

Chapter

# 9

## Mobile Platform

**CHAPTER AUTHORS**

Gopalakrishnan Kadambari

Koh Zi Han

Sneha Girish Tilak

Vu Viet Quynh Huong



<b>CONTENTS</b>	
1	Introduction ..... 5
2	Core Concepts..... 5
2.1	Platforms and operating systems..... 5
2.2	Application frameworks and applications..... 6
3	Mobile application development..... 6
3.1	Choose your platform..... 6
3.2	Know your application framework..... 9
3.3	Know your device..... 11
3.4	Explore available tools..... 11
3.5	Support during development..... 11
3.6	Publish your application ..... 12
3.7	Monetize on your application ..... 13
4	From desktop development to mobile development..... 14
4.1	Interaction paradigm..... 14
4.2	Resource constraints ..... 14
4.3	Resource opportunities ..... 16
5	Looking ahead..... 18
5.1	Superphones..... 18
5.2	Detecting the pressure..... 18
6	Summary ..... 19
7	Bibliography..... 20
	Appendix..... 21



## 1 INTRODUCTION

*“Mobile is the future”*

- Eric Schmidt (Ex-CEO, Google)

Mobiles are rapidly becoming an integral part of people’s lives. Gartner expects the mobile applications industry to increase to a massive 21.6 billion sales figure by 2013 generating revenue of 29.5 billion dollars(Gartner Newsroom 2010). Therefore, as developers, we can look at this phenomenon as an opportunity to develop software that has a potential of reaching a vast audience.

Also, for individual mobile developers, this industry could offer very lucrative benefits. Even today, the possibility of making profits is huge. Success stories of developers, both novice and expert, becoming instant millionaires after creating hit application(s) are plenty. The job opportunities for mobile developers are very good too. According to ACM career news, there is a high demand for mobile application developers but not enough skilled people are available(ACM CareerNews 2010). Hence, if, as a developer, your resume boasts mobile development knowledge, you have an advantage over those who don’t. This knowledge will be useful even if you do not want to focus your career just on mobile application development. More and more businesses are entering the mobile application industry and introducing mobile applications for their users (at&t Media Newsroom 2011); hence you can use this knowledge in almost any field you enter.

Thus, this chapter aims to be a starting point for the readers into the world of mobile application development by giving an introduction and basic overview of different aspects of mobile platforms and applications.

The chapter has been organized as follows:

Firstly, we will cover some basic concepts important to mobile development. Then, we will give a step-by-step guide of developing applications, from designing to publishing. Simultaneously, we will introduce and explain the related concepts. Later, we cover the differences between desktop development and mobile development - mainly the constraints and opportunities mobile development has. To conclude, we look at what the future of mobile holds and then give a summary of the concepts covered in the chapter.

## 2 CORE CONCEPTS

Before we can delve into mobile application development, let’s look at some of the core concepts.

### 2.1 Platforms and operating systems

To be able to develop mobile applications, we need to first understand the underlying principles of mobile platforms and how they are related to mobile applications. Mobile platforms are basically those that allow software and services to be run on devices (Fling 2009). Examples of mobile platforms include Palm, BlackBerry, iPhone, Android and Windows Mobile.

Mobile operating systems provide tools that allow application to share data and services. Examples of mobiles OS includes Palm OS, Symbian, Windows Mobile, Mac OS X and Android.

(Note: For simplicity, ‘mobile platforms’ and ‘mobile operating system’ will be used interchangeably in the rest of this chapter).

For this chapter, we will be focusing on three platforms:



## 2.2 Application frameworks and applications

Mobile application frameworks are set of class libraries that provide developers an interface to build applications. Examples of application frameworks include Java ME, Cocoa Touch, Android SDK and WebKit.

Applications are built using application frameworks on the given platform. The next section covers steps the developers needs to take while developing mobile applications.

## 3 MOBILE APPLICATION DEVELOPMENT

In this section we will look at all the steps you need to consider in designing, developing, publishing and maintaining your application. The major concepts and ideas are explained as and when they are introduced.

