To tackle with this problem Smartphone app developers took the route to smartphone app development across multi-platforms. So, it was time and cost saving. Depending on the type of applications which is being developed, the number of platforms that would be integrated is decided. Suppose, the app which is not specific, say, a weather app would be developed so that users of multiple platforms can benefit from it, but a specific app need not be developed to support all the major operating systems. Now a day's mobile are replacing the use of laptops & desktops on large scales. But as there are various operating system, with different configuration and programming language it becomes problem for developer to develop the application for each operating system individually. So this has been solved by the cross platform mobile application development tool which provides more scope in less time. With the introduction of Android from Google and the Open Handset Alliance, developers have a robust platform to run their applications, and extend their reach to millions of mobile users. The development of these applications is much easier than earlier .however, developing an application is still complicated. Whenever a developer has to create a new application for smart phones, developers will simply have to write a code only once, and by using PHONEGAP, developer can convert the application code so as to run it can run on other OS.

The current scenario is that due to the current restrictions and differences between iPhone and Android platform, applications that need to be deployed on both the platforms needs to be developed twice. Hence this doubles the effort and time needed for development of the application. This increases the demand for a translator to convert code from Java to Objective C for iPhone and Java for Android. Due to the above mentioned drawbacks there is a need to have a design for the code convertor by which we can run applications on Android and iPhone platforms.

II. LITERATURE SURVEY

IDE (Integrated Development Environment) where we can code in a language we like and deploy our application on multiple platforms. Many times the time required for developer to learn different languages for developing an application cant be tolerable . So that it was the need to think something new. And this is the cross platform application development A Smartphone is a high end mobile phone that combines the functions of a personal digital assistant (PDA) and a mobile phone. The iPhone is a line of Internet and multimedia enabled smart phones marketed by Apple. An iPhone can function as a video camera (video recording was not a standard feature until the iPhone 3GS was released), a camera phone, a portable media player, and an Internet client with email and web browsing capabilities, can send texts and receive visual voicemail, and has both Wi-Fi and 3G connectivity. Android is an operating system for mobile devices such as Smart phones and tablet computers. It is developed by the Open Handset Alliance led by Google. Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run compiled Java code. The main aim of our project is to develop an application which is to

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

develop an application which would support various mobile operating systems. In this Project, we are developing a tool for "Online Voting System". A change in model world all the things are going to be automated know a day all work done with the help of computer. Which makes our work more easy, fast (time consuming) & reliable.

III. BASICS OF MOBILE OS (ANDROID AND IOS)

***Android* :**

Android is a Linux based operating system designed for touch screen mobile device such as Smartphones and tablets. Android is an open source and Google releases the code under Apache License. Android application written in a customized version of the java programming language. Android's user interface is based on direct manipulation using touch input by user. The response to the user input provides immediate a fluid touch interface. Android device boot to the home screen. Android home screen are typically made up of app icons and widgets. Android consist of a kernel based on Linux kernel version 2.4 and from ice cream sandwich version 4.0 onwards, with middleware, libraries and API written in C and application is running on an application framework which includes java compatible libraries based on apache harmony. Android is designed to manage RAM to keep power consumption at minimum. Android manages the apps stored in memory automatically. Android application run on SandBox, an isolated area of system that does not allow the rest of the system's resource, unless permission has granted by user when application is installed.

***Architecture* :**

Android Architecture is based on the Linux 2.6 Kernel (Red Section of Figure 1). It is used as the hardware abstraction layer. The Linux Kernel provides a driver model, memory management, process management and other robust features that have been proven over time[11]. The Android libraries (Green Section of Figure 1), written in C and C++, provide much of the core functionality and power of the Android platform. For example, SQLite is used as the core for the majority of the data storage needs. Webkit is an open-source browser engine and is the same engine that powers Apple's Safari. The main component of the Android Runtime (Yellow Section of Figure 1) is the Dalvik virtual machine. The Android platform was designed to meet the needs of an embedded environment, where battery, memory and CPU limitations exist. The Dalvik VM converts .jar and .class files into .dex files at run-time for a much more efficient byte-code. Additionally, .dex files are CPU optimized and designed to be shared across processes, resulting in the ability for multiple, concurrent Dalvik machines to be running on a single device. The Core Libraries (Blue Area of Figure 1) provide all the utilities that basic programmers require, such as File I/O, UI and basic functionality. The Application Framework (Lower Blue Section of Figure 1) is the toolkit that all applications, Google or third-party developers, use. An example would be the activity manager. This application manages the application life cycle and a common back stack, enabling applications running in separate processes to have a smoothly integrated navigation experience. Content Providers are a unique piece of the Android Platform; they allow application to share data. An example of this is the sharing of contacts to any application with correct permissions. The Application Layer (Top Section of Figure 1) is the actual implementation of the applications, such as the phone, messaging or our NPS Portal[6].

