Embedded
COMPUTING DESIGN

February 8, 2013

Eating Right at the Open Source Buffet

Resources and Best Practices for Choosing OSS Components for Embedded Development

Bill Weinberg, Senior Director, Olliance Consulting, a division of Black Duck

Open Source Software (OSS) offers intelligent systems designers a veritable smorgasbord of tools and technology. Spanning the entire software stack, from boot code and drivers to OSes, executives to middleware, and application components to development tools, OSS provides readily available alternatives to both legacy commercial software and also to in-house code developed from scratch.

But dining at the open source table is not an embedded bean feast – code gathered *à la carte* may not always integrate easily to make a well-formed "meal." While literally millions of OSS projects are available on popular forges and hubs, developers must take care to choose the right technology ingredients and tidbits to fit project and intellectual property needs.

This article presents resources and tools for discovering OSS projects and metrics for those projects. It examines factors to consider when choosing OSS projects and components for embedded designs. And it serves up heuristics for choosing the OSS technology most appropriate to real-world embedded development needs.

The IT-ification of Embedded Development

Embedded Linux, Android and BSD, and also high-level proprietary embedded OSs like QNX, LynxOS and Windows Embedded/Phone, have turned embedded systems development from a niche discipline into a branch of the larger IT space. These applications-level platforms, unlike their RTOS predecessors, easily run hundreds of thousands of ready-to-consume software components originally developed for servers and desktops, as opposed to dedicated embedded hardware.

This IT-ification of embedded design presents a good news / bad news scenario for developers. On one hand, it lets developers focus on value-added programming. On the other, the menu of options for OSs, libraries, middleware, utilities and other components can prove bewildering. How can embedded developers find, evaluate and integrate this bounty of available code?

Discovering Embedded Open Source

Finding open source software is easy. Finding the right piece of OSS can be much harder. Luckily, options for finding and evaluating OSS are plentiful, and come in five categories: search engines, hosting sites, individual project sites, dedicated OSS discovery tools, and embedded platform distributions.

Search Engines

Google, Yahoo!, Bing, Baidu and other general-purpose search engines actually do an okay job at ferreting out OSS projects. A quick search on the string "open source embedded database," for example, yields a rich mix of references and actual project sites and repositories. But while search engines are an okay starting point, using them can yield scattershot results.

Hosting Sites, Foundations

Another path is to go right to the source – the forges and hubs that host multiple projects. Until a few years ago, SourceForge would have been a developer's prime destination, with its collection of 450,000 project repositories. But today, new projects are likely to find homes on GitHub (with 2.4M unique repositories), CodePlex (32,000 projects), Google Code (10,000 projects), Gitorius, and a long tail of other sites.

Yet another type of locale for project hosting is the gamut of open source foundation forges – the Apache Foundation, CodePlex the Eclipse Foundation, and others. These sites bring together usually related bodies of code (e.g., IDE elements and plug-ins for Eclipse) and can boast several hundred hosted projects.

While repository aggregations and foundation sites are searchable by themselves, each still constitutes a distinct silo; however vast their portfolios may be, they don't cover the entire universe of open source.

Project Sites

Some projects eschew the crowded forges and build their own dedicated web sites and repositories. These may be projects of broad community interest, of greater maturity, or merely the result of technical vanity. In any case, the main challenge is still finding the project, not in the relatively limited haystack of a forge but in the larger universe of the world wide web.

Discovery Portals and Tools – The *Michelin Guide* of OSS

Probably the shortest path to finding and also evaluating open source projects lies in portals that help developers discover, track and compare open source code and the projects behind them. The most comprehensive of these free portals is Ohloh.net (owned by Olliance Consulting parent company, Black Duck), followed by Google Code Search and others. These services track the full gamut of open source software, and like the projects they monitor, they are themselves open, letting users introduce new project repositories for cataloguing and analysis.

There also exist tools to help developers discover suitable homemade open source as well as "in the wild." At companies with established policies for the use and deployment of OSS, developers can use these tools to peruse directories of vetted/approved open source code documented and/or maintained by their employers. These portfolios can also include code built and managed under the umbrella of "inner-source" and "corporate source" programs. One such tool is the Black Duck® Suite.

Embedded Platform Distributions – Prix Fixe Meals

If your organization has already committed to a pre-packaged embedded platform distribution – a commercial or community-based Linux tool kit, an Android SDK, or equivalent – then you already have a library of applications, middleware and utilities at your fingertips. Embedded distributions typically comprise 250-500 packages, with each package containing one or more unique, ready-to-use pieces of project code. Unlike downloading code directly from project sites, embedded distributions and SDKs usually include pre-built versions of project code, tested and vetted for integration compatibility across packages. In many cases, these versions may not be the latest and greatest, and you may need to turn to the original project sites to access the more current features and bug fixes. However, switching to newer versions of projects, while attractive, can break compatibility with other code in your stack, and also fall outside SLAs from commercial suppliers.

