A Brief History of Mobile Software Development

To understand what makes Android so compelling, we must examine how mobile development has evolved and how Android differs from competing platforms.

Way Back When

Our mobile phones keep us safe and connected. Now we roam around freely, relying on our phones not only to keep us in touch with friends, family, and coworkers, but also to tell us where to go, what to do, and how to do it.

Even the simplest events seem to involve a mobile phone these days.

Figure 1.1 Mobile phones have become a crucial shopping accessory.

Today's mobile devices aren't called smartphones for no reason. They can solve just about any problem—and we rely on them for everything these days.

Notice that half a dozen different mobile applications were used over the course of this story.

Each application was developed by a different company and had a different user interface. Some were well designed; others not so much.

Some of the applications were purchased, and others came on the phone free of charge.

From the user perspective, the experience was functional, but not terribly inspiring. From the perspective of a mobile developer, there seemed to be an opportunity to create a more seamless and powerful application that could handle all the tasks performed and more—to build a better bat trap, if you will.

Before Android, mobile developers faced many roadblocks when it came to writing applications. Building the better application, the unique application, the competing application, the hybrid application, and incorporating many common tasks, such as messaging and calling, in a familiar way was often unrealistic.

“The Brick”

The Motorola DynaTAC 8000X was the first commercially available portable cell phone.

First marketed in 1983, it was 13 × 1.75 × 3.5 inches in dimension, weighed about 2.5 pounds, and allowed you to talk for a little more than half an hour. It retailed for $3,995, plus hefty monthly service fees and per-minute charges.[1 2 3]

We called it “The Brick,” and the nickname stuck for many of those early mobile phones we alternately loved and hated.

About the size of a brick, with battery power just long enough for half a conversation, those early mobile handsets were mostly seen in the hands of traveling business execs, security personnel, and the wealthy.

First-generation mobile phones were just too expensive. The service charges alone would bankrupt the average person, especially when roaming.

Early mobile phones were not particularly full featured (although even the Motorola DynaTAC, shown in Figure 1.2, had many of the buttons we’ve come to know well, such as the SEND, END, and CLR buttons).

These early phones did little more than make and receive calls, and if you were lucky, there was a simple contacts application that wasn’t impossible to use.

The first-generation mobile phones were designed and developed by the handset man- ufacturers. Competition was fierce and trade secrets were closely guarded.

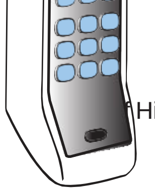

Figure 1.2 The first commercially available mobile phone: the Motorola DynaTAC.

Manufacturers didn’t want to expose the internal workings of their handsets, so they usually developed the phone software in-house.

As a developer, if you weren’t part of this inner circle, you had no opportunity to write applications for the phones.

It was during this period that we saw the first “time-waster” games begin to appear. Nokia was famous for putting the 1970s video game Snake on some of its earliest mono- chrome phones. Other manufacturers followed suit, adding games such as Pong, Tetris, and Tic-Tac-Toe.

These early devices were flawed, but they did something important—they changed the way people thought about communication. As mobile device prices dropped, batteries improved, and reception areas grew, more and more people began carrying these handy devices. Soon mobile devices were more than just a novelty.

Customers began pushing for more features and more games. But there was a problem. The handset manufacturers didn’t have the motivation or the resources to build every application users wanted.

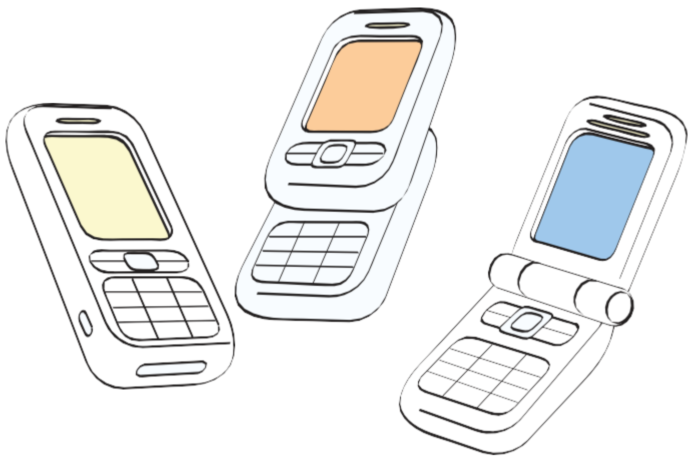

Figure 1.3 Various mobile phone form factors: the candy bar, the slider, and the clamshell.

They needed some way to provide a portal for entertainment and information services without allowing direct access to the handset.

What better way to provide these services than the Internet?

