Android Platform Code – Turtles Most of the Way Down

Bill Weinberg | Senior Director and Analyst, Open Source Strategy | The Linux Foundation | March 8, 2011

This week Android application developers from around the world are gathering in San Francisco at AnDevCon – the Android Developers Conference. While they are soaking up tutorials on UI haptics and building apps with Ruby and HTML5, I find myself pondering the particulars of the Android platform.

A quick glance at the conference curriculum (and Gingerbread documentation) reveals Android as ever more resource-rich, with an growing repertoire of APIs and capabilities to leverage emerging hardware (like the barometer on the Motorola XOOM) and to meet developer community requirements. In providing the underpinnings for its burgeoning app portfolio (approaching 300,000 – AndroLib.com), Google and its Open Handset Alliance (OHA) partners have created an increasingly complex mobile applications platform.

A Daunting Integration Task

The underlying complexity of Android platform code can be daunting to developers, especially to software teams at chipset suppliers, device manufacturers (OEMs) and integrators. Anyone needing to integrate Android platform code with hardware and system software will be concerned about

- Managing the 165 different packages that comprise the Android GIT repository
- Tracking changes in over 80,000 source code files
- Integrating Android internal system code, device drivers, Dalvik code, middleware and applications with myriad external repositories
- Maintaining, integrating and QA-ing company-specific additions to the platform (e.g., UI customizations and Dalvik performance enhancements)
- Reconciling the rights and obligations represented in at least 19 different licenses
- Repeating this exercise every 3-4 months (hello Gingerbread and Android 3.0!)

Licensing Kumbaya

Most investments around the Android platform focus on technical and marketing issues – making an Android-based device work and delivering a compelling user experience. Every bit as important but less visible are issues like license choice and its implications.

When Google launched Android and brought together OHA, they licensed the platform under Apache 2.0. Choice of this "business friendly" license is credited with spurring unprecedented OEM adoption – there are dozens of shipping Android phone models today and over 60 new Android gadgets appeared at CES 2011 in Las Vegas alone.

Choosing Apache 2.0 as the top-level Android license has advanced Google/OHA commercial and technical goals, but beneath this neat licensing aspiration lies a messier reality:

- Android components actually reference 19 different detectable licenses as well as code of uncertain license status
- Key external components, such as Linux and Webkit, are distributed under GPLv2 and LGPL
- Over 30 Android internal components also use reciprocal licenses – GPL, LGPL, CPL, etc.
- Additional components use non-OSI approved licenses including OpenSSL and Bzip2

This complex mélange of licenses implies very specific compliance activities for each software component covered as well as more comprehensive purview and practices to understand and handle interactions among those licenses.

FYI - to address Android licensing challenges, Black Duck Software just announced a new "Android Fast Start Program" for hardware manufacturers and integrators to speed Android platform ramp up and to ease incremental re-integration with each new Android release.

To Be Continued

Tomorrow, I'll continue the discussion of hidden complexity in Android with Part 2 – *Kinks in the Android Supply Chain*.