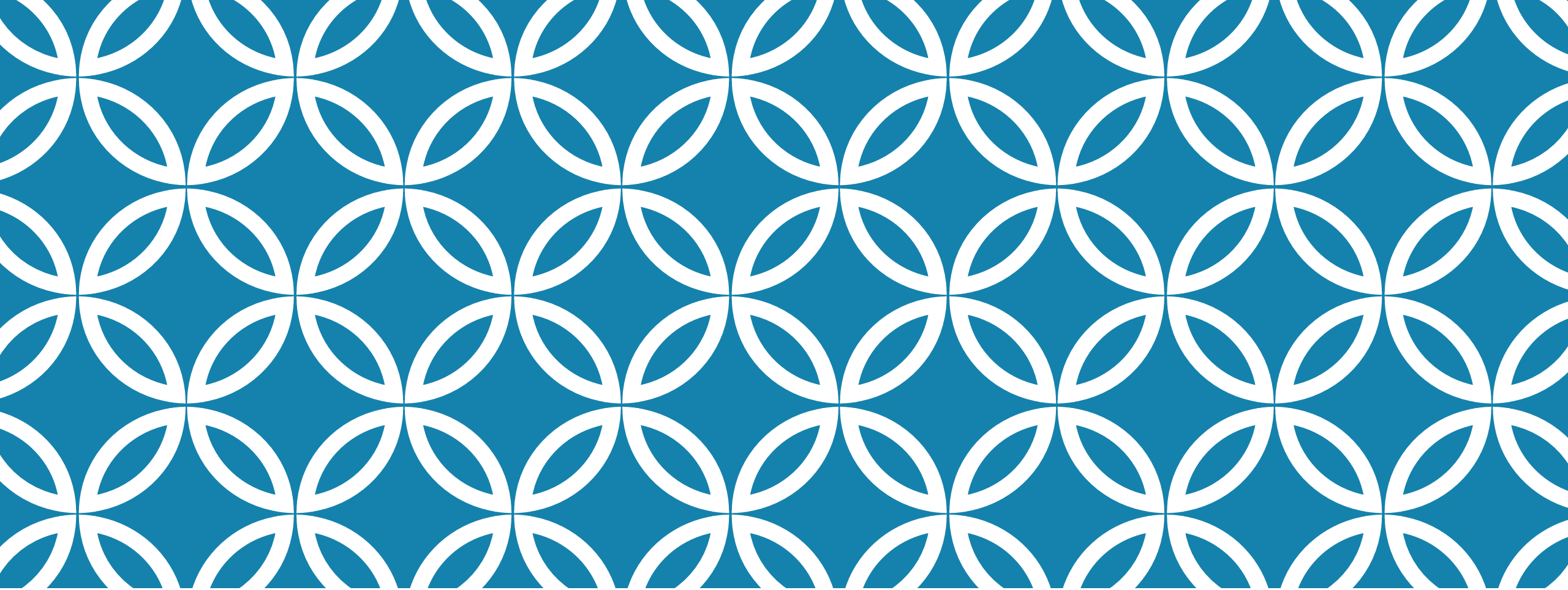


ANDROID PRIMER

CS 175 Mobile Software Dev

by Dr. Angus Yeung

ANDROID SYSTEM ARCHITECTURE

Android system architecture is designed for mobile programming.

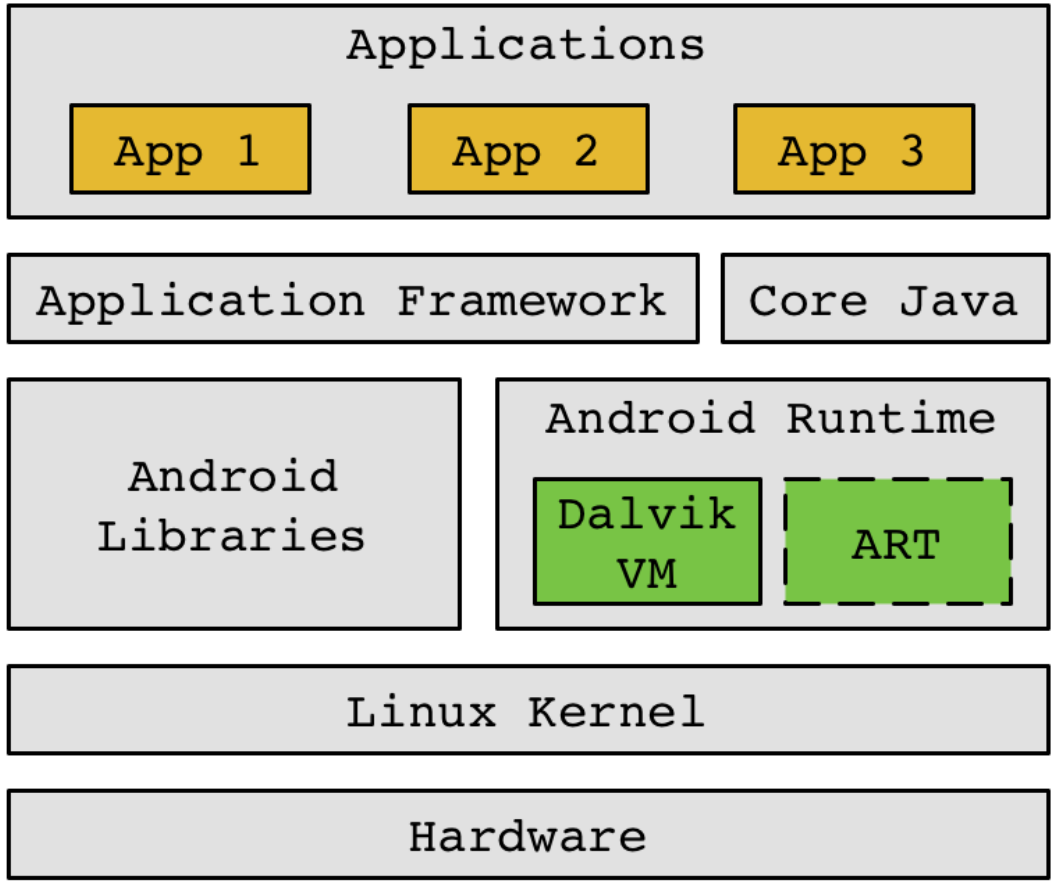
It consists of several distinct modules: Applications, Application Framework, Core Java, Android Native Libraries, Android Runtime, and Linux Kernel.

