Android:

The past, present, and future

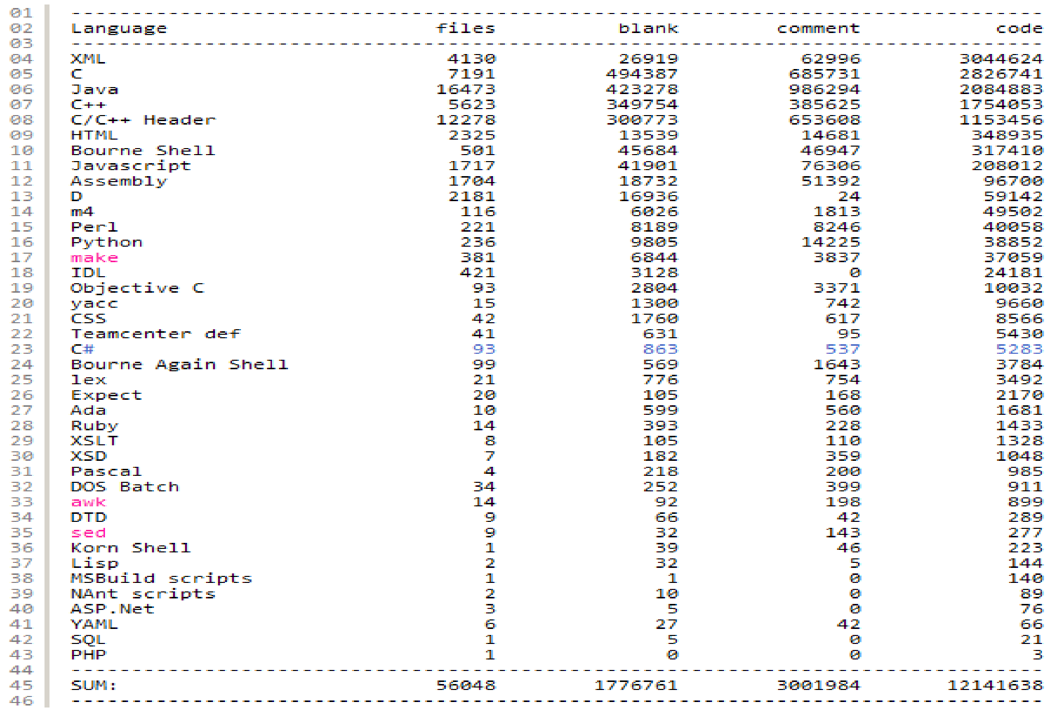
Dillon Welch

CSC 345

04/28/2011

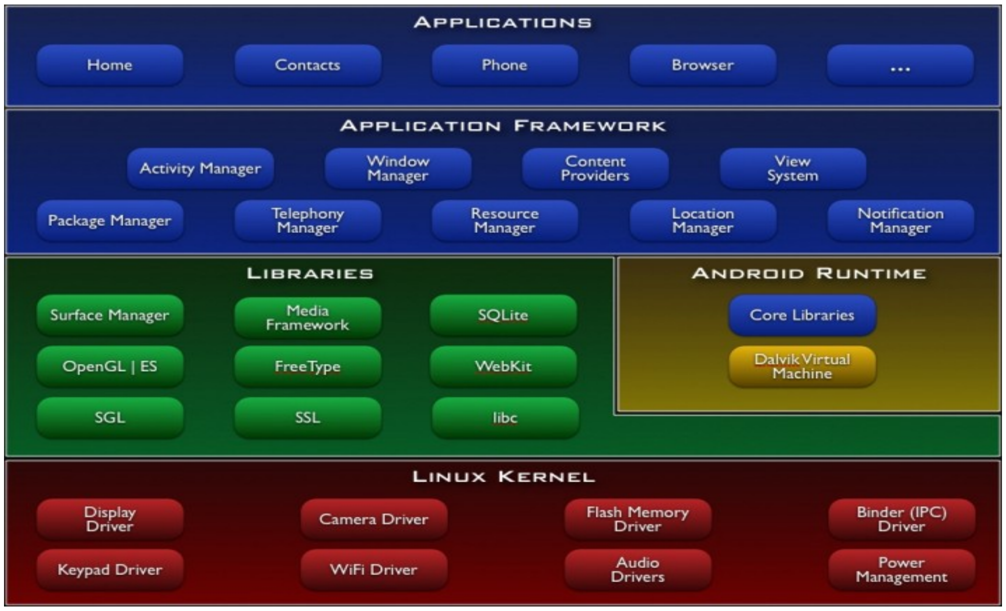
What is Android? According to the developer's website, “Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.” (Android Developers). This paper will go into detail about how Android works on the inside, and then talk about it's history, its current popularity, and where it may go in the future.

