Internal structure of iOS and Building tools for iOS apps

Miss Priyanka V. Kanoi
2nd yr Information and Technology
JDIET , Yavatmal
kanoi.priyanka@gmail.com

Miss Payal N. ingole
3rd yr Computer sc and Engineering
JDIET, Yavatmal
ingole.payal@gmail.com

Abstract: **iOS is world's most advanced operating system. With its easy-to-use interface, amazing features, and rock-solid stability, iOS is the foundation of iPhone, iPad, and iPod touch. It is the most user friendly operating system. With better GUI performance. iOS consists of four layers – cocoa touch layer, media layer, core service layer, core OS layer. Some of the developmental tools used for preparing iOS apps include XCODE, Instruments and developer library. With thousands of apps in practically every category, iOS is the platform for the world's largest collection of mobile apps. And every app starts with the right DNA. That's because it gives third-party developers a rich set of tools and APIs to create apps and games designed to take advantage of the technology inside every iOS device.**

Keywords: **XCODE, iCLOUD, SDK, cocoa touch layer, media layer, core service layer, core OS layer, OS X.**

I. INTRODUCTION

iOS which was imagination of Steve Jobs, is now reality of young minds. A mobile operating system, iOS (previously iPhone OS) , was developed and distributed by Apple Inc. Originally released for the iPhone and iPod Touch in 2007, it has been extended to support other Apple devices such as the iPad and Apple TV. Unlike Microsoft's Windows Phone and Google's Android, Apple does not license iOS for installation on non-Apple devices. The user interface of iOS is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of buttons, sliders and switches. Interaction with the OS includes gestures such as tap, swipe, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS and its multi-touch interface. iOS is derived from OS X,with which it shares the Darwin foundation. iOS is Apple's mobile version of the OS X used on Apple computers.

II. HISTORY

The operating system was unveiled with the iPhone at the Macworld Conference & Expo, January 9, 2007, and released in June of that year.[1] At first, Apple marketing literature did not specify a separate name for the operating system, stating simply that the "iPhone runs OS X".[2] Initially, third-party applications were not supported. Steve Jobs' reasoning was that developers could build web applications that "would behave like native apps on the iPhone". [3][4] On October 17, 2007, Apple announced that a native Software Development Kit (SDK) was under development and that they planned to put it "in developers' hands in February". On March 6, 2008, Apple released the first beta, along with a new name for the operating system: "iPhone OS".In June 2010, Apple rebranded iPhone OS as "iOS". The trademark "IOS" had been used by Cisco for over a decade for its operating system, IOS, used on its routers. To avoid any potential lawsuit, Apple licensed the "IOS" trademark from Cisco. Apple provides major updates to the iOS operating system approximately once a year over iTunes and also, since iOS version 5.0, over the air. The latest major update is iOS 6, publicly announced on June 11, 2012 and released on September 12, 2012. Over 200 new features debut in iOS 6, including Apple's new Passbook service, Apple-sourced Maps, and full Facebook integration.

III. VERSIONS OF IOS IN IPHONE

VERSIONS	iOS
iPhone	iOS 1.0
iPhone 3G	iOS 2.0
iPhone 3GS	iOS 3.0
iPhone 4	iOS 4.0 (GSM model) iOS 4.25 (CDMA model)
iPhone 4s	iOS 5.0
iPhone 5	iOS 6.0

Table 3.1 : Version history

IV. IOS 6

iOS 6 is the latest major version of the iOS mobile operating system from Apple Inc. The latest version of iOS 6 is 6.1. It was preceded by iOS 5 (final version was 5.1.1).
Apps supported by iOS 6 are Siri, maps, music, facetime, safari, message, airplay, game center, imovie, iphoto, garrage band, keynotes,pages, numbers, ibooks, itunes u, cards , podcasts, passbook, find my iPhone, remote.