ANDROID SYSTEM ARCHITECTURE

As we can see, many modules such as Linux Kernel and Core Java are based on on the existing and mature technologies.

This strategically helps Android tap into the rich talent pools of Linux and Java development communities when Android was first rolled out to mobile programmers in 2008.

MODULES IN ANDROID SYSTEM



DESCRIPTION OF ANDROID SYSTEM MODULE

Applications - this module contains the applications contributed by a mobile programmer.

Typically a programmer writes Java code that work with Android frameworks with various built-in features. Some of these applications, such as web browser, phone dialer, email app, etc., are native applications that come with the Android device.

A mobile programmer may start a built-in application from custom application through Android's Intent mechanism.

DESCRIPTION OF ANDROID SYSTEM MODULE

Application Framework - this module provides several Android frameworks to your custom application.

Each framework abstracts and communicates with the underlying native libraries to accomplish different tasks required by your custom application.

Rather than dealing with low level native code in Android libraries, now you can focus on the new features you want to implement in an object oriented approach using Java.

DESCRIPTION OF ANDROID SYSTEM MODULE

Core Java - Java is the official programming language for Android. Android supports most but not all of the features provided by Java libraries. However, Android sometimes lags the support of new JDK features in the latest Android SDK release.

For example, Android N does not support all JDK 8 features but it supports Lambda expression which is a very popular feature for developer communities.

DESCRIPTION OF ANDROID SYSTEM MODULE

Android Native Libraries - this module includes native libraries written in C/C++. Many library code is highly optimized for specific hardware architecture and must be ported to support another hardware platform specifically.

Some Android phone makers may modify this module heavily to expose or enhance many new phone features empowered by their hardware.

Except working with the legacy C/C++ code, you are recommended not to write native code for Android applications.

DESCRIPTION OF ANDROID SYSTEM MODULE

Android Runtime - this modules contains Dalvik virtual machine (VM) for older Android devices or ART (Android Runtime) for newer Android devices.

Dalvik VM is similar to normal Java virtual machine but highly optimized for mobile environment (e.g., designed specifically for resource constraint environment).

A Java VM compiles your code into byte code class file which is then converted to .dex files that will be installed on Android device.

DESCRIPTION OF ANDROID SYSTEM MODULE

Linux Kernel - Like any other Linux kernels, this module provides essential kernel features such as security management, memory management, process management, drivers, file system, network I/O, etc.

Android added many mobile specific features to the kernel, including power management, and interprocess communication between processes.

ANDROID VERSIONS

At the time of this writing, the latest version is a developer preview of Android N.

Android 4.4, code named "KitKat" is the single most widely used Android version at about 35.5% of total shares.