### 3.1 Choose your platform

The very first step in mobile application development is to decide on which platform to build the application for. This may depend on the number of factors including the below:

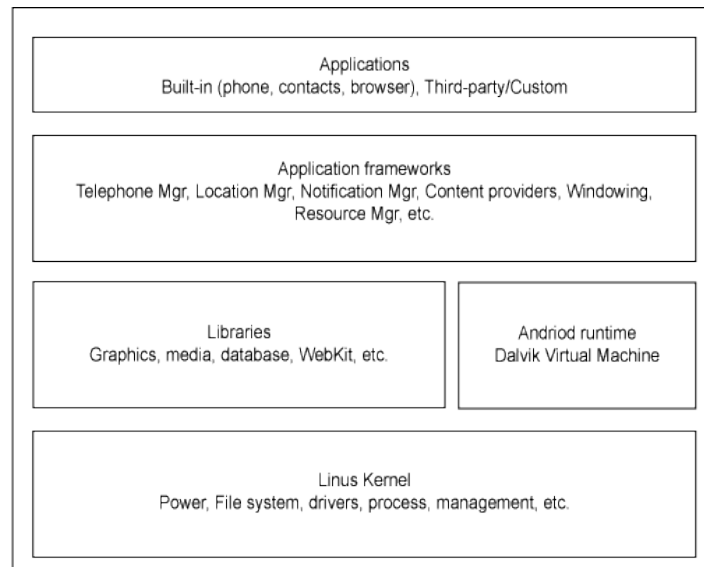
1. Target users: The BlackBerry platform is known to be dominant in the enterprise market and continues to do so - though other platforms are gradually gaining market share (Gohring 2010) . Therefore, if your application is targeted at enterprise users, BlackBerry may be your best choice, for now. Whereas, if your application is to be used by the mass market, you can decide to develop your application for platforms such as Android and iOS.
2. Market share: Analyze the current and the predicted market share to determine which platform will have the highest market share in the current and the coming years. Refer to 0 Appendix, point 1 for Gartner's forecast for various platforms' market share.
3. Features: Each platform has some unique features which you can leverage upon in your application. Consider these when designing your applications to best optimize your application.

Let's take a look at our chosen platforms:



Basic info: Android was founded by Android Inc. in October, 2003 and later purchased by Google Inc. in August, 2005.

Below is the basic architecture of Android: **Invalid source specified.**



**Linux Kernel:** The Linux Kernel handles core system services and acts as a hardware abstraction layer (HAL) between the physical hardware of the device and the Android SDK. Some of the major functionalities include low level memory management, process management, networking and other OS related services.

**Native Libraries:** The Android native libraries are all shared libraries written in C or C++, compiled for the particular hardware architecture used by the phone. Some of the important libraries include OpenGL for 2D and 3D graphics, SQLite database and the WebKit library for browsing HTML content.

**Android Runtime:** This includes the Dalvik Virtual Machine (DVM) and the core java libraries. Based on java VM, the Dalvik design has been optimized for the low memory requirements of mobile devices. It allows multiple VMs to run concurrently.

**Application Framework:** This layer provides the high-level building blocks with which applications are created. The important classes are Activity manager that controls the life cycle of the applications, Location manager that provide location sensing options and Resource manager.

**Applications:** The highest layer in the Android architecture is the applications themselves. This consists of applications and widgets. Widgets are gadgets that operate only on a small area of the Home screen.

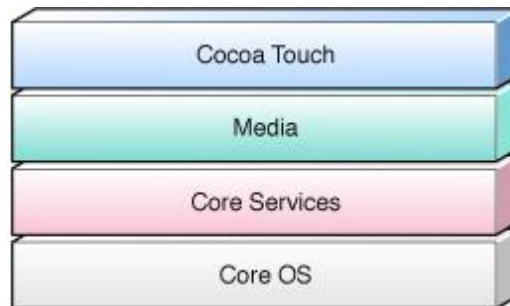
Unique features:

1. Open: The Android platform is provided through open source licensing. Developers have unprecedented access to the handset features when developing applications.
2. Free: Android applications are free to develop. It uses familiar and inexpensive development tools and the software development kit (SDK) is also freely available for download. It requires only a registration charge for publishing your applications for the first time. Consequent updates to the application are free of charge.



Basic info: iOS (formerly known as iPhone OS) was released on June, 2007 by Apple Inc.

Below is the basic architecture of iOS:



The kernel in iOS is based on the same variant of the Mach kernel in Mac OS X. On top of this kernel are layers of services that are used to implement applications on the platform. These layers include the Core OS, Core Services, Media services, and the Cocoa Touch layer. Layering as such gives application developers different degrees of control. The intention is for developers to use the higher-level frameworks for general development, as they free developers of the burden of implementing standard system behaviors. Where this is insufficient, the lower-level frameworks give developers more control.