Wireless Application Protocol (WAP)

As it turned out, allowing direct phone access to the Internet didn't scale well for mobile.

By this time, professional websites were full color and chock full of text, images, and other sorts of media.

These sites relied on JavaScript, Flash, and other technologies to enhance the user experience, and they were often designed with a target resolution of 800 × 600 pixels and higher.

When the first clamshell phone, the Motorola StarTAC, was released in 1996, it merely had an LCD ten-digit segmented display. (Later models would add a dot-matrix- type display.) Meanwhile, Nokia released one of the first slider phones, the 8110—fondly referred to as "The Matrix Phone" because the phone was heavily used in films. The 8110 could display four lines of text with 13 characters per line. Figure 1.3 shows some ofthe common phone form factors.

With their postage-stamp-size low-resolution screens and limited storage and process- ing power, these phones couldn't handle the data-intensive operations required by tradi- tional Web browsers.

The bandwidth requirements for data transmission were also costlyto the user.

The Wireless Application Protocol (WAP) standard emerged to address these con- cerns. Simply put, WAP was a stripped-down version of Hypertext Transfer Protocol (HTTP), which is the backbone protocol of the World Wide Web. Unlike traditional

Web browsers, WAP browsers were designed to run within the memory and bandwidth constraints of the phone. Third-party WAP sites served up pages written in a markup lan- guage called Wireless Markup Language (WML). These pages were then displayed on the phone's WAP browser. Users navigated as they would on the Web, but the pages were much simpler in design.

The WAP solution was great for handset manufacturers. The pressure was off—they could write one WAP browser to ship with the handset and rely on developers to come up with the content users wanted.

The WAP solution was great for mobile operators. They could provide a custom WAP portal, directing their subscribers to the content they wanted to provide, and rake in the data charges associated with browsing, which were often high.

For the first time, developers had a chance to develop content for phone users, and some did so, with limited success. Few gained any traction in the consumer market be- cause the content was of limited value and the end user experience left much to be desired.

Most of the early WAP sites were extensions of popular branded websites, such as CNN.com and ESPN.com, which were looking for new ways to extend their readership. Suddenly phone users could access the news, stock market quotes, and sports scores on their phones.

Commercializing WAP applications was difficult, and there was no built-in billing mechanism. Some of the most popular commercial WAP applications that emerged dur- ing this time were simple wallpaper and ringtone catalogs that enabled users to personalize their phones for the first time.

For example, a user browsed a WAP site and requested a specific item. He filled out a simple order form with his phone number and his handset model.

It was up to the content provider to deliver an image or audio file compatible with the given phone. Payment and verification were handled through various premium- priced delivery mechanisms such as Short Message Service (SMS), Enhanced Messaging Service (EMS), Multimedia Messaging Service (MMS), and WAP Push.

WAP browsers, especially in the early days, were slow and frustrating. Typing long URLs with the numeric keypad was onerous. WAP pages were often difficult to navigate. Most WAP sites were written one time for all phones and did not account for individual phone specifications. It didn’t matter if the end user’s phone had a big color screen or a postage-stamp-size monochrome screen; the developer couldn’t tailor the user’s experience. The result was a mediocre and not very compelling experience for everyone involved.

Content providers often didn’t bother with a WAP site and instead just advertised SMS short codes on TV and in magazines. In this case, the user sent a premium SMS message with a request for a specific wallpaper or ringtone, and the content provider sent it back. Mobile operators generally liked these delivery mechanisms because they receiveda large portion of each messaging fee.

WAP fell short of commercial expectations. In some markets, such as Japan, it flour- ished, whereas in others, such as the United States, it failed to take off. Handset screens were too small for surfing. Reading a sentence fragment at a time, and then waiting seconds for the next segment to download, ruined the user experience, especially because every second of downloading was often charged to the user. Critics began tocall WAP “Wait and Pay.”

Finally, the mobile operators who provided the WAP portal (the default home page loaded when you started your WAP browser) often restricted which WAP sites were ac- cessible. The portal enabled the operator to restrict the number of sites users could browse and to funnel subscribers to the operator’s preferred content providers and exclude com- peting sites. This kind of walled garden approach further discouraged third-party develop- ers, who already faced difficulties in monetizing applications, from writing applications.

Proprietary Mobile Platforms

It came as no surprise that users wanted more—they will always want more.

Writing robust applications with WAP, such as graphics-intensive video games, was nearly impossible. The 18-to-25-year-old sweet-spot demographic—the kids with the disposable income most likely to personalize their phones with wallpapers and ringtones— looked at their portable gaming systems and asked for a device that was both a phone and a gaming device or a phone and a music player. They argued that if devices such as Ninten- do’s Game Boy could provide hours of entertainment with only five buttons, why not just add phone capabilities? Others looked to their digital cameras, Palms, BlackBerries, iPods, and even their laptops and asked the same question. The market seemed to be teetering onthe edge of device convergence.