Before going into the details, a few more clarifications need to be made. A software stacks is “a set of software subsystems or components needed to deliver a fully functional solution, for example a product or service” which in this case is a working device such as a tablet PC or cell phone (“Solution Stack”). Android is also open source, meaning that (most of) the source code for the operating system is fully available to the public. Android's kernel comes from Linux but Google has made substantial changes. For example, Android does not support the full GNU libraries. More detail about this will be discussed in the history section. Android has over 12 million lines of code, with 3 million being in XML, 2.8 million in C, 2.1 million in Java, and 1.75 million in C++ along with smaller amounts of lots of other types of code [listed in the picture below] (“How many lines of code does it take to create the Android OS?”).



```
--------------------------------------------------------------------------------
Language                     files          blank        comment           code
--------------------------------------------------------------------------------
XML                           4130          26919          62996        3044624
C                             7191         494387         685731        2826741
Java                         16473         423278         986294        2084883
C++                           5623         349754         385625        1754053
C/C++ Header                 12278         300773         653608        1153456
HTML                          2325          13539          14681         348935
Bourne Shell                   501          45684          46947         317410
Javascript                    1717          41901          76306         208012
Assembly                      1704          18732          51392          96700
D                             2181          16936             24          59142
m4                             116           6026           1813          49502
Perl                           221           8189           8246          40058
Python                         236           9805          14225          38852
make                           381           6844           3837          37059
IDL                            421           3128              0          24181
Objective C                     93           2804           3371          10032
yacc                            15           1300            742           9660
CSS                             42           1760            617           8566
Teamcenter def                  41            631             95           5430
C#                              93            863            537           5283
Bourne Again Shell              99            569           1643           3784
lex                             21            776            754           3492
Expect                          20            105            168           2170
Ada                             10            599            560           1681
Ruby                            14            393            228           1433
XSLT                             8            105            110           1328
XSD                              7            182            359           1048
Pascal                           4            218            200            985
DOS Batch                       34            252            399            911
awk                             14             92            198            899
DTD                              9             66             42            289
sed                              9             32            143            277
Korn Shell                       1             39             46            223
Lisp                             2             32              5            144
MSBuild scripts                  1              1              0            140
NAnt scripts                     2             10              0             89
ASP.Net                          3              5              0             76
YAML                             6             27             42             66
SQL                              1              5              0             21
PHP                              1              0              0              3
--------------------------------------------------------------------------------
SUM:                         56048        1776761        3001984       12141638
--------------------------------------------------------------------------------
```

Android has many features, including a virtual machine (VM) optimized for mobile devices, an integrated browser based on the WebKit open source engine, a custom 2D graphics library and a 3D graphics library based on OpenGL, SQLite, common media file support, and support for hardware dependent items such as Bluetooth and cameras. It ships with a set of core applications for common smartphone tasks such as e-mail, SMS, calendars, maps, and contacts, all written in Java. Developers of apps have the same access to application framework APIs as the core applications, so components can be re-used and whole applications can be replaced by the user if the default one is unsatisfactory. Included with Android is a set of C/C++ libraries that are used by Android and able to be accessed by developers through APIs. Besides what was already mentioned, there is a BSD type implementation of the standard C system library, media libraries for common formats, and a manager for 2D and 3D graphics. Linux 2.6 is used for low level services such as security and memory managements, and acts as a layer of abstraction between the hardware and applications ("What is Android?").