Code Name	Version Number	Release Date	API Level
	1.0	09/23/08	1
	1.1	02/09/09	2
Cupcake	1.5	04/27/09	3
Donut	1.6	09/15/09	4
Eclair	2.0-2.1	10/26/09	5-7
Froyo	2.2-2.2.3	05/20/10	8
Gingerbread	2.3-2.3.7	12/06/10	9-10
Honeycomb	3.0-3.2.6	02/22/11	11-13
Ice Cream Sandwich	4.0-4.0.4	10/18/11	14-15
Jelly Bean	4.1-4.3.1	07/09/12	16-18
KitKat	4.4-4.4.4,4.4W-4.4W.2	10/31/13	19-20
Lollipop	5.0-5.1.1	11/12/14	21-22
Marshmallow	6.0-6.0.1	10/05/15	23
N	Developer Preview 1		

THE EVOLUTION OF ANDROID

TABLE 1-1 Common Android Platforms

Platform Version	API Level(s)	Year	Version Name	Important Features
5.0	23	2014	Lollipop	Tuned to work on devices with 64-bit ARM, Intel, and MIPS processors
4.4	19	2013	KitKat	Optimized to run with a minimum of 512 MB of RAM
4.3 4.2 4.1	18 17 16	2013, 2012	Jelly Bean	Efficient and refined user interface Google Voice Search
4.0.4, 4.0.3, 4.0.2, 4.0.1, 4.0	15 14	2012 2011	Ice Cream Sandwich	Compatibility with most Android 2.3.x API fixes and refinements
3.2 3.1 3.0	13 12 11	2011	Honeycomb	Tablet-only support

THE EVOLUTION OF ANDROID

2.3.4, 2.3.3, 2.3.2, 2.3.1, 2.3	10 9	2011 2011	Gingerbread	Enhancements for game developers Support for multicore processors
2.2	8	2010	Froyo	Android Cloud support
2.1 2.0.1 2.0	7 6 5	2010	Eclair	Google Maps Navigation New camera features Bluetooth Speech-to-text
1.6	4	2009	Donut	Integrated camera and camcorder Expanded gestures Introduction of Quick Search Box
1.5	3	2009	Cupcake	Soft keyboard Copy and paste Video recording
1.1	2	2009		Bug fixes to Android 1.0
1.0	1	2008		Gmail Notifications Widgets

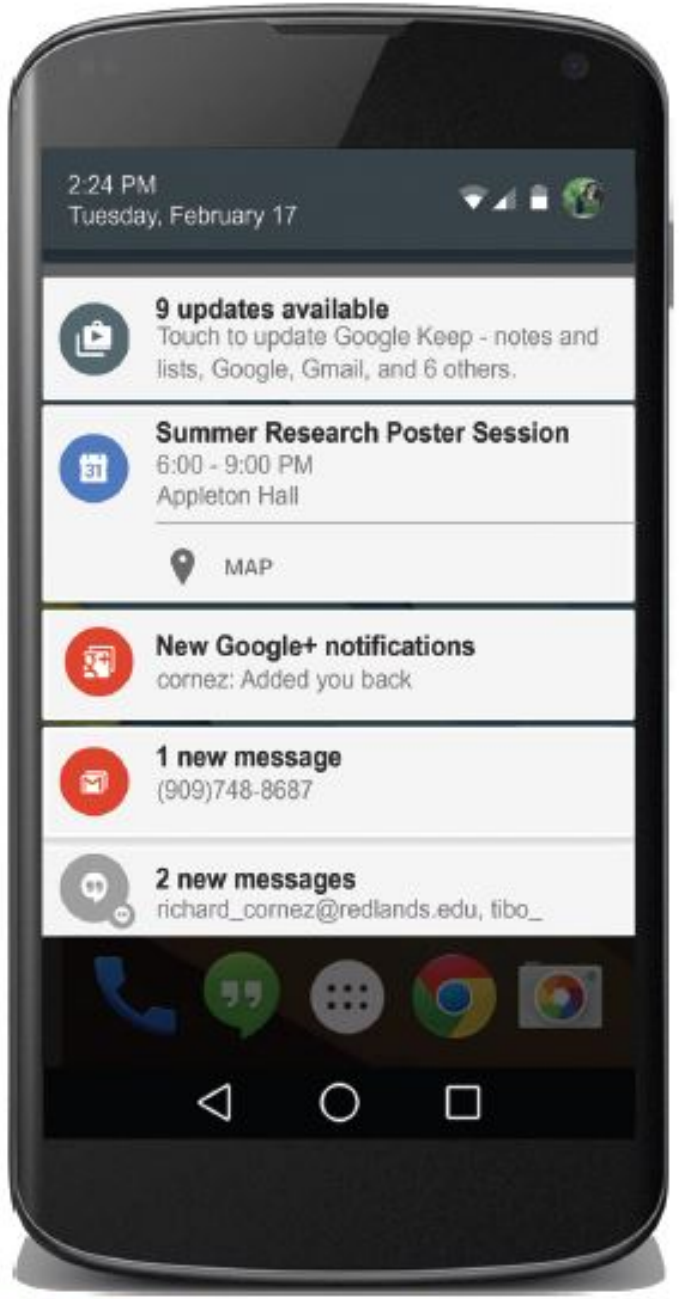
THE EVOLUTION OF ANDROID

Google services were integrated as part of the Android operating system since the first installment

Two of the most substantial inclusions of Android 1.0 were **Google Maps** and **Notifications**

Notifications remains an important part of the Android user interface

In current releases of Android, this feature has been refined to include chat messages and calendar events that support synchronization across all Android devices



