LECTURE NOTES

on

MOBILE APPLICATIONS AND SERVICES

M.Tech III SEMESTER

(IARE - R18)

Prepared by

Ms. B Vijaya Durga

Assistant Professor

Computer Science and Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal- 500 043, Hyderabad

UNIT 1

Introduction to mobile computing

Mobile Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be connected to a fixed physical link. The main concept involves −

Mobile communication

Mobile hardware

Mobile software

Mobile communication

The mobile communication in this case, refers to the infrastructure put in place to ensure that seamless and reliable communication goes on. These would include devices such as protocols, services, bandwidth, and portals necessary to facilitate and support the stated services. The data format is also defined at this stage. This ensures that there is no collision with other existing systems which offer the same service.

Since the media is unguided/unbounded, the overlaying infrastructure is basically radio wave-oriented. That is, the signals are carried over the air to intended devices that are capable of receiving and sending similar kinds of signals.

Mobile Hardware

Mobile hardware includes mobile devices or device components that receive or access the service of mobility. They would range from portable laptops, smart phones, tablet Pc's, Personal Digital Assistants.

These devices will have a receptor medium that is capable of sensing and receiving signals. These devices are configured to operate in full- duplex, whereby they are capable of sending and receiving signals at the same time. They don't have to wait until one device has finished communicating for the other device to initiate communications.

Above mentioned devices use an existing and established network to operate on. In most cases, it would be a wireless network

Mobile software

Mobile software is the actual program that runs on the mobile hardware. It deals with the characteristics and requirements of mobile applications. This is the engine of the mobile device. In other terms, it is the operating system of the appliance. It's the essential component that operates the mobile device.

Since portability is the main factor, this type of computing ensures that users are not tied or pinned to a single physical location, but are able to operate from anywhere. It incorporates all aspects of wireless communications.

Introduction to Android Development Environment

Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team). Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code. Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others. This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phenomenon of Apple's iPhone, a hugely successful product that revolutionized the smartphone industry. Such companies include Motorola and Sony Ericsson, which for many years have been developing their own mobile operating systems. When the iPhone was launched, many of these manufacturers had to scramble to find new ways of revitalizing their products. These manufacturers see Android as a solution — they will continue to design their own hardware and use Android as the operating system that powers it. The main advantage of adopting Android is that it offers a unified approach to application development. Developers

need only develop for Android, and their applications should be able to run on numerous different devices, as long as the devices are powered using Android. In the world of smartphones, applications are the most important part of the success chain. Device manufacturers therefore see Android as their best hope to challenge the onslaught of the iPhone, which already commands a large base of applications. Android Versions Android has gone through quite a number of updates since its first release. Table 1-1 shows the various versions of Android and their codenames.

Table 1-1: A Brief History of Android Versions

Android Version	Release Date	Codename
1.1	9 February 2009	
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 October 2009	Éclair
2.2	20 May 2010	Froyo
2.3	6 December 2010	Gingerbread
3.0	Unconfirmed at the time of writing	Honeycomb

Features of Android As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

- Storage — Uses SQLite, a lightweight relational database, for data storage. Chapter 6 discusses data storage in more detail.
- Connectivity — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX. Chapter 8 discusses networking in more detail.
- Messaging — Supports both SMS and MMS. Chapter 8 discusses messaging in more detail.
- Web browser — Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- Media support — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- Hardware support — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS Multi-touch — Supports multi-touch screens Multi-tasking — Supports multi-tasking applications
- Flash support — Android 2.3 supports Flash 10.1.

- Tethering — Supports sharing of Internet connections as a wired/wireless hotspot Architecture of Android In order to understand how Android works, take a look at Figure 1-1, which shows the various layers that make up the Android operating system (OS).

Architecture of Android

In order to understand how Android works, take a look at Figure 1-1, which shows the various layers that make up the Android operating system (OS).

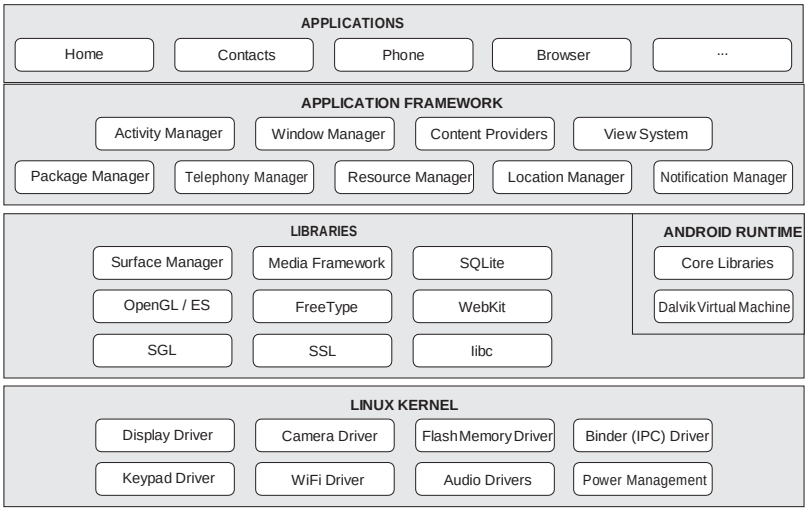


Figure 1-1

The Android OS is roughly divided into five sections in four main layers:

Linux kernel — This is the kernel on which Android is based. This layer contains all the low- level device drivers for the various hardware components of an Android device.

Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android appli- cation to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

Applications — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

Android devices in the market

Android devices come in all shapes and sizes. As of late November 2010, the Android OS can be seen powering the following types of devices:

- Smart phones
- Tablets
- E-reader devices
- Net books
- MP4 players

Internet TVs Chances are good that you own at least one of the preceding devices. Figure 1-2 shows (clockwise) the Samsung Galaxy S, the HTC Desire HD, and the LG Optimus One smartphones.

Another popular category of devices that manufacturers are rushing out is the tablet. Tablet sizes typically start at seven inches, measured diagonally. Figure 1-3 shows the Samsung Galaxy Tab and the Dell Streak, which is a five- inch phone tablet

Besides smartphones and tablets, Android is also beginning to appear in dedicated devices, such as e-book readers. Figure 1-4 shows the Barnes and Noble's NOOKcolor, which is a color e-Book reader running the Android OS.

Figure 1-3

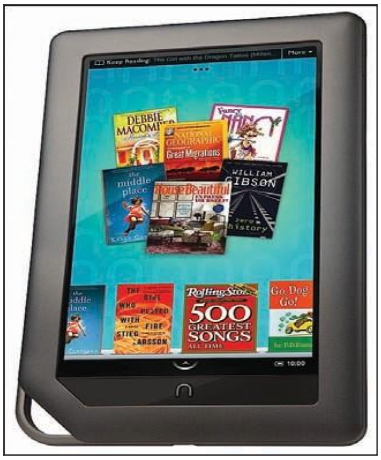


Figure 1-4

In addition to these popular mobile devices, Android is also slowly finding its way into your living room. People of Lava, a Swedish company, has developed an Android-based TV, call the Scandinavia Android TV (see Figure 1-5).

Google has also ventured into a proprietary smart TV platform based on Android and co-developed with companies such as Intel, Sony, and Logitech. Figure 1-6 shows Sony's Google TV.

Figure1-5

Figure 1

As mentioned earlier, one of the main factors determining the success of a smartphone platform is the applications that support it. It is clear from the success of the iPhone that applications play a very vital role in determining whether a new platform swims or sinks. In addition, making these applications accessible to the general user is extremely important. As such, in August 2008, Google announced the Android Market, an online application store for Android devices, and made it available to users in October 2008. Using the Market application that is preinstalled on their Android device, users can simply download third-party applications directly onto their devices. Both paid and free applications are supported on the Android Market, though paid applications are available only to users in certain countries due to legal issues. Similarly, in some countries, users can buy paid applications from the Android Market, but developers cannot sell in that country. As an example, at the time of writing, users in India can buy apps from the Android Market, but developers in India cannot sell apps on the Android Market. The reverse may also be true; for example, users in South Korea cannot buy apps, but developers in South Korea can sell apps on the Android Market. Chapter 11 discusses more about the Android Market and how you can sell your own applications in it.

Obtaining the required tools

Now that you know what Android is and its feature set, you are probably anxious to get your hands dirty and start writing some applications! Before you write your first app, however, you need to download the required tools and SDKs.

For Android development, you can use a Mac, a Windows PC, or a Linux machine. All the tools needed are free and can be downloaded from the Web. Most of the examples provided in this book should work fine with the Android emulator, with the exception of a few examples that require access to the hard- ware. For this book, I will be using a Windows 7 computer to demonstrate all the code samples. If you are using a Mac or Linux computer, the screenshots should look similar; some minor differences may be present, but you should be able to follow along without problems.

So, let the fun begin!

jAvA jdk

The Android SDK makes use of the Java SE Development Kit (JDK). Hence, if your computer does not have the JDK installed, you should start by downloading the JDK from `www.oracle.com/technetwork/java/javase/downloads/index.html` and install- ing it prior to moving to the next section.

Eclipse

The first step towards developing any applications is obtaining the integrated development environment (IDE). In the case of Android, the recommended IDE is Eclipse, a multi-language software development environment featuring an extensible plug-in system. It can be used to develop various types of applica- tions, using languages such as Java, Ada, C, C++, COBOL, Python, etc.

For Android development, you should download the Eclipse IDE for Java EE Developers(www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr1). Six editions are available: Windows (32 and 64-bit), Mac OS X (Cocoa 32 and 64), and Linux (32 and 64-bit). Simply select the rel- evant one for your operating system. All the examples in this book were tested using the 32-bit version of Eclipse for Windows.

Once the Eclipse IDE is downloaded, unzip its content (the eclipsefolder) into a folder, say C:\Android\. Figure 1-7 shows the content of the eclipsefolder.

Figure 1-7

Android Sdk

The next important piece of software you need to download is, of course, the Android SDK. The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.

You can download the Android SDK from http://developer.android.com/sdk/ index.html.

Once the SDK is downloaded, unzip its content (the android-sdk-windowsfolder) into the C:\Android\folder, or whatever name you have given to the folder you just created.

Android development tools (Adt)

The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- Create new Android application projects.
- Access the tools for accessing your Android emulators and devices.
- Compile and debug Android applications.
- Export Android applications into Android Packages (APK).
- Create digital certificates for code-signing your APK.

To install the ADT, first launch Eclipse by double-clicking on the eclipse.exefile located in the eclipsefolder.

When Eclipse is first started, you will be prompted for a folder to use as your workspace. In Eclipse, a workspace is a folder where you store all your projects. Take the default suggested and click OK.

Once Eclipse is up and running, select the Help ➪ Install New Software… menu item

In the Install window that appears, type **http://dl-ssl.google.com/android/eclipse** in the text box (see Figure 1-9) and click Add….

After a while, you will see the Developer Tools item appear in the middle of the window (see Figure 1-10). Expand it, and it will reveal its content: Android DDMS, Android Development Tools, and Android Hierarchy Viewer. Check all of them and click Next.

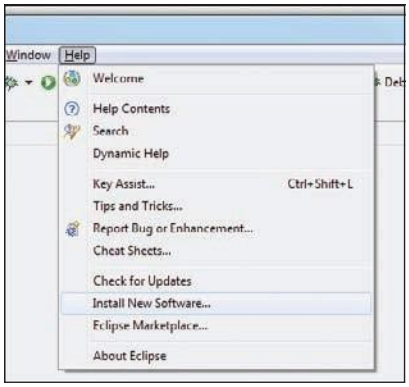
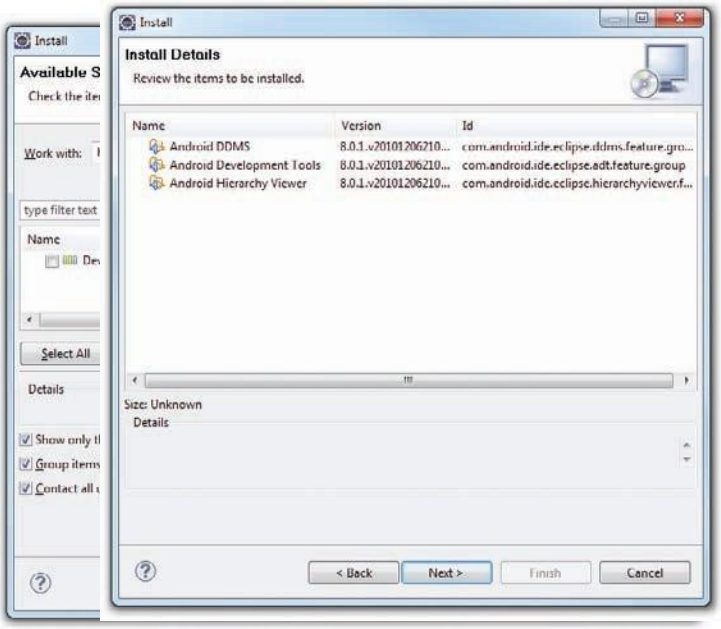


Figure 1-8



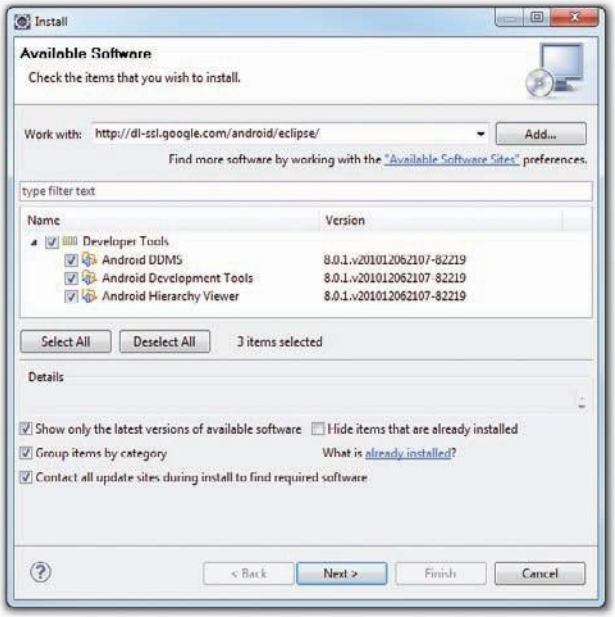


Figure 1-10

When you see the installation details, as shown in Figure 1-11, click Next.