The major components of the Android Architecture

Once an application is written (in Java), the SDK complies the code and any data or other resources into what is called an Android package. This is an archive file that has the suffix of .apk, and all code in an .apk is considered one application and is what is used to install the application. After an application is installed, it lives in a "security sandbox." Each application is considered a different user by Android, which assigns the application a unique user ID that is known only to the system. The system sets all file permissions such that only the user ID associated with the application can access the files. Each process runs in it's own VM in isolation from all the other running applications. The default setting is that every application runs its own process, which starts when a component needs to be executed and is shut down when the process is no longer needed or the memory is needed by other applications. This is what is called “the principle of least privilege” or that each application has access to only what is needed and no more. This allows for a very secure environment, as applications cannot access parts they do not have permission for. There are ways to allow applications to share data and access other system services though. Two applications can have the same user ID, so they can therefore access each other's files, and can be run in the same process and VM to conserve resources. At install time, an application can request permission to access resources such as SMS and contacts, and they must be granted by the user (“Application Fundamentals”).

Application components are what make up any Android application. There are four different type of components: activities, services, content providers, and broadcast receivers. First, an activity is generally a single screen with a user interface. Each activity is for a specific purpose, and an application may have several. For example, an e-mail application may list e-mails, compose e-mails, and read e-mails. Each one of these would be a separate activity. While they work together to make an application, they are independent and other applications can start the specific activity without starting the entire application. Second, a service runs in the background and is for long running applications, and does not provide a user interface. Two examples would be playing music while the user is in another application and retrieving data while doing some activity. An activity can start a service and either let it run or bind the service to itself in order to interact. Third, a content provider manages shared application data. Though a content provider, an application can access and, if allowed, modify the data in storage locations such as the file system, and SQLite database, or the Internet. An example of this would be the content provider that manages contact information. Content providers are useful for accessing data that needs to be private. Finally, a broadcast receiver responds to system-wide broadcasts and acts based on them. Broadcasts usually originate from the system, but can come from applications as well. Some examples of broadcasts would be that the screen is off, the battery is low, and that data has been downloaded and is ready for use. They do not have a user interface, but they may put notifications on the status bar to alert the user. Generally broadcast receivers are normally just there to activate other components once a broadcast is received ("Application Fundamentals").

Before a component can be started by Android, the system must see that the component exists by looking in what is called the manifest file. This is a file called AndroidManifest.xml, contains a declaration of all the components of the application, and is located at the root directory of the application. In addition to the component declarations, the manifest file contains all user permissions the application needs, a declaration of the levels of APIs that the program uses, any software or hardware used by the application, and any libraries the application needs to be linked to. Activities, services, and content providers not declared in the manifest are not visible and therefore can never be run. Broadcasts receivers however have an additional method of creation, in that they can be created dynamically as BroadcastReceiver objects and then registered with the system in the code ("Application Fundamentals").

In the manifest file, the developer can define a profile that declares the software and device requirements. The system does not read them, but the Android Market does and it filters users who do not meet the requirements. If a device is declared as used but not required, the application must do a runtime check to see that the hardware (phone, tablet, etc) has the required device and disable any unusable features. There are four main device characteristics: screen size and density, input configurations, device features, and the platform version. All programs are by default compatible with all screen sizes and densities, as Android makes the adjustment to the UI and images. Developers can create specialized layouts based on specific sizes and densities, and these are declared in the manifest. Different devices have different methods of input mechanisms, and if an application requires a specific kind it should be declared in the manifest. Different devices also have different device features, and if an application uses a specific one this should be declared in the manifest. There are many versions of Android, and each newer version generally has additional APIs and are referenced to as API levels. Any APIs added after version 1.0 that are used should also be declared in the manifest file. In addition to these software and device requirements, there are resources that an application may require such as images and menus. These should be defined with XML files, so they can be updated without modifying code and to make it easier for an application to be have different resources to be used on a greater variety of hardware (called alternate resources). For every resource included, the SDK creates a unique integer ID, which can be referenced in the application code or from other resources. Android supports qualifiers for alternate resources, which are short strings that define the device configuration conditions in which the resource would be used. For example, two different layouts can be created for when the hardware is held vertical or horizontal, and the system applies the appropriate layout based off the current orientation ("Application Fundamentals").