Evaluating OSS Options – Refining Your Free Software Palate

Finding potentially useful code represents only half the challenge. Developers must also vet discovered code across a variety of parameters to determine if it is technically and legally viable. Factors to consider include code size and quality, community history and dynamics, software licensing and provenance.

Code Size

Legacy embedded designs faced severe constraints on code size. While tumbling DRAM and flash memory prices have made parsimonious provisioning a concern of the past, embedded software still benefits from compact code. Memory and storage eaten up by utility and infrastructure code is unavailable for differentiating software and for end-user content.

Because OSS starts with source code, the memory footprint of a given project or software component isn't always obvious. Moreover, today's device-based software stacks can contain ingredients cooked up in traditional compiled/assembled languages (C, C++, assembly), byte-code executed Java, and scripted/interpreted languages (PHP, Python, Lua, etc.).

The sites and tools mentioned above report both the language of projects and the lines of code (LoC) in each. If your project is truly size-sensitive, the best approach is to download and build the source code to determine actual binary size (or just examine the total size of scripted/interpreted code). Figure 1. uses Ohloh reports to compare source code growth in three database projects over time.

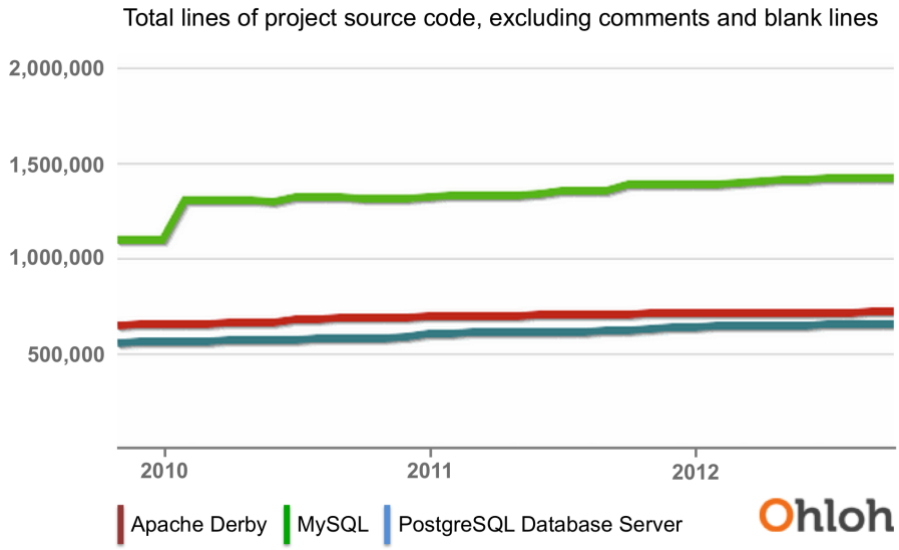

Figure 1. – Comparing Code Size (LoC) Over Time for Three Database Projects

Language

Implementation language is as important as functionality and size. If you are developing in C, projects in Java or Python probably won't integrate well into your existing or planned software stack.

Code Quality

Code quality can prove rather more difficult to gage. OSS discovery portals do report how well commented/documented OSS projects are. Other tools exist to vet the quality of code contained within a project, e.g., open source Sonar and the popular Coverity suite.

Community Dynamics

An important metric of the health and quality of open source projects lies in the size and activity of the community behind it. Some hosting sites offer historical participation metrics, and sites like Ohloh include contributor data and activity over the lifetime of a project. Figure 2. uses Ohloh reports to compare the waxing and waning of the developer community over time for three database projects.

Commit History

Tied to community dynamics is the commit history for a project – how often are changes committed to project repositories, over the project lifetime and for recent timeframes? In an immature project, change can appear to be fast and furious; for moribund projects, commits drop away to zero. Viable, stable, mature projects lie somewhere in between.