Figure 1-11

You will be asked to review the licenses for the tools. Check the option to accept the license agree- ments (see Figure 1-12). Click Finish to continue.

Figure 1-12

Eclipse will now proceed to download the tools from the Internet and install them (see Figure 1-13). This will take some time, so be patient.

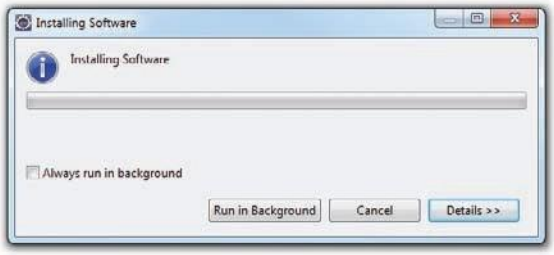


Figure 1-13

Once the ADT is installed, you will be prompted to restart Eclipse. After doing so, go to Window ➪ Preferences (see Figure 1-14).

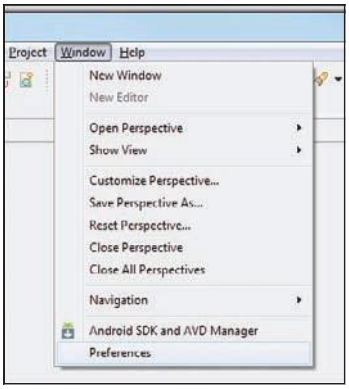


Figure 1-14

In the Preferences window that appears, select Android. You will see an error message saying that the SDK has not been set up (see Figure 1-15). Click OK to dismiss it.

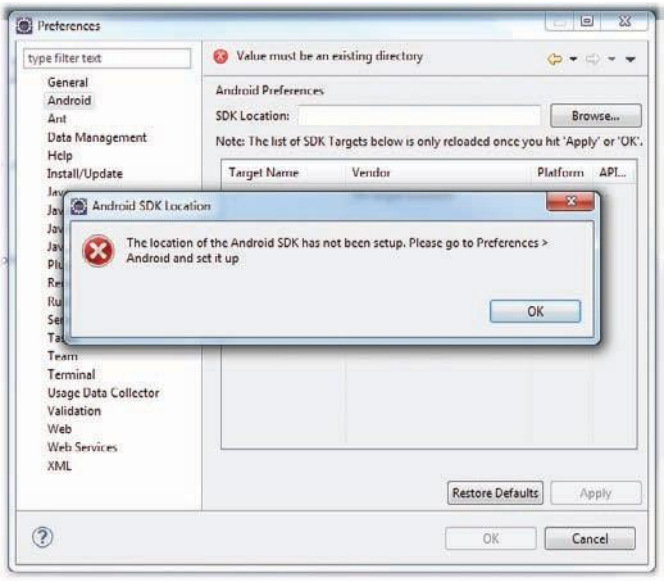


Figure 1-15

Enter the location of the Android SDK folder. In this example, it would be C:\Android\ android-sdk-windows. Click OK.

Anatomy of an Android Application

Now that you have created your first Hello World Android application, it is time to dissect the innards of the Android project and examine all the parts that make everything work.

First, note the various files that make up an Android project in the Package Explorer in Eclipse (see Figure 1-30).

The various folders and their files are as follows:

src— Contains the .javasource files for your project. In this example, there is one file, MainActivity.java. The MainActivity.javafile is the source file for your activity. You will write the code for your application in this file.

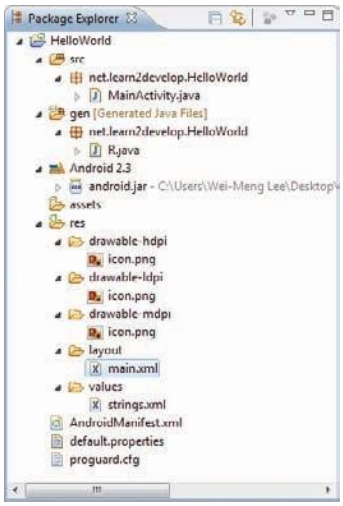
Android 2.3 library — This item contains one file, android.jar, which contains all the class libraries needed for an Android application.

gen— Contains the R.java file, a compiler-generated file that references all the resources found in your project. You should not modify this file.

assets— This folder contains all the assets used by your application, such as HTML, text files, databases, etc.

res — This folder contains all the resources used in your application. It also contains a other subfolders: drawable-<*resolution*>, layout, and values. Chapter 3 talks more about how you can support devices with different screen resolutions and densities.

AndroidManifest.xml — This is the manifest file for your Android application. Here you spec- ify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.).



Unit 2

VUI and Mobile Development

Voice user interaction represents a unique challenge for mobile developers. As human beings, we are social, vocal creatures, which means that using our voice to communicate should be natural. However, voice interaction is still a significant departure from the keyboards and touchscreens that you might be accustomed to using. All the traditional guidelines for designing for iOS or Android, or graphical user interfaces in general, have very little relevance to VUIs.

One of the biggest challenges of VUI design is understanding your users' expectations. Human beings have certain innate assumptions about how communication takes place between two speakers, which can often lead to ambiguity as the computer struggles to compensate for this missing context. For example, a query of "One lump or two?" would be perfectly understandable to a person asking for tea, but could easily trip up a virtual assistant.

Companies and developers, especially those with an app already in place, need to think hard about the ways that they can use VUI to truly enhance the user experience. When it comes to VUI, it's all too easy to get it wrong by trying to hop on the bandwagon of the newest, hottest technology. For example, a company might decide to overhaul its telephone support line by installing a VUI just because it's "easier" for callers, but fail to consider how callers will interact differently with the new VUI than they would with the old push-button keypad interface. Instead, consider how you can use VUI to supplement your existing brand and make it more approachable and helpful.

To address these shortcomings, mobile developers need to be explicit about what the voice recognition system can and cannot do. For example, a sports app with a VUI should inform users on launch that it can retrieve football and basketball scores from the past month, as well as individual players' statistics for the season. Of course, you should also avoid overwhelming your users with too many options.

In addition, the app needs to make it very clear what question it's answering so that users know that their query has been understood. Rather than simply giving a score such as "28–14," the app should provide context such as the names of the teams and the date on which the game was played.

During conversations between two humans, it's easy to tell whether someone is listening to you through body language cues like eye contact. Mobile devices, however, have no such form of expression (at least not yet). Instead, you should use simple visual feedback to inform your users that the VUI is listening to them as they speak.

In order to get the best performance out of your VUI app, it's crucial to integrate it with conversational APIs such as api.ai or Amazon's Alexa Skills Kit API. Services like Dropsource are able to connect to any REST API, making the integration process a snap.

Of course, good VUI design is much harder than it looks. Some of the practices to avoid while building your VUI include:

• Not asking the user a question when the app expects a response.

• Not being clear about the user's options. For example, "Which sport do you want scores for, football or basketball?" is a better question than "Do you want scores for football or basketball?", since the user might simply respond "Yes."

• Giving the user too many choices or being too verbose (for example, "Say 'football' for football. Say 'basketball' for basketball…").

• Confirming the user's query too often. Confirmations should be reserved for important actions such as sending a message or making a purchase.

Hey Siri…Call Mom

One of the most successful examples of VUI design, according to the above principles, is Apple's Siri virtual assistant. When users start Siri, they're presented with a list of ideas such as "Call Hannah" and "Play some rock music." In addition, the system provides both visual feedback (in the form of a glowing white line) and tactile feedback to let users know when it's listening. And the rumor is that millions of Moms across the world have been called a little more often thanks to Siri's help.

Designing great VUIs isn't easy, but there's little doubt that voice interaction is the way of the future. As more and more users use voice-controlled virtual assistants and apps on their mobile devices, getting your VUI right will be ever more important.

Text-to-Speech Techniques

Text-to-speech (TTS) synthesis ultimate goal is to create natural sounding speech from arbitrary text. Moreover, the current trend in TTS research calls for systems that enable production of speech in different speaking styles with different speaker characteristics and even emotions. Speech synthesis generally refers to the artificial generation of human voice – either in the form of speech or in other forms such as a song. The computer system used for speech synthesis is known as a speech synthesizer. There

are several types of speech synthesizers (both hardware based and software based) with different underlying technologies. For example, a TTS (Text to Speech) system converts normal language text into human speech, while there are other systems that can convert phonetic transcriptions into speech. The goal of a text-tospeech system is to automatically produce speech output from new, arbitrary sentences. The text-to-speech synthesis procedure consists of two main phrases. The first is text analysis, in which the input text is transcribed into a phonetic or some other appropriate representation, and the second is the actual generation of speech waveforms, in which the acoustic output is produced from the information obtained from the first phase [2]. A simplified version of the synthesis procedure is presented in figure 1. Figure 1: Phases of Text -to -speech system The quality of a speech synthesizer is measured based on two primary factors – its similarity to normal human speech (naturalness) and its intelligibility (ease of understanding by the listener). Ideally, a speech synthesizer should be both natural and intelligible, and speech synthesis systems always attempt to maximize both characteristics [3]. A typical text to speech system has two parts – a front end and a back end. The front end is responsible for text normalization (the pre-processing part) and text to phoneme conversion. Text normalization or tokenization is the phase where numbers and abbreviations in the raw text are converted into written words. Text to phoneme conversion or grapheme-to-phoneme conversion is the process of assigning phonetic transcriptions to each word and dividing them into prosodic units such as phrases, clauses, and sentences. The output of the front-end system is the symbolic linguistic representation of the text. It is composed of the phonetic transcriptions along with the prosody information. This output is then passed on to the back-end system or the synthesizer, which converts it into sound [1]. This paper is organized as follows. This section gives an introduction about text to speech synthesis. In section II, a review about various methods for text to speech synthesis is explained in detail. The conclusion is given in section III. 2. METHODS OF TEXT TO SPEECH SYNTHESIS Various methods of text to speech synthesis are explained below. 2.1 Formant Synthesis Formant synthesis is based on the source-filter-model of speech. There are two basic structures: parallel and cascade, but for better performance some kind of combination of these is usually used. Formant synthesis also provides infinite number of sounds which makes it more flexible than concatenation methods. In this approach, at least three formants are generally required to produce intelligible speech and to produce high quality speech up to five formants are used. Each formant is modelled with a two-pole resonator which enables both the formant frequency (pole-pair frequency) and its bandwidth to be specified. The input parameters may be the open quotient that means the ratio of the open-glottis time to the total period duration: Voicing fundamental frequency (F0), Voiced excitation open quotient (OQ) , Degree of voicing in excitation (VO), Formant frequencies and amplitudes (F1...F3 and A1...A3), Frequency of an additional low-frequency resonator (FN) and Intensity of low and high-frequency region (ALF, AHF). 2.1.1 A Cascade Formant Synthesizer: It consists of band-pass resonators

connected in series and the output of each formant resonator is applied to the input of the next one. The cascade structure needs only formant frequencies as control information. The main advantage of the promising and this model has been incorporated into several present TTS systems, such as MITalk, DECtalk, Prose-2000, and Klattalk[1]. 2.1.4 PARCAS (Parallel-Cascade) model: In the model, the transfer function of the uniform vocal tract is modelled with two partial transfer functions, each including every second formant of the transfer function. Coefficients k1, k2, and k3 are constant and chosen to balance the formant amplitudes in the neutral vowel to keep the gains of parallel branches constant for all sounds[1]. The PARCAS model uses a total of 16 control parameters: F0 and A0 - fundamental frequency and amplitude of voiced component, Fn and Qn - formant frequencies and Q-values (formant frequency / bandwidth), VL and VH - voiced component amplitude, low and high, FL and FH - unvoiced component amplitude, low and high, QN - Q-value of the nasal formant at 250 Hz[1]. The used excitation signal in formant synthesis consists of some kind of voiced source or white noise. The correct and carefully selected excitation is important especially when good controlling of speech characteristics is wanted. The formant filters represent only the resonances of the vocal tract, so additional provision is needed for the effects of the shape of the glottal waveform and the radiation characteristics of the mouth. Usually the glottal waveform is approximated simply with -12dB/octave filter and radiation characteristics with simple +6dB/octave filter[1]

UI Design Principles

When used together, design principles make the UI designer's job much easier. They remove a lot of the guesswork and make interfaces more predictable and, therefore, easier to use.

Chris Mears from The UX Review gave us this piece of advice on designing for mobile:

"One of the main use cases for mobile is killing time. But that doesn't mean you should waste that of your users. Make sure you understand the main tasks they want to accomplish on your app through research and make those the focus of the interface."

Before we go any further, let's define six of the most common user interface design principles; the structure principle, the simplicity principle, the visibility principle, the feedback principle, the tolerance principle, and finally, the reuse principle.

The Structure Principle

Design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and

separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.

The Simplicity Principle

The design should make simple, common tasks easy, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.

The Visibility Principle

The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with alternatives or confuse them with unnecessary information.

The Feedback Principle

The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.

The Tolerance Principle

The design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions.