**Core OS, Core Services:** These provide fundamental low-level services like file I/O and network sockets.

**Media:** This layer provides services used to support 2D and 3D drawing, as well as audio and video.

**Cocoa Touch:** This layer provides the fundamental infrastructure used by applications. Besides providing object-oriented programming support for collections, file management and network operations, this layer also provides access to the device user's profile information, accelerometer, and other hardware features.



**Basic info:** Windows Phone 7 was launched in February, 2010 by Windows Inc. it is Microsoft's first mobile operating platform to focus on the consumer market instead of the enterprise market

The architecture details of Window Phone 7 have not been released for the public by Microsoft yet.

**Unique features:**

One significant distinguishing component of Windows Phone 7 is in the UI, dubbed the Metro UI utilizing the Metro Design Language by Microsoft. The Metro UI utilizes simple, clean and modern as its design language, and revolves around having easily accessible and glance-able information.

Two concepts that Metro uses to achieve these goals are *live tiles* and *hubs*. Any application can be pinned to "Start" as a live tile, and applications can choose to animate or update the live tiles with key information pertaining to the app (e.g. the number of unread e-mails for an E-mail app). This allows the user to see at a glance at all times if there are items that are in need of his attention. The second are Hubs, of which functions of a similar nature are grouped together. The People hub for example, not only acts as an address book across the user's phone, mail, exchange server accounts, but also integrates with Facebook and shows Facebook statuses and updates for the users' contacts. Likewise, Pictures hub do not just contain pictures and videos



that the user has taken on his phone, but also surfaces photos shared by the users' friends on Facebook.

This approach taken by Metro differs greatly from the other two platforms, for instead of taking an app-centric approach where the phone is just a gateway to applications, Metro focuses more on the user experience and in making it easy for information that the user cares about to surface up and be easily accessible.

### 3.2 Know your application framework

Having chosen your platform, the next step is to familiarize yourself with the application framework. Mobile application frameworks are set of class libraries that provide developers an interface to build applications. Examples of application frameworks include Java ME, Cocoa Touch, Android SDK and WebKit.



Android applications are written in **Java** programming language using the Android libraries provided by the SDK. Developers can also build software components in C and C++ using the Android Native Development Kit (NDK) but it is not intended as a full alternative to the java development model.

Android being an open platform, developers have access to the same application framework as the core phone applications. The main components of the framework are **Invalid source specified.:**

1. Views – basic building block for UI components like layouts and menus.
2. Content Providers – enables sharing data between applications such as access to the user contacts.
3. Resource Manager
4. Notification Manager – enables applications to display alerts in the status bar.
5. Activity Manager – manages the activity lifecycle (Refer to 0, 2)



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 applications, and so on) needed to support that library (iOS Technology Overview n.d.).

The frameworks have been arranged similar to the layer in the architecture of iOS.

1. Core OS Layer:
  - a) Accelerate Framework: Provides interface for performing mathematic calculations.
  - b) External Accessory Framework: This framework is used to communicate with any hardware accessories attached to an iOS-device.
  - c) Security Framework: Provides support for adding security features like managing certificate, keys, trust policies and so on.
2. Core Services Layer

Some of the frameworks under this layer include:

- a) Address Book Framework: Provides support for accessing contacts' information on the user's contact database.
- b) CFNetwork Framework: Set of C-based interfaces that provide support for working with network protocols.
- c) Core Data Framework: Manages applications that use the Model-View-Controller application.
- d) Core Foundation Framework: Set of C-based interfaces that provides basic data management and service features for applications like date/time management, preferences management and so on.
- e) Event Kit Framework: Provides an interface to access calendar events on a user's device.

### 3. Media Layer

Some of the frameworks under this layer include:

- a) Assets Library Framework: Supports retrieving photos and videos from the user's device.
- b) AV Foundation Framework: Provides classes for playing audio content.
- c) OpenGL ES Framework: Provides interface for drawing 2D and 3D content.

### 4. Cocoa Touch Framework

Some of the frameworks under this layer include:

- a) Address Book UI Framework: Provides interface for creating new contacts and editing current contacts.
- b) Event Kit UI Framework: Provides interface for viewing and editing calendar-related data.
- c) iAd Framework: Lets the developer add advertisements to the applications
- d) Map Kit Framework: Provides an interface to add a scrollable map to the existing view.
- e) UIKit Framework: Supports accessing some device specific features like accelerometer data, camera, battery life information and so on



This platform provides two frameworks: **Invalid source specified.**

The Silverlight framework: For event-driven, XAML-based application development that allows developers to develop creative mark-up based user experiences.

