

Developing Commercial Products for Open Source Platforms

by Lawrence Rosen*

When open source first came to broad public awareness it was often conflated with Linux, the cadillac of brand names in the open source world. When we spoke then about open source, our audiences thought only of Linux and its license, the GPL, and most of their questions were about whether it was safe to build commercial products on top of Linux.

Most people are no longer so confused. For one thing, there are now many different brand name distributions of the Linux operating system, and several other open source operating systems besides Linux—and enough other “open source platforms” to reassure developers of commercial products that the open source world is now much larger than Linux and the licensing issues are now much larger than the GPL.

Furthermore, most people now recognize that it is indeed safe to build commercial products on open source platforms. Regardless of the license, open source software doesn't reach out to “infect” other software or hardware in ways that destroy the commercial viability of those commercial products, despite early FUD to that effect. Companies can safely write applications that run on top of any operating system, virtual machine, web browser, or other open source platform, without risking their own code. Most companies already do just that.

Of course, care must be taken to follow the rules of the open source licenses that apply to the platform itself, particularly when copies of the platform are delivered to customers along with the commercial product. Free software doesn't imply that there can be a free-for-all attitude toward respecting the intellectual property embodied in open source platform software.

Drawing Lines

For many purposes, an analysis of the interaction between a commercial product and the open source platform on which it is built must start by understanding the dividing line (if any) between them. It is helpful to begin with an architectural diagram of the commercial product that clearly indicates what functions are performed by the platform and what functions are performed by the unique and perhaps proprietary software or hardware of the commercial product.

The reason for this software architectural diagram is obvious: As long as the software works are independent in the copyright sense, any product developments on the platform side of the dividing line are subject to its open source copyright license, while the commercial side of the line is yours to license as you wish.

Consider for example the boundary between a commercial web application and the open source browser in which it is designed to operate. Nobody would suggest that a simple HTML- or CGI-based web application is subject to the browser's open source license, even if they are

*Copyright © 2007 Lawrence Rosen. Licensed under the Open Software License (OSL 3.0).

Lawrence Rosen is both an attorney and a computer specialist. He is founding partner of Rosenlaw & Einschlag, a technology law firm that specializes in intellectual property protection, licensing, and business transactions for technology companies. In addition to this law practice, Larry also served for many years as general counsel and secretary of the non-profit Open Source Initiative (OSI). He currently advises many open source companies and non-profit open source projects including the Apache Software Foundation. Larry's book, *Open Source Licensing: Software Freedom and Intellectual Property Law*, was published by Prentice Hall in 2004. He taught at Stanford Law School in 2005-2006.

both distributed to consumers on the same disk. But the analysis may be different in a commercial application where a particular browser plug-in is downloaded and integrated into the browser, or where the browser itself is modified to support a commercial product and is distributed along with the product. In those latter cases, it might be more difficult to argue that the commercial application—or at least those portions of the commercial product where the open source platform is most directly modified or enhanced—is distinct from the platform on which it operates.

An architectural diagram will help identify the extent to which the platform is independent of the commercial product for copyright licensing purposes.

The application interfaces of open source platforms are usually defined by published specifications, sometimes even industry standards, for interoperation with commercial products. When a commercial product interacts with an open source platform through those standard interfaces, either exchanging data or linking through standard techniques, there is no confusion about the boundary line between the two. The intellectual property of the underlying open source platform is being used for its intended purpose and remains under its license; the intellectual property of the overlying commercial product is separate and distinct, and can be sold or given away under its own license.

Attorneys recognize that the drawing of such boundary lines between interacting software is often more a matter of technical and legal judgment rather than scientific precision. Some platform authors, recognizing the difficulties, have explicated their own licenses so as to reassure their customers on this point. Linus Torvalds, for example, has written many times about his interpretation of the GPL, under which independent applications written to standard Linux operating system interfaces are not themselves subject to the GPL. And Sun now distributes much of its Java platform under the “GPL+Classpath Exception” license that expressly provides that applications that link to the platform in standard ways may remain proprietary.

I am personally not a fan of the “licenses plus authors’ exceptions” method of explaining licenses, but if the platform software you need is available only under such a license, you don’t always have a choice. Be careful, however, that the exceptions apply to your version of the open source platform, and that the lines you draw between your application and the platform leaves your proprietary commercial software on the correct side of the exception. To be candid, my preference for clarity is my own Open Software License (OSL 3.0), which is now also available in a Non-Profit OSL 3.0 version; this license unambiguously allows you to combine OSL 3.0 components with your own software without any obligation to disclose your own independent software.

Several open source licenses, including the Mozilla Public License and others similar to it, are file-based. For those licenses, be careful not to include your commercial software in the same file as your platform software, even for distributing it, because the open source license may then apply to the entire file.

Some platform software, fortunately, is distributed under simpler open source licenses, where line drawing can comfortably be less precise. All software from the Apache Software Foundation, for example, including their Java and web server platforms, is distributed under the Apache License 2.0. That license authorizes combinations of Apache software with proprietary

software to be distributed under any license of your choice. So, too, platforms (or platform components) licensed under BSD-style licenses, or the Academic Free License (AFL 3.0), can be safely combined with proprietary products wherever you draw the architectural dividing lines.

The Ethics of License Reciprocity

There are two fundamentally different philosophies of open source licensing. The first, characterized by the BSD license, offers free software with almost no obligations of reciprocity; most important, you needn't give back the source code of your software improvements to the community. The second model, championed by the Free Software Foundation, is based on a reciprocal set of obligations they call "copyleft," which require you to return your changed software to the world for free use by others.

Most open source licenses nowadays, however, including even the very generous and intentionally non-copyleft Apache License 2.0, impose at least a few conditions of reciprocity. Your commercial product must reciprocally satisfy the license conditions for the platform on which your commercial product runs, or it may be in breach of the open source license.

Rather than speak of breach, however, it may be more helpful to explain these reciprocity obligations in terms of the ethical norms of the open source community. These norms aren't often enforced in litigation, but they are effectively enforced nonetheless through means of public and private pressure within the community upon those who appear to be gaining selfish commercial advantage by avoiding their obligations.

- Most important for some authors of open source software is attribution. They expect to see notice of their contribution, perhaps in the form of a copyright notice in product documentation or an ABOUT box, or more likely in any source code that licensees are required to distribute. Some companies now satisfy this obligation by posting information about the open source components of their products in pages on their own websites, satisfying at once the authors' need for attribution and their own internal requirements for product manufacturing documentation. Beyond that, however, some open source licenses now require more substantial acknowledgement of the authors of the open source platform you use and distribute, so read the license terms carefully. Generally, the more open you are to giving attribution credit where credit is deserved, the more ethical you will be perceived to be in the community.
- The authors of open source software may not always demand reciprocal disclosure of your source code, but they always appreciate it as a sign of good will and generosity. After drawing the architectural lines between the open source platform and your commercial products, consider how much of what you own on the commercial side of the line might reasonably be contributed back to open source anyway. Building your commercial products on top of open source platforms to which you contribute also helps share the burden of supporting and enhancing that platform to everyone's benefit.
- Many open source licenses nowadays impose patent reciprocity. Their goal, expressed in legal language that ranges from safely narrow to dangerously broad, is

to create a zone free of patent infringement litigation where open source software can thrive. You need to read the defensive termination provisions carefully, recognizing that they are designed to protect their authors. Open source licenses may in effect impose reciprocal promises that you won't sue the author of the open source software for patent infringement and that you will reciprocally license some of your own patent claims if they are necessary for