The Reuse Principle

The design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

Hannah Alvarez of UserTesting gave this advice for UI beginners in the ebook, Getting Out of the Office: Testing Mobile App Prototypes With User:

"Ever heard the saying, 'Measure twice, cut once?' Well, that idea applies to building apps, too. When you're designing an app, you can verify you're on the right track and avoid doing costly rework by user testing your prototypes. You can iron out the kinks in the design before you've written any code, saving

your team the time and hassle of making changes in development. Plus, you'll already know how users will receive your product before it hits the app store."

Multichannel and Multimodal UIs

Multimodality in mobile computing has become a very active field of research in the past few years. Soon, mobile devices will allow smooth and smart interaction with everyday life's objects, thanks to natural and multimodal interactions. In this context, this chapter introduces some concepts needed to address the topic of pervasive and ubiquitous computing. The multi-modal, multi-channel and multidevice notions are presented and are referenced by the name and partial acroynm "multi-DMC". A multi-DMC referential is explained, in order to understand what kind of notions have to be sustained in such systems. Next we have three case studies that illustrate the issues faced when proposing systems able to support at the same time different modalities including voice or gesture, different devices, like PC or smartphone and different channels such as web or telephone.

Developed at Xerox PARC in 1973 and popularized by the Macintosh, this type of graphical user interfaces is still largely used today, on most computers. However, in recent years, numerous scientific researches focus on post-WIMP interfaces. It is no longer limited to a single way of interacting with a computer system, but considering the different solutions to offer user interfaces as natural as possible. With the introduction of many types of mobile devices, such as cellphones, Personal Digital Assistant (PDA), pocket PC, and the rise of their capabilities (Wifi, GPS, RFID, NFC...) designing and deploying mobile interactive software that optimize the human-computer interaction has become a fundamental challenge. Modern terminals are natively equipped with many input and output resources needed for multimodal interactions, suchas camera, vibration, accelerometer, stylus, etc. However, the main difference between multimedia and multimodal interaction lies in the semantic interpretation and the time management. Multimodality in mobile computing appears as an important trend, but a very few applications allow a real synergic multimodality. Yet, since the famous Bolt's ("put that there") paradigm (Bolt 1980), researchers are studying models, frameworks, infrastructure and multimodal architecture allowing relevant use of the multimodality, especially in mobile situations. Multimodality tries to combine interaction means to enhance the ability of the user interface adaptation to its context of use, without requiring costly redesign and reimplementation. Blending multiple access channels provides new possibilities of interaction to users. The multimodal interface promises to let users choose the way they would naturally interact with it. Users have the possibility to switch between interaction means or to multiple available modes of interaction in parallel. Another field of research in which multimodality is playing an important role is in the Computer Supported Cooperative Work domain (CSCW). CSCW is

commonly seen as the study of how groups of people can work together using technology in a shared time, space hardware and software relationship. “In the context of ubiquitous and mobile computing, this situation of independent and collocated users performing unrelated tasks is however very likely to occur.” (Kray & al. 2004). Even if there is a risk of overlapping categories, design issues are often classified into management, technical and social issues. • Management: mainly deals with registration and later identification of users and devices as they enter and leave the workspace environment. • Technical: issue occurs with the control of specific device features and also the technical management of services offering the possibility to introduce (discover) or remove specific components from an interaction. The design problems is then related to fusion (i.e. combining multiple input types) and fission (i.e. combining multiple output types) mechanisms, synchronization and rules management between heterogeneous devices. • Social issues are more related to social rules and privacy matters. As we know, some devices are inherently unsuitable for supporting privacy, such as microphones, speakers and public displays. The ubiquitous role of the computer makes each day more unsuitable for the screen-keyboardmouse model posed on a corner of a desk. In fact, the large success and rise of the Internet networks have complemented computing communication due to the technical standards used and their adoption of languages such as HTML, WML, or VoiceXML. Yet, we observe a few incompatibilities even though promulgated standards subsist. An example of this incompatiblity can be found in computers with different operating systems that process various types of media (texts, graphics, sounds, and video). Though the information can be easily transmitted through the networks, the formats of the coded data are incompatible. As a direct result the end-user bears addional cost and time lost when trying to obtain or utilize product and service based on their particular platform. This creates the urgent need for easier access to information — whether at the office, home, or on the train, etc. This need is felt all the more with the constant new arrival of soft/hardware materials, the success of the pocket computers and mobile telephones. In fact, the current trends are leaning towards the transformation of end-user’s interface for “anyone and anywhere” (Lopez & Szekely 2001). With the multiplicity of the means of connecting to Internet, it is necessary to conceive generic interfaces and mechanisms of transformation to obtain concrete interfaces for each platform.

Storing and Retrieving Data

Data and file storage overview

Android provides several options for you to save your app data. The solution you choose depends on your specific needs, such as how much space your data requires, what kind of data you need to store, and whether the data should be private to your app or accessible to other apps and the user.

This page introduces the different data storage options available on Android:

- **Internal file storage**: Store app-private files on the device file system.
- **External file storage**: Store files on the shared external file system. This is usually for shared user files, such as photos.
- **Shared preferences:** Store private primitive data in key-value pairs.
- **Databases**: Store structured data in a private database.

Except for some types of files on external storage, all these options are intended for app-private data—the data is not naturally accessible to other apps. If you want to share files with other apps, you should use the FileProvider API. To learn more, read Sharing Files.

If you want to expose your app's data to other apps, you can use a ContentProvider. Content providers give you full control of what read/write access is available to other apps, regardless of the storage medium you've chosen for the data (though it's usually a database). For more information, read Content Providers.

Internal storage

By default, files saved to the internal storage are private to your app, and other apps cannot access them (nor can the user, unless they have root access). This makes internal storage a good place for internal app data that the user doesn't need to directly access. The system provides a private directory on the file system for each app where you can organize any files your app needs.

When the user uninstalls your app, the files saved on the internal storage are removed. Because of this behavior, you should not use internal storage to save anything the user expects to persist independently of your app. For example, if your app allows users to capture photos, the user would expect that they can access those photos even after they uninstall your app. So you should instead use the MediaStore API to save those types of files to the appropriate media collection.

Internal cache files

If you'd like to keep some data temporarily, rather than store it persistently, you should use the special cache directory to save the data. Each app has a private cache directory specifically for these kinds of files. When the device is low on internal storage space, Android may delete these cache files to recover space. However, you should not rely on the system to clean up these files for you. You should always

maintain the cache files yourself and stay within a reasonable limit of space consumed, such as 1MB. When the user uninstalls your app, these files are removed.

For more information, see how to write a cache file.

External storage

Every Android device supports a shared "external storage" space that you can use to save files. This space is called external because it's not guaranteed to be accessible—it is a storage space that users can mount to a computer as an external storage device, and it might even be physically removable (such as an SD card). Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.

So before you attempt to access a file in external storage in your app, you should check for the availability of the external storage directories as well as the files you are trying to access.

Most often, you should use external storage for user data that should be accessible to other apps and saved even if the user uninstalls your app, such as captured photos or downloaded files. The system provides standard public directories for these kinds of files, so the user has one location for all their photos, ringtones, music, and such.

You can also save files to the external storage in an app-specific directory that the system deletes when the user uninstalls your app. This might be a useful alternative to internal storage if you need more space, but the files here aren't guaranteed to be accessible because the user might remove the storage SD card. And the files are still world readable; they're just saved to a location that's not shared with other apps.

For more information, read how to save a file on external storage.

Shared preferences

If you don't need to store a lot of data and it doesn't require structure, you should use SharedPreferences. TheSharedPreferences APIs allow you to read and write persistent key-value pairs of primitive data types: booleans, floats, ints, longs, and strings.

The key-value pairs are written to XML files that persist across user sessions, even if your app is killed. You can manually specify a name for the file or use per-activity files to save your data.

The API name "shared preferences" is a bit misleading because the API is not strictly for saving "user preferences," such as what ringtone a user has chosen. You can use SharedPreferences to save any kind of simple data, such as the user's high score. However, if you *do* want to save user preferences for your app, then you should read how to create a settings UI, which uses the AndroidX Preference Library to build a settings screen and automatically persist the user's settings.

To learn how to store any type of key-value data, read Save Key-Value Data with SharedPreferences.

Databases

Android provides full support for SQLite databases. Any database you create is accessible only by your app. However, instead of using SQLite APIs directly, we recommend that you create and interact with your databases with the Room persistence library.

The Room library provides an object-mapping abstraction layer that allows fluent database access while harnessing the full power of SQLite.

Although you can still save data directly with SQLite, the SQLite APIs are fairly low-level and require a great deal of time and effort to use. For example:

There is no compile-time verification of raw SQL queries.

As your schema changes, you need to update the affected SQL queries manually. This process can be time consuming and error prone.

You need to write lots of boilerplate code to convert between SQL queries and Java data objects.

The Room persistence library takes care of these concerns for you while providing an abstraction layer over SQLite.

Synchronization and Replication of Mobile Data

Synchronization and Replication The following scenario will show, how mobile computing, synchronization and replication are linked to each other. Imagine a sales person, who is travelling from customer to customer and is collecting orders from them. The sales person is collecting all data on a laptop where he is also able to access data that indicates how long an order will take to be delivered and provides the possibility to calculate customer and order specific conditions. When the sales person is at a customer site, most of the time communication between the laptop and the central sales person’s corporate database, where all required data should be entered or accessed, is not possible. This is where

data replication comes in. As already stated, the needed data has to be copied onto the sales person's device in order to provide the required functionality. On a daily basis the sales person needs to send the new orders to the central system by any means of communication link or medium. When reconnecting to the corporate database or application the two data sets need to be synchronized. That way, mobile devices become mobile databases as [8] understands them. Mobile databases appear and disappear over time in a network of connected and stationary databases as well as other mobile databases. 1.3 Topics covered and their dependencies In the following, first data synchronization techniques will be covered, before data replication techniques within databases will be described. This order is applied, since the discussion of data replication techniques will refer to concepts of data synchronization. Therefore it is necessary to gain some insight into data synchronization, before being able to fully understand all aspects of data replication. 2 Synchronization Techniques Data items that are located within a database, a file system or in memory as a variable within a program, can be modified by processes. If only one process at a time is trying to modify a data item, there is no need for synchronization. If more than one process at a time tries to modify a data item, it is necessary to ensure that no two processes access the data item at the same time. Accesses to the data item are put into a sequential order, which is called process synchronization.[5] In distributed systems, identical data items may be managed at physically distributed places or machines. These data items can be manipulated independently from each other, resulting in different versions These logically connected data items need to be merged back into a consistent state. This process is called data synchronization. Data synchronization is at the centre of the following discussions and is meant by the term synchronization.[5] 2.1 Aspects Synchronisation can take place between two or more systems, namely in a 1:1 or 1:n relation. That means that there is always one active system that starts and controls the synchronization process. The active system can be any of the participating systems. Synchronization can be triggered manually or by certain events, which requires? an event notification mechanism. A system can explicitly inform another system that it wants to synchronize with it, which is called push synchronization, or pull synchronization where a system is being asked to engage in a synchronization process.[5] 2 Another aspect is which amount of data is being exchanged during the synchronization process. Incremental synchronization just exchanges the data items that have been modified on all systems since the last synchronization happened. In contrast to this method, full synchronization exchanges always all data items from one system to the other. Incremental synchronization can be achieved by storing timestamps of the last modification and comparing them between the participating systems. This introduces the need to synchronize the local time of a replica with the other replicas in the system. An alternative to that approach is to compare and exchange data structures within data items which is mostly done when synchronizing files. It might also be necessary to transform data items between not structural equal data structures that are equivalent.[5] When data items are exchanged, it is

possible that both data items have been changed within their respective location and cannot easily be merged. A conflict resolution mechanism has to be applied to put the data items back into a consistent state.[5] One aspect that also has to be mentioned is the mapping between distributed data items and their dependent or referenced items. Within relational databases this can mostly be done by foreign and primary keys if the synchronization is done within homogeneous databases, which is also known as global unique identifier approach, that can be generalized to all kinds of data items.[5] 2.2 Layers of synchronization and solutions As already described data items can be managed in different storage systems. The file system is one possible storage solution. Synchronization in this layer can be achieved in many ways. Tools and protocols that exist can be grouped into categories. Two categories are: - file system tools like the UNIX tool rsync, that is especially designed for an efficient synchronization of files, or the coda file system, that is a distributed file system - version control systems like CVS or the WEBDav protocol, that enables file transfer via http and incorporates locking and versioning Synchronization mechanisms within databases are better known as replication mechanisms and will be described in the replication techniques section. Special constraints are imposed on mobile clients, as described in the introduction. These have to be considered in the case of synchronization. A couple of solutions exist, that are mostly concerned with synchronization in the application layer. Microsoft is offering a product called Active Sync, that can be used to synchronize between PC applications and Windows CE based mobile clients.[5] 2.3 Synchronization Protocol In order to actually implement synchronization between two or more systems it is necessary to have a well-defined specification of the steps and the overall workflow of this process. The afore mentioned synchronization solutions have one big disadvantage. They are all using proprietary synchronization protocols that increase the efforts for server or PC application providers and mobile application providers to make their respective products interoperable.[5] 3 2.3.1 Standard SyncML Based on the open extensible mark-up language (XML) standard, the SyncML specification forms the basis for specifying an open data synchronization protocol. Two parts of the specification can be distinguished, the representation protocol and the synchronization protocol. SyncML uses the client server communication model, where conceptually two components on both sides that are called SyncML Adapter are exchanging XML formatted messages, that are defined by the representation protocol. The Adapter also implements the protocol workflow via a variety of transport protocols. The SyncML Adapter provides a procedural interface for server and client side components, as there are the Sync Engine on the server side and client applications on the client side. The sync Engine is responsible for modifying the data repositories as well as providing conflict resolution services. The server side application is interacting with the Sync Engine. On the client side there is no concept like the Sync Engine. Instead, the client side application is directly interacting with the SyncML Adapter. The concrete synchronization mechanisms are not specified by the SyncML specification. For an in-depth