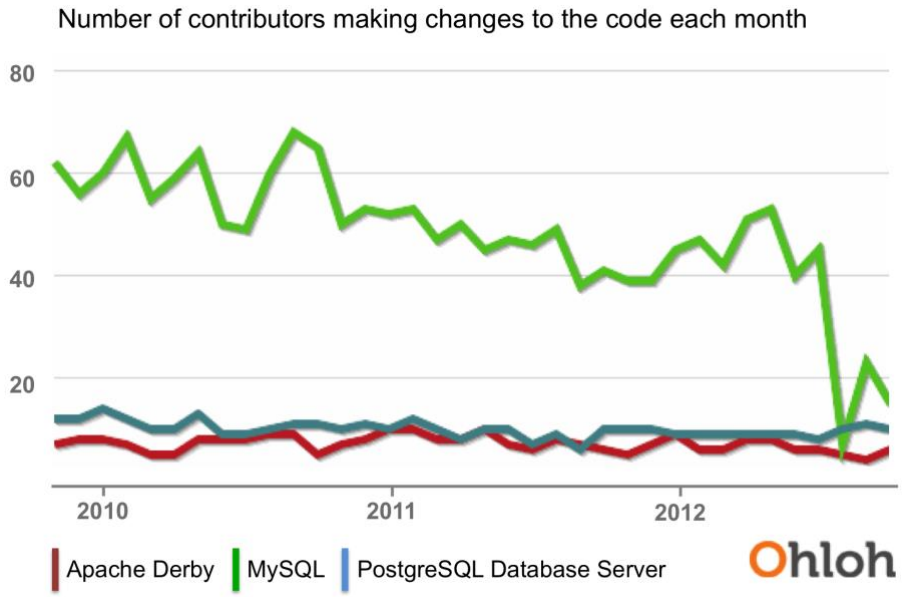

Figure 2. – Comparing Project Contributors over Time

Licensing

Dealing with the diversity of open source license types and requirements is beyond the scope of this article. Of the 2,200+ recognized licenses, you are most likely to encounter perhaps a dozen (see Black Duck's list of the top 20 open source licenses, which account for 90% of all projects). The most important open source licenses are the GNU General Public Licenses (GPL, LGPL, AGLP), the Apache License (APL), the BSD license, the Mozilla Public License (MPL), the Eclipse Public License (EPL), and a handful of others. Learn more about these and others at the Open Source Initiative, OpenSource.org.

A larger challenge lies in reconciling project licensing with your company's intellectual property rights (IPR) governance and compliance programs. A related challenge is reconciling the requirements of different licenses for diverse code integrated into a single software stack.

Provenance

Knowing the actual origins of code can also help in finding support for code, as well as protecting your company from potential legal challenges. Many useful and important projects are associated with commercial organizations that help maintain the project and provide support for it. Most projects have a unified copyright (note: the Linux kernel does not!) and many have established processes for determining provenance (e.g., certificates of origin for code submission).

Open Source, Multi-Source – Welcome to the Food Court

In your search for open source technology for your next intelligent device project, you are likely to discover multiple sources for the same code. These sources can include:

- The top-level project repository

- Prototype / experimental versions from project maintainers
- Redistribution of project code in other open source distributions
- Commercial versions of a project from support and productization suppliers

A good example lies in the MySQL database, popular in both enterprise and embedded settings (about 1/3 of MySQL deployments were embedded). Originally created, maintained and supported by the company MySQL, the MySQL code, copyright and primary commercial support channel passed through acquisition to Sun Microsystems and subsequently to Oracle. Later, the project forked into Drizzle (for cloud applications) and into MariaDB, recently spawning the MariaDB Foundation.

As a result, device OEMs can take their pick from multiple sources and support channels for this popular software component, including:

- Oracle for commercial MySQL product and support
- The Monty Program for packaging and support of MariaDB
- Linux distribution and tools suppliers for pre-packaged embeddable versions and support
- Direct access to the MySQL and MariaDB project repositories, and membership in the MariaDB Foundation

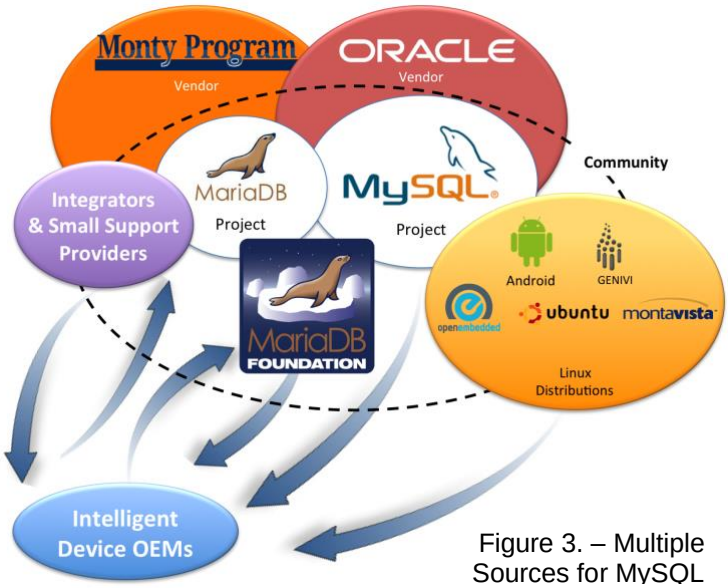

Figure 3. – Multiple Sources for MySQL

Not every OSS project or technology offers developers quite so many sourcing options, but many projects of interest to device developers do come with multiple channels for support and commercialization.

The Choosy OSS Diner

The goal of this article has been to serve code-hungry developers useful pointers for discovering, vetting and ingesting open source software. The diversity of options and the surfeit of licenses need not require a particularly adventuresome technology palate – OSS is today truly mainstream and it is a rare embedded project that does not use and/or deploy open source software tools and components.

Matching the right OSS technology to your project is less like rocket science and more like pairing wines and food. More time "tasting" OSS will teach you where to look for compatible coding languages, licenses and support options, and will, with time, impart a sense of how OSS can enhance your home-made projects.