

Reasons for Root

Benefits to Manufacturers for Replaceable Firmware

by Mark L. Murphy

11 March 2010



Today, Android devices are appliances, by and large. By “appliances”, I mean they are closed units, running firmware from when the device was manufactured or via controlled update paths (OTA, signed firmware downloads, etc.). Some device manufacturers actively prevent replacement firmware. Some device manufacturers do not make it easy to replace the firmware but do not get in the way of such replacements. Few device manufacturers embrace replacement firmware.

There are any number of reasons for the current state of affairs, from culture and history to support and maintenance issues. Some developers view this ability – to replace their phone’s operating system as easily as their PCs – as an emotional or ethical issue.

In this paper, I will outline some business reasons why Android device manufacturers should allow, and perhaps support, replacement firmware. This paper, therefore, is targeted at Android device manufacturers, principally those creating today’s crop of “smartphones”, though others may find the paper of interest.

This paper splits these business reasons into two groups:

- First, we examine how supporting replacement firmware will benefit Android as a whole, as a stronger Android ecosystem will have benefits for all Android device manufacturers
- Second, we examine how supporting replacement firmware can benefit a specific manufacturer, one in competition with other manufacturers (of Android and non-Android devices)

Making Android a Success

As the saying goes, “a rising tide lifts all boats”. A stronger Android ecosystem benefits everyone in that ecosystem, including all Android device manufacturers. After all, Android is not exactly “king of the (mobile) hill”, beset on all sides by competitors that are open (Symbian, Meego) and closed (Blackberry, iPhone, WebOS, Windows Phone).

In this section, we look at ways that replacement firmware can make Android stronger in competition with these other platforms.

Expand Pool of Firmware Developers

Right now, Android firmware development expertise resides largely with the device manufacturers and select outside organizations (e.g., PacketVideo). While some intrepid developers have been working on firmware, it is still a very small niche of people.

This is bad, because Android's long-term success dictates the need for a bigger pool of people comfortable with working on the firmware.

A lot of firmware developers will be needed for **customizing Android for enterprises**. Android, at present, is a very consumer-focused operating system, and enterprises are going to want changes as part of a mass install of devices. These could range from blocking application installs (to reduce support expenses) to using encrypted partitions (for regulatory compliance, such as HIPAA). These firmware developers may be ISVs, consultants, or employees of the enterprises themselves. More importantly, they are all going to need to learn how to manipulate the firmware, meaning they need a previous generation of firmware developers to show the way. If the only devices which can be used for this are non-consumer "developer phones", fewer people will try picking up the skills, because they cannot obtain one of these devices, or they view the situation as being too limiting.

Android is not without bugs. There are over 7,000 issues on the public issue tracker¹, and while many are closed or are feature requests, bugs abound. While there are many developers, in Google and in some device manufacturers, working on fixing these things, the sheer pace of growth of the issue count suggests that we are losing ground. Android, being an open source project, holds the promise of **community-contributed bug fixes**, but that implies that people can readily become comfortable with Android firmware development and that they can run their own fixes – after all, most bug fixes are "scratching one's own itch". Right now, this will only work if the person with the itch to scratch has a "developer phone", which is not terribly common. Result: fewer contributions.

Similarly, the development community should be able to serve as a **force multiplier for feature implementation**. Of the 7,000 issues in the issue tracker, the non-bug, non-closed ones are all feature requests of one form or another. While some features can be implemented as reusable components that reside on top of Android, there are still many that have to be firmware modifications, such as continuing progress on Bluetooth APIs. As with the bug fixes, though, fewer people will bother trying to add features if they cannot use those features themselves via their own firmware.

For Android to greatly expand the number of firmware contributors and experts, more Android devices need to allow replacement firmware.

Device Upgrade Paths

Historically, cell phones, even smartphones, were not upgradeable. Just because Microsoft came out with, say, Windows Mobile 5, did not mean that users with earlier Windows Mobile phones would have an opportunity to run the newer operating system.

Along comes Apple and iPhone. Now, millions of people are used to being able to enjoy new versions of the iPhone OS, upgraded like clockwork once a year. They may not get access to every new feature without replacing their phone (e.g., video recording with the 3GS), but they also do not feel left behind and forced to keep replacing their phones to stay current.

1 <http://b.android.com>

Eventually, Apple may stumble and abandon early-generation iPhones from upgrades. Until that time comes, though, the new standard for phones is that they get upgraded.

For Android device manufacturers, particularly those with a fleet of devices and Android operating system versions, Apple's standard and Android users' dream is going to be a nightmare.

We see this already as device manufacturers struggle to get their devices upgraded to Android 2.1, courtesy of a flurry of Android releases in late 2009. And, we already see signs of device users getting extremely irritated – to the point of swearing off certain manufacturers, or Android outright – because they believe their specific device will not get upgraded. And, of course, application developers get frustrated when they have extra overhead for supporting a wide range of releases.