description of the SyncML XML format please consider [11]. Only the main elements of this format will be described here. A SyncML Message element is composed of one SyncHdr and one SyncBody element. The former hold information about the source and the target of the synchronization process, therefore specifying the direction of the process, as well as a unique message identifier that is being used throughout the whole communication interaction. Within the SyncBody element the payload of the message, the actual data items are encoded, together with a SyncML Command element. An alert Command is used to initialise a synchronization process. Commands can also have one of the values add, copy, delete, put or replace. These Commands are defining modification operations. Values like read and search are specifying query operations. In responses, the values status or results are used. Sequence and atomic values specify how subcommands should be executed, here the order and the transactional behaviour. A possibility to map data items from the server to the client and vice versa is provided with the SyncML XML format. What is more interesting in the context of this discussion, are the synchronization models that SyncML describes and that are defined by the synchronization protocol. The following scenarios are described within the specification. - Two-way Sync: This model allows both sides, server and client to exchange modifications. The client sends its modifications first. - Slow Sync: The server checks the complete data repository of the client for modifications. 4 - One-way Sync: Only one side sends its modifications, which results in two possible scenarios, from the client side only, or from server side only. - Refresh Sync: One of the both sides sends its complete data repository to the respective other side for actualisation. Again, this gives two possible scenarios where the only the client sends its data repository or the server. - Server-alerted Sync: the server informs the client, that it should issue synchronization. 3 Replication Techniques Replication is used to achieve better availability and performance by using multiple copies of a server system, where availability is the main goal. If one of the copies, or replicas is not running, the service which is provided by a set of replicas can still be working. It is also of great use if the communication link between two systems is only intermittently available, which is most often the case in mobile applications. [10] In the distributed systems community, software based replication, which is being described here, is seen as a cost effective way to increase availability. In the database community, however, replication is used for both performance and fault-tolerant purposes. [7] Replication is both used in the database community as well as the distributed system community. In [13] an abstract functional model is introduced that is used to describe existing solutions in both fields. The abstract model is also used to compare the replication protocols. The classification that [13] introduces is semantically equivalent to the one that will be used in the following review of replication techniques. 3.1 Replication in Databases Server systems usually depend on a resource, mostly on a database system. Replicating the server without the database improves availability but not performance and leaves the single point of failure problem. Performance is not improved since all kinds of access, query and update

operations are done on one single database resulting in query-update problems. Therefore in addition to the replication of the server system, the resource needs to be replicated, to gain substantial advantages over systems that rely on one resource.[10] According to [13] databases are collections of data items that are controlled by a database management system. Replicated databases are therefore a collection or set of databases that store copies of identical data items. A data item can be referred to as the logical data item or by it's physical location. 5 There are many technical implications that come with resource replication, or data replication as it will also be referred to. The main requirement in data replication is, that replicas should functionally behave like servers, that are not replicated. This strict requirement can not always be fulfilled since the mechanism to solve this problem - synchronous replication or eager replication, where every transaction updates all replicas - produces heavy distributed transaction load. Therefore synchronous replication is considered to be very hard to achieve in an efficient way [7]. Fortunately there is another way for propagating updates to all replicas. This mechanism is called asynchronous replication, since replicas are updated independently from each other and updates are sent from one server to all other replicas. This solution can lead to situations, where transactions are updating the same data items and may result in conflicts, which can be solved by ensuring the right time dependent order of the original transactions. That way all replicas take in the intermediate states the whole system got through and, at some point in time, get into a common, identical state. Keeping the order of updates requires synchronization, which imposes performance implications that require sophisticated synchronization techniques, resulting in a high degree of complexity [10]. Replication is therefore imposing a constant trade-off between consistency and efficiency [7]. Since we are dealing with a distributed system of replicas, situations have to be handled in which servers go down or network connections fail, introducing network partition problems. How these major problems are solved, introduces a classification schema that will be described in the next section. If not explicitly mentioned the subjects of the discussion are relational databases. Therefore basic knowledge in the field of relational database technology, the transaction concept and management are required in order to understand the following discussion. It should be mentioned that replication, especially in databases with high transaction rates is a complex topic. In the simple case of only a few replica nodes, that apply the update anywhere, anytime, anyway replication scheme, combined with low transaction rates, this simple replication scheme works well with respect to deadlocks and reconciliation, that will be described later. When the system scales up, problem complexity grows drastically at cubic rates leading to inconsistent replicas and soon to system delusion, where there is no way to repair the inconsistencies [4].

Working with a Content Provider

Content provider basics

A content provider manages access to a central repository of data. A provider is part of an Android application, which often provides its own UI for working with the data. However, content providers are primarily intended to be used by other applications, which access the provider using a provider client object. Together, providers and provider clients offer a consistent, standard interface to data that also handles inter-process communication and secure data access.

Typically you work with content providers in one of two scenarios; you may want to implement code to access an existing content provider in another application, or you may want to create a new content provider in your application to share data with other applications. This topic covers the basics of working with existing content providers. To learn more about implementing content providers in your own applications, see Creating a content provider.

This topic describes the following:

How content providers work.

The API you use to retrieve data from a content provider.

The API you use to insert, update, or delete data in a content provider.

Other API features that facilitate working with providers.

Overview

A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database. A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for an instance.

A content provider coordinates access to the data storage layer in your application for a number of different APIs and components as illustrated in figure 1, these include:

Sharing access to your application data with other applications

Sending data to a widget

Returning custom search suggestions for your application through the search framework using Search Recent Suggestions Provider

Synchronizing application data with your server using an implementation of Abstract Threaded Sync Adapter

Loading data in your UI using a Cursor Loader

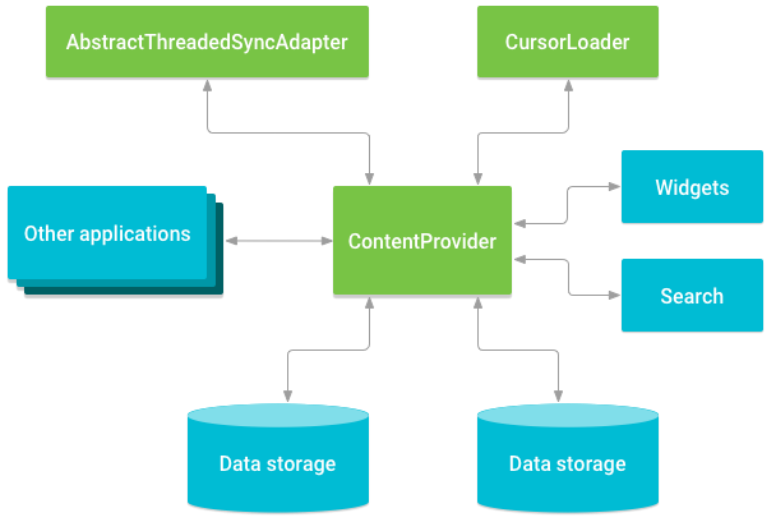


Figure 1. Relationship between content provider and other components.

Accessing a provider

When you want to access data in a content provider, you use the ContentResolver object in your application'sContext to communicate with the provider as a client. The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider. The provider object receives data requests from clients, performs the requested action, and returns the results. This object has methods that call identically-named methods in the provider object, an instance of one of the concrete subclasses of ContentProvider. The ContentResolver methods provide the basic "CRUD" (create, retrieve, update, and delete) functions of persistent storage.

A common pattern for accessing a ContentProvider from your UI uses a CursorLoader to run an asynchronous query in the background. The Activity or Fragment in your UI call a CursorLoader to the query, which in turn gets the ContentProvider using the ContentResolver. This allows the UI to continue to be available to the user while the query is running. This pattern involves the interaction of a number of different objects, as well as the underlying storage mechanism, as illustrated in figure 2.

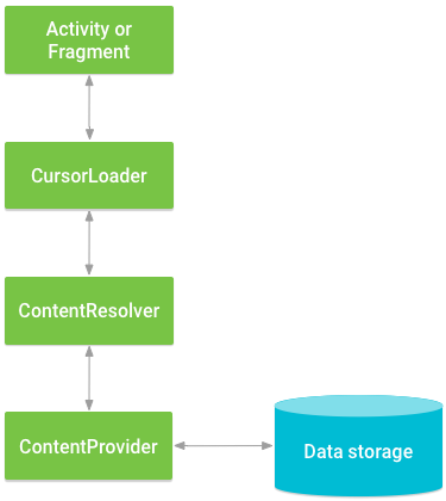


Figure 2. Interaction between ContentProvider, other classes, and storage

Unit 3

NETWORK AND THE WEB:STATE MACHINE

State machine models can be learned from a set of existing traces (passive learning) or a set of traces that are generated while learning (active learning). The traces are generated by interacting with an application and observing input and output combinations. The drawback of passive learning is that the model is incomplete in the variety of existing traces, i.e., when the set of traces does not describe particular application behavior, the inferred model does not specify this behavior either. Active learning overcomes the shortcoming by querying for the information it needs to know. Software systems, and therefore also Android applications, can function as a data source to generate the required traces because the applications can respond interactively. The drawback of applying active learning to software systems is that interacting with an application consumes time, as each input needs to be simulated and each output is required to be observed. To improve the learning process, different active learning algorithms have been developed, such as the L* and TTT algorithm. Active state machine learning is often implemented according to the Minimally Adequate Teacher (MAT) framework as first proposed by Angluin [9], which is composed of a learner and a teacher. The goal of the learner is to infer the state machine model of a system under test (SUT), by posing membership queries and equivalence queries to the teacher. Membership queries ask whether the SUT recognizes a specific behavior input. The combination of a query and answer form a trace, and a hypothesized model can be constructed after a sufficient amount of traces are generated. The model learner poses an equivalence query to the teacher for the built hypothesis. The query determines if the model correctly describes the SUT's input/output behavior. Learning halts when the hypothesized model is equivalent. If the model incorrectly describes the SUT's behavior, the teacher provides a trace which distinguishes the model and the SUT. The trace is also called a counterexample because it invalidates the hypothesized model. The learner utilizes the counterexample to refine the hypothesis. The process repeats itself until an equivalence query yields success. Figure 1.1 visually depicts the discussed methods of the MAT framework.

Correct Communications Model:

The term "communication" has been derived from the Latin "communis," that means "common"1 . Thus "to communicate" means "to make common" or "to make known", "to share" and includes verbal, non-verbal and electronic means of human interaction. Scholars who study communication analyze the development of communication skills in humans and theorize about how communication can be made more effective. It is the meaningful exchange of information between two or a group of people. Communicative competence designates the capability to install intersubjective interactions, which means

that communication is an inherent social interaction2 . One definition of communication is "any act by which one person gives to or receives from another person information about that person's needs, desires, perceptions, knowledge, or affective states. Communication may be intentional or unintentional, may involve conventional or unconventional signals, may take linguistic or non-linguistic forms, and may occur through spoken or other modes."3 This act of making common and known is carried out through exchange of thoughts, ideas or the like. The exchange of thoughts and ideas can be had by gestures, signs, signals, speech or writing. People are said to be in communication when they discuss some matter, or when they talk on telephone, or when they exchange information through letters. Basically, communication is sharing information, whether in writing or orally

Humans convey information through a variety of methods: speaking, telephones, email, blogs, TV, art, hand gestures, facial expressions, body language and even social contexts5 . Communication can occur instantaneously in closed, intimate settings or over great periods of time in large public forums, like the Internet. However, all forms of communication require the same basic elements: a speaker or sender of information, a message, and an audience or recipient. The sender and recipient must also share a common language or means of understanding each other for communication to be successful. As such, a study of communication often examines the development and structure of language, including the mathematical languages used in computer programming. The act of communicating draws on several interpersonal and intrapersonal skills. These include speaking, listening, observing, questioning, processing, analyzing and evaluating. Recipients of a message must be able to identify the sender's intent, take into account the message's context, resolve any misunderstandings, accurately decode the information and decide how to act on it. Such skills are essential to learning, forming healthy relationships, creating a sense of community and achieving success in the workplace. As a field of study, communication spans a broad, rich array of subjects, including sociology, psychology, philosophy, political science, linguistics, history, literature, criticism and rhetoric. Although much of the field's subject matter is theoretical in nature, communication studies have proven applicable to business, film, theatre, composition, advertising, education, foreign policy and computer science. In today's globalized, media-driven world, communication studies have become more relevant and exciting than ever. Web developers seek new, inventive ways to draw Internet users to their websites. Public policy writers debate society's most pressing issues. Through linguistics, computer scientists are developing programming languages that may someday allow humans to interact directly with computers. Students who earn degrees in communication often hold highly influential positions as journalists, editors, university professors, public relations officers, marketing consultants, speech writers, filmmakers, motivational speakers and political campaign managers. To communicate is to shape the world. Communication requires a sender, a message, and a

recipient, although the receiver doesn't have to be present or aware of the sender's intent to communicate at the time of communication; thus communication can occur across vast distances in time and space. Communication requires that the communicating parties share an area of communicative commonality . The communication process is complete once the receiver has understood the message of the sender . Language issues and Cultural Differences: the receiver(s) may not (fully) understand the language used by the transmitter. This may occur if the transmitter's language is foreign to the receiver. There may also be language problems (that the communication process) if the message contains technical information and the receiver's is not familiar with the technical terms used. Cultural differences created by an individual's background and experience affect their perception of the world. Such cultural differences may affect the interpretation (decoding) of the message sent . Environmental issues: If the environment that the transmitter or receiver are in, is noisy and full of sound, the sounds may prevent the message being fully understood. Background noise is often created by colleagues or machinery . Channel issues: If the channel used to transfer the information is poor it may prevent all or some of the information being transferred. Examples include a faulty fax machine, a crackling phone, handwriting that cannot be read or in the case of oral messages incorrect facial gestures. Receivers Attitude and behaviour: If the receiver(s) is not interested in the message (or unable to give their full attention to decoding) this may reduce the amount of information received or the accuracy of the information transmitted to them. Similarly the receiver(s) may misinterpret the message by "jumping to conclusions" or reading the message in a manner that suits their own interests/objectives and distort the true meaning of the message10. Transmission journey: i.e. steps in the message. If the message is complicated or there are lots of steps taken to transfer the message it may affect the accuracy or interpretation11. Comparing with the leaky bucket if the leaky bucket has to carry water over a longer distance more water will probably lost than if the journey was shorter.