The XNA framework: for loop-based games that enables immersive and fun gaming and entertainment experiences.

### 3.3 Know your device

Knowing the specifications of the device you will be developing the application for, is extremely essential. The device will be the point of interaction between the users and your application and hence incorporating these specifications in your application will allow you to provide a better user experience. Things to keep in mind include features like screen dimensions, position and orientation of buttons, processing power, graphics capabilities, multi-touch capabilities and so on.

### 3.4 Explore available tools

Explore the tools at hand and determine which will best help you in developing the applications. For mobile applications, emulators can be used to simulate mobile conditions on desktops to do some preliminary testing.



The most popular and familiar IDE is Eclipse which provides full support for Android Applications through the Android Development Tools (ADT) plug-in. Android applications can also be developed from the command-line or a simple text editor with the appropriate tools installed. There are third party tools also available like the IntelliJ IDEA which ensures support for the latest Android SDK and provides additional features like quick navigation between code and resources, graphical debugger and unit testing support.



Below are the tools available for developing applications for iPhone.

Xcode: Complete development environment provides project management, a powerful source editor, and a graphical debugger.

iOS Simulator: Run, test, and debug your application locally on your Mac using a simulated iPhone and iPad.

Instruments: Collect, display, and compare performance data graphically in real-time to optimize your application.

Interface Builder: Interface Builder makes designing a user interface as easy as drag and drop.



All tools for Windows Phone 7 development are provided as a complete downloadable package from the Getting Started page on App Hub. The tools include Visual Studio 2010, Microsoft Expression Blend 4.0, XNA Game Studio 4.0, Windows Phone Emulator, Silverlight and .NET Framework 4. If you already have a copy of these software, the installer will just install the SDK and Windows Phone development support on top of your existing software. If not, a free, express edition of the tools will be installed instead.

### 3.5 Support during development

You can use various resources to help you with any problems or doubts encountered during development. Platforms have official forums where developers can help each others with any development issue. You can also make use of IRC channels which provide an easy and quick way to finding solutions.



<http://developer.android.com/resources/community-groups.html>

The above link is the official page of resources for the Android developers. It has links to various discussion forums and mailing lists.

IRC: Provides two channels.

1. #android on irc.freenode.net which is for any general discussion on the Android platform.
2. #android-dev on irc.freenode.net which is for discussions specific to developing applications on the Android platform.



Apple Developer Forums: Post iOS SDK development topics and questions for an open discussion with other iOS developers and Apple engineers.

Getting started videos and documents: Cover a range of topics, from tools and frameworks to development best-practices and design methods.

iOS Reference Library: A rich collection of documentation, guides, and articles categorized so you can quickly find the information you're looking for.

IRC: There is currently one widely used channel for iOS developers

1. #iphonedev on irc.freenode.net



<http://forums.create.msdn.com/forums/default.aspx?GroupID=19>

The above link is the official page of resources for the Window Phone developers.

IRC: There is currently no dedicated channel for Window Phone 7 development.

### 3.6 Publish your application

After your application is developed and tested, it needs to be published in the application stores. Application stores are a single point of contact for downloading applications and updates.(Online App Store definition n.d.). Developers add their applications, updates and price whereas users download and add reviews for the applications on the application stores.

To successfully publish your applications on the application stores, you need to follow the rules and regulations set by the respective platforms. For some platforms like iOS and Windows Phone 7, your application needs to get approved before you can be publish it in the marketplace.



Android Market: Developers need to follow a number of steps before they can publish their applications on the android market. The first step is digitally signing the application with a certificate using a private key held by the application developer. After signing the applications, the next step in deployment is publishing the application. Android applications can be distributed to end users in any way including through the Developer's web server. Android Market is a service provided by Google for users to find and download Android applications to their devices and for developers to distribute their applications for end users all around the world. To publish an application on the Android Market developer's need to first register and agree to the terms and conditions specified. A registration fee of 25\$ is required for the first time to create an account. Subsequently publishing updates and creating a license are free of charge.





**App Store:** Apple's App Store is the first smartphone application store. As of January 2011, it had 350,000+ available applications and reached 10B downloads. In March 2011, Apple CEO Steve Jobs announced that Apple had paid a total of US\$2B to its app developers (supposedly 70% of apps revenue).



Apple's App Store is famous for their tight control over submitted applications. Each application submitted will be reviewed by Apple employees before it gets approved. In order to pass this code review process, the application needs to show that (1) it works as advertised; (2) it does not crash the iDevice and (3) it does not use private APIs.