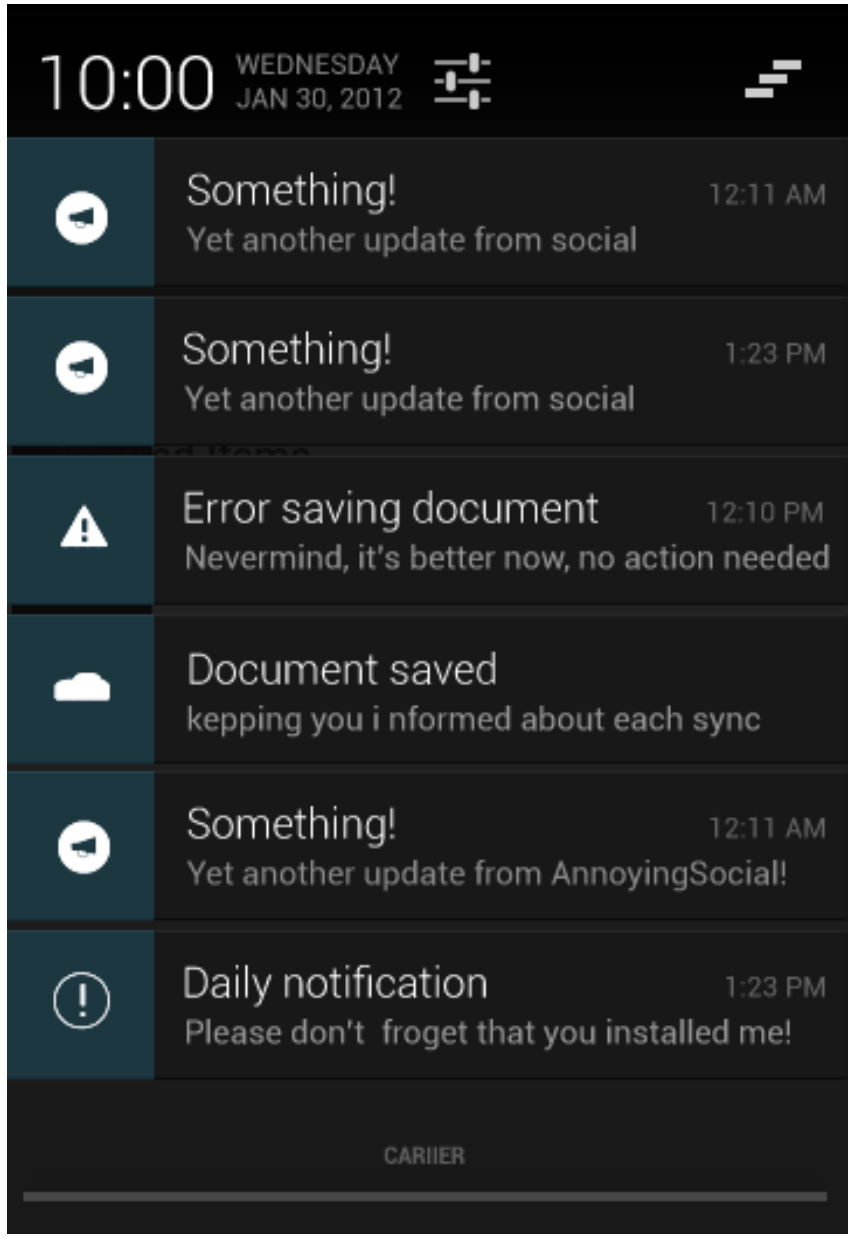


FIGURE 1-1 Android Lollipop Notifications.

CUPCAKE

By the time **Cupcake** appeared in 2009, refinements, such as the inclusion of self-refreshing widgets and copy-and-paste, had been implemented

One of the most significant features that emerged with **Cupcake** was the introduction of a soft keyboard, which also brought the first forms of keyboard skins

Keyboard skins are created by third-party developers to provide enhancements to an existing soft keyboard

CUPCAKE

Unique to Android, skins allow users to personalize their keyboard

Skins have evolved alongside Android and have shifted beyond basic appearance and into more extensive behavior than originally seen in Cupcake

Today, Android device manufacturers regularly offer skins that enhance the user experience by adding functionality to the design.

FROYO

Android 2.2 was created with the ability to execute far faster than previous versions. This was due to the introduction of the new Dalvik Just-In-Time (JIT) compiler, first seen on Froyo

Dalvik allowed for better CPU performance, which significantly enhanced processing power

Froyo's browser came with a new JavaScript engine, making Internet browsing nearly three times faster than the previously released version of Android

Froyo also brought native support for tethering.