Android Networking and Web

Android lets your application connect to the internet or any other local network and allows you to perform network operations.

A device can have various types of network connections. This chapter focuses on using either a Wi-Fi or a mobile network connection.

Checking Network Connection

Before you perform any network operations, you must first check that are you connected to that network or internet e.t.c. For this android provides **ConnectivityManager** class. You need to instantiate an object of this class by calling **getSystemService()** method

this connected states, there are other states a network can achieve. They are listed below −

Sr.No	State
1	Connecting
2	Disconnected
3	Disconnecting
4	Suspended
5	Unknown

Performing Network Operations

After checking that you are connected to the internet, you can perform any network operation. Here we are fetching the html of a website from a url.

Android provides **HttpURLConnection** and **URL** class to handle these operations. You need to instantiate an object of URL class by providing the link of website

this connect method, there are other methods available in HttpURLConnection class. They are listed below −

S. No	Method & description
1	**disconnect()** This method releases this connection so that its resources may be either reused or closed
2	**getRequestMethod()** This method returns the request method which will be used to make the request to the remote HTTP server
3	**getResponseCode()** This method returns response code returned by the remote HTTP server
4	**setRequestMethod(String method)** This method Sets the request command which will be sent to the remote HTTP server

5	**usingProxy()** This method returns whether this connection uses a proxy server or not

It creates a basic application that allows you to download HTML from a given web page.

To experiment with this example, you need to run this on an actual device on which wifi internet is connected .

Steps	Description
1	You will use Android studio IDE to create an Android application under a package com.tutorialspoint.myapplication.
2	Modify src/MainActivity.java file to add Activity code.
4	Modify layout XML file res/layout/activity_main.xml add any GUI component if required.
6	Modify AndroidManifest.xml to add necessary permissions.
7	Run the application and choose a running android device and install the application on it and verify the results.

Wireless Connectivity and Mobile Apps technology users and consumers are interacting with more devices than ever. Using a smart phone, you can sync a smart watch, make point-of-sale purchases, control your home's alarm system, or even fly a drone. Depending on the application, you could be using Bluetooth, Wi-Fi, NFC, or mobile data networks to connect to these devices

As a developer, you may want to create applications for a specific device, or perhaps even develop your own connected device that can be controlled with a smart phone. Before you break out your IDE and start tinkering with hardware, let’s take a quick tour of Bluetooth, Wi-Fi, NFC and mobile data to learn how these technologies work, and which applications each one is best suited for.

NFC

Near Field Communication (NFC) describes a set of protocols that enable two devices in close proximity (up to 10 cm apart)to communicate with each other. Like many other proximity card technologies, it employs electromagnetic induction when two devices exchange information in the 13.56MHz range. One device can be an unpowered NFC tag in a label or poster, which means that

the reader would simultaneously initiate data reads and provide power to the tag. Data rates typically range between 106-424 kbit/s. NFC supports three different modes:

Card Emulation allows a device to act like a smart card. (This is, in fact, the mechanism used for Apple Pay and Android Pay.)

Reader/Writer Mode enables a device to read and write data to NFC tags.

Peer-to-Peer Mode facilitates information exchange between devices in ad-hoc fashion.

NFC has a wide range of applications, but of late, mobile payment solutions have been the main drivers for adoption on Android and iOS devices. Generally speaking, NFC is best suited for applications that require very close proximity between a phone and tag or device and secure data exchange.

At present, only a limited set of devices support NFC. Compatible iOS devices are iPhone 6/6S/6+/6S+ and Apple Watch models. Apple has not yet opened development access to these sensors via API, so they can only be used for Apple Pay. Android, on the other hand, does offer a development API, but, again, not many Android phones on the market include NFC support, especially not at lower price points.

Bluetooth

Bluetooth was conceived as a protocol for exchanging data over short distances. It uses the same frequency range as 2.4 GHz Wi-Fi but with 79 channels (for versions below4.0), and 40 channels for 4.0 and above. However, unlike most Wi-Fi implementations, it uses a technology called frequency-hopping spread spectrum which regularly changes the channel in order to avoid congestion caused by competing networks. Data throughput rates are also a lot higher, ranging from 1 Mbit/s for Bluetooth 1.2 to 24 Mbit/s for 4.0 and above. All iOS and most Android devices support Bluetooth. Some applications of standard Bluetooth include wireless headsets and data tethering.

Bluetooth LE

Bluetooth LE (Low Energy) is a subset of the Bluetooth 4.0 specification designed for low energy applications. It has a maximum data throughput rate of 0.27 Mbit/s and a power consumption that can be as low as 0.01 W, up to a maximum of 0.5 W (about half of what a standard Bluetooth device consumes.) To put this in perspective: Most smartphone batteries have capacities that range from 1400-3000 mAh (milliampere hours) but struggle to get more than a day or two of battery life, yet it is not uncommon for a Bluetooth LE-based fitness tracking device with a 100mAh battery to last up to a week or longer. Applications that use Bluetooth LE include fitness trackers, smartwatches, and many medical devices.

The LE stack also has a number of predefined generic attribute (GATT) profiles that specify communication mechanisms for various types of devices such as heart rate monitors, blood pressure

gauges, or proximity sensors. This means that if a company implements a device to follow the specification, it wouldn't require proprietary software to retrieve data from it.

While not quite as ubiquitous as standard Bluetooth support, device support for Bluetooth LE is, nonetheless, very good. All iPhone models upward of 4S support it,along with iPad 3rd generation or newer. On the Android side, things are a bit more complicated, since the support implementation that first appeared in Android 4.3 was somewhat buggy and limited to central mode (meaning that interaction is restricted to other peripheral Bluetooth LE devices) but the phone cannot act like a peripheral. Android4.4 provided more stable functionality, and Android 5.0 added support for peripheral advertising. By default, any device that supports Bluetooth 4.0 also supports Bluetooth LE, including older flagship phones such as the Galaxy Nexus and other more recent releases in the Nexus range, as well as tablets like the Nexus 7 2013 and newer models. These days, even inexpensive Android phones such as the LG Lucky, which retails for approximate $10, also offers Bluetooth 4.0 support.

Wi-Fi and Cellular Technology

Most people have used, and are familiar, with these technologies. Cellular networks use radio signals served by fixed-location transceivers (cell base towers) to transmit voice and data and connect mobile devices to the publicInternet at speeds that range from the slow 2G to lightning-fast LTE. A cellular connection is, generally speaking, more expensive from a cost per Mb perspective, but is more readily available when you're on the move. All Android and iOS phones and select models of iPad and Android tablets support some form of cellular network connectivity.

Wi-Fi is defined as a wireless local area network connection,which may, or may not, connect to resources like the Internet. These networks usually offer faster transmission than cellular connections and are much less expensive per Mb, or even Free. Virtually all iOS and Android devices support Wi-Fi.

For applications that consume lots of data or need to communicate with devices local to a network, Wi-Fi is the recommended solution. As such, it is the typical mechanism used for home automation and video casting.

So, What Should I Use?

NFC has compelling use cases, but unless you're creating an application that needs to scan NFC tags, or allows you to control the hardware and have rigorous security requirements, you may not want to use it in your project. It is obviously the perfect technology for mobile payment applications, but until Apple opens their APIs to application developers, you'll only reach a limited audience.

Bluetooth is a great solution when you need to communicate with a device in close proximity to a phone. A smart device, such as a printer, might use Bluetooth to configure its Wi-Fi settings in order

to connect to a network or wireless headphones. When you have a small device with a limited battery capacity that doesn't need to transfer lots of data, such as a wearable fitness monitor or smartwatch, Bluetooth LE would be ideal.

For applications that consume lots of data or need to communicate with devices local to a network, Wi-Fi is the recommended solution. As such, it is the typical mechanism used for home automation and video casting. Some applications may be configured to only download data when connected to a Wi-Fi network, but when an application doesn't consume a lot of data or, alternatively, needs to maintain a constant network connection, it's usually best to let the operating system decide whether to use Wi-Fi when available, or to switch to cellular coverage.

User interface (UI) performance testing ensures that your app not only meets its functional requirements, but that user interactions with your app are buttery smooth, running at a consistent 60 frames per second without any dropped or delayed frames, or as we like to call it, *jank*. This document explains tools available to measure UI performance, and lays out an approach to integrate UI performance measurements into your testing practices.

Measure UI performance

In order to improve performance you first need the ability to measure the performance of your system, and then diagnose and identify problems that may arrive from various parts of your pipeline.

dumpsys is an Android tool that runs on the device and dumps interesting information about the status of system services. Passing the gfx info command to dumpsys provides an output in logcat with performance information relating to frames of animation that are occurring during the recording phase.

```
> adb shell dumpsys gfx info <PACKAGE_NAME>
```

This command can produce multiple different variants of frame timing data.

Aggregate frame stats

With Android 6.0 (API level 23) the command prints out aggregated analysis of frame data to logcat, collected across the entire lifetime of the process. For example:

These high level statistics convey at a high level the rendering performance of the app, as well as its stability across many frames.

Precise frame timing info

With Android 6.0 comes a new command for gfxinfo, and that's framestats which provides extremely detailed frame timing information from recent frames, so that you can track down and debug problems more accurately.

```
>adb shell dumpsys gfxinfo <PACKAGE_NAME> framestats
```

This command prints out frame timing information, with nanosecond timestamps, from the last 120 frames produced by the app. Below is example raw output from adb dumpsys gfxinfo <PACKAGE_NAME> framestats:

Each line of this output represents a frame produced by the app. Each line has a fixed number of columns describing time spent in each stage of the frame-producing pipeline. The next section describes this format in detail, including what each column represents.

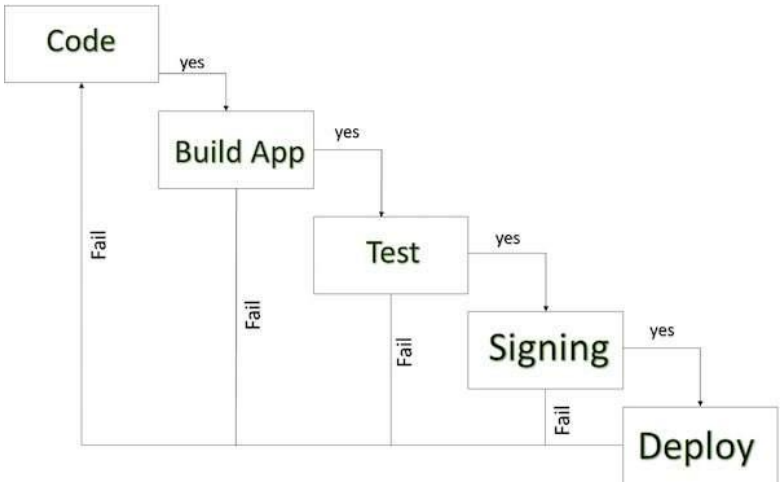
Framestats data format

Since the block of data is output in CSV format, it's very straightforward to paste it to your spreadsheet tool of choice, or collect and parse with a script. The following table explains the format of the output data columns. All timestamps are in nanoseconds.

UNIT-IV

PUTTING IT ALL TOGETHER AND MULTIMEDIA

Android application publishing is a process that makes your Android applications available to users. Infect, publishing is the last phase of the Android application development process.



Android development life cycle

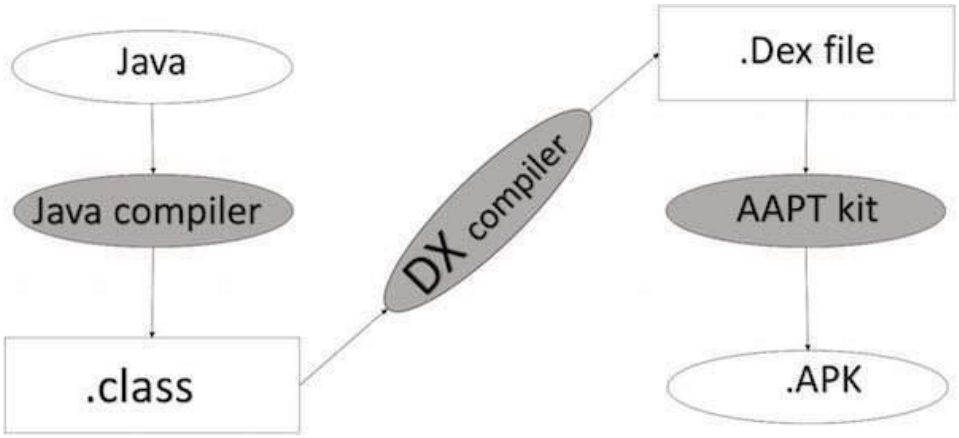
Once you developed and fully tested your Android Application, you can start selling or distributing free using Google Play (A famous Android marketplace). You can also release your applications by sending them directly to users or by letting users download them from your own website.

