Mobile application development experiences on Apple's iOS and Android OS

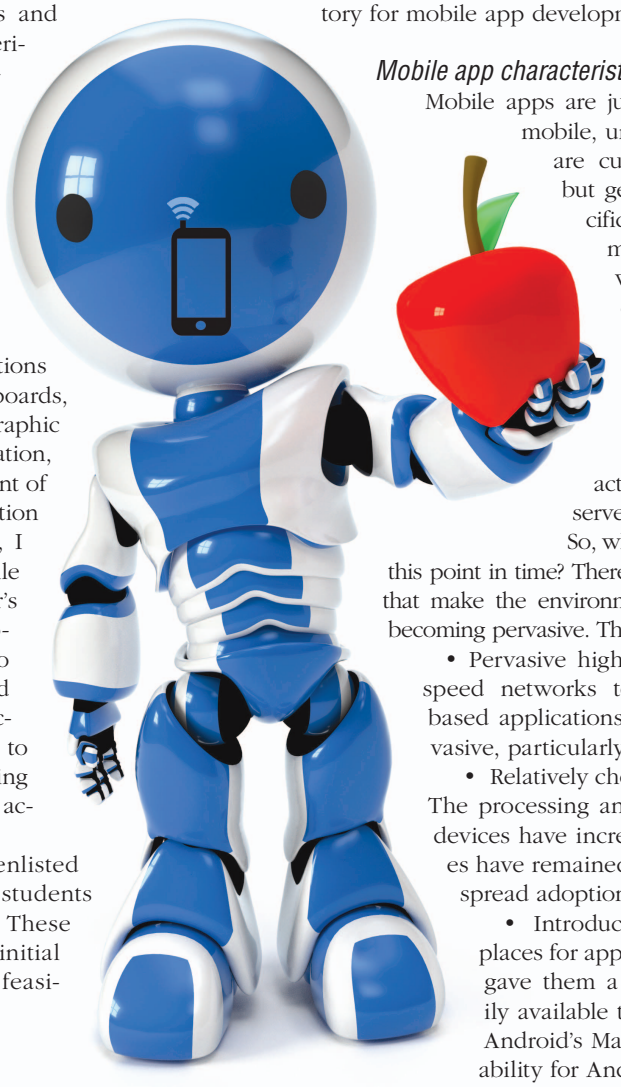
Kim W. Tracy

Mobile application deployment and use has exploded since Apple's deployment of the iPhone and the release of Google's Android operating system. The development of these applications is much easier than earlier mobile application development platforms but still carries some of the same complexities and issues. This article details recent experiences in developing an official IEEE mobile application on both Apple's iOS as well as the Android operating system. Some history of mobile application development is included. The mobile application developed starts with leveraging IEEE.tv content, but the vision is to expand the application to a more engaging application that IEEE members will enjoy and use.

This project idea came from interactions with various IEEE committees and boards, mostly related to Member and Geographic Activities (MGA). As a member organization, increasing the interaction and involvement of members is key to making the organization vital. As I also teach computer science, I was already familiar with various mobile apps and had directed several master's projects in that space. Given that, I proposed (as part of the MGA Challenge) to help IEEE in developing an app focused on member engagement. After MGA accepted the project, we refined the idea to focus on IEEE.tv initially, as it has engaging content and should be relatively easy to access with an app.

Given the acceptance of the idea, I enlisted the help of two computer science students (Adil Mezghouti and Jon Urbanski). These two students did the bulk of the initial application development to prove the feasibility of the idea.

Digital Object Identifier 10.1109/MPOT.2011.2182571
Date of publication: 27 July 2012



ROBOT © CAN STOCK PHOTO/LEO BLANCHETTE. SMART PHONE COURTESY OF STOCK.XCHNG/ LORA WILLIAMS.

The apps are currently freely available in the Apple App Store and in the Google Play Store.

Mobile app background

This section details some of the general challenges and history for mobile app development.

Mobile app characteristics

Mobile apps are just applications that run on a mobile, untethered device. Mobile apps are currently evolving very rapidly but generally are targeted at a specific task, run on a mobile device, make heavy use of the data network, and have a simple delivery mechanism (i.e., Google Play Store or Apple Store). The more interesting mobile apps tend to use your location, make heavy use of the network for content and interactions, and have a device and server component.