GINGERBREAD

Gingerbread the first version of Android that backed multi-core processing on mobile devices

For application developers, **Gingerbread** brought support for new technologies, such as NFC (Near Field Communication), and SIP (Session Initiation Protocol)

The SIP API provides developers with tools to create applications that perform video conferencing and instant messaging

HONEYCOMB

Honeycomb was released as the first version of Android specifically implemented for tablets

Prior to **Honeycomb**, Android tablets were running on phone operating systems that were stretched to fit the screen of a larger tablet

ICE CREAM SANDWICH

Ice Cream Sandwich combined the best characteristics of **Gingerbread** and **Honeycomb** into a single operating system that would work for tablets and phones

Ice Cream Sandwich was able to bring many of the design elements of **Honeycomb** to smartphones, while refining the Android experience

JELLY BEAN & KITKAT

Jelly Bean it was a faster and smoother Android version

Along with its Google Voice Search feature, **Jelly Bean** was a jump in magnitude of performance

This version of Android was often referred to as a turning point for Android, where services and customization options met responsive design guidelines

KitKat looked similar to **Jelly Bean**; however, due to a 512 MB size it was able to run on a much larger array of devices. Previously Android required1GB to 3GB to run.

LOLLIPOP

Jelly Bean it was a faster and smoother Android version

Along with its Google Voice Search feature, **Jelly Bean** was a jump in magnitude of performance

This version of Android was often referred to as a turning point for Android, where services and customization options met responsive design guidelines

KitKat looked similar to **Jelly Bean**; however, due to a 512 MB size it was able to run on a much larger array of devices

ANDROID RUNTIME

Runtime is an environment which provides libraries, virtual machine and other components to run applications written in the Java programming language.

The virtual machine is an implementation for executing compiled Java files in .class on a specific operating system.

ANDROID RUNTIME

Android runtime includes Dalvik virtual machine to execute and manage Android processes.

Dalvik is Google's implementation of JVM that is optimized for mobile.

ANDROID RUNTIME

For Jelly Bean version of Android, Google also released another runtime called ART (Android Runtime).

Developer can select either Dalvik or ART from Developer options.

ANDROID RUNTIME

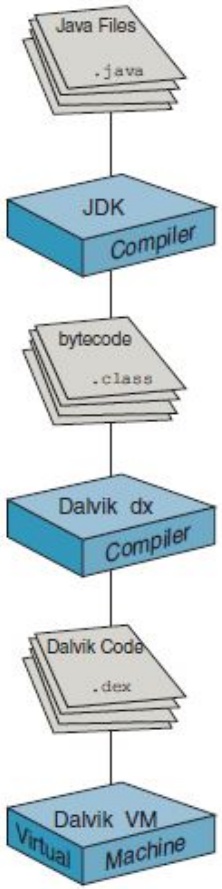


FIGURE 1-3 Java bytecode is compiled into Dalvik bytecode and executed on a Dalvik VM.

The .class and .jar intermediate files from .java source code are further converted to .dex files.

The .dex file format is more compact and efficient, making it a good storage option for memory resource limited phone devices.

ANDROID RUNTIME

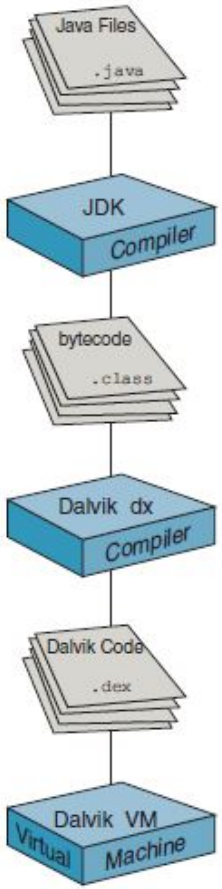


FIGURE 1-3 Java bytecode is compiled into Dalvik bytecode and executed on a Dalvik VM.

Bytecode loaded by the Dalvik virtual machine should not be confused with Java bytecode.

The Android SDK includes the Dalvik dx tool, used to translate java bytecode into Dalvik bytecode.