V. TECHNOLOGICAL OVERVIEW AND MAKING OF IOS

iOS is the operating system that runs on iPhone, iPod touch, and iPad devices. The operating system manages the device hardware and provides the technologies required to implement native apps. The operating system also ships with various system apps, such as Phone, Mail, and Safari, that provide standard system services to the user.

The iOS Software Development Kit (SDK) contains the tools and interfaces needed to develop, install, run, and test native apps that appear on an iOS device's Home screen.[5] Native

apps are built using the iOS system frameworks and Objective-C language and run directly on iOS. Unlike web apps, native apps are installed physically on a device and are therefore always available to the user, even when the device is in Airplane mode. They reside next to other system apps and both the app and any user data is synced to the user's computer through iTunes.

In addition to native apps, it is possible to create web apps using a combination of HTML, cascading style sheets (CSS), and JavaScript code. Web apps run inside the Safari web browser and require a network connection to access your web server. Native apps, on the other hand, are installed directly on the device and can run without the presence of a network connection. The iOS SDK provides the resources you need to develop native iOS apps. Therefore, understanding a little about the technologies and tools that make up this SDK can help you make better choices about how to design and implement your apps.

VI. THE LAYERED ARCHITECTURE OF IOS

At the highest level, iOS acts as an intermediary between the underlying hardware and the apps that appear on the screen. The apps you create rarely talk to the underlying hardware directly. Instead, apps communicate with the hardware through a set of well-defined system interfaces that protect your app from hardware changes. This abstraction makes it easy to write apps that work consistently on devices with different hardware capabilities.
The implementation of iOS technologies can also be viewed as a set of layers, which are shown in Figure 4.1. At the lower layers of the system are the fundamental services and technologies on which all apps rely; higher-level layers contain more sophisticated services and technologies.

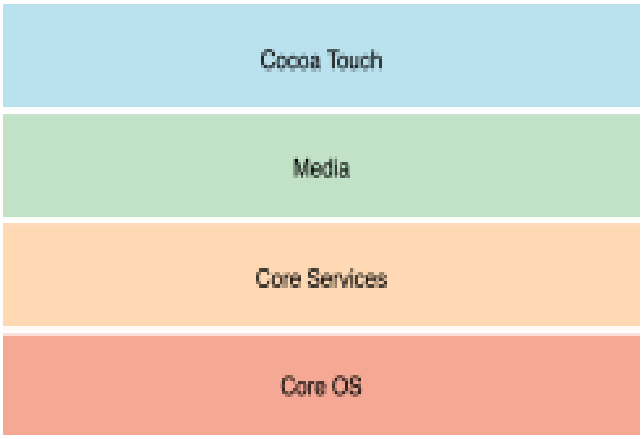

Figure 4.1 Layers of iOS

As you write your code, you should prefer the use of higher-level frameworks over lower-level frameworks whenever possible. The higher-level frameworks are there to provide object-oriented abstractions for lower-level constructs. These abstractions generally make it much easier to write code because they reduce the amount of code you have to write and encapsulate potentially complex features, such as sockets and threads. Although they abstract out lower-level technologies, they do not mask those technologies from you. The lower-level frameworks are still available for developers who prefer to use them or who want to use aspects of those frameworks that are not exposed by the higher layers.

VII. LAYERS OF IOS

A. *Cocoa Layer*

The Cocoa Touch layer contains the key frameworks for building iOS applications. This layer defines the basic application infrastructure and support for key technologies such as multitasking, touch-based input, push notifications, and many high-level system services. When designing your applications, you should investigate the technologies in this layer first to see if they meet your needs.[6]

The following sections describe some of the key technologies available in the Cocoa Touch layer.

1) AutoLayout - Introduced in iOS 6, auto layout improves upon the "springs and struts" model previously used to lay out the elements of a user interface. With auto layout, you define rules for how to lay out the elements in your user interface. These rules express a larger class of relationships and are more intuitive to use than springs and struts.