And all of this has occurred without any phones being officially “end-of-lived”, with a formal announcement that they will never be upgraded.

With the ability to replace firmware, some of this pressure can be relieved. While users may not be able to upgrade to newer Android releases via official channels, they will undoubtedly have options to **keep their devices current via community-baked firmware**. After all, there are a handful of such firmware releases now².

Some may worry that this will depress new device sales. After all, while the phrase “planned obsolescence” may be overstating the situation, device manufacturers certainly must hope that owners of today's devices exchange those for tomorrow's devices. However:

- Compared to alternatives (e.g., iPhone) that are obviously upgradeable, devices lacking such support will be considered weaker, depressing sales
- Devices that get replaced do not magically vanish into a puff of smoke, but rather get trickled down to people who might not otherwise have bought one of your devices, and if you want them to someday buy a new one of your devices, they need to be happy with the one they have
- The iPhone has demonstrated that it is possible to sell millions of handsets to people who already have your handset, perhaps merely a year old

With Apple and the iPhone setting expectations for continuous upgrades, for Android not to suffer by comparison, more Android devices need to allow replacement firmware.

Encroachment from More Open Platforms

When Android debuted, it was the only commercially viable open source smartphone operating system. As a result, a lot of the press coverage – encouraged, no doubt, by Google and the Open Handset Alliance – emphasized this openness and the ability for people to work with the operating system code itself.

2 <http://www.openeclair.com/>

However, times have changed since November 2007. Since then, Symbian has released their operating system as open source, and Nokia and Intel are combining other open source mobile platforms into the Meego initiative, under the auspices of the Linux Foundation.

It may be easy to write these off. After all, neither Maemo nor Moblin (the predecessors to Meego) achieved wide adoption, and Symbian application development has historically been considered painful.

However, Symbian and Meego are used by Nokia, holder of the global market share lead in phones (by some measures). Maemo allowed replacement firmware. Only when devices ship with the open source Symbian and Meego will we see how Nokia, and others, address replacement firmware on those platforms.

If Nokia ships devices supporting replacement firmware, then they will, in one fell swoop, establish Symbian and Meego as having comparable market potential as Android **while being more open**.

Already, there is a cadre of developers and device enthusiasts who view Android (and Google/OHA by extension) as being “two-faced” on open source. Yes, the operating system is open source, but it does people little good if they cannot use that source. These people are likely to jump ship to Symbian and Meego if those platforms prove equally promising and more open. After all, for those interested in openness, whether Android succeeds or fails matters not a whit if there are other strong open alternatives.

For Android’s openness meme to continue to pay dividends, and not be stolen by Symbian and Meego, more Android devices need to allow replacement firmware.

Making You a Success

While a stronger Android begets an easier time for Android device manufacturers in their struggle against non-Android competition, there is still the matter of competition within the Android ecosystem. Android is being adopted by device manufacturers across the spectrum, from the (HTC, Motorola, LG, Samsung, SonyEricsson, etc.) and small. Some Android devices will not necessarily be seen as direct competitors, such as smartphones and ebook readers. But, clearly, particularly in smartphones, Android devices are competing against other Android devices as well as the legion of other platforms.

This section examines how adopting a pro-firmware stance can help an Android device manufacturer even against Android competition.

The WRT Scenario

Linksys (now a division of Cisco) shipped a home wireless router and switch, the WRT54G, back in late 2002. In the years that followed, many enterprising developers realized that you could flash alternative Linux-based firmware, and used the WRT54G as a platform for more powerful routers, integrating VPNs, or supporting public “hotspots”, and so forth. These alternative firmware distributions – OpenWrt, DD-WRT, Tomato, etc. – became very popular. In fact, they became so popular that when Linksys later modified the WRT54G to no longer run their firmware, demand fell to the point where they elected to re-release the older model as the WRT54GL.

Not only have these alternative firmware distributions – and the hardware that runs them – valued in their own right, but they in turn become platforms for yet more innovations. How far the innovations can take you will be dependent on the power of the hardware, so we are unlikely to see any WRT54GLs taking on the role of TV set-top box, pumping out 1080p video off of Blu-Ray. But “makers” have experimented with everything from X10 home automation to simply enabling the office door to be opened via an iPhone or Android app³.

This is not only appliance platform to be similarly repurposed. Many other home routers, from the likes of Buffalo and Asus, also run OpenWrt and kin. Linksys’ NSLU2, designed as a small NAS platform, is used for everything from firewalls to streaming media servers. Multiple replacement firmwares were made for Sharp’s Zaurus line of PDAs. And so on. In each case, the replacement firmware helped sales of devices that otherwise might have been moribund.

You never know who is going to come up with an idea that can result in device sales. Device manufacturers, talented as they are, do not hold monopolies on brainpower. The benefit of what Prof. Jonathan Zittrain refers to as “generative platforms” is that others, besides the device manufacturer, can come up with and implement these ideas.