You can check a detailed publishing process at Android official website, but this tutorial will take you through simple steps to launch your application on Google Play. Here is a simplified check list which will help you in launching your Android application −

Step	Activity
1	**Regression Testing** Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2	**Application Rating** When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3	**Targeted Regions** Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.

4	**Application Size** Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.
5	**SDK and Screen Compatibility** It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.
6	**Application Pricing** Deciding whether you app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7	**Promotional Content** It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8	**Build and Upload release-ready APK** The release-ready APK is what you you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9	**Finalize Application Detail** Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.

Export Android Application Process



Apk development process

Before exporting the apps, you must some of tools

- **Dx tools**(Dalvik executable tools): It going to convert **.class file** to **.dex file**. it has useful for memory optimization and reduce the boot-up speed time

- **AAPT**(Android assistance packaging tool):it has useful to convert **.Dex file** to**.Apk**
- **APK**(Android packaging kit): The final stage of deployment process is called as .apk.

You will need to export your application as an APK (Android Package) file before you upload it Google Play marketplace.

To export an application, just open that application project in Android studio and select **Build → Generate Signed APK** from your Android studio and follow the simple steps to export your application

Multimedia: Mobile Agents and Peer-to-Peer Architecture, Android Multimedia

In recent times the development of the internet has shifted from traditional client-server model to peer-to-peer networks, also known as p2p-networks. In this model, computers (peers) act as both server and client, thus being able to communicate directly without central servers. This kind of model has several advantages, some of which are discussed later in this paper. Mobile agents are agents that have the capability of migrating from one host to another. When migrating, the state of the program is saved, transported to next host and continued as normal process. Mobile agents can also produce other mobile agents by duplicating or cloning themselves or even by creating totally new agents. Both of these technologies have been researched a lot, but the combination of mobile agents and p2p-networks have just lately taken under research. With these technologies together, many of the applications using the traditional client server model could be modified to work a lot more efficiently. The structure of the paper is as follows: First, a brief introduction of the technologies is presented in chapter 2; p2pnetworks and mobile agents presents some possible usage scenarios for mobile agents and p2p-networks and finally some problems and difficulties are presented.

Peer-to-peer networks Peer-to-peer networks are one of the trends in the field of internetworking. In p2p-networks, the hosts, or peers, act as client and server. lists some characteristics for p2p-system. These are

• Decentralization
• Scalability
• Anonymity
• Self-Organization
• Cost of Ownership
• Ad-Hoc connectivity
• Performance
• Security
• Transparency and Usability
• Fault Resilience

• Interoperability Probably the most significant of these are Decentralization, Scalability and Ad-Hoc connectivity. As the clients act as servers in p2p-systems, there is no need for central management, which in traditional client-server model is done by servers. Scalability is achieved, as hosts can join or leave the network easily without having to register into a database. As hosts can also be up or down at every instant, ad-hoc connectivity is achieved. Peer-to-peer networks can be divided into pure p2pnetworks and hybrid networks. In pure p2p-networks, there is no kind of central servers as in hybrid model where some servers are offered for e.g. locating the resources. The most known p2p-systems are probably file-sharing networks, such as Napster [16], Kazaa [10], DC++ [4], Gnutella [7] and Bittorrent [2] but p2p is also used in e-commerce, distributed computing (such as SETI@home [19], which can be considered as p2p-system) and in instant messaging (such as MSN Messenger [14]).

Mobile agents Mobile agents are defined as programs that can migrate from host to host. The program is able to save its state, "jump" to another host, and continue the processing in the new location. Mobile agents are also capable of multiplicating or cloning themselves. Mobile agents can also produce new agents by duplication, cloning or by creating totally new agents. As Reddy says in, "An agent is defined as a person, whose job is to act for, or manage the affairs of, another people." The same principle holds true for mobile agents. With mobile agents, one can define a job for the agent to carry out. These jobs could be for example to find the cheapest item in an e-commerce network, or to find the possibilities of flights for a vacation. A person could configure a mobile agent to perform this kind of task, launch it from a home machine, and wait for the agent to return with the information. With the use of mobile agents lots of intelligence can be added to the use of networks. As mobile agents can be programmed to perform certain tasks, the functionality of them can also be programmed in advance. This could mean for example for mobile agents the ability to act differently in different kinds of networks. Reddy lists some applications for mobile agents:

• Parallel computing

• Data collection

• E-commerce

• Mobile computing In parallel computing, mobile agents could locate hosts with enough resources, do the computing, and return home with data. For data collection, mobile agent could travel through the network and collect data that the user had it configured to look for. In the field of e-commerce, mobile agents could locate the most suitable item for the user. With mobile computing, mobile agents could help the host to save battery and bandwith. But as always, there are also drawbacks for mobile agents. Probably the biggest problem with them is the security. When agents travel the network, there is a big possibility for malicious hosts attacking them before the agent returns home

Mobile agents and p2p-networks Resource discovery In file sharing networks, such as Napster, Kazaa or Gnutella, locating the shared resources is one of the key-features. Present technologies use various methods for locating the resources. Napster used Centralized Directory model, a hybrid p2p-model, where one centralized server included information about the resources. When someone wanted to find a mp3-file, he asked the server, got the address of a peer having the desired file, and contacted that peer for downloading. The method Gnutella uses, is flooded request model. In this model the query is flooded through the p2p-network, and the peer that has the wanted resource, answers the query. But the resources to be found from the network does not need to be only files. Also for example computing power can be shared as a resource. In traditional resource sharing and downloading model the hosts establishes a communication channel, through which all of the information is sent and received. With this kind of system bandwith is needed continuously thus capturing bandwith from other processes. With the use of mobile agents, this need of communication can be packaged to a single packets, and sent to the network. Mobile agents thus allow all the interactions take place locally on the home host, as the agent gathers the information from the network and returns back to home host. Dunne also lists other reasons why to use mobile agents in resource discovery. Dasgupta presents a mobile agent based method for discovering resources in a network. Dasgupta uses both, stationary and mobile agents. The agents in Dasguptas method are

- Task agents (stationary)
- Reconnaissance agents (mobile)
- Search agents (mobile)
- Download agents (mobile or stationary)
- Information agents (stationary)
- Interface agents (stationary)
- Security agents

The most interesting agents for this paper are reconnaissance, search and download agents. The procedure for discovering and downloading a resource from network is conducted by following method: Task agent sends reconnaissance agents to all of the neighbor peers of the task agent. The task for these agents is to discover the availability of the resources including for example network connectivity and computational capabilities of that node. After taking all the necessary information, reconnaissance agent visits all the neighbor peers of the peer. When reconnaissance agent has visited a limited number (which is decided when launching the agent) of nodes, it returns to home and tells the task agent the addresses and other information obtained from hosts it had visited. Second, when a resource (file etc) is wanted to find, task agents creates a search agent, with an itinerary that consists of nodes reconnaissance agent had visited. On each of the visited hosts, search agent saves all relevant information if the resource it is trying to find is

found on the host. After visiting all the hosts on the itinerary, the agent returns home and tells the task agent about the information. Finally, the task agent decides from what peer the resource is going to be fetched, and creates a download agent whose job it is to download thee resource. Download agent tries to open a connection without going mobile, but if this is not feasible, it travels to the peer and negotiates the method to be used for downloading. Dunne describes another method for resource discovery using mobile agents. In this method, mobile agents are also launched from home host to the network to find the resources, but the agents are also allowed to clone themselves and thus being able to search more efficiently from larger network. By reproducing more agents, more reliability is also achieved, as one or more of the agents can be destroyed while others still trying to locate the resources. These methods for locating resources in the network can also be applied if the resource to be searched is not a file. For example the search for computational power for distributed computing could be done by mobile agents. File Sharing File sharing networks can easily me modified to use mobile agents as part of them. As presented , mobile agents can be used efficiently of locating not only files but all kinds of resources from the network. After finding the desired file from the network, it must somehow be downloaded. With mobile agents support for mobile devices can also be achieved. Presents a method for supporting mobile devices with mobile agents in Gnutella file sharing network. In this model the network consists of mobile devices, p2p-hosts and hosts with execution environment for mobile agents. First, when a user wants to join Gnutella file sharing network, he launches a software on the mobile device. The software then creates an agent with information about the host (ip-address, connection, list of files user wants to share) and sends the agent to a host in the network with the execution environment. As the mobile agent reaches the host, it sends its current location back to home host and joins the Gnutella network. For searching the files, First, the user defines what he wants to search, and this query is transmitted to the agent using lightweight protocol. The query is then transformed into Gnutella search query and sent to the network by the mobile agent. After results arrive to the mobile agent, it sends them, or part of them depending of for example bandwith, back to home host. The agent also must answer to queries from other hosts in the network, and if someone wants to download an item from the list of the mobile agent, it must forward the query to the home host. If the home host is unreachable, the query is rejected. If the user wants to download a file from the network. In the first scenario, the user can download directly from the host sharing the resource. The answers to the query that were sent by the agent to the mobile host includes the IP-address of the host sharing the resource, and thus the user is able to download the file directly. Alternatively, the mobile agent can download the file from the network, and save it on the host it is running or in some other place in the network. For this to happen, the home host must send the information about the file to be downloaded using the lightweight communication protocol to the mobile agent. When the home host wants to get the file, it asks for the mobile agent to send it. With this method

support for mobility is achieved as both, the mobile device and the mobile agent, are able to move. Only thing needed for mobility is that the agent and home host must know the address of the other party. Dasguptas method, described in more, is also good possibility for searching and downloading files from a network. With Dasguptas method, more intelligent downloading action can be achieved, as there is an agent whose responsibility it is to conduct the downloading operation. This way the operation does not need to be only simple fetching operation, but the agent can negotiate other possibilities for downloading.. defines the operation only for Gnutella network, Dasguptas method works in any type of p2p-network. Of course, the method Hu et al. describes could be implemented in other p2p file sharing networks. Downloading files from network could be developed even further. As intelligence can be built into the agent, the agent could perform many tasks for the file to be downloaded. For example, the file the user might want to download could be downloaded to a ftp-server from which the user could fetch it at some time. The agent could also only update a www-page containing for example links and information about possible files to be downloaded. In this scenario, the agent could only try to locate possible files, and the user decides according to given information what file he wants to download. In "traditional" file sharing networks some servers and lists of the shared files must be maintained. With the usage of mobile agents, these lists are no more needed. For authorities trying to prevent illegal distribution of copyright material, this leads to problems. When only the hosts in the network include information about the material they are sharing, it is difficult for authorities to find these peers. Of course, they could use the same methods presented in this paper, but it is not so feasible for them. 3.3 Distributed computing The computing power of modern computers have grown dramatically in the last few years. But still even the most powerful computers does not have the power needed for some problems. Lübke and Gómez gives Seti@home, cracking encryption keys and DNA sequence problems as such problems. But the problems does not have to be so big. A normal user might want to perform multiple tasks simultaneously without using all own capacity. The problem of resource usage can be solved by using p2p-networks. In the traditional distributed computing model, there is a central server whose job it is to split the problem to small individual problems, that are sent to hosts with the capacity. After the computations are done, the results are sent back to the server. [15] With mobile agents the usage of shared resources in other hosts can be done more efficiently. A user could define the task to perform for the mobile agent, and send it to the network to perform the task. The agent then could find the most suitable host to perform the task and return back with the results. In case that the host agent is performing its task goes down, the agent could jump to next host and continue the process there. Of course, the agent can migrate only if it has time to do it. If the host crashes without prior notification, the agent is lost. But the usage of mobile agents could easily be developed even further. If the resources the agent is using are not enough, or the problem is still too big to be performed, the agent could produce other agents, and send them to find suitable hosts for

performing the task. For locating the shared resources, mobile agents could use the same kind of system as Dasgupta uses for discovering resources from the networks, as defined in section 3.1. Mobile agents could also be used for managing the distributed computing. Mobile agents could be sent to travel the network and gather information about the hosts doing the computing. By this way the peer acting as the manager of the computing could easily get all the necessary information about the state of the distributed computing. For this to work feasibly, the manager must know all the hosts that are doing the computing. Of course, if the manager knows exactly the nodes doing computation, the need for a mobile agent is not so big, as manager can ask directly the state from the host doing computation. 3.4 Offline applications Some of the problems of the present mobile devices such as mobile phones is the lack of bandwith, the fees of bandwith usage and battery. For example in file sharing networks such as Gnutella [7] the hosts join and maintain their memberships in the network by sending periodically messages to other hosts [8]. This leads to problems with mobile devices with limited bandwith and battery. In some cases, the device might not be connected to the internet at all. With mobile agents the problem can be solved easily. The agent could be sent to the network to do the task, and the home host does not need to be connected to the network, or it may even be shut down.

After the agent has performed the task it is supposed to do, Lübke and Gómez [13] give two alternative implementations for the agent:

• The agent waits on a node somewhere in the network, until the home node comes online. If the node where the agent is waiting goes down, the agent should jump to another host to wait. If the agent has no time to migrate, it is destroyed.