2) Storyboards - Introduced in iOS 5, storyboards supplant nib files as the recommended way to design your application's user interface. Unlike nib files, storyboards let you design your entire user interface in one place so you can see all of your views and view controllers and how they work together. An important part of storyboards is the ability to define segues, which are transitions from one view controller to another. Applications can define these transitions visually in Xcode or initiate them programmatically in Xcode. These transitions allow you to capture the flow of your user interface in addition to the content.

3) Document Support - Introduced in iOS 5, the UIKit framework introduced the UIDocument class for managing the data associated with user documents. This class makes implementing document-based applications much easier, especially applications that store documents in iCloud. In addition to providing a container for all of your document-related data, the UIDocument class provides built-in support for asynchronous reading and writing of file data, safe saving of data, automatic saving of data, support for detecting iCloud conflicts, and support for flat file or package file representations. For applications that use Core Data for their data model, you can use the UIManagedDocument subclass to manage your data stores.

4) Multitasking - Applications built using iOS SDK 4.0 are not terminated when the user presses the Home button; instead, they shift to a background execution context. The multitasking support defined by UIKit helps your application transition to and from the background state smoothly.To

preserve battery life, most applications are suspended by the system shortly after entering the background. A suspended application remains in memory but does not execute any code.

5) UI State Preservation - Introduced in iOS 6, state preservation makes it easier for apps to restore their user interface to the state it was in when the user last used it. When an app moves to the background, it is asked to save the semantic state of its views and view controllers. Upon relaunch, the app uses this state to restore its interface and make it seem as if the app had never quit. Support for state preservation is integrated into UIKit, which provides the infrastructure for saving and restoring your app's interface.

6) Standard System View Controllers - Many of the frameworks in the Cocoa Touch layer contain view controllers for presenting standard system interfaces. You are encouraged to use these view controllers in your applications to present a consistent user experience. Whenever you need to perform one of the following tasks, you should use a view controller from the corresponding framework:

a) Display or edit contact information—Use the view controllers in the Address Book UI framework.

b) Create or edit calendar events— Use the view controllers in the Event Kit UI framework.

c) Compose an email or SMS message— Use the view controllers in the Message UI framework.

d) Open or preview the contents of a file— Use the UIDocumentInteractionController class in the UIKit framework.

e) Take a picture or choose a photo from the user's photo library— Use the UIImagePickerController class in the UIKit framework.

f) Shoot a video clip — Use the UIImagePickerController class in the UIKit framework.

Other technologies supported by cocoa touch layer are – Printing , Apple Push Notification Service , Local Notifications , Gesture Recognizers , Peer-to-Peer Services, External Display Support

B. Media Layer

The Media layer contains the graphics, audio, and video technologies geared toward creating the best multimedia experience available on a mobile device. The technologies in this layer were designed to make it easy for you to build applications that look and sound great.[7]

The following sections describe some of the key technologies available in the media layer.

1) Graphics Technologies - High-quality graphics are an important part of all iOS applications. The simplest way to create an application is to use prerendered images together with the standard views and controls of the UIKit framework and let the system do the drawing. However, there may be situations where you need to go beyond simple graphics.

2) Audio Technologies - The audio technologies available in iOS are designed to help you provide a rich audio experience for your users. This experience includes the ability to play high-quality audio, record high-quality audio, and trigger the vibration feature on certain devices.The system provides several ways to play back and record audio content.

3) Video Technologies - Whether you are playing movie files from your application or streaming them from the network, iOS provides several technologies to play your video-based content. On devices with the appropriate video hardware, you can also use these technologies to capture video and incorporate it into your application.The system provides several ways to play and record video content that you can choose depending on your needs.

4) AirPlay - AirPlay is a technology that lets your application stream audio to Apple TV and to third-party AirPlay speakers and receivers. AirPlay support is built in to the AV Foundation framework and the Core Audio family of frameworks. Any audio content you play using these frameworks is automatically made eligible for AirPlay distribution. Once the user chooses to play your audio using AirPlay, it is routed automatically by the system.