Memory was getting cheaper, batteries were getting better, and PDAs and other em- bedded devices were beginning to run compact versions of common operating systems such as Linux and Windows. The traditional desktop application developer was suddenlya player in the embedded device market, especially with smartphone technologies such asWindows Mobile, which they found familiar.

Handset manufacturers realized that if they wanted to continue to sell traditional hand-sets, they needed to change their protectionist policies pertaining to handset design and expose their internal frameworks to some extent.

A variety of different proprietary platforms emerged for traditional handsets. Some smartphone devices ran Palm OS (later known as WebOS) and RIM BlackBerry OS. Sun Microsystems took its popular Java platform and J2ME emerged (now known as JavaMicro Edition [Java ME]). Chipset maker Qualcomm developed and licensed its Binary Runtime Environment for Wireless (BREW). Other platforms, such as Symbian OS, were developed by handset manufacturers such as Nokia, Sony Ericsson, Motorola, and Samsung. The Apple iPhone OS (OS X iPhone) joined the ranks in 2008.

Many of these platforms operate associated developer programs. These programs keep the developer communities small, vetted, and under contractual agreements on what they can and cannot do. These programs are often required, and developers even pay to participate.

Each platform has benefits and drawbacks. Of course, developers love to debate about which platform is “the best.” (Hint: It’s usually the platform we’re currently developing for.)

The truth is that no one platform has emerged victorious. Some platforms are best suited for commercializing games and making millions—if your company has brand backing.
Other platforms are more open and suitable for the hobbyist or vertical market applications. No mobile platform is best suited for all possible applications. As a result, the mobile phone market has become increasingly fragmented, with all platforms sharing parts of the pie.

For manufacturers and mobile operators, handset product lines quickly became com- plicated. Platform market penetration varies greatly by region and user demographic. In-stead of choosing just one platform, manufacturers and operators have been forced to sellphones for all the different platforms to compete in the market. We’ve even seen some handsets supporting multiple platforms. (For instance, BlackBerry 10 phones provide a runtime to support Android applications.)

The mobile developer community has become as fragmented as the market. It’s nearly impossible to keep track of all the changes. Developer specialty niches have formed. The platform development requirements vary greatly. Mobile software developers work with distinctly different programming environments, different tools, and different program- ming languages. Porting among the platforms is often costly and not straightforward.

Furthermore, application testing services, signing and certification programs, carrier rela- tionship management services, and application marketplaces have become complex spin- off businesses of their own.

It’s a nightmare for the ACME Company that wants a mobile application. Should it develop an application for BlackBerry? iPhone? Windows Phone? Everyone has a differ- ent kind of phone. ACME is forced to choose one or, worse, all of the platforms. Some platforms allow for free applications, whereas others do not. Vertical market application opportunities are limited and expensive.

As a result, many wonderful applications have not reached their desired users, and many other great ideas have not been developed at all.

The Open Handset Alliance

Enter search advertising giant Google. Now a household name, Google has shown an interest in spreading its vision, its brand, its search and ad revenue–based platform, and its suite of tools to the wireless marketplace. The company’s business model has been amaz- ingly successful on the Internet and, technically speaking, wireless isn’t that different.

Google Goes Wireless

The company’s initial forays into mobile were beset with all the problems you would expect. The freedoms Internet users enjoyed were not shared by mobile phone subscrib- ers. Internet users can choose from a wide variety of computer brands, operating systems, Internet service providers, and Web browser applications.

Nearly all Google services are free and ad driven. Many applications in the Google Labs suite directly compete with the applications available on mobile phones. The appli- cations range from simple calendars and calculators to navigation with Google Maps and

the latest tailored news from News Alerts—not to mention corporate acquisitions such as Blogger and YouTube.

When this approach didn’t yield the intended results, Google decided on a different approach—to revamp the entire system upon which wireless application development was based, hoping to provide a more open environment for users and developers: the Internet model. The Internet model allows users to choose among freeware, shareware, and paid software. This enables free-market competition among services.

Forming the Open Handset Alliance

With its user-centric, democratic design philosophies, Google has led a movement to turn the existing closely guarded wireless market into one where phone users can move be- tween carriers easily and have unfettered access to applications and services. With its vast resources, Google has taken a broad approach, examining the wireless infrastructure— from the FCC wireless spectrum policies to the handset manufacturers' requirements, application developer needs, and mobile operator desires.