In order to publish apps to the AppStore, developers need to enroll into iOS Developers Program and pay an annual fee of \$99 (for individual membership). The application needs to go through complex code signing process before it is ready for release. Once it is ready, you can submit the app to Apple with basic information including Name, Category, Summary and Screenshots. After that all you can do is to wait. The review process often takes weeks to finish.



To deploy applications onto actual Windows Phone devices, the phone has to first be unlocked as a developer phone. This can be done via the Windows Phone Developer Registration tool that is installed with the SDK, though a developer has to first be formally registered with Microsoft as a member of App Hub. Membership for App Hub comes with an annual subscription of \$99.00 USD, though students can have the fee waived through the Dreamspark program

### 3.7 Monetize on your application

Having developed your application, you need to think of pricing strategies to monetize on your application. You can look at similar applications already present in the market and any prevalent market trends to determine how and where you want to position your application in the market.

Below are factors you need to consider:

#### **Free vs. Paid**

Decide whether you want to make your application available for free or impose some price. You can combine this aspect with the next two aspects to explore more monetization options.

If you plan on pricing on your application, experiment with the amount and the number of users downloading your application and aim to get the right price-user ratio.

#### **In-app purchases**


In-app purchases, as the name suggests, allow users to buy extra features from within your application. For e.g. you can offer a free level in a game application, or provide extra tools in a photo-editing application.


You can use this in pricing your application. For eg, you can distribute your application at no cost, and then earn money by having some priced in-app features.

While Android and iOS both support in-app purchases, Windows Phone 7 has not enabled this feature yet.

#### **Advertising**

This feature allows you to add advertisement to your applications. Most platforms have their own official ad networks through which you add the advertisements. You can only use third-party network that perform the same function.

 Official: AdMob; Third-party: Flurry, GreyStripe

 Official: iAd; Third-party: AdMob, Super Rewards

 Official: Microsoft Advertising pubCenter; Third-party: AdMob, AdDuplex

Some official ad-networks make adding advertisements to your applications as easy as drag-and-drop.

## 4 FROM DESKTOP DEVELOPMENT TO MOBILE DEVELOPMENT

As a desktop application developer, you need to be aware of the subtle and not-so-subtle differences between mobile application development and desktop application development. This section aims to explore these differences – the way users interact with your application, the environment in which they do so and the constraints and opportunities that mobiles' resources bring about.

### 4.1 Interaction paradigm

To build applications that will ultimately be consumed by users, we as developers need to understand the environment that the user will be using the applications in. While desktop users are usually in a static environment – either at home, office, college and so on, mobile users are on the go – using applications in transit, train rides, walking, and so on. Hence mobile users will have many factors vying for their attention while using your application. Therefore, the application needs to be easier to use, simple and allowing the user to use it even with divided attention.


### 4.2 Resource constraints


Mobiles bring about some constraints when compared to desktops and this needs to be considered when developing applications. Below are some of those constraints:

#### Runtime Memory

For desktops generally, each process may take around 50 – 200 MB whereas for mobile this available runtime memory is very less. For e.g. Nexus S has a 512MB RAM, but it enforces a 32 MB cap on the amount of memory an application can use.

Below are the limits for our chosen platforms,

 Android enforces a hard limit of 16 to 32MB on application, depending on the devices.

 The limit for iOS is not officially released to the public yet, but a safe range for your application n will be from 15 to 20 MB



Window Phone 7 sets the limit at 90MB for devices with less than 256MB of memory. No limits for devices with better memory.

It is important to follow the memory limits, failing which the OS will terminate your application. You can query the memory limits before running your applications or handle any warnings the OS may issue if your application is nearing the memory limit. You can take some measure to make sure you do not overshoot the limits.

1. Avoid memory leaks: Nullify all the references that are not needed any more, and the garbage collectors will reclaim this memory. For iOS, there are no garbage collectors hence your manually need to take care of memory allocations.
2. Load lazily: When handling large amounts of data, load only a chunk of data that is currently needed by the user.
3. Optimize resources: Focus on using the compressed format instead of the raw data format. For e.g., use binary instead of xml and 'png' instead of 'jpeg'.

### Storage

Mobiles have a very limited storage space as compared to desktops. For e.g., Samsung Focus has an 8GB ROM but this is still much smaller than the storage space on desktops.

You can take some steps to increase or save storage space.