C. Core Services Layer

The Core Services layer contains the fundamental system services that all applications use. Even if you do not use these services directly, many parts of the system are built on top of them. [8]

The following sections describe some of the key technologies available in the Core Services layer.

1) iCloud Storage - Introduced in iOS 5, iCloud storage lets your application write user documents and data to a central location and access those items from all of a user's computers and iOS devices. Making a user's documents ubiquitous using iCloud means that a user can view or edit those documents from any device without having to sync or transfer files explicitly. Storing documents in a user's iCloud account also provides a layer of safety for that user. Even if a user loses a device, the documents on that device are not lost if they are in iCloud storage.There are two ways that applications can take advantage of iCloud storage, each of which has a different intended usage:

a) iCloud document storage - Use this feature to store user documents and data in the user's iCloud account.

b) iCloud key-value data storage - Use this feature to share small amounts of data among instances of your application.

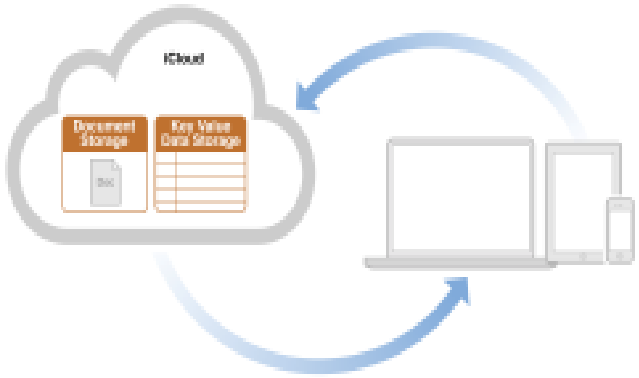

Figure 7.C.1 : iCloud Storage

2) Automatic Reference Counting - Introduced in iOS 5, Automatic Reference Counting (ARC) is a compiler-level feature that simplifies the process of managing the lifetimes of Objective-C objects. Instead of you having to remember when to retain or release an object, ARC evaluates the lifetime requirements of your objects and automatically inserts the appropriate method calls at compile time. ARC replaces the traditional managed memory model style of programming found in earlier versions of iOS. Any new projects you create automatically use ARC. And Xcode provides migration tools to help convert existing projects to use ARC.

3) SQLite - The SQLite library lets you embed a lightweight SQL database into your application without running a separate remote database server process. From your application, you can create local database files and manage the tables and records in those files. The library is designed for general-purpose use but is still optimized to provide fast access to database records. The header file for accessing the SQLite library is located in <iOS_SDK>/usr/include/sqlite3.h, where <iOS_SDK> is the path to the target SDK in your Xcode installation directory.

4) XML Support - The Foundation framework provides the NSXMLParser class for retrieving elements from an XML document. Additional support for manipulating XML content is provided by the libXML2 library. This open source library lets you parse or write arbitrary XML data quickly and transform XML content to HTML. The header files for accessing the libXML2 library are located in the <iOS_SDK>/usr/include/libxml2/ directory,
where <iOS_SDK> is the path to the target SDK in your Xcode installation directory.

5) Block Objects - Introduced in iOS 4.0, block objects are a C-level language construct that you can incorporate into your C and Objective-C code. A block object is essentially an anonymous function and the data that goes with that function, something which in other languages is sometimes called a *closure* or *lambda*. Blocks are particularly useful as callbacks or in places where you need a way of easily combining both the code to be executed and the associated data.

In iOS, blocks are commonly used in the following scenarios:

a) As a replacement for delegates and delegate methods

b) As a replacement for callback functions

c) To implement completion handlers for one-time operations

d) To facilitate performing a task on all the items in a collection

e) Together with dispatch queues, to perform asynchronous tasks

Other technologies supported by core service layer are – Data Protection , File-Sharing Support , Grand Central Dispatch , In-App Purchase.