Components (except content providers) are activated by a message called an intent. Intents serve to connect various components at runtime. An intent can be used to activate either a specific component or a specific type of component. This second action is called an intent action, and if there are multiple components that can process the intent, the user chooses which one. This is done by looking through intent filters that can (optionally) be declared in the manifest file. For activity and service components, an intent tells them the action they will perform and if necessary the data they will need to perform that action. An activity can receive a result and then return the result as an intent. For broadcast receivers, an intent defines the broadcast to be received. Content providers are not activated by intents, but has an additional layer of abstraction with a ContentResolver object. This object handles direct communication with the content provider so the component that is communicating with the content provider instead calls appropriate methods on the ContentResolver. This is done for security reasons ("Application Fundamentals").

While Android is derived from the Linux kernel, in the words of Google engineer Patrick Brady "Android is not Linux." Android trades compatibility with Linux for internal compatibility and ease of use of applications with Android devices. Android does not support native C, so many third-party Linux applications can't be used with Android without a rewrite. Other types of programs, such as MIDP applications, can't be run due to Android's specifically tailored VM. In exchange, the universal nature of the Android system means that applications should run on nearly all devices without much further modification (Dream(sheep++): A developer's introduction to Google Android”).

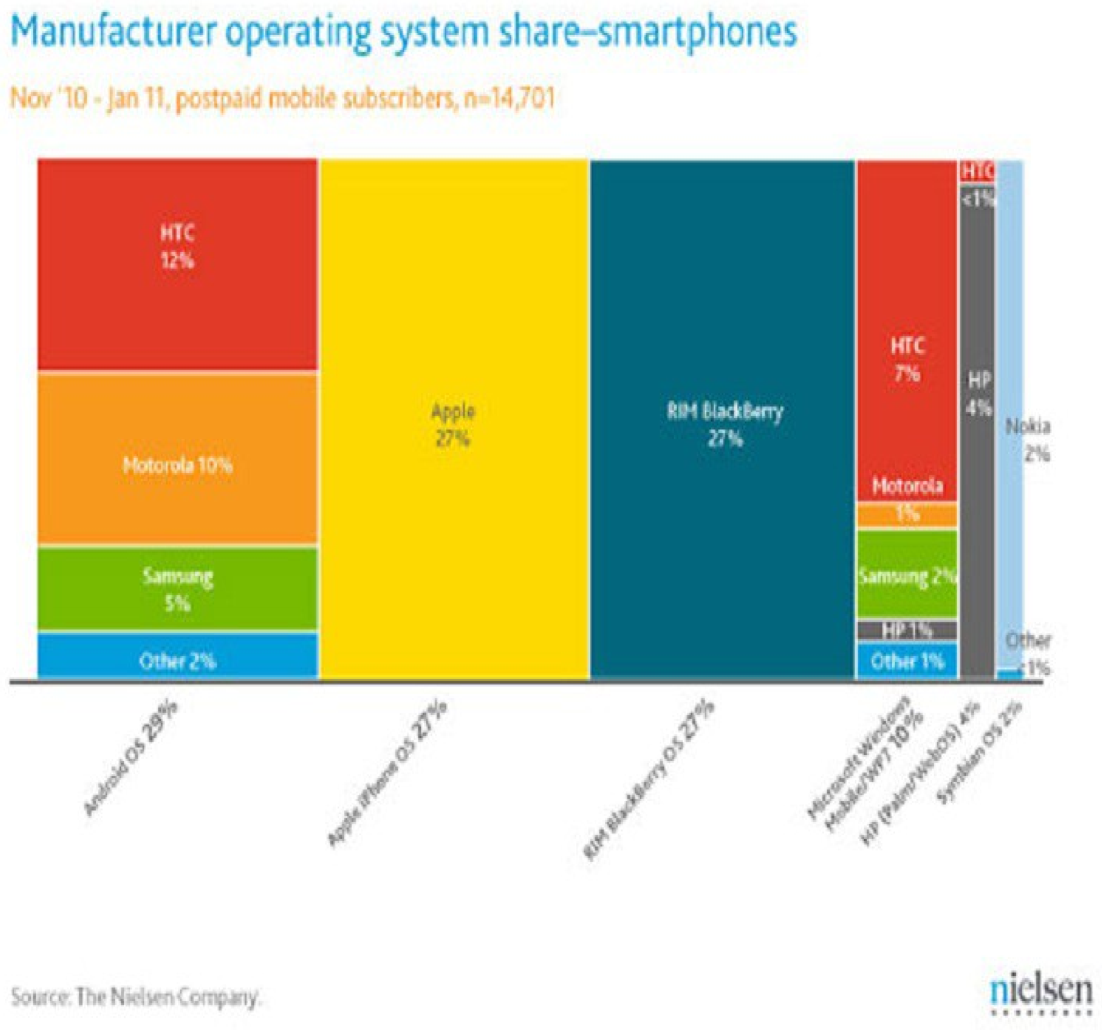
In October 2003, Android Inc. was founded by Andy Rubin. “ In a 2003 interview with *BusinessWeek*, just two months before incorporating Android, Rubin said there was tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences. 'If people are smart, that information starts getting aggregated into consumer products,' said Rubin.” (“Google Buys Android for Its Mobile Arsenal”). Previous to the creation of Android, Andy Rubin had a history of being a top engineer helping to create innovative new devices. He worked in robotics until 1989, when by chance he was offered a job at Apple Inc. after offering an Apple engineer named Bill Caswell a place to stay during a vacation in the Cayman Islands after Caswell's girlfriend had kicked him out. At Apple, he was eventually transferred to R&D, where he worked on projects such as the Quadra and an early software modem. In 1992, he joined a company Apple had spun off to work on hand-held computing called General Magic. There, he and other engineers developed Magic Cap, a “groundbreaking” operating system for hand-helds. It was ahead of its time though, and the company eventually dissolved. He bounced from idea to idea, and in 2002 he and some fellow engineers created a company called Danger Inc. to work on an idea that had grown from a digital scanning device to a smartphone. They called it the Sidekick, and unlike with the Magic Cap the idea found funding and popularity. Rubin eventually left Danger in 2004, and again bounced from idea to idea. Eventually, he got together a group of people to work on an idea he had to design an open source mobile platform, using a domain he owned for year as a lover of robots and technology called Android.com. This idea gained great popularity, and Google eventually acquired it in 2005, with Andy Rubin and others staying on (“I, Robot: The Man Behind the Google Phone”).