Next, Google joined with other like-minded members in the wireless community and posed the following question: What would it take to build a better mobile phone?

The Open Handset Alliance (OHA) was formed in November 2007 to answer that very question. The OHA is a business alliance composed of many of the largest and most successful mobile companies on the planet. Its members include chip makers, handset manufacturers, software developers, and service providers. The entire mobile supply chain is well represented.

Andy Rubin has been credited as the father of the Android platform. His company, Android, Inc., was acquired by Google in 2005. Working together, OHA members, including Google, began developing an open-standard platform based on technology developed at Android, Inc., that would aim to alleviate the aforementioned problems hindering the mobile community. The result is the Android project.

Google's involvement in the Android project has been so extensive that who takes responsibility for the Android platform (the OHA or Google) is unclear. Google provides the initial code for the Android open-source project and provides online Android docu- mentation, tools, forums, and the Software Development Kit (SDK) for developers. Most major Android news originates from Google. The company has also hosted a number of events at conferences (Google IO, Mobile World Congress, CTIA Wireless) and the An- droid Developer Challenge (ADC), a series of contests to encourage developers to write killer Android applications—for millions of dollars in prizes to spur development on the platform. That's not to say Google is the only organization involved, but it is the driving force behind the platform.

Manufacturers: Designing Android Devices

More than half the members of the OHA are device manufacturers, such as Samsung, Motorola, Dell, Sony Ericsson, HTC, and LG, and semiconductor companies, such as Intel, Texas Instruments, ARM, NVIDIA, and Qualcomm.

Figure 1.4 A sampling of Android devices on the market today.

The first shipping Android handset—the T-Mobile G1—was developed by handset manufacturer HTC with service provided by T-Mobile. It was released in October 2008. Many other Android handsets were slated for 2009 and early 2010. The platform gained momentum relatively quickly. By the fourth quarter of 2010, Android had come to dom- inate the smartphone market, gaining ground steadily against competitive platforms suchas RIM BlackBerry, Apple iOS, and Windows Mobile.

Google normally announces Android platform statistics at its annual Google IO con- ference each year and at important events, such as financial earnings calls. As of May 2013, Android devices were being shipped to more than 130 countries, more than

million new Android devices were being activated every day, and about 900 million devices have been activated in total. The advantages of widespread manufacturer and car-rier support appear to be really paying off at this point.

Manufacturers continue to create new generations of Android devices—from phones with HD displays, to watches for managing exercise programs, to dedicated e-book read- ers, to full-featured televisions, netbooks, and almost any other “smart” device you can imagine (see Figure 1.4).

Mobile Operators: Delivering the Android Experience

After you have the devices, you have to get them out to the users. Mobile operators from North, South, and Central America as well as Europe, Asia, India, Australia, Africa, and

the Middle East have joined the OHA, ensuring a worldwide market for the Android movement. With almost half a billion subscribers alone, telephony giant China Mobile is a founding member of the alliance.

Much of Android’s success is also due to the fact that many Android handsets don’t come with the traditional smartphone price tag—quite a few are offered free with activa- tion by carriers. Competitors such as the Apple iPhone have struggled to provide competi- tive offerings at the low end of the market. For the first time, the average Jane or Joe can afford a feature-full smart device. We've heard so many people, from wait staff to grocery store clerks, say how much their lives have changed for the better after receiving their first Android phone. This phenomenon has only added to the Android's underdog status.

Manufacturers have contributed significantly to the growth of Android. In January 2013, Samsung announced that its Galaxy S line has sold more than 100 million units worldwide (*http://www.samsungmobilepress.com/2013/01/14/Samsung-GALAXY-S-Series- Surpasses-100-Million-Unit-Sales*). Within the first month after the release of the Gal-
axy S4, Samsung achieved sales of more than 10 million of the devices.

Google has also created its own Android brand known as Nexus. There are currently three devices in the Nexus line, the Nexus 4, 7, and 10, each created in partnership with the handset manufacturers LG, Asus, and Samsung, respectively. The Nexus devices pro- vide the full, authentic Android experience as Google intends. Many developers use these devices for building and testing their applications because they are the only devices in
the world that receive the latest Android operating system upgrades as they are released. If you, too, would like your applications to work on the latest Android operating system version, you should consider investing in one or more of these devices.

Apps Drive Device Sales: Developing Android Applications

When users acquire Android devices, they need those killer apps, right?

Initially, Google led the pack in developing Android applications, many of which, such as the email client and Web browser, are core features of the platform. They also de- veloped the first successful distribution platform for third-party Android applications: the Android Market, now known as the Google Play store. The Google Play store remains the primary method by which users download apps, but it is no longer the only distribu- tion mechanism for Android apps.