1. Use external storage: The storage space can be extended by using external storage like SD cards. Currently, iOS does not cater to external storage.
2. Allow users to control their data: Due to a lack of proper file explorers, allow users to delete, backup and restore data hence giving them more control on which data they want to use currently.

### Battery

As mentioned in the interaction paradigms, desktop users are in a static environment and mostly have easy access to electricity as opposed to mobile users who are mostly on the go and need their batteries to last for a much larger period of time. Battery life will be affected by the usage of Wi-Fi, Bluetooth, GPS, sensors like accelerometers and magnetometers, the mobile's backlight and so on. Hence you can take the following measures to ensure that your application does not drastically reduce the mobile's battery.

1. Avoid accessing hardware: Access resources only as and when required. For e.g., write only the updated version to a file, update user locations not more frequently than required and so on.
2. Use dark background: Use a darker background as far as your application design allows it because screen illumination drains the battery. Hence, a brighter screen will reduce battery life faster than a darker screen.
3. Use push: Try to use the push model as opposed to the pull model.  
Push model: The network access is performed only when required.

Pull model: The mobile will keep querying to check for any updates and hence make many redundant and unnecessary network accesses.

### **Screen Size**

Mobile users are operating on a much smaller screen than desktop users. Some ways you can make it easier for the users to use your application are:

1. Use bigger font: Owing to the smaller screen size and the distracting environment, you a font that can make the information easily visible to the user.
2. Make the right thing visible: As the screen size is very limited, make only the important and currently required things visible on the screen. For e.g., functionalities used less frequently can be hidden behind a single button and accessible only when the user specifically asks for that information.

### **Input Device**

Desktops generally have separate input (keyboard and mouse) and output (screen) devices, whereas for mobiles the screen acts as both the input and output device and keyboards, if present, are much tinier than the desktop keyboards. To make it easier for users to use your application, you can make use of the following tips:

1. Use gestures: Do not just rely on the keyboard to get input. You can also use different gestures and sensor to take the inputs. This will be explored deeper in the next section.
2. Have good fault-tolerance: Working on a smaller screen increases the chances of users pressing something unintentionally. Therefore, always make sure that the users mean to perform the pressed actions. For e.g., have warning or re-confirming messages for critical actions like move and delete. For touchable inputs, keep reasonable space between screen buttons to allow for the finger size.

## **4.3 Resource opportunities**

Mobiles have features that allow users to interact with the device in more ways than those in a desktop do and hence provide many opportunities for developing applications that may not have been possible on a desktop. This section explores such features and the advantages they bring about from the developer's point of view.

### **Touch**

Touch is a feature that is not yet common in a desktop computer or laptop. Touch allows the users to directly interact with the mobile with no need of any additional devices such as a mouse or a keyboard. As opposed to using a mouse which is a learned skill, touch is very intuitive and hence allows mobile usage without much hassle. Another point is that, a mouse provides just one point of contact with the screen whereas touch incorporates multi-touch which allows interaction over a much larger area.

1. Single-touch

Single-touch includes gestures such as tap, double tap, slide, scroll etc. Platforms provide API support for these single-touch interactions.

2. Multi-touch



Multi-touch include gestures such pinch, reverse pinch (for zooming in and out of the screen), rotating and so on. Platforms provide API support for some of the standard multi-touch gestures and also allow developers to design other multi-touch gestures.

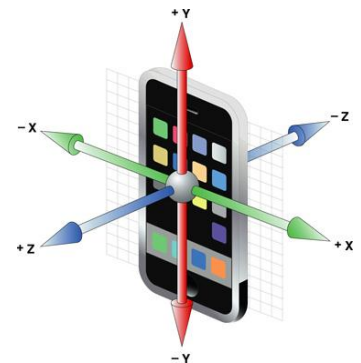
The things that developers need to take care of when handling multi-touch is:

1. Recognizing events of touch.
2. The information about the touch: its position (X/Y co-ordinates), size and pressure.
3. The type of action: to determine the movement i.e. to see if the touch has moved up, down etc.
4. Manipulating individual touches: Individual touches can be retrieved using identifiers that are assigned uniquely to each touch.

Based on the above info, developers can define their own gestures and how the application behaves with them.

### Accelerometer

An accelerometer is an in-built sensor in smartphones that is capable of measuring the tilt and the motion of the phone (Accelerometer - Mobile terms glossary n.d.). Unlike desktops, accelerometers allow users a no-touch interaction i.e. navigating the device with touching the input screen.