D. Core OS Layer

The Core OS layer contains the low-level features that most other technologies are built upon. Even if you do not use these technologies directly in your applications, they are most likely being used by other frameworks. And in situations where you need to explicitly deal with security or communicating with an external hardware accessory, you do so using the frameworks in this layer.[9]

VIII. iOS Developer Tools

To develop applications for iOS, you need an Intel-based Macintosh computer and the Xcode tools. Xcode is Apple's suite of development tools that provide support for project management, code editing, building executables, source-level debugging, source-code repository management, performance tuning, and much more. At the center of this suite is the Xcode application itself, which provides the basic source-code development environment. Xcode is not the only tool, though, and the following sections provide an introduction to the key applications you use to develop software for iOS.[10]

A. XCODE

The focus of your development experiences is the Xcode application. Xcode is an integrated development environment (IDE) that provides all of the tools you need to create and manage your iOS projects and source files, assemble your user interface, build your code into an executable, and run and debug your code either in iOS Simulator or on a device. Xcode incorporates a number of features to make developing iOS applications easier, including the following:

1) GCC compilers supporting C, C++, Objective-C, Objective-C++, and other languages.

2) A project management system for defining software products.

3) A code-editing environment that includes features such as syntax coloring, code completion, and symbol indexing.

4) An integrated editor for creating storyboard and nib files.

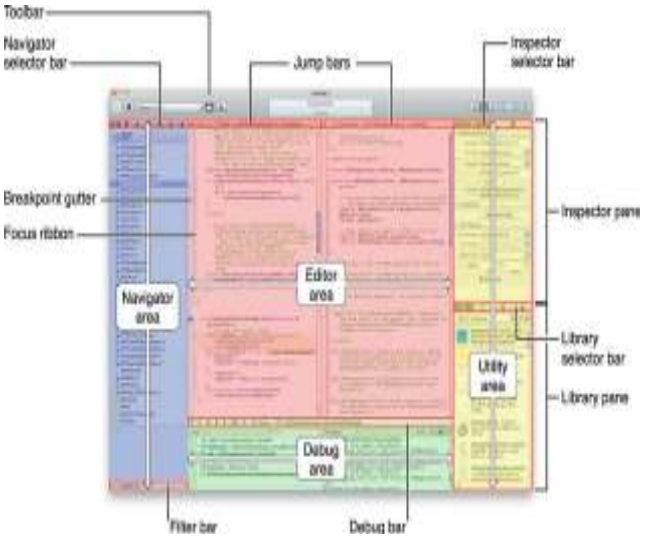

Figure 8.A.1 : XCODE Workspace

B. Instruments

To ensure that you deliver the best user experience for your software, the Instruments environment lets you analyze the performance of your iOS applications while running in Simulator or on a device. Instruments gather data from your running application and presents that data in a graphical display called the *timeline view*. You can gather data about your application's memory usage, disk activity, network activity, and graphics performance. The timeline view can display all the types of information side by side, letting you correlate the overall behavior of your application, not just the behavior in one specific area.

C. The Developer Library

The iOS Developer Library contains the documentation, sample code, tutorials, and other information you need to write iOS applications. Because the developer library contains thousands of pages of documentation, ranging from high-level getting started documents to low-level API reference documents, understanding how to find the information is an important step in the development process. The developer library uses a few techniques for organizing content that should make it easier to browse.

IX. HARDWARE SECURITY FEATURES IN IOS

On every device, speed and power efficiency are critical. Cryptographic operations are complex and can introduce performance problems if not designed and implemented correctly. Every iOS device has a dedicated AES 256 crypto engine built into the DMA path between the flash storage and main system memory, making file encryption highly efficient. Along with the AES engine, SHA-1 is implemented in hardware, further reducing cryptographic operation overhead. The device's unique ID (UID) and a device group ID (GID) are AES 256-bit keys fused into the application processor during manufacturing. No software or firmware can read them directly; they can see only the results of encryption or decryption operations performed using them. The UID is unique to each device. The GID is common to all processors in a class of devices, and is used as an additional level of protection when delivering system software during installation and restore. Burning these keys into the silicon prevents them from being tampered with or bypassed, and guarantees that they can be accessed only by the AES engine. The UID allows data to be cryptographically tied to a particular device. The UID is not related to any other identifier on the device. Apart from the UID and GID, all other cryptographic keys are created by the system's random number generator (RNG) using an algorithm based on Yarrow. System entropy is gathered from interrupt timing during boot, and additionally from internal sensors once the device has booted.[11]