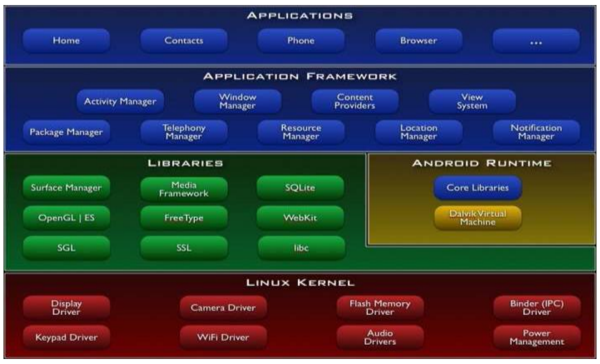

Fig 1. Android System Architecture [11]

***iOS* :**

iOS is a mobile operating system developed and distributed by Apple Inc, released in 2007 for the iPhone and iPad Touch. Unlike Microsoft's Windows Phone (Windows CE) and Google's Android, Apple does not allow nonapple hardware to install license iOS. The user interface of iOS is based on the concept of Direct Manipulation, using multi-touch gesture which consists of sliders, switches and buttons. iOS is divided into four different layers such as the Core OS layer, the Core Service layer, the media layer and the Cocoa Touch layer. Features those are provided by iOS are as follow: Home Screen, Folders, Notification Center, Multitasking, Switching Application, Siri, Game Center. iOS application must be written and compiled specifically for iOS and the ARM architecture. XCode is the development environment for the iOS SDK and application are written in Objective-C language.

***Architecture* :**

iOS architecture is similar to that of the Mac OS X; at the highest level it acts as an intermediary between the hardware and on-screen applications. Applications do not directly communicate with the device hardware. Instead the applications communicate through a set of pre-defined system interfaces, allowing developer applications to work seamlessly on different hardware configurations. Although each application is protected against the different hardware configurations, developers still need to account for the configurations in their code. For example, a developer creating a photography application must account for the lack of a camera in the 1st generation iPod touch. This application will be able to use all the functionality of the app, except for taking pictures. Within the iPhone OS (Blue Section of Figure 2), there are four major layers:

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Cocoa Touch, Media, Core Services and Core OS. The Core OS is built on the system level encompassing the kernel environment, drivers and low-level UNIX interfaces of the OS. The kernel, based on Mach, manages virtual memory, threads, file system , networking, and inter-process communications. Also, composing the Core OS layer is the Security Framework, External Accessory Framework and Accelerate Framework. These frameworks provide the necessary security, access to external hardware, math functionality (basic math, big-number, and DSP calculations), and are optimized for iOS device hardware configurations. The Core Services layer contains all fundamental system services; many parts of the system are built on top of this layer.This layer also includes support for SQLite databases, xml, grand central dispatch, and in-app purchases. Core service frameworks, such as the Address Book Framework, CF Network Framework (which manages Wi-Fi, cell, and Bluetooth networking), and Core Data Framework (model view controller data management) are contained in this layer. The Media layer contains support for all audio, video and graphics technologies. This layer supports 2D and 3D graphics through the Quartz graphics engine. Various codecs, capable of decoding audio, video and AirPlay support, are also included in the media Layer. Finally, the Cocoa Touch layer contains the key frameworks for building iOS applications. This layer defines basic application infrastructure and supports key technologies, such as touch-based input, multi-tasking and other high-level system services. This layer contains the implementation of the Apple view controllers. Developers will use support and services from this layer to create application UIs. Since this layer provides all functionality required by the typical developer, we will initially focus on this layer to begin identifying common programming aspects of iOS and Android[6].