After this, Rubin and the team worked hard to develop what would be known as the Android OS. They eventually developed a platform based on the Linux kernel. As this was being developed, Google was marketing the idea to various hardware vendors and phone carriers based on the idea of a mobile platform that had flexibility and was easy to upgrade. *InformationWeek* reported that patents in the field of mobile telephony had been filed by Google in September 2007. A month later, on November 5th, the Open Handset Alliance announced its existence. This is a consortium of many hardware/carrier companies all with the goal of developing open standards for mobile devices. The Open Handset Alliance consisted of companies such as Broadcom, Google, HTC, and Sprint Nextel. On this day, they also announced Android as their first product. On December 9th, 2008, 14 new members joined the Open Handset Alliance. Android has been available under an open source software license since October 21st, 2008. The first phone to use Android was the G1 (also HTC Dream) and was released on October 22nd, 2008. The entire source code is published under an Apache license and can be found at http://source.android.com/. While Android is open source, the Android trademark may not be used by devices unless they pass Google's Compatibility Definition Document. They must also pass this to be able to used the closed source applications such as the Android market. Each new version of Android focuses on fixing bugs and adding new features. Generally, each has a code name based on a type of dessert, such as Cupcake, Donut, and Eclair ("Android (operating system").

As stated on the Android Open Source Project (AOSP) website, "We created Android in response to our own experiences launching mobile apps. We wanted to make sure that there would always be an open platform available for carriers, OEMs, and developers to use to make their innovative ideas a reality. We wanted to make sure that there was no central point of failure, where one industry player could restrict or control the innovations of any other. The solution we chose was an open and open-source platform. The goal of the Android Open Source Project is to create a successful real-world product that improves the mobile experience for end users." While the project is open source, the AOSP has an Android Compatibility Program, which defines what "Android Compatible" is as the standards device builders have to meet to use Android and its applications. At any given moment, there is a version of Android that is considered the latest release, viewed as a new branch in the tree of versions. In parallel, contributors work on new devices and applications for the current version, and Google works on the next version of Android in the given direction they want to go in (“Philosophy and Goals”).