As of July 2013, there have been more than 50 billion application installations from the Google Play store. This takes into account only applications published through this one marketplace—not the many other applications sold individually or on other markets. These numbers also do not take into account all the Web applications that target mobile devices running the Android platform. This opens up even more application choices for Android users and more opportunities for Android developers.

The Google Play store has recently received a significant redesign, and there has been a growing effort to increase the exposure and sales of game applications. One way Googleplans to increase the distribution of games is by providing the new Google Play Game Services SDK. This SDK will allow developers to add real-time social features to games,

and application programming interfaces (APIs) for implementing leaderboards and achieve- ments to help drive new users to applications, while continuing to engage existing users.

Another reason for the Google Play redesign is to help drive sales of content. Users are always looking for new music, movies, TV shows, books, magazines, and more, and Google Play's focus on content has placed it in a position to keep up with user demand for such services.

Taking Advantage of All Android Has to Offer

Android's open platform has been embraced by much of the mobile development community—extending far beyond the members of the OHA.

As Android devices and applications have become more readily available, many other mobile operators and device manufacturers have jumped at the chance to sell Android devices to their subscribers, especially given the cost benefits compared to proprietary platforms. The open standard of the Android platform has resulted in reduced operator costs in licensing and royalties, and we are now seeing a migration to more open devices. The market has cracked wide open; new types of users are able to consider smartphones for the first time. Android is well suited to fill this demand.

The Android Marketplace: Where We Are Now

The Android marketplace continues to grow at an aggressive rate on all fronts (devices, developers, and users). Lately, the focus has been on several topics:

- **Competitive hardware and software feature upgrades:** The Android SDK developers have focused on providing APIs for features that are not available on competing platforms to move Android ahead in the market. For example, recent releases of the Android SDK have featured significant improvements such as Google Now contextual services.
- **Expansion beyond smartphones:** Tablet usage is on the rise with Android users. There are many new tablets on the market with Android Jelly Bean that come in many different sizes and form factors. Some hardware manufacturers are even using Android for gaming consoles, smart watches, and TVs, in addition to many other types of devices that require an operating system.
- **Improved user-facing features:** The Android development team has shifted its focus from feature implementation to providing user-facing usability upgrades and "chrome." They have invested heavily in creating a smoother, faster, more responsive user interface, in addition to updating their design documentation with excellent trainings with best practices for developers to follow. Those principles are centered around three goals and focused on the user experience: "Enchant me," "Simplify my life," and "Make me amazing." Following these principles should help any applica- tion increase usability.

Note

Some may wonder about various legal battles surrounding Android that appear to involve almost every industry player in the mobile market. Although most of these issues do not affect developers directly, some have—in particular, those dealing with in-app purchases. This is typical of any popular platform. We can't provide any legal advice here. What we can recommend is keeping informed on the various legal battles and hope they turn out well, not just for Android, but for all platforms they impact.

Android Platform Differences

The Android platform itself is hailed as “the first complete, open, and free mobile platform”:

- **Complete:** The designers took a comprehensive approach when they devel- oped the Android platform.They began with a secure operating system and built a robust software framework on top that allows for rich application development opportunities.
- **Open:** The Android platform is provided through open-source licensing. Develop- ers have unprecedented access to the device features when developing applications.
- **Free:** Android applications are free to develop.There are no licensing fees for de- veloping on the platform. No required membership fees. No required testing fees. No required signing or certification fees. Android applications can be distributed and commercialized in a variety of ways.There is no cost for distributing your ap- plications on your own, but to do so in the Google Play store requires registration and paying a small, one-time $25 fee. (The term *free* implies there might actually be costs for development, but they are not mandated by the platform. Costs for design-ing, developing, testing, marketing, and maintaining are not included. If you provide all of these, you may not be laying out cash, but there is a cost associated with them.The $25 developer registration fee is designed to encourage developers to create quality applications.)

Android: A Next-Generation Platform

Although Android has many innovative features not available in existing mobile plat- forms, its designers also leveraged many tried-and-true approaches that have been proven to work in the wireless world. It’s true that many of these features appear in existing proprietary platforms, but Android combines them in a free and open fashion while simultaneously addressing many of the flaws on these competing platforms.

The Android mascot is a little green robot, shown in Figure 1.5. This little guy (girl?) is often used to depict Android-related materials.