#### How it works

Using this example from Apple, the accelerometer uses XYZ axes to detect the motion of the phone. Developers can use the changes in these xyz values to know what kind of motion has occurred.

#### Usage

Accelerometers are commonly used in the auto-rotation of the mobile view when the orientation is changed. It is also widely used in racing gaming where the user can navigate the mobile to control the wheels in the game.

Accelerometers can also be used to design interesting applications. For e.g., there are apps that undo text when the phone shakes, locks it when it's faced down, scrolls up and down through the text depending on if it's a tilted forward or backward and so on.

### Location-based services

Mobile users are almost always on-the-go so as developers, you get more opportunities to interact with the user as compared to desktops and using their location can help you to personalize this interaction with the users.

#### How it works

As opposed to desktop which uses IP addresses to determine the location, mobile phones can also obtain the location using information from the GPS and network operators. Though GPS is the most widely used method, it is not totally accurate. Some ways can be implemented to get more accurate information about location.

1. Obtain several GPS readings and calculate the average.
2. iPhone uses A-GPS, short for assisted-GPS, which combines location readings from GPS, Wi-Fi signals and cell towers to give a more accurate reading.

## Camera

Developers also can make use of the in-built camera and combine it with other features like location or accelerometers to enhance the application experience. For e.g. geo-tagging combines location-based services and the camera to allow users to tag photos with their location.

## 5 LOOKING AHEAD...

*"Mobile is the future"*. So what's in store for smartphones in the future and hence for your mobile applications?

### 5.1 Superphones

(O'Dell 2010)

Coined by Google Inc., superphones are supposed to be smarter and more powerful than smartphones. Features of superphones include:

1. Allowing developers much deeper access to hardware capabilities, hence allowing developers more control over their applications and what they can do.
2. Larger screen
3. Camera – between five and eight megapixels.
4. Multiple microphones: multiple microphones will allow for better call and recording quality.
5. Powerful processors: Phones with gigahertz processor. The current best include phones of 1Ghz processors.

These features and others will allow for better quality applications and their interaction with the user.

### 5.2 Detecting the pressure

Imagine if your touch-screen can detect the pressure of your touch and is able to differentiate between a gentle tap and a hard poke. How it works is that there is an internal switch that can detect the bend in the touch screen. These readings and the data about the touch can tell you exactly where and how hard the screen was pressed **Invalid source specified..** This, combined with multi-touch feature can allow you many possible ways of designing your application. Though not mainstream yet, research is on-going and maybe widely used in the upcoming years.

Technology is constantly changing, and it is best to keep track of those so that you can incorporate the latest technology in your application.

6 SUMMARY

Platforms: A Comparison



Google Inc.

Java



Android Market

\$25 registration fee

Easily deploy on emulator

Easily deploy on device

Apple Inc.

Objective-C



App Store

\$99/year to publish

Easily deploy on emulator

Require iOS University Accounts to deploy

Microsoft Corp.

.NET Languages



App Hub

\$99/year to publish Free for students through Dreamspark

Easily deploy on emulator

Must register to unlock phone and deploy on phone

## Desktop to Mobile

In this section we discussed about the differences brought about when switching from desktop application development to mobile application development.

1. We covered the **interaction paradigm** explaining how a mobile user is different from a desktop user.
2. We covered some of the **constraints** mobile resources bring about such as runtime memory, storage, battery life, screen size, input devices and showed some tips that could be implemented to overcome these constraints. We then covered the **opportunities** brought about by mobile resources like the touch features, accelerometer, location-based services and in-build cameras.

## 7 BIBLIOGRAPHY

*Accelerometer* - *Mobile terms glossary*.  
<http://www.gsmarena.com/glossary.php3?term=accelerometer> (accessed 18 March, 2011).

*ACM CareerNews*. 21 December, 2010.  
<http://plone.acm.org/membership/careernews/archives/acm-careernews-for-tuesday-december-21-2010/>.

Amit Bahree, Dennis Mulder, Shawn Cicoria, Chris Peiris, Nishith Pathak. *Pro WCF: Practical Microsoft SOA Implementation [Paperback]*. Apress, 2007.

*at&t Media Newsroom*. 15 March, 2011. <http://www.att.com/gen/press-room?pid=19326&cdvn=news&newsarticleid=31689&mapcode=enterprise>.

Chappell, David. *Introducing Windows Communication Foundation in .NET Framework 4*. March, 2010. <http://msdn.microsoft.com/library/ee958158.aspx> (accessed 9 March, 2011).