Android had some early bumps in development and adoption by phone carriers, but by 2010 Android was growing in popularity. HTC's second quarter revenue increased by 58% from the year before, and 150,000 new phones were being activated every day (“HTC quarterly sales up 58% thanks to Android”). The release of the iPad allowed fans of Apple technology to have a portable Apple device without having to have an iPhone, so they could switch and test the waters of Android. Others switched as they considered the benefits of Google's open approach to development relative to the the “walled garden” of the iPhone (“Testing the Android Waters”). By early 2011, Android accounted for 29% of all active smartphones, as compared to Apple and Blackberry's share at 27% each. This data, along with the data for the other OS's, are shown in the picture on the next page. This data did not account for the addition of the iPhone to the Verizon network. Apple and Blackberry were in the lead though for creation of phones with their OS on them, as they were also creating the phones as well instead of relying on third party vendors like Android and Windows. The age group of 18-24 was more likely to use Android, and those 25-34 were evenly split between Android, iOS, and Blackberry (“Android Pulls Ahead in smartphone Race, Report Says”).

Google is continuing to push Android in new directions to make sure it stays a viable and competitive OS long into the future. Android 3.0, codenamed Honeycomb, is at the moment designed as only a tablet based OS, but with many features such as improved multitasking and notifications that many hope are ported to smartphones. The Army has decided that it wants each of its soldiers to have a smartphone to stay connected, and have initially decided to have a version of Android as the OS on this smartphone. A prototype device called the Joint Battle Command-Platform is running Android, and the development kit is being released in July. Envisioned apps include a mapping function, a tracker program for friendly forces, and critical messaging for services like medevac (“US Army picks Android to power its first smartphone”). As of April 2010, there were 82 devices that were either released or in development, and by 2011 that has probably passed 100 (“Google Android: Its history and uncertain future”).



The Android OS is an open source project owned by Google that provides a more flexible and customizable solution as compared to iOS and the iPhone. While there were some flaws in development, Android has come to surpass the iPhone in terms of popularity, and continues to expand into new markets such as tablet PCs and e-book readers. The future success of Android will depend on how innovative Google can continue to be with it and how much people can be convinced to use Android over other solutions that companies such as Apple may invent for future devices.

Works Cited

Ackerman, Spencer. "US Army Picks Android to Power Its First Smartphone." *Ars Technica*. 23 Apr. 2011. Web. 26 Apr. 2011.

"Android (operating System)." *Wikipedia*. 26 Apr. 2011. Web. 26 Apr. 2011.

"Application Fundamentals." *Android Developers*. 1 Apr. 2011. Web. 26 Apr. 2011.

Bilton, Nick. "Android Pulls Ahead in Smartphone Race, Report Says." *Technology - Bits Blog - NYTimes.com*. 3 Mar. 2011. Web. 26 Apr. 2011.

Bilton, Nick. "Testing the Android Waters." *Technology - Bits Blog - NYTimes.com*. 14 July 2010. Web. 26 Apr. 2011.

Elgin, Ben. "Google Buys Android for Its Mobile Arsenal." *BusinessWeek*. 17 Aug. 2005. Web. 26 Apr. 2011.

Geere, Duncan. "Google Android: Its History and Uncertain Future." *Wired.co.uk Future Technology News and Reviews (Wired UK)*. Wired.co.uk, 15 Apr. 2010. Web. 26 Apr. 2011.

Hardawar, Devindra. "HTC Quarterly Sales up 58% Thanks to Android." *VentureBeat*. 6 July 2010. Web. 26 Apr. 2011.

"How Many Lines of Code Does It Take to Create the Android OS?" *Gubatron.com*. 23 May 2010. Web. 26 Apr. 2011.

Markoff, John. "I, Robot: The Man Behind the Google Phone." *The New York Times*. 4 Nov. 2007. Web. 26 Apr. 2011.

Paul, Ryan. "Dream(sheep++): A Developer's Introduction to Google Android." *Ars Technica*. 23 Feb. 2009. Web. 26 Apr. 2011.

"Philosophy and Goals." *Android Open Source Project*. Web. 26 Apr. 2011.

"Solution Stack." *Wikipedia*. 12 Apr. 2011. Web. 26 Apr. 2011.

"What Is Android?" *Android Developers*. 1 Apr. 2011. Web. 26 Apr. 2011.

Wodajo, Felasfa. "Is Google Android or the iPhone the Future of Mobile Smartphones?" *KevinMD.com*. Web. 26 Apr. 2011.