WORKFLOW FOR BUILDING AN APP

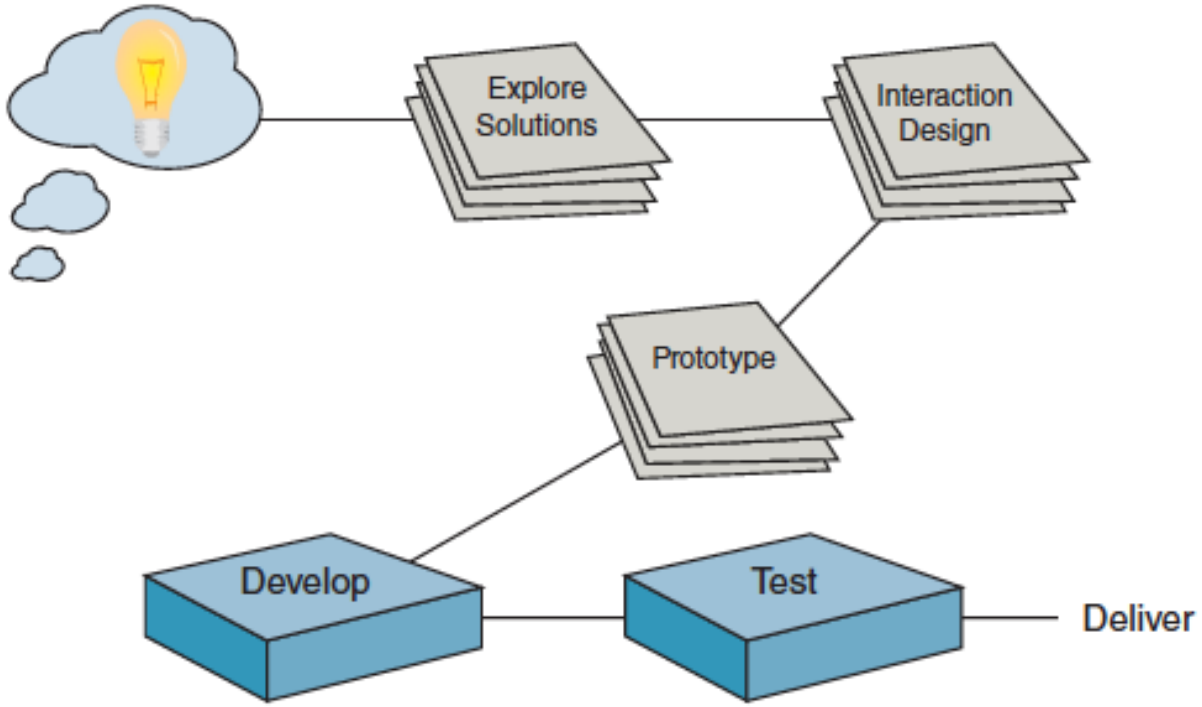


FIGURE 1-4 The process for building begins with an idea and ends with delivery.

WORKFLOW FOR BUILDING AN APP

A project built in Android Studio, will be structured similar to Figure 1-6

Project structure is organized by source code, the application's resources, manifest settings, and build files

All of these files are eventually packaged into an APK file, which represents the final