So, why are mobile apps taking off at this point in time? There are several convergent factors that make the environment attractive to mobile apps becoming pervasive. The major factors are as follows:

- Pervasive high-speed data networks. High-speed networks to support interesting, data-based applications have become relatively pervasive, particularly in metropolitan areas.
- Relatively cheap, high-performing devices. The processing and storage capacities of small devices have increased dramatically and devices have remained affordable enough for widespread adoption.
- Introduction of easy-to-use marketplaces for apps. Apple's success with iTunes gave them a platform to make apps easily available through the Apple App Store. Android's Market also supports easy availability for Android apps.
- Support for many third-party mobile app development. The easier it is for others to

0278-6648/12/$31.00 © 2012 IEEE

create and publish an app (as well as make money on them), the more likely a developer will create and publish the app.

- Underlying need for simple, targeted applications while mobile. A mobile user is usually doing something else (such as trying to get to a restaurant). So many mobile apps are in support of that other activity or helping to accomplish a task while mobile. Additionally, many organizations with traditional Web applications have a desire to make their applications more available and easy to use on small mobile devices where a regular Web application may be hard to use.

However, running on a mobile device inherently introduces a number of considerations that a programmer does not have to deal with on a regular application program. These include the following characteristics:

- Varied network speeds. As these devices are mobile, the wireless network that the device is using will have varying network characteristics as the device moves (and even if the device doesn't move—due to increased usage, added interference, etc.). In addition, many cellular network technologies will fall back to earlier technologies with less capacity (such as LTE to EV-DO to 1X).
- Network failure. The lack of a viable data network is more likely for mobile devices as there are always places where the network will be unavailable. In this case, the application either needs to have an offline mode or gracefully fail based on the lack of being currently attached to the network.
- Varying platform performance. Ideally, your app will be available on as many devices as possible. This directly implies supporting different devices and, usually, different platforms. For example, your app may be designed to run on Android v2.2, but each device that runs Android v2.2, may have a different CPU, different amount of memory, and different hardware architectures. Additionally, given the ability of users to run multiple applications at the same time, those processing resources may be further tapped and less predictable. Related is the fact that the underlying operating system release may vary. Given that the Android OS is open source, device manufacturers can modify it to better use their device, introducing more variation for app developers.
- Varying screen size (and resolution) and other functionality. Related to the performance concern, different devices have different sensors and abilities, with one of the most obvious being screen size. The development platforms for both Android and iOS do a good job of helping deal with those differences, but not paying attention to this issue may make a great application on one device appear clunky on another.
- Difficult to test applications fully. Given the above variation in devices, it is difficult to test all the current devices, let alone all the new ones coming out. Some of the more effective testing is done using models such as crowdsource testing such as uTest, but it is still very difficult.

Some of these challenges were faced in the current project. Depending on the platform, these will be noted in the following sections.

We did not want to cover all aspects and make the initial app too complex, so we wanted to focus on what we knew was doable.

History of mobile apps

Mobile apps are not a new idea, and various versions and platforms have been around for some time. In particular, the following is a rough timeline for the development of apps on mobile devices on cell phone wireless networks (see Table 1, roughly based on *Mobile Design and Development* by Fling).

"Candy bar" phones had a relatively small display, were often shaped like a candy bar (and about the size of a large candy bar), and had a very limited data service to use. So, applications in this era were targeted to specific phones and very rudimentary.

Wireless application protocol (WAP) based phones are able to support wireless markup language (WML), a subset of HTML, but it was still difficult to deploy these applications and the programmer had to take into account the varying screen sizes, etc., as well as developing an application in addition to the Web site.

Smart phones are the prior generation of phones that were less "app" based. That is, they didn't have quite the same delivery method as the current generation of touch screen phones. Some people continue to lump all the current generation of phones together into "smart phones." Smart phones tend to have much tighter control over the applications that are provided and are focused on messaging and personal information management (PIM) features such as calendaring and contacts.