Conder, Shane, and Lauren Darcey. *Android wireless Application Development*. New Jersey: Addison-Wesley Professional, 2010.

Corporation, Oracle. *The Java EE 5 Tutorial*.  
<http://download.oracle.com/javaee/5/tutorial/doc/bnazq.html> (accessed 10 March, 2011).

*Creating an iPhone Application*.  
[http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/Creating\\_an\\_iPhone\\_App/\\_index.html](http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/Creating_an_iPhone_App/_index.html) (accessed 18 March, 2011).

Eichengreen, Barry. *One Economy, Ready or Not: Thomas Friedman's Jaunt Through Globalization*. May/June, 1999. <http://www.foreignaffairs.com/articles/55017/barry-eichengreen/one-economy-ready-or-not-thomas-friedman-s-jaunt-through-globaliz> (accessed 6 March, 2011).

Erl, Thomas. *SOA Principles of Service Design*. Prentice Hall, 2007.

Fling, Brian. *Mobile Design and Development*. O'Reilly Media Inc. , 2009.

*Gartner Newsroom*. 18 January, 2010. <http://www.gartner.com/it/page.jsp?id=1282413>.

Gohring, Nancy. *Smartphones in the enterprise: A changing landscape*. 18 December, 2010. [http://www.computerworld.com/s/article/print/9201298/Smartphones\\_in\\_the\\_enterprise\\_A\\_changing\\_landscape?taxonomyName=Mobile%20and%20Wireless&](http://www.computerworld.com/s/article/print/9201298/Smartphones_in_the_enterprise_A_changing_landscape?taxonomyName=Mobile%20and%20Wireless&)

Goncalves, Antonio. *Beginning Java(TM) EE 6 with GlassFish(TM) 3: From Novice to Professional*. Apress, Inc., 2009.

Hewitt, Eben. *Java SOA Cookbook*. O'Reilly Media, 2009.

*iOS Technology Overview*.  
[http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSOverview/iPhoneOSOverview.html#//apple\\_ref/doc/uid/TP40007898-CH4-SW6](http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSOverview/iPhoneOSOverview.html#//apple_ref/doc/uid/TP40007898-CH4-SW6).

Jane Laudon, Kenneth Laudon. *Essentials of Management Information Systems*. Prentice Hall, 2007.

Judith Hurwitz, Robin Bloor, Carol Baroudi, Marcia Kaufman. *Service Oriented Architecture for Dummies*. Wiley Publishing, Inc., 2007.

Kalin, Martin. *Java Web Services: Up and Running*. O'Reilly Media, Inc., 2009.

Klein, Scott. *Professional WCF Programming: .NET Development with the Windows Communication Foundation*. John Wiley & Sons, 2007.

O'Dell, Jolie. *What makes a Smartphone a Superphone?* 12 July, 2010.  
<http://mashable.com/2010/07/12/superphone/>.

*Online App Store definition*.  
[http://www.pcmag.com/encyclopedia\\_term/0,2542,t=online+app+store&i=62644,00.asp](http://www.pcmag.com/encyclopedia_term/0,2542,t=online+app+store&i=62644,00.asp).

## APPENDIX

Forecast for mobile platform sales.

<b>Forecast: Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units)</b>				
<b>OS</b>	<b>2009</b>	<b>2010</b>	<b>2011</b>	<b>2014</b>
Symbian	80,876.3	107,662.4	141,278.6	264,351.8
Market Share (%)	46.9	40.1	34.2	30.2
Android	6,798.4	47,462.1	91,937.7	259,306.4
Market Share (%)	3.9	17.7	22.2	29.6
Research In Motion	34,346.8	46,922.9	62,198.2	102,579.5
Market Share (%)	19.9	17.5	15.0	11.7
iOS	24,889.8	41,461.8	70,740.0	130,393.0
Market Share (%)	14.4	15.4	17.1	14.9
Windows Phone	15,031.1	12,686.5	21,308.8	34,490.2
Market Share (%)	8.7	4.7	5.2	3.9
Other Operating Systems	10,431.9	12,588.1	26,017.3	84,452.9
Market Share (%)	6.1	4.7	6.3	9.6
<b>Total Market</b>	<b>172,374.3</b>	<b>268,783.7</b>	<b>413,480.5</b>	<b>875,573.8</b>
Source: Gartner (August 2010)				

Activity Lifecycle **Invalid source specified.**

Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity -- the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

- If an activity in the foreground of the screen (at the top of the stack), it is *active* or *running*.
- If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is *paused*. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.
- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.