X. IOS TECHNOLOGIES PACKAGED AS FRAMEWORKS

Apple delivers most of its system interfaces in special packages called frameworks. A framework is a directory that contains a dynamic shared library and the resources (such as header files, images, helper apps, and so on) needed to support that library. To use frameworks, you link them into your app project just as you would any other shared library. Linking them to your project gives you access to the features of the framework and also lets the development tools know where to find the header files and other framework resources.

XI. IOS AND OS X SHARE MANY OF THE SAME FRAMEWORKS

If you are an existing Cocoa developer, writing iOS apps should feel familiar. Many of the technologies found in OS X can also be found in iOS. The biggest differences between the two platforms occur at the user interface level but even then there are similarities in how you present and manage your app's interface. As a result, porting apps from OS X to iOS is possible with a little bit of work.

XII. CONCLUSION

Each component of the iOS security platform, from hardware to encryption to device access, provides organizations with the resources they need to build enterprise-grade security solutions. The sum of these parts gives iOS its industry-leading security features, without making the device difficult or cumbersome to use. Study of the layers of iOS helps users to understand the building blocks of apps used in this operating system. Using the simple basic language like c and

c++ , apple has developed this operating system. Along with the employees working in apple .inc, it also gives ample opportunity to common users to create, develop and upload their apps. With the above study of iOS in iPhone, we can conclude that the system which is the centre of attraction for every intelligent mind around is in a real sense a block builds out of very basic technologies which was worked hard for years.

XIII. ACKNOWLEDGMENT

We thank Prof. Dr. Rajkishor Tugnayat sir (H.O.D) I.T Dept JDIET,Yavatmal for inspiring and guiding us. We extend our thanks to Ankita Patil mam and Kinjal Patel mam (Assistant Prof. CSE Dept JDIET,Yavatmal) for helping us out.

XIV. REFERENCES

[1] Honan, Matthew (January 9, 2007). "Apple unveils iPhone". *Macworld*. Retrieved January 16, 2010.

[2] "Apple – iPhone – Features – OS X". Archived from the original on January 11, 2008. Retrieved June 15, 2010.

[3] Gonsalves, Antone (October 11, 2007). "Apple Launches iPhone Web Apps Directory". *InformationWeek*. Retrieved February 16, 2010.

[4] Ziegler, Chris (June 11, 2007). "Apple announces third-party software details for iPhone". Engadget. Retrieved June 9, 2010.

[5] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html

[6] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html

[7] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/MediaLayer/MediaLayer.html#//apple_ref/doc/uid/TP40007898-CH9-SW4

[8] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/CoreServicesLayer/CoreServicesLayer.html#//apple_ref/doc/uid/TP40007898-CH10-SW5

[9] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1

[10] http://developer.apple.com/library/ios/#documentation/miscellaneous/conceptual/iphoneostechoverview/CoreOSLayer/CoreOSLayer.html#//apple_ref/doc/uid/TP40007898-CH11-SW1

[11] http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf

Creating the iOS library. iOS problems and pitfalls. Where to go from here. Created. 27 August 2012. Requirements.Â Ensuring support for iOS 4.* devices. By default Xcode sets the deployment target to match the SDK you are using. Xcode now only comes with the iOS 5.0 SDK, so by default your library project targets 5.0. To remedy this, click the project name in the bar on the left side. In Build Settings search for "deployment" and set iOS Deployment Target to 4.0.Â That being said, you should name the internal initializer functions in a unique way. Appending the name of your extension is an easy way to accomplish this. Here's an example