• There are nodes in the network, that provides a persistent storage for agents. This way agents could wait for the home host to connect to the network again. Implementation is defined in more detail in [13]. With mobile agents the usage of bandwith and battery can thus be reduced. Usually the applications need constant interaction between the hosts and thus the device must be continuosly online in the network. With mobile agents the computation is shifted to the remote location instead of moving the data to the host. This way there is no need for the device to stay up all the time which leads to savings in battery and bandwith. One of the advantages of using mobile agents in mobile devices is achieved when the network used is not reliable . After sending the agent to the network, even if the home host wanted to stay running and connected to network, the connection can go down. As in the case when user wanted to disconnect, the agent can wait in the network after the task was performed to be able to get back to home host. Collaboration Today the most known p2p-systems after file sharing networks are probably instant messaging systems, such as MSN Messenger and Skype. But online games and shared applications can also be considered as collaboration applications. At least in the case of instant messaging systems the usage of mobile agents could be very efficient. Arbeeny introduces the term "active messages". As the

messages, the agents, are able to deliver the program with them, the messages become more than just strings of text that are delivered through the network. For example, the messages can be defined to route themselves to the target host. This feature is very useful for example in wireless ad-hoc p2p-network, where hosts can be out of range without connection to other hosts. When messages are sent as mobile agents, the messages can wait in the network for the host to come into range for sending. This way messages can travel through a long chain of hosts before getting to the target host. As instant messaging services are becoming more and more popular, with mobile agents more intelligence can be added to messages. For example, more intelligent smileys could be delivered to other users, if that is something users want.Calendaring and scheduling Mobile agents could be used efficiently in for example corporations where personnel have personal computers that include calendars and schedules in the company's intranet. Normally, companies use some commercial software to centralize the information for scheduling appointments etc. Usually these software are not feasible for small companies because of the high prices of the software. For scheduling purposes communities and companies could form p2p-networks, where mobile agents can be used for scheduling purposes. In this kind of system a mobile agent is programmed to reserve a time for an appointment or meeting and sent to network. The agent then travels all the hosts involved, check their calendars, and travel to next host. After the agent has found the time, it makes another round through the involved hosts, and make the reservation. The advantage of using mobile agents in calendaring and scheduling is that access priviledges can be granted more efficiently. A user can define easily who are allowed to see the contents of his calendar without publishing the whole calendar to the network. E-Commerce E-commerce is defined simply as the business based on Internet. The E-commerce network must not be exactly p2p-network, but the agent based e-commerce network can be applied to both. Whatever the case is, the hosts in the network have some assets or services they are selling or offering. Wang et al. defines a method for mobile agents to be used in E-marketplace. In this method there are several servers working for the transaction to happen. These servers are Coordinator server, Marketplace, Buyer Agent server and Seller server. The function of the model is described in more detail in. If the model would be transformed into p2pnetwork, there would not be need for so many servers, as hosts could act as marketplaces and seller servers, and buyer agents could travel only on these nodes. In the E-commerce mobile agents could be very useful. One could program a mobile agent to find the cheapest item, to reserve a vacation that matches users preferences or just travel the network and try to find something the users wants to find

Android Multimedia

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called **Media Player**.

Android is providing Media Player class to access built-in media player services like playing audio, video e.t.c. In order to use Media Player, we have to call a static Method **create()** of this class. This method returns an instance of Media Player class. Its syntax is as follows −

Media Player media Player = Media Player.create(this, R.raw.song);

The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name **raw** and place the music file into it.

Once you have created the Mediaplayer object you can call some methods to start or stop the music. These methods are listed below.

```
MediaPlayer.start();
mediaPlayer.pause();
```

On call to **start()** method, the music will start playing from the beginning. If this method is called again after the **pause()** method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, you have to call **reset()** method. Its syntax is given below.

mediaPlayer.reset();

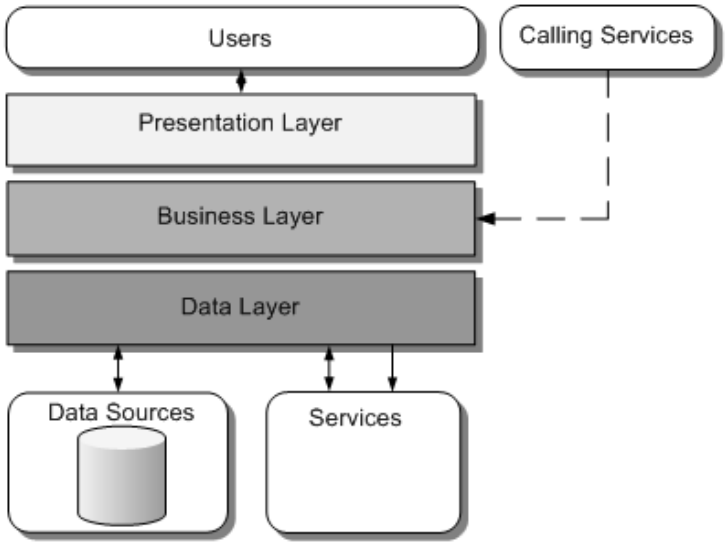
Apart from the start and pause method, there are other methods provided by this class for better dealing with audio/video files. These methods are listed below −

S. No	Method & description
1	**isPlaying()** This method just returns true/false indicating the song is playing or not
2	**seekTo(position)** This method takes an integer, and move song to that particular position millisecond
3	**getCurrentPosition()** This method returns the current position of song in milliseconds
4	**getDuration()** This method returns the total time duration of song in milliseconds
5	**reset()** This method resets the media player
6	**release()** This method releases any resource attached with MediaPlayer object
7	**setVolume(float leftVolume, float rightVolume)** This method sets the up down volume for this player

8	**setDataSource(FileDescriptor fd)** This method sets the data source of audio/video file
9	**selectTrack(int index)** This method takes an integer, and select the track from the list on that particular index
10	**getTrackInfo()** This method returns an array of track information

PLATFORMS AND ADDITIONAL ISSUES, SECURITY AND HACKING

Application architecture is a set of technologies and models for the development of fully-structured mobile programs based on industry and vendor-specific standards. As you develop the architecture of your app, you also consider programs that work on wireless devices such as smart phones and tablets.



Mobile app architecture design usually consists of multiple layers, including:

Presentation Layer - contains UI components as well as the components processing them.

Business Layer - composed of workflows, business entities and components.

Data layer - comprises data utilities, data access components and service agents.

Things to consider before attempting mobile app architecture development

As building a better application architecture is crucial to the success of your project, there are several things to keep in mind before you start designing your app architecture:

There are different types of smart phones and it is important to evaluate the device type and its characteristics before choosing a specific app architecture. You should keep in mind the following device features:

- Screen resolution
- Screen size
- CPU Features
- Storage Space
- Memory
- Availability of the development framework

Wondering why we should determine the device type when choosing the architecture of the application? Because an app's intended features may have some specific software and hardware requirements.

Android Hacking Applications

There are several popular applications that are used by developers to hack Android devices to make them faster, increase battery life, and customize screensavers, ringtones, alerts, and more. The list of hacks available to make improvements to an Android is large and growing every day. Tweaks or hacks can be either surface ones or the deep-system kind, depending on what the hack can do. Popular surface tweaks or hacks are:

Tusker - for location based automation

Ability to install custom keyboards like Swype and Swift Key

Deep system tweaks include downloading new kernels and radios to increase speed and battery life

Unfortunately, there are many hackers with malicious intent that can and do break into an Android device to steal valuable personal information or to profit from illegal financial transactions. While it may be hard (or even impossible) to make your Android un-hackable, there are things you can do to make your device more secure.

Three Biggest Hacking Threats to Your Android

Data in transit: Android devices and mobile devices in general are especially susceptible because they use wireless communications exclusively and often public WiFi, which can be insecure. An attack that is used frequently by hackers is a man-in-the-middle attack where an attacker breaks into the device and redirects data to exploit the resources on it before forwarding it to the original destination. This method

allows the hacker to spy on Internet browsing activity, steal keystrokes to identify passwords and isolate the individual's physical location, along with potentially listening to calls and intercepting texts.

Third party apps: In a recent study, 57% of malicious apps in the Android marketplace were found in third party app stores.

SMS Trojans: By including premium dialing functionality into a Trojan app an attacker can run up the victim's phone bill and get the mobile carriers to collect and distribute the money to them. Another malicious usage of SMS involves using an infected device to send out SMS text messages to all contacts in the address book with a link to trick the recipients into downloading and installing the worm, thereby infecting many devices at one time.

Three Steps you can take to protect your Android device

SSL encryption for the device: SSL is one of the best ways to secure sensitive data in transit.

Test third party apps: Try to install Apps from first party vendors like Google. If you do buy apps from a third party store, vet the security/authenticity of any third party code/libraries used in your mobile application by using a mobile security vendor. Read the permissions that apps require before downloading them. Examples of permissions apps can request that may raise red flags are permission to reveal your identity or location or send messages to the Internet.

Be wary of SMS Trojans: Implement controls to prevent unauthorized access to paid-for resources. If an application asks for a payment via SMS, exercise additional caution.

UNIT-V

PLATFORMS AND ADDITIONAL ISSUES, SECURITY AND HACKING

Android is one of the most popular operating systems for mobile devices. It is the operating system of choice for companies across developing markets. Android app development process is a sum-mean of critical phases. It comprises designing, creation, development, and post-development. To develop a successful Android app a robust and fail-proof mobile app development process is much required. With this article, we aim to share some of those proven strategies. It helps create opportunities for Android developers to create unique and powerful apps.

Here are some of the app development process guidelines for developing Android applications.

Android Application Development Process Steps:

Conceptualization – Refining the app idea into a solid base of the application is the first and most significant stage in the development process of Android application. The initial analysis of the app must include the demographics, behavior patterns, and goals of the buyer persona as all the other stages of app creation will depend on the said traits of users.

During this stage, all the necessary groundwork for the following process is laid down. It is beneficial to do substantial research and brainstorming before jumping to the next step. Another pivotal part of this stage is the competitor analysis to figure out what features can make the app stand out in the market.

Feasibility Assessment – Enterprises can gain a clear understanding of the app visuals through wireframes, detailed sketches of the conceptualized product to refine their ideas and arrange design components in a precise manner. To assess whether the concept of the application is technically feasible or not, the app developers need access to public data through public APIs sourcing. By the end of

feasibility testing, the team may have a completely different app idea if their original functionality is not feasible.

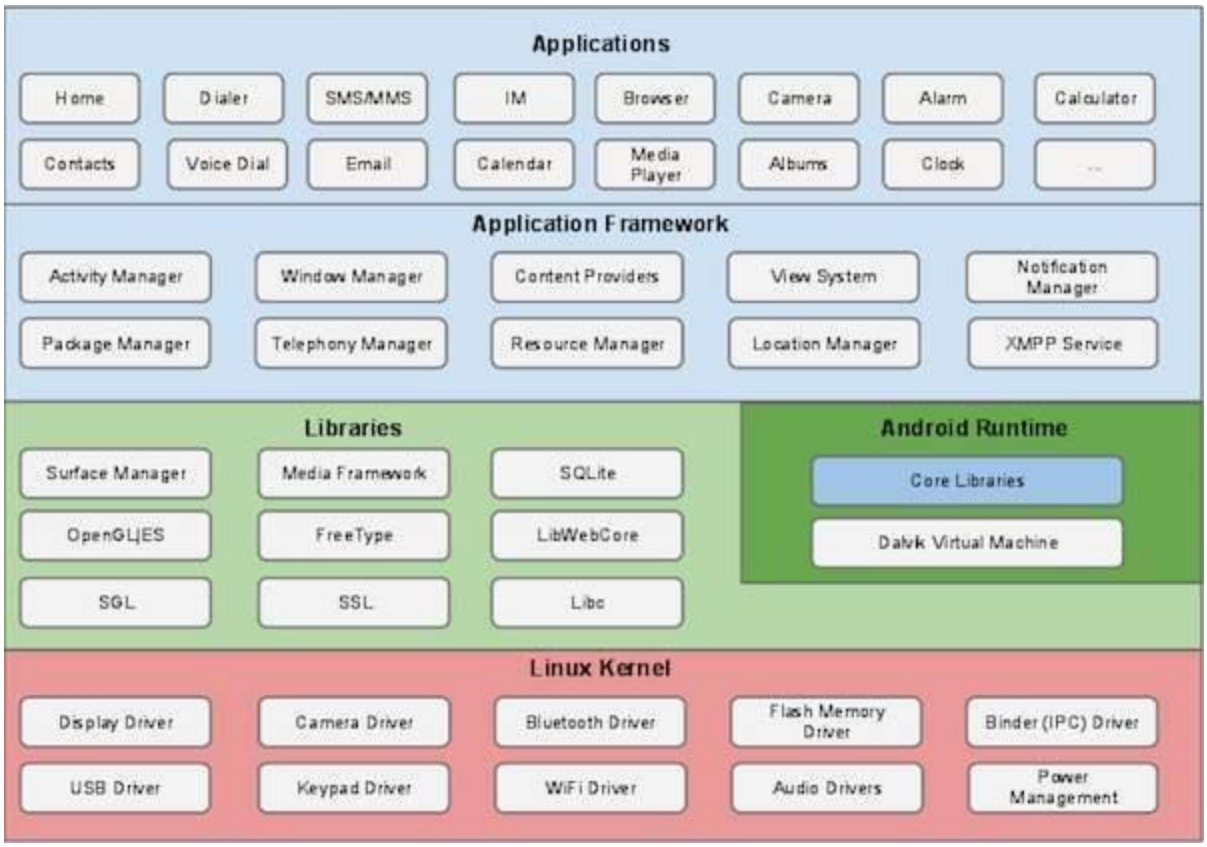
Design – In this phase of the Android development, the UX (user experience) designer architects the design elements' interaction, while the UI (user interface) designer builds the app's persona by keeping in mind the modern user's preference. Application designing is a multi-step process for drawing clear visual directions and offering an abstract of the final product.

Development – During this stage, a working prototype is developed to validate the functionality, assumptions, and understanding of the project scope. The app goes through a broad set of steps as the development progresses from core functionality development to light testing and then releasing the app for an external group of users for further field testing the concept. If an application has a broader scope than the usual, the creation process gets divided into smaller modules through agile methodology, and the entire mobile app development process is applicable for each of these small parts. **Testing & Deployment** – One of the critical components of the app development process, it is a good idea to test at early stages, often for usability, interface & security checks, stress, compatibility, and performance. After fixing the bugs, the app moves to the deployment phase and is ready for release via a formal launch. Different application stores have different policies of application launch, and therefore the deployment phase plan is aligned according to the app store. However, keep in mind that this is not the end.