Android is the first in a new generation of mobile platforms, giving its developers a dis-tinct edge over the competition. Android’s designers examined the benefits and drawbacks

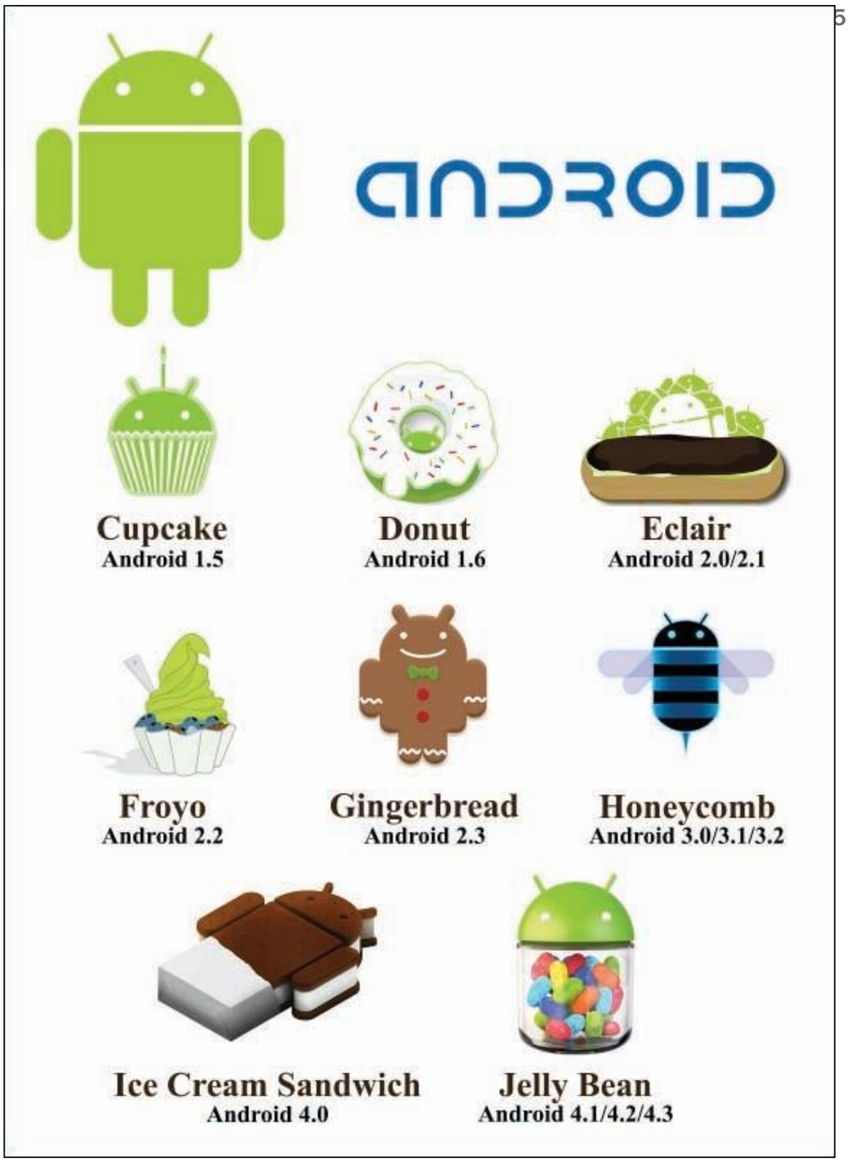

Figure 1.5 Some Android SDKs and their code names.

of existing platforms and then incorporated their most successful features. At the same time, Android's designers avoided the mistakes others made in the past.

Since the Android 1.0 SDK was released, Android platform development has contin-ued at a fast and furious pace. For quite some time, a new Android SDK came out every couple of months! In typical tech-sector jargon, each Android SDK has had a project name. In Android's case, the SDKs are named alphabetically after sweets (see Figure 1.5).

The latest version of Android is code-named Jelly Bean, and the next version of An-droid is a highly anticipated release.

Free and Open Source

Android is an open-source platform. Neither developers nor device manufacturers pay royalties or license fees to develop for the platform.

The underlying operating system of Android is licensed under GNU General Public License Version

2 (GPLv2), a strong “copyleft” license where any third-party improve- ments must continue to fall under the open-source licensing agreement terms. The Android framework is distributed under the Apache Software License (ASL/Apache2), which allows for the distribution of both open- and closed-source derivations of the source code. Platform developers (device manufacturers, especially) can choose to en- hance Android without having to provide their improvements to the open-source community. Instead, platform developers can profit from enhancements such as device- specific improvements and redistribute their work under whatever licensing they want.

Android application developers also have the ability to distribute their applications under whatever licensing scheme they prefer. They can write open-source freeware or traditional licensed applications for profit and everything in between.