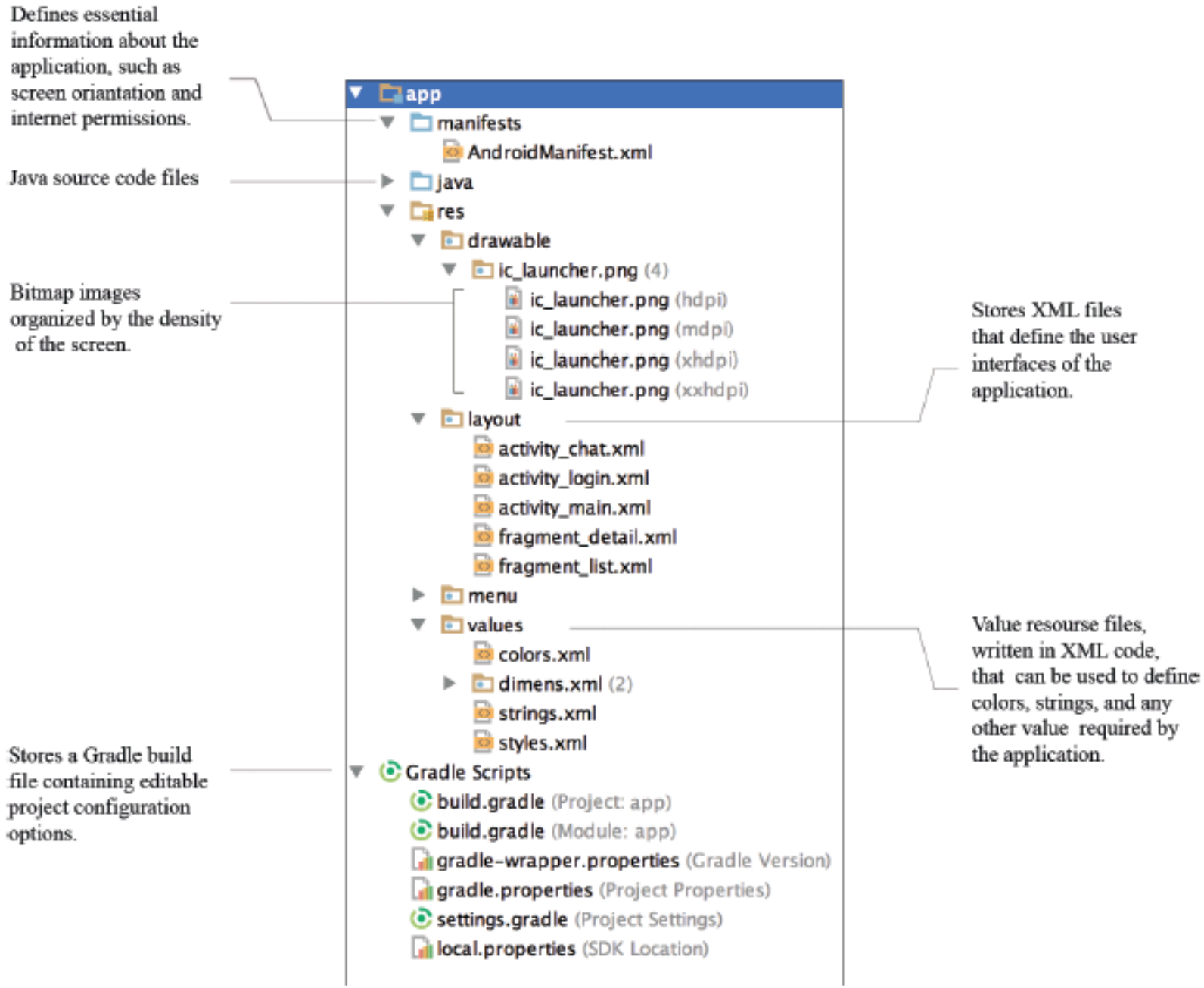


FIGURE 1-6 Sample Android App Project Structure.

ANDROID MANIFEST FILE

Manifest file plays an important role in Android application project because it provides Android operating system necessary information about the application.

The file must be named as AndroidManifest.xml and placed in the project's root directory.

ANDROID MANIFEST FILE

You may use AndroidManifest.xml to configure the following settings for your Android application:

- Minimum API support level
- Hardware and software features
- User permissions <activity> <users-permission>
- Dependencies of Google API libraries
- Device compatibility
- Declaration of App Requirements
- Declaration of Components and Capabilities, e.g., Intent Filter

RESOURCES IN ANDROID PROJECT

Resources in an Android project may include icons, images, localized strings, layouts, etc.

Android resource compiler compresses and packs the resource items, and creates the **R** class that provides references and identifiers for you to reference the resource in your application code.

DRAWABLE RESOURCES

The drawable folder is located in the res directory

res contains the application resources

Drawable resources are image files, such as application icons, buttons, and background textures

DRAWABLE RESOURCES

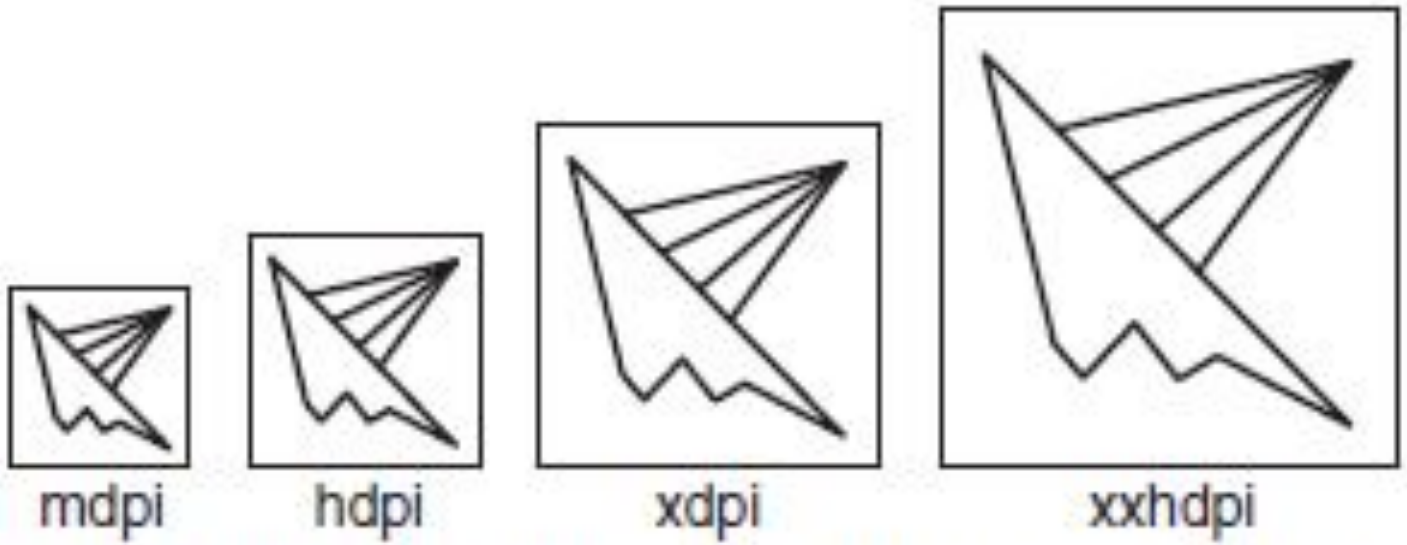


FIGURE 1-7 Pixel density of graphic images ranges from medium to extra-high density.

GENERALIZED SCREEN SIZE

xlarge screens are at least 960dp x 720dp

large screens are at least 640dp x 480dp

normal screens are at least 470dp x 320dp

small screens are at least 426dp x 320dp

LAYOUT XML FILES

User interface screens are visually designed and coded as XML layout files

The design and arrangement of the elements on the screen are implemented using XML code in a layout file.