Touch-screen phones really came into popularity due to the success of the iPhone. These phones are based on the app concept, have special markets for the apps, and are treated as a development platform. The number and kind of applications have mushroomed quickly with the release of touch screen phones with relatively large displays. Relatively lightweight tablets (iPad, Samsung Galaxy Tab, Dell Streak) are becoming widespread as well and are (for the most part) larger versions of touch screen phones. Such devices are viewed as general computing devices, and messaging and PIM features are just apps rather than core functionality. These devices also usually support Wi-Fi 802.11 networks rather than being purely designed for cellular 3G/4G networks.

Requirements and platforms

From the initial idea of developing an app to help IEEE in attracting, engaging, and retaining members, we needed specific requirements on where to start. Additionally, we did not want to cover all aspects and make the initial app too complex, so we wanted to focus on what we knew was doable.

One of the first requirements was platform selection. In terms of market share, both iOS and Android OS were continuing to increase the number of available devices and the bulk of app development was focused on these two platforms at that point in time. So we decided that we really had to support

Table 1. Mobile device application eras.

Era	Timeframe	Characteristics
Candy bar phones	1988–1998	Very limited data service to phones
Phones supporting WAP	1998–2002	WML, MMS/SMS, small screens, etc.
Smart phones	2002–present	Java-ME, Windows Mobile, Palm OS, RIM, etc.
Touch screen era	2009–present	iPhone, Android, etc.

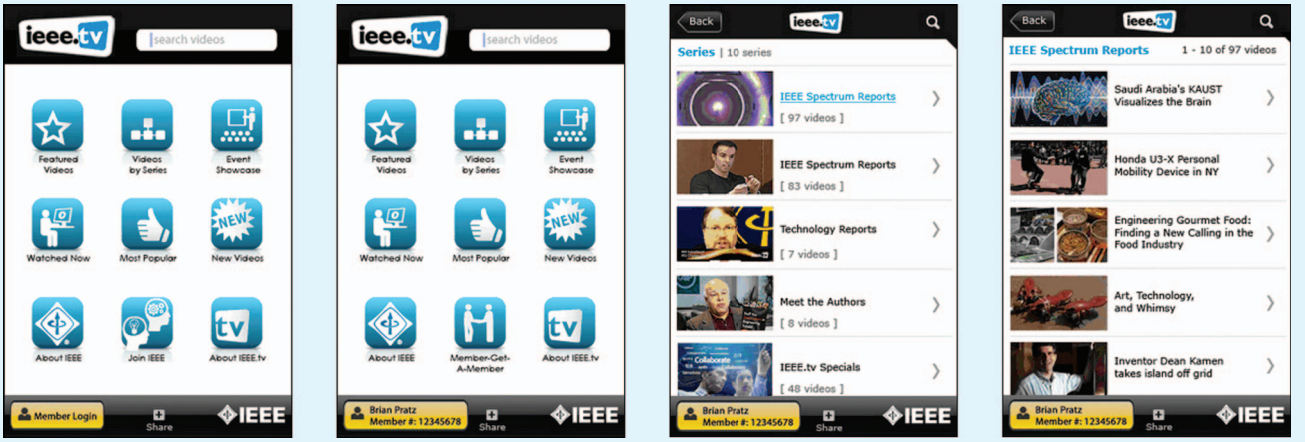


Fig. 1 Screen shots for the prototype app.

both of these platforms to have a viable app that could be widely deployed to both IEEE Members and nonmembers. Having chosen that platform, it was also key that the app have the same features and close to the same experience on both platforms. Other platforms were considered such as RIM Blackberry, Windows Mobile, and Palm but were considered less critical for the initial app.

Another key requirement was what content to focus on first. Given some of MGA's directions and input, IEEE.tv was considered to be a must-have to deliver some engaging content from the get-go. Other content that we wanted to include was an easy method for nonmembers to become IEEE Members and to leverage the Member-get-a-Member program (which gives a reward to existing members for finding and referring new members). So the focus became the delivery of content available on IEEE.tv, initially only public videos (not requiring login).

We did not want to be dependent on IEEE's IT staff, but after talking to staff there, we were able to add additional features including

- the ability to use IEEE XML to dynamically present IEEE.tv video lists
- access to IEEE Member-only content by providing a secure way to log-in using an IEEE Member's Web login (which required some development on the IEEE end)
- availability of video formats that mobile devices would support (already in the works at IEEE).