Familiar and Inexpensive Development Tools

Unlike some proprietary platforms that require developer registration fees, vetting, and expensive compilers, there are no up-front costs to developing Android applications.

Freely Available Software Development Kit

The Android SDK and tools are freely available. Developers can download the Android SDK from the Android website after agreeing to the terms of the Android Software De- velopment Kit License Agreement.

Familiar Language, Familiar Development Environments

Developers have several choices when it comes to integrated development environments (IDEs). Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications. Eclipse is one of the most popular IDEs for Android development, and an Android Eclipse plugin known as the Android Developer Tools (ADT) is available for facilitating Android development. Furthermore, the Android SDK may be downloaded as a bundle that includes the Eclipse IDE preconfigured with the ADT plugin, known as the Android IDE. In addition, the Android team released an

alternative to Eclipse named Android Studio, which is based on the Community Edi- tion of IntelliJ IDEA. Android applications can be developed on the following operating systems:

- Windows XP (32-bit), Windows Vista (32-bit or 64-bit), and Windows 7 (32-bit or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (Ubuntu Linux 8.04 or later is required)

Reasonable Learning Curve for Developers

Android applications are written in a well-respected programming language: Java.

The Android application framework includes traditional programming constructs, such as threads and processes and specially designed data structures to encapsulate objects com- monly used in mobile applications. Developers can rely on familiar class libraries such as java.net and java.text. Specialty libraries for tasks such as graphics and database man- agement are implemented using well-defined open standards such as OpenGL Embedded Systems (OpenGL ES) and SQLite.

Enabling Development of Powerful Applications

In the past, device manufacturers often established special relationships with trusted third- party software developers (OEM/ODM relationships). This elite group of software de- velopers wrote native applications, such as messaging and Web browsers, that shipped on the device as part of its core feature set. To design these applications, the manufacturer would grant the developer privileged inside access and knowledge of the internal software framework and firmware.

On the Android platform, there is no distinction between native and third-party ap- plications, thus helping maintain healthy competition among application developers. All Android applications use the same APIs. Android applications have unprecedented access to the underlying hardware, allowing developers to write much more powerful applica- tions. Applications can be extended or replaced altogether.

Rich, Secure Application Integration

Recall from the bat story previously shared, where one of the authors accessed a variety of phone applications in the course of a few moments: text messaging, phone dialer, cam- era, email, picture messaging, and the browser. Each was a separate application running on the phone—some built in and some purchased. Each had its own unique user inter- face. None were truly integrated.

Not so with Android. One of the Android platform’s most compelling and innovative features is well-designed application integration. Android provides all the tools neces- sary to build a better “bat trap,” if you will, by allowing developers to write applications that seamlessly leverage core functionality such as Web browsing, mapping, contact

management, and messaging. Applications can also become content providers and share their data with each other in a secure fashion.

In the past, platforms such as Symbian have suffered from setbacks due to malware.
Android’s vigorous application security model helps protect the user and the system from malicious software, although Android devices are not immune to malware.

No Costly Obstacles for Development

Android applications require none of the costly and time-intensive testing and certifica- tion programs that other platforms such as iOS do. A one-time low-cost ($25) developer fee is required to publish applications within the Google Play store. To create Android applications, there is no cost whatsoever other than your time. All you need is a com- puter, an Android device, a good idea, and an understanding of Java.

A “Free Market” for Applications

Android developers are free to choose any kind of revenue model they want. They can develop freeware, shareware, trial-ware, ad-driven applications, and paid applications. Android was designed to fundamentally change the rules about what kind of wireless applications could be developed. In the past, developers faced many restrictions that hadlittle to do with the application functionality or features:

- Store limitations on the number of competing applications of a given type
- Store limitations on pricing, revenue models, and royalties
- Operator unwillingness to provide applications for smaller demographics

With Android, developers can write and successfully publish any kind of application they want. Developers can tailor applications to small demographics, instead of just large- scale moneymaking ones often insisted upon by mobile operators. Vertical market appli- cations can be deployed to specific, targeted users.

Because developers have a variety of application distribution mechanisms to choose from, they can pick the methods that work for them instead of being forced to play by others’ rules. Android developers can distribute their applications to users in a variety of ways:

- Google developed the Google Play store (formerly the Android Market), a ge- neric Android application store with a revenue-sharing model.The Google Play store now has a Web store for browsing and buying apps online (see Figure 1.6). Google Play also sells movies, music, and books, so your application will be in goodcompany.
- Amazon Appstore for Android launched in 2011 with a lineup of exciting Android applications using its own billing and revenue-sharing models. A unique feature of Amazon Appstore for Android is that it allows users to demo certain applications in the Web browser or from within the Amazon Appstore app before purchasing them. Amazon uses an environment similar to the emulator that runs right in the browser.