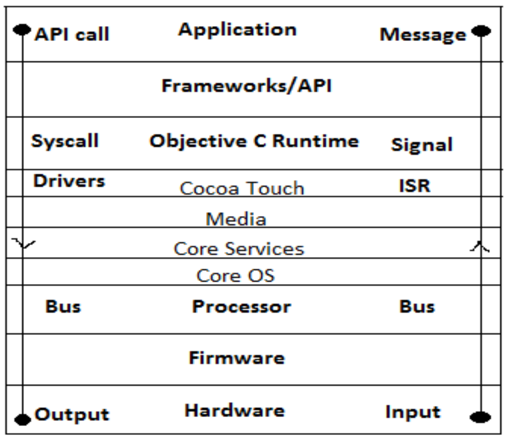

Fig 2 . iOS System Architecture

Advantages of iOS :

1) You can easily interact and manipulate your screen in many ways.
2) The unique interface is provided for the smart phones.
3) Eight integration with social networking and sophisticated music experience.

Disadvantages of iOS :

1) Flash or shockwaves videos won't be able to run.
2) Hearing regular gloves One cannot intermingle with the screen.
3) Customization is Required.

Advantages of Android :

1) Any application is freely upgradable.
2) Android smart phone can be used as a USB storage device.
3) Notification is easy.
4) Any application can be downloaded from google android app market that is too free of cost.
5) A variety of settings can be easily and quickly accessible using widgets.

Disadvantages of Android :

1) Android software containing an application often hang or crashed.
2) Application design is Inconsistence.
3) Continuous internet connection required.

IV. PROBLEM WITH CROSS-PLATFORM DEVELOPMENT

Applications written to iOS and Android platforms are written in different languages. Applications written for iOS are written in Objective-C while for Android devices are written in Java. The most obvious difference between platforms is the language with which the applications are written. So the unique coding is required. Apart from that, developers must be aware of different hardware capabilities such as external SD cards and forward facing cameras. Finally, each platform has its individual User Interface (UI) styles to which users are used to. Users expect that each application will adhere to the platform standard UI style. This final aspect is that the cross-platform applications maintain separate UIs for each targeted platform. In order to remain relevant in today's application market places, developers must embrace multiplatform development concepts to ensure that the applications are targeted to as many different platforms as possible. To that end it becomes necessary that any application developed for one platform also be made available for other not only existing platforms but also to future platforms. Each platform has its own hardware profits, including processor and memory, as well as screen size and other options such as cameras and Bluetooth. These hardware profiles cause the platform specific APIs to differ between platforms. In addition to language and hardware differences, each platform provides unique user interface guidelines with which users have become accustomed and developers are expected to maintain in any application they develop These factors result in increased costs in terms of time and money spent on the re-design process and the opportunity cost of that development time not being spent addressing new application design, or at the least, maintaining and upgrading the already released application. Additionally, the second design process often results in drastically different code bases that increases maintenance costs and may lead to applications with different features.

V. PROPOSED SOLUTION

An ideal method to accomplish this cross-platform dilemma is the ability to design and write a single application that runs on all platforms, or an interpretive multiplatform solution such as OpenGL and Web Applications. Using these solutions application can be run directly on each platform by using an interpreter. In the mobile realm the interpreters for Web Applications are the web browsers and for OpenGL it is the graphics libraries. These solutions fall short of the desired cross platform solution because they do not adhere to the platform specific UI Styles. A single UI is developed which may either correspond only to a single platform, or implement a platform neutral UI. Users from one or more platforms will be forced to adhere to a UI and navigation style which is unfamiliar to them. Products such as Appacelerator's Titanium and Corona provide third party APIs that result in separate but related applications tailored to specific platforms. While they have shown extremely promising results, we decided to pursue a solution that relied only on native platform APIs. We believe that developers are better able to handle security issues related to their applications by using native APIs and implementing applications in native platform languages without the aid of third party tools. While these methods are accepted, and potentially cost efficient solution to the problem, we find that it lacks the customization that users of different platforms have come to expect from applications running on their devices .OpenGL and Web Applications force developers to choose a single UI and navigation schema that will be presented to users of any platform on which the application is hosted, which can often lead to user confusion on one or more of the platforms, as they do not provide common "look-and-feel" features with which users have become accustomed. Cross-platform applications should utilize interfaces specification of each platform. They are targeting to avoid such user confusion. Short of relying on such "neutral" navigation solutions, we need to understand key similarities and differences of the platforms we plan to use. By identifying key similarities between the platforms we will be able to leverage those similarities to develop a design process that will use the aspects common between platforms while minimizing the differences, allowing for applications to be built for multiple platforms from a common set of design documents.

VI. SYSTEM ARCHITECTURE

Online Voting System has four major components.

1) Database Preparation,
2) Voting System,
3) Voting Monitoring,
4) The Auto Counting System

Database Preparation : This component is used to manage the data bank of the students or candidates. Data base is opened to the Administrator to add the students or candidates details for the Voting.