MODEL-VIEW-CONTROLLER

Android applications tend to rely on the Model-View-Controller design architecture

This architecture assigns one of three roles that objects can play in an application

For example, an object can be a Model, View, or Controller

MODEL-VIEW-CONTROLLER

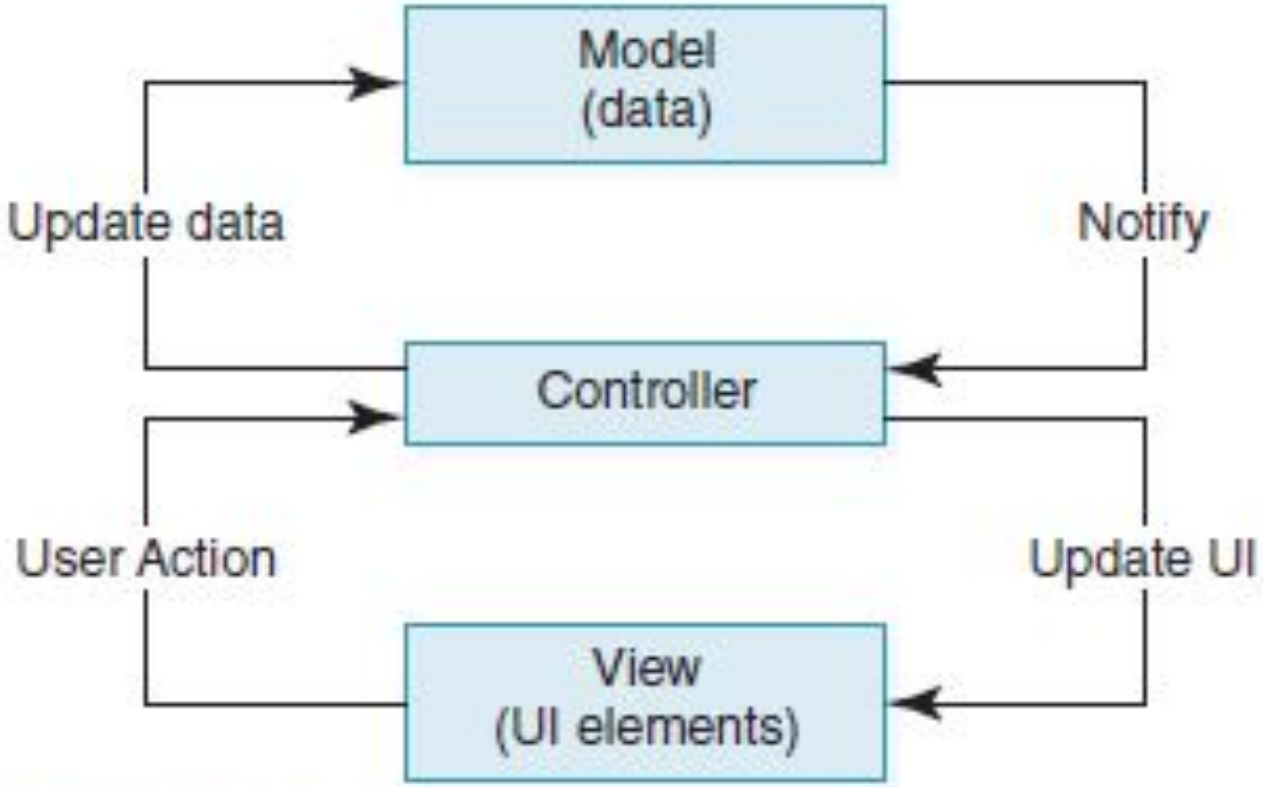


FIGURE 1-61

SHARING YOUR ANDROID APPLICATIONS

Android requires the application to be digitally signed with a certificate

Android uses this certificate to identify the author

Android apps typically use self-signed certificates, in which the app developer holds the certificate's private key

BUILDING BLOCKS OF ANDROID

The essential building blocks of Android application are **app components**.

Each app component is an entity that provides a set of features with distinct purpose.

Combination of all app components makes up to the overall behavior of your app.

BUILDING BLOCKS OF ANDROID

There are four types of app components defined in Android:

- Activities
- Services
- Broadcast Receivers
- Content Providers

BUILDING BLOCKS OF ANDROID

Each app component is executed in **its own process** but it may trigger the execution of another app component within the same app or from another app.

For example, if a user touches the address of an real estate app, it starts the process for the Google Map app if it is not already running and instantiates all the classes required by the app component in Google Map.

BUILDING BLOCKS OF ANDROID

Such **remote invocation across different apps** requires specific permissions so your app can activate a component from another app.

A way to avoid dealing with the permissions is to route your app's message through Android system to another app, using the Android message routing mechanism **Intent**.