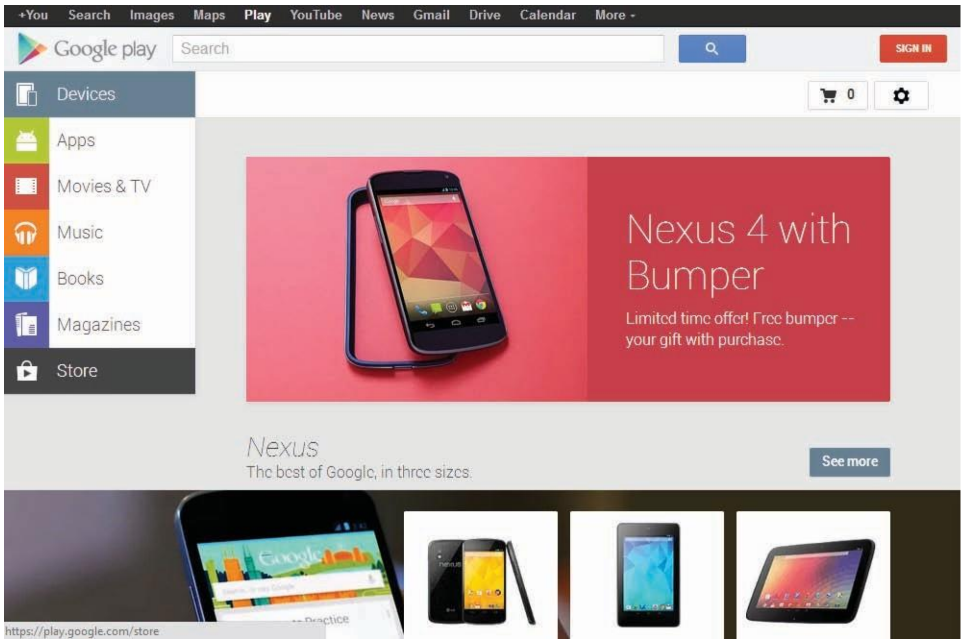
+You Search Images Maps Play YouTube News Gmail Drive Calendar More
Google play
Search
SIGN IN
Devices
Apps
Movies & TV
Music
Books
Magazines
Store
Nexus 4 with Bumper
Limited time offer! Free bumper -- your gift with purchase.
Nexus
The best of Google, in three sizes.
See more
https://play.google.com/store

Figure 1.6 The Google Play store online, showing the Devicestab.

- Numerous other third-party application stores are available. Some are for niche markets; others cater to many different mobile platforms.
- Developers can come up with their own delivery and payment mechanisms, such asdistributing from a website or within an enterprise.

Mobile operators and carriers are still free to develop their own application stores and enforce their own rules, but these will no longer be the only opportunities developers have to distribute their applications. Be sure to read any application store agreements carefully before distributing your applications on them.

A Growing Platform

Early Android developers have had to deal with the typical roadblocks associated with a new platform: frequently revised SDKs, lack of good documentation, market uncertain- ties, and mobile operators and device manufacturers that have been extremely slow in rolling out new upgrades of Android, if ever. This means that Android developers oftenneed to target several different SDK versions to reach all users. Luckily, the continuously

evolving Android development tools have made this easier than ever, and now that Android is a well-established platform, many of these issues have been ironed out. The Android forum community is lively and friendly and very supportive when it comes tohelping one another over these bumps in the road.

Each new version of the Android SDK has provided a number of substantial im- provements to the platform. In recent revisions, the Android platform has received some much-needed UI "polish," in terms of both visual appeal and performance. Popular types of devices such as tablets and Internet TVs are now fully supported by the platform, in addition to new devices such as Google Glass and smart watches. Screen sharing of An- droid devices and applications is now possible with the release of API Level 18 and a new device known as Chromecast.

Although most of these upgrades and improvements were welcome and necessary, new SDK versions often cause some upheaval within the Android developer commu- nity. A number of published applications have required retesting and resubmission to the Google Play store to conform to new SDK requirements, which are quickly rolled out to all Android devices in the field as a firmware upgrade, rendering older applications obso- lete and sometimes unusable.

Although these growing pains are expected, and most developers have endured them, it's important to remember that Android was a latecomer to the mobile marketplace compared to the RIM and iOS platforms. The Apple App Store boasts many applications, but users demand these same applications on their Android devices. Fewer developers can afford to deploy exclusively to one platform or the other—they must support all the popular ones.