Voting System : This component is used to control all the functionality of the system. Like client side Control, Time Control, Security of the system. This component is very important because it used to keep the control on the security and time.

Voting Monitoring : This component is used for real time monitoring which can help to keep the track of voting processing, student ID statistic and collect the result.

The Auto Counting System : This Component is used to count the votes automatically with the help of this system. This section provides an introduction into the Android and iPhone platforms, integral differences, and the issues related to the topic of multiplatform development of the two platforms. Additional considerations specific to platform architectures, Integrated Development Environments (IDEs), and design patterns will also be addressed.

System Features :

The idea with a language cross compiler is that it allows the developer to write code in one language and then it is translated into another language. For example, the developer writes Java code and uses a translator to create Objective-C source code. Web development was first introduced in the desktop world and only later applied to mobile development in order to handle a few unique considerations for device specific hardware. With this approach, applications are coded in a browser-supported language and run inside the mobile browser but rely on server side components to produce the views. SenchaTouch web application development framework offers a rich JavaScript based SDK for mobile development.

A single code base :

A single stream of source code simplifies app writing and maintenance. A single SDK provides the interface to multiple platform.

System Architecture and Business Logic :

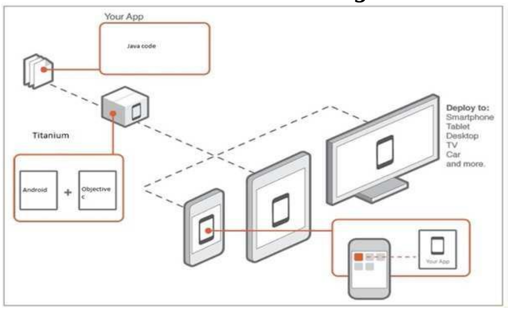

Fig 3. System Architecture and Business Logic

VII. CONCLUSION

This application software helps to acquire a code suitable for the two mentioned platforms. It can also be applied to various other software systems which demand such conversions. It can also be implemented to overcome the drawbacks of the two mentioned platforms with respect to their phone specific applications. Thus the application code has to be developed only once reducing the code development time and energy by half.

ACKNOWLEDGEMENT

We would like to express our gratitude to all friends for co-operating with us and we acknowledge sincere thanks towards our institute Pune Vidhyarthi Griha's College of Engineering and Technology, University of Pune.

REFERENCES

[1] Kim W. Tracy, "Mobile application development experiences on Apple's iOS and Android OS", ABET computing accreditation commissioner, Northeastern Illinois University (NEIU), 27 July 2012.
[2] Yonathan Aklilu Redda "Cross platform Mobile Applications Development", June 2012.
[3] Martin Cronje, "Creating Cross-Platform Mobile Applications with .NET and Mono", Software Engineering Coach and co-Founder nReality Systems, 2011 Microsoft Corporation.
[4] Anuja H. Vaidya, Sapan Naik, "Comprehensive Study and Technical Overview of Application Development in iOS, Android and Window Phone8", International Journal of Computer Applications (0975 – 8887) Volume 64– No.19, February 2013.
[5] Mrs. V.C. Kulloli , Ashish Pohare, Sujit Raskar, Tania Bhattacharyya, Shashikant Bhure ,"Cross Platform Mobile Application Development", Lecturer of Department of Information Technology, Pune University Pimpri Chinchwad College of Engineering.Sector-26, Pradhikaran Nigdi, Pune-44, May 2013.
[6] Christian G. Acord, Corey C. Murphy, "Cross-Platform Mobile Application Development: A Patten-Based Approach", March 2012. 37.
[7] Clement Guerin" Introduction to cross-platform mobile development with Appcelerator Titanium", March 2012.
[8] Eero Maasalmi, Panu Pitkanen "Comparing Google;s Android and Apple's iOS Mobile Software Development Environments", 2011.
[9] Henning Heitkotter, Tim A. Majchrzak, "Comparing Cross-Platform Development Approaches for Mobile Applications ".
[10] Otso Kassinen, Erkki Harjula, Timo Koskela, Mika Ylianttila," Guidelines for the Implementation of Cross-platform Mobile Middleware", International Journal of Software Engineering and Its Applications Vol. 4, No. 3, July 2010,43.
[11] Android. What is android? http://developer.android.com/ guide/basics/what-is-android.htm, Mar 2012.