Who knows? Somebody might take an Android device and turn it into a remote medical analysis unit, with replacement firmware to enable better access to the microphone and for USB input from sensors. That sort of thing saves lives...and sells some devices. Somebody might take an Android device and create a cryptographically-secure communications node for pro-democracy activists in a totalitarian regime, with custom firmware vetted against government-supplied malware. That sort of things promotes freedom...and sells some devices. And so on.

The question is: would you rather these uses and stories be about your competitors’ devices, or yours?

Recruiting Sneezers to Your Platform

Seth Godin calls them “sneezers”. Malcom Gladwell calls them “connectors” and “mavens”. You might call them “thought leaders”. Whatever the term, there is no question that there are people who, by intent or by side-effect, promote things and drive others to use them.

³ <http://sunlightlabs.com/blog/2010/our-door-opener-science-project/>

Android as a whole has many “sneezers” and “thought leaders” on its side, from Michael Arrington (TechCrunch) to Linus Torvalds (Linux). However, a lot of “sneezing” is done on a much smaller scale, within micro-communities of specialized interest or within today’s social networks. And, a lot of those “sneezers” – the Glyn Moodys⁴ and Tim Brays⁵ of the world – are interested in Android for its openness, as much as they are for its feature set.

Hence, one strategy to help sales of a product is to specifically **promote its openness, and attract the attention of these technology “sneezers”**, no different than you might promote a high-end car or designer handbag via “sneezers” among the glitterati. That’s part of what has helped the Nexus One – when Linus Torvalds, a well-regarded proponent of open platforms, says he likes his Nexus One, that makes a difference.

Sometimes, the community of a “sneezers” is an enterprise. The decision on what handset to roll out in a major firm may start based on features, functions, and pricing, but can easily turn into an emotional decision. Those who see your device as a way to get a custom-tailored Android experience to their workforce are more likely to try to evangelize on your behalf, compared to those who view your devices as an impediment to that experience.

The Farm System

As noted, device manufacturers do not hold a monopoly on talent. They need talent around them, as consultants, as sources of new hires, and so forth.

Making open devices is one way to get that talent.

First, firmware developers working on your hardware will already have a decent understanding of what makes your devices tick.

Second, firmware developers may develop interesting features that you can use, both from an actual implementation standpoint (courtesy of open source), as an evaluation method (measuring skill level on something of value), and as a recruiting tool (“Come join us! You can work on this stuff and get paid!”).

Trend Line to Openness

Mobile devices started out as open as your average rock. They only did what the device manufacturer and carrier said they could do, period.

Then, Windows Mobile, Symbian, and Palm’s Treo started the move towards downloadable apps, perhaps ones not even authorized by carriers.

Next, Apple and the iPhone turned such installable apps from being a “nice to have” to being essential, courtesy of clever marketing (“there’s an app for that”).

Now, Android and other platforms are aiming at making the operating system itself more open.

4 <http://twitter.com/glynmoody/status/10314965123>

5 <http://twitter.com/timbray/status/10108352122>

One clear way to take advantage of a trend line is to get out in front of it. The steady progression towards open suggests that, over time, the more open the device, the more valued and desired it will be. Leaders in delivering such value get sustained benefits – witness how long the power of the Netscape brand outlasted its actual relevance.

The Way

Diving completely into “open” is probably impossible for many device manufacturers, as there are too many partial barriers in the way, such as:

- Some carriers view openness the way the rest of us view leprosy
- Some third-party components you might want to include on the device will have licensing problems that preclude allowing root access, a typical precursor to replacing firmware
- Existing devices may not readily be “open-able” after the fact

Here are some initial steps to consider:

- Make “developer phones”, designed for firmware replacement. Make them easy to be purchased in as many countries as possible, probably unlocked and without carrier subsidy. Feel free to take alternative approaches for tech support (e.g., steer developers to a **well-staffed** online support area vs. costly live support). While not ideal – limited support and no carrier subsidy will not make them popular among ordinary consumers – they can help you gain some of the benefits described in this paper.
- Allow replacement firmware on as many devices as possible, after you have declared their “end of life”. A lot of the angst about Android devices and firmware comes from uncertainty in upgrades. If you are no longer going to upgrade the phone, let others do it. Some devices may not qualify (e.g., the third-party component license issue), but so long as you are up front about that fact, it should keep anger to a minimum.
- Have a strategy for openness. Whatever you do or announce, it will be assumed to be a strategy by the public. Hence, you may as well go ahead and craft a real strategy. Develop a plan for taking advantage of openness while not adding undue support burdens. Then, communicate that strategy and stick to it. Given a reasonable strategy (some devices open immediately, some open at end-of-life, some permanently closed), reasonable developers will understand and accept it.