Android application development service does not end with the launch. As the application gets in the users' hands, feedback will start pouring in, and enterprises will need to incorporate the feedback to develop the future versions of the app. As soon as the first version of the app is out, the app development cycle commences incorporating new updates and features. As soon as the first version of the app is out, the app development cycle commences yet again incorporating new updates and features, by ensuring that the organization has the required resources to maintain the viability of the product. It is about sustaining a long-term commitment, that is beyond the significant amount of money invested in building the said digital product.

Architecture:

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



Linux kernel

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Libraries

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that

facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows −

- **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing and messaging between applications and application components.
- **android.database** − Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.
- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

- **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** − Allows applications to publish and share data with other applications.
- **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** − Allows applications to display alerts and notifications to the user.
- **View System** − An extensible set of views used to create application user interfaces.

Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

Mobile App Development Hurdles:

Here are some unfeasible challenges faced by mobile app developer as a startup founder.

1. **App Discovery**

The fundamental aim of developing an app is to make life simple, productive and pleasant. Keeping it in mind, creating an app which gets noticed is a grueling task. There are a way more good apps than there are successful ones, and that's because many of the good ones don't get investment. App discovery is extremely concerned with who are your users, what type of service they are expecting, their financial background and many others factors. Make sure while choosing developer team, it must be chosen gingerly.

2**. Development Approaches**

Of course, the world isn't small so the field of mobile app development. There are numerous development approaches i.e. Hybrid mobile app development (combination of native app & web app) Cross platform native app (app can be available for two or more platform like iOS, Android and even JavaScript, HTML5 etc) Platform specific native app (made for particularly one platform).

Deciding development approach will decide framework & mock-ups, UI & UX and many other imperative entities that mobile app is made of. So user interaction is a total concern with choosing right development approach.

3. **Investment Required**

Once you have decided the development approach, to develop your app in real, it requires ample amount of money (to hire the developers in case of you don't belong to programming background). There is no one-time investment as after completion of development, to bring a new variety of features and for Iteration, adequate amount of money is necessary.

4. **Device Compatibility**

Our world is seemingly teeming with high-tech mobile brands. Challenges you might face are inevitable i.e. screen resolution, OS requirements, RAM and other factors and whether your app works on a smart phone or specific tablet or phablet. The main issue is choosing OS (either Android or iOS). For different OS you have to go for different SDK (software development kit), UI & UX, framework & Mock-ups and different iteration processes. Your app should run on a latest available version of particular OS as well as on an older version of similar OS. The empirical solution is to develop an app for each different platform available if you have enough investment. My personal opinion in the case of selection of OS goes for iOS, though both OS have their pros and cons.

5. **Performance vs. Battery**

Parameters such as an app design, UI, user interaction are important, but the main factors you should not forget are Performance & battery consumption. If you can develop a good performing, bug-free app which runs on minimum power, this challenge can be overcome. Developer team has to be specific about choosing right development tool (such as SDK) and be precise about device specifications as well.

6. **Competition**

Once you developed an app, you need to launch it into the app market, where as I mentioned before the gargantuan figure of apps are launched every day. As a startup, the biggest challenge is to stand out from the rest while creating an app. Even popular apps and games developers are struggling hard to make their mark.

7. **ASO**

ASO stands for App Store Optimization, is a process of optimizing mobile apps to rank higher in the search result. Just like SEO (search engine optimization for articles, images, and video content) is for

websites, ASO works for mobile apps. Better your ASO, the more likely your app will reach thousands of devices. This is a most fastidious challenge that people have forgotten about. It also includes an imperative component of 'App store rating' & 'App store ranking'.

8**. Marketing & Promoting**

Last but not least, as a startup, you should have precise knowledge about marketing and empirical way of promoting your app. It includes PR & media plan, social & viral marketing & internationalization of app which mean the development of an app that enables easy localization for targeted audience, regions, and language. You can put an app on the market in a weird but attractive manner that attracts consumers. There are several challenges that you will face: a crowded market, the same service provider as you, investment for promoting an app and others. To simplify marketing for you there are numerous tools like AppTamin, AppScend, MobileDevHQ, and Some other you should know about.

There are many other challenges like debugging, beta testing, prototype simulation & distribution of app to make it available in different languages. An auspicious app is about 90% marketing and 10% development (yeah you read it right 90%) but that does not mean development isn't important.

1. **Preparing the security testing plan**

One of the challenges of pen testing mobile applications involves applying the correct methodology. The OWASP iOS cheat sheet provides an overview of the attack vectors that can be considered in your test plan. This one, has been specifically created for iOS devices, although the methodology applied can be used for other platforms.

As shown on the above tree, each major attack surface contains specific areas that apply to the assessment. For each major attack, consider an appropriate technique:

Application mapping ⇒ Information gathering

Client attacks ⇒ Runtime, binary, and file system analysis

Network and server attacks ⇒ Network analysis and insecure data storage

For each section, a specified set of tools and skills are necessary. In addition to understanding the OS to be attacked, you should pay close attention to the type of mobile application. Each one has a different attack vector. In the case of a native iOS app, it probably has been written in Objective-C or Swift language, but browser-based or hybrid apps are built using traditional web application technologies, created to be run on the device's web browser.

Furthermore, compared with web applications, analysis of mobile applications has a major difference: binary and file system analysis. This stage requires reverse-engineering skills and the use of debugging techniques with programs such as IDA Pro, as well as disassembling files with GCC, for example. In contrast, reverse engineering represents a small portion of traditional web application penetration techniques, due to the architecture and deployment of the thin client.

2. **Preparing the testing environment**

Web applications run on all kinds of platforms and browsers, but that is not the case for mobile apps. Therefore, a specific device-driven testing environment must be configured. In the case of an iOS device, for example, it will be necessary to jailbreak the device, given the security imposed by Apple, which will not permit you to observe and analyze or respond to the attack? Using the evasi0n7 jailbreak allows you to boot the device and have root/admin access to the OS. In the case of Android, rooting a device by installing One Click Root will supply such access.

3. **Building the attack arsenal**

Once the device is ready, it will require some extra tools to be installed for analysis and information-gathering purposes. These should be deployed in the testing environment and the device. Cydia is the app store of jailbroken iOS, and through it, it's possible to download the necessary tools for hacking. Debuggers and decrypters can help you understand the mechanics of the application.

For binary analysis, using Android Apktool or a more robust Android Reverse Engineering virtual machine is highly recommended. If the app is encrypted, decrypting it using tools such as AppCrack or Dump Decrypted is a must. In addition, Burp Proxy, Android Proxy, OWASP ZAP, Wireshark, and Tcpdump are just a few of the tools available for network analysis.

4. **Preparing the test cases: Application mapping**

In this phase, observing the application at the functional level and analyzing its behavior, including decrypting it if the application has been obfuscated, is essential. Extraction of what kind of frameworks have been used, taking notes on the following, will help create a proper threat modeling, applying the same principles for creating a test suite as explained in the OWASP testing guide:

Identity, authentication, and access control => Key chains, brute-force attacks, parameter tempering

Input validation and encoding => Malicious input, fuzzing

Encryption => SQLite database password fields, configuration file encryption

User and session management => Session IDs, time lockouts

Error and exception handling

Auditing and logging => Logs, access control to logs.

5. **Attacking the client: Binary and file analysis**

During this phase, the main goal is to discover insecure API calls and files not properly secured with adequate access controls. This can be achieved by debugging and analyzing the code using IDA Pro or the Hopper App. Buffer overflows should not be discarded. To uncover vulnerabilities such as SQL injections, you can use techniques like fuzzing the application or applying malicious inputs. Techniques used to actually uncover vulnerabilities within a native application are similar to pen testing web apps with this difference: Instead of using a proxy to understand the inner workings of the app, debugging software is used.

Some of these techniques involve testing approaches similar to those used in the OWASP testing guide. During web pen testing, we are most certainly assisted by the use of an attack proxy to inject malicious input. In the case of a native mobile application, a tool such as iOKit can support this task. To assess risks related to local data storage, database browsing with the SQLite database browser applies in the case of Android and iOS, to verify how the data has been secured. Eventually, if encrypted, you can verify the type of encryption used in sensitive data fields. Analyze proper storing of API key chains' location and access control to them, among some of the tests applied to client attacks.

6. **Network attacks: Install traffic, run traffic**

When the mobile application has been designed with a clear client-server tier architecture, network attacks are one of the major concerns. Using sniffers to capture network traffic and investigate transport layer protection is essential. Attack proxies such as ZAP can assist with that. Other tests that should be included during this phase are:

Authentication. By observing the request and responses between client and server, it is possible to uncover vulnerabilities involving authentication. In cases where the application is using HTTP basic authentication, this represents a risk. If used, it should always be done through SSL.

Authorization. Roles and access controls between them can be uncovered through parameter tampering. Securing the API key properly in an inaccessible folder can be uncovered by file analysis (native apps) or spidering the application (web-based apps).

Session management. Session ID tokens sent through GET methods and placed in the URL are visible while proxying the application or sniffing the network.

Weak encryption and protocols. Mobile applications are more vulnerable at these areas. Wireless vulnerabilities revolving around encryption protocols used by the device must be categorized.

7**. Staging server attacks**

Testing the infrastructure, specifically the server hosting the mobile web app, requires tools like Nmap and similar pen testing armor designed to map and discover potential vulnerabilities and exploitation threats. Also, unrestricted file upload, open redirect, and cross-origin resource sharing should be included as part of the tests.

Testing on hybrid and web-based mobile apps should focus on conducting attacks that try to bypass authentication mechanisms between the thin client and the server. For instance, implementation of web services security can cause vulnerabilities such as XML and XPath Injections. A clear example of a server-side attack occurred in 2013, when Apple ID iCloud accounts could be easily hijacked by resetting the password, providing only the email and date of birth of the owner of the account. Weak authorization controls were the major cause of this security vulnerability.

8. **Learning more about mobile vulnerabilities**

Practice makes perfect. One of the resources available to testers willing to learn more about how security vulnerabilities occur in mobile applications are vulnerable mobile applications designed with this purpose in mind.

Some interesting dummy vulnerable applications are:

Damn Vulnerable iOS Application (DVIA)

MobiSec

Androick Project Page

The Damn Vulnerable iOS app provides complete documentation, including some instruction articles that discuss setting up a testing environment, runtime analysis, and network traffic, among many other detailed examples.

A false sense of mobile security is pervasive

There clearly is a false sense of security among mobile users. Mobile apps are plagued with vulnerabilities that affect them in ways that are similar or identical to those in web applications. There are real challenges in securing them, which demands an understanding of proper security controls so that

applications are developed with security in mind. Testing the apps involves areas such as reverse engineering, decryption, and file analysis, which require skills that are not so simple to find.

Mobile developers should also become aware of these techniques to build more secure applications. Some interesting documentation is available, as well as other resources focused on creating awareness around the risks of vulnerable mobile applications. In addition, continuous learning and practice can help you better understand the security risks associated with them. In the meantime, do not just click and download. Rather, execute due diligence by understanding who developed an application and what kind of information is provided regarding security controls, including proper two-factor authentication.

Android Hacking Applications

There are several popular applications that are used by developers to hack Android devices to make them faster, increase battery life, and customize screensavers, ringtones, alerts, and more. The list of hacks available to make improvements to an Android is large and growing every day. Tweaks or hacks can be either surface ones or the deep-system kind, depending on what the hack can do. Popular surface tweaks or hacks are:

Tusker - for location based automation

Ability to install custom keyboards like Swype and SwiftKey

Deep system tweaks include downloading new kernels and radios to increase speed and battery life

Unfortunately, there are many hackers with malicious intent that can and do break into an Android device to steal valuable personal information or to profit from illegal financial transactions. While it may be hard (or even impossible) to make your Android un-hackable, there are things you can do to make your device more secure.

Three Biggest Hacking Threats to Your Android

Data in transit: Android devices and mobile devices in general are especially susceptible because they use wireless communications exclusively and often public WiFi, which can be insecure. An attack that is used frequently by hackers is a man-in-the-middle attack where an attacker breaks into the device and redirects data to exploit the resources on it before forwarding it to the original destination. This method allows the hacker to spy on Internet browsing activity, steal keystrokes to identify passwords and isolate the individual's physical location, along with potentially listening to calls and intercepting texts.

Third party apps: In a recent study, 57% of malicious apps in the Android marketplace were found in third party app stores.

SMS Trojans: By including premium dialing functionality into a Trojan app an attacker can run up the victim's phone bill and get the mobile carriers to collect and distribute the money to them. Another malicious usage of SMS involves using an infected device to send out SMS text messages to all contacts in the address book with a link to trick the recipients into downloading and installing the worm, thereby infecting many devices at one time.

Three Steps you can take to protect your Android device

SSL encryption for the device: SSL is one of the best ways to secure sensitive data in transit.

Test third party apps: Try to install Apps from first party vendors like Google. If you do buy apps from a third party store, vet the security/authenticity of any third party code/libraries used in your mobile application by using a mobile security vendor. Read the permissions that apps require before downloading them. Examples of permissions apps can request that may raise red flags are permission to reveal your identity or location or send messages to the Internet.

Be wary of SMS Trojans: Implement controls to prevent unauthorized access to paid-for resources. If an application asks for a payment via SMS, exercise additional caution.

Android Hacking Resources

XDA Developers Forums is a great resource for learning about Android hacking applications, and Android hacking tools to customize Android devices. It is a mobile software development community of over 4 million users worldwide, started in 2003. The site's main purpose is discussion, troubleshooting and development of Android among other devices.