The ability to link into this IEEE IT-provided functionality not only makes the app immediately more functional, but also provides a great base for adding additional content that requires an IEEE Web login.

Figure 1 gives a snapshot of the prototype interface for the planned functionality. We focused on providing access to the lists of videos (as preexisting at IEEE.tv) and in providing a few other features such as the ability to login and view limited membership information.

Given some of MGA's directions and input, IEEE.tv was considered to be a must-have to deliver some engaging content from the get-go. Other content that we wanted to include was an easy method for nonmembers to become IEEE Members and to leverage the Member-get-a-Member program.

Android OS and iOS overviews

App development on the Android OS and iOS platform requires different approaches, programming languages, and has different methods of app publication for the different platforms.

Google android OS platform

For the Android platform, we chose to use Eclipse with its plug-ins to support the Android platform. Google and Android developers have provided a lot of documentation on the Android developer site. In order to actually publish the app to the Google Play Store, we will need to pay a small fee (US$25 at this time) and register (see https://play.google.com/apps/publish/signup) so that we can publish apps. However, to create, install on actual Android-based devices, and test, we do not need to register.

In order to install our own prototype Android apps on physical devices, we only needed to adjust a few settings on each device. These are usually under the "settings" on the device and "applications" settings category. There you can allow "unknown sources" for apps and set "USB debugging" under the "development" item. These settings can vary a bit depending on the version of the Android operating system and the manufacturer's changes.

The Android architecture is well described on the Android developer site and is designed as a specialized version of Linux for phone-type devices. It contains a virtual machine called the Dalvik Virtual Machine, which most apps use (rather than writing native applications directly against the Linux kernel—though that is possible). Most of the environment for developing apps is centered on using Java.

The Android platform grew out of development with the Open Handset Alliance and Google. Google purchased Android, Incorporated in 2005 and made the code open source in 2008. Since then, Google has released many versions of Android and continues to enhance the code base to offer more features and support for a wider array of devices (such as tablets). Google has released a number of significant versions of Android

since 2009 that have been deployed to many devices. These include: 1.5 (project name Cupcake, in 2009), 1.6 (Donut in 2009), 2.0 (Éclair in 2009), 2.2 (Froyo in 2010), 2.3 (Gingerbread in 2010), and 3.0 (Honeycomb in 2011). Our goal was to support all releases since Android 1.6, but decided to back off that goal a bit as the video player support was much enhanced in 2.0 and beyond.

Besides the Eclipse Smart Development Environment with the Android plug-in (which includes an Android simulator), there are many other tools to support development for Android-based devices. We used DroidDraw (Fig. 2), which is a free tool that helps in creating the XML for producing the interface. Other Android programming resources include Menieks et al., Meier, and Steele and To.

Apple iOS platform

The Apple iOS platform is a based on a proprietary model with the best development environment being on an Apple Mac running OS X and using the Xcode development environment. Apple maintains an extensive Web site to help developers. It also provides a nice simulator for iOS-based devices (iPhones and iPads). Additionally, to develop apps, the Objective C programming language is the most supported programming language. In order to create apps to deploy on an actual device, you need to be able to create keys for that device and application. To do that, you need to have access to Apple to develop the keys for specific devices. (Universities can get that for free with an agreement with Apple.) Similarly to Android, to publish apps to the Apple Store, you need to pay an annual fee (US$99 at this time) and register with Apple. Apple also has a more extensive process for reviewing apps before they are released on the store.

The iOS platform (which architecturally looks similar to OS X) runs with four layers: Core OS/Kernel, Core Services, Media Support, and the Cocoa Touch Interface layer (see Fig. 3). The Cocoa Touch layer provides support for using a touch-based interface. OS X supports other interface models where iOS is tuned to support touch-based devices and just support the touch-based interface.

Apple developed the iOS platform for the iPhone with the initial releases coming out with the iPhone in 2008. Later versions included support for the iPad and most recently (version 4.0 and greater) support multiprocessing. The strength of the iOS platform is that it has a relatively small number of versions, all supported by Apple. There is also a small number of different devices, again all supported by Apple. These factors reduce the complexity a great deal for the support. For another good resource for iOS development, see Stark.

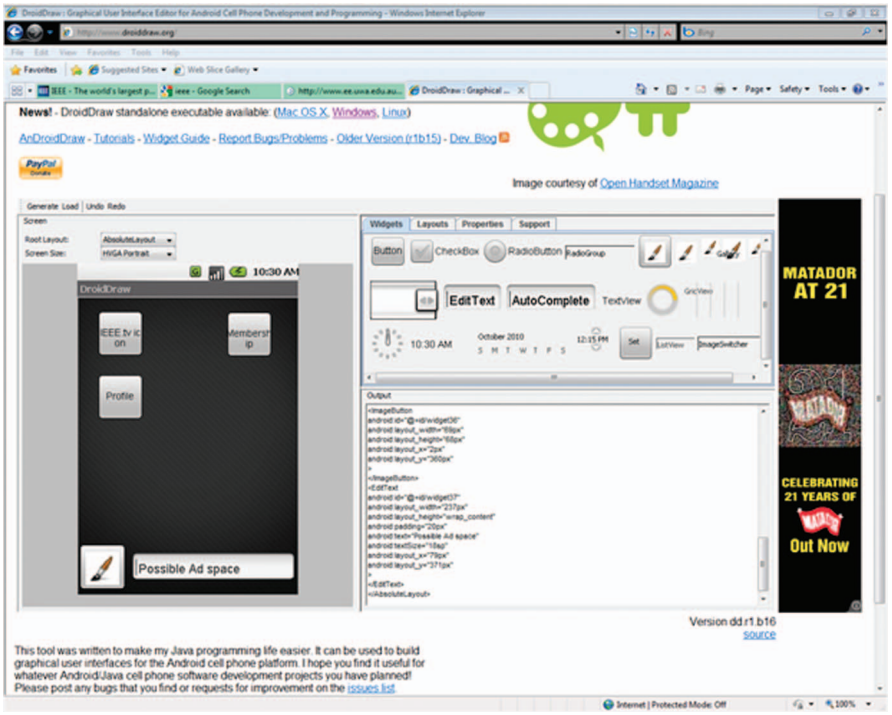


Fig. 2 Using DroidDraw to create an interface.

One thing that really helped our development was focusing on video playback, where both iOS and Android have embedded in their development environment standard ways to play video. Those existing methods for video playback helped tremendously in terms of handing some of the challenges noted in "Mobile app background" such as handling different network characteristics and having an offline mode.

Differences between iOS and android

This section highlights some of the differences that we ran across during this project. Most significantly, the development platforms are very different and require different tools and techniques. As a result, each developer had to become an expert on his/her platform and we had to communicate regularly to share details so that the apps performed similarly. Besides the development platform little differences like the default size of the icons, the format the icons had to be in, and the fact that some Android-based devices have menu buttons meant we had to have some differences in the app (i.e., what should the menu button do?). Android devices also usually have a back button, which is not on iOS devices.

Mobile app challenges

We experienced many of the challenges in the "Mobile app background"

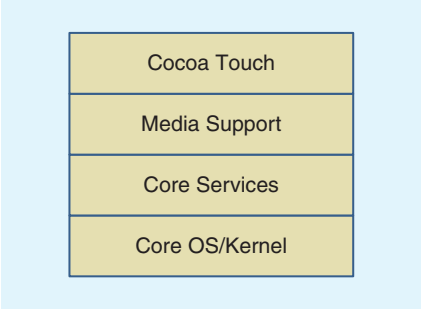


Fig. 3 iOS high-level architecture.

section above. One thing that really helped our development was focusing on video playback, where both iOS and Android have embedded in their development environment standard ways to play video. Those existing methods for video playback helped tremendously in terms of handing some of the challenges noted in "Mobile app background" such as handling different network characteristics and having an offline mode.

In order to do real development for the iOS platform, we had to get an agreement with Apple so that we could deploy the app to real devices for testing. In addition, we had to have an Apple Mac in order to do development for iOS.

Future development and predictions

Both the iOS version and the Android version of the app have been published and are available for free download. A screenshot of the final app on an iPad is given in Fig. 4, showing one of the dynamically loaded lists of videos from IEEE.tv.

From the start, this app was viewed as a starting point for continued development of an IEEE app focused on Member engagement. Other content from IEEE would be a natural extension (such as subscribed content like *IEEE Spectrum* and *IEEE Potentials* magazines), particularly for larger tablet devices. Given we've already included a method of authentication, this content could be added. Additionally, there are a number of tools focused on IEEE volunteers (vTools) that may also make sense to incorporate to help volunteers. These features may be helped by having further "hooks" put on the IEEE corporate end to allow the app to easy pull and present content.

Due to a lot of the difficulties we faced as well as the rapid changes in the mobile device market, I expect the nature of mobile apps to change over the next few years. My expectation is that most of the interesting and dynamic content will continue to be housed on enterprise servers, with applications becoming more flexible. Those applications will become more capable of flexibly leveraging the capabilities of the device and the device will dynamically download the app based on its capabilities and what the user needs to do at that point in time. Given the difficulty of developing a specialized application to run on a multitude of devices, I expect devices will become more capable of communicating their capabilities and the server side more capable of delivering the app as a result of the user request.

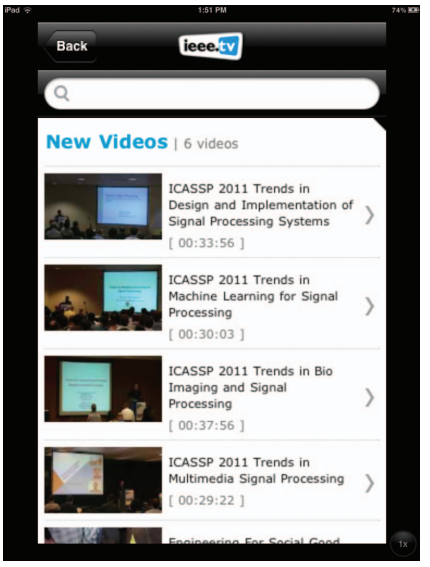


Fig. 4 Apple iPad running the published app.

I expect the nature of mobile apps to change over the next few years. My expectation is that most of the interesting and dynamic content will continue to be housed on enterprise servers, with applications becoming more flexible.

Acknowledgments

Adil Mezghouti (iPhone version) and Jon Urbanski (Android version) did the bulk of the work in getting the initial apps to work. IEEE IT staff (Brian Pratz, Susan Hutton, and Tom Smith) was extremely helpful in working on hooks in the back-office applications (and providing example code) so that features such as using the IEEE Web login would work as well as identifying some requirements that were needed.

Read more about it

- uTest Corporation. *Crowdsourcing Essentials* and *Crowdsourcing Usability*. [Online]. Available: http://www.utest.com/whitepapers
- B. Fling, *Mobile Design and Development*. Sebastaopol, CA: O'Reilly Media Inc., 2009.
- Android Open Source Project. Installing the SDK. [Online]. Available: http://developer.android.com/sdk/installing.html
- Google. Google Play Android Develop Console. [Online]. Available: http://market.android.com/publish/signup
- Z. Mednieks, L. Dornin, G. B. Meike, and M. Nakamura, *Programming Android*. Sebastopol, CA: O'Reilly Media Inc., July 26, 2011.
- R. Meier, *Professional Android 2 Application Development, Wrox* (Programmer to Programmer Series), 2nd ed. New York, NY: John Wiley and Sons, 2010.
- J. Steele and N. To, *The Android Developer's Cookbook: Building Applications with the Android SDK*. Reading, MA: Addison-Wesley, 2010.
- Apple. Apple Developer Web site. [Online]. Available: http://developer.apple.com
- Apple. App Store Review Guidelines. [Online]. Available: http://developer.apple.com/appstore/guidelines.html
- J. Stark, *Building iPhone Apps with HTML, CSS, and JavaScript*. Sebastopol, CA: O'Reilly Media Inc., 2010.

About the author

Kim W. Tracy (k.w.tracy@ieee.org) is currently at Northeastern Illinois University (NEIU) overseeing all information technology functions for the university. He also teaches computer science at NEIU and has directed more than two dozen computer science master's projects. He is a current ABET computing accreditation commissioner. He is also quite active in IEEE, including serving on the editorial board for *IEEE Potentials* (past editor-in-chief) and incoming vice chair for information management on the MGA board. He has a master's degree in computer science from Stanford University and bachelor's degrees in computer science and applied mathematics from the Missouri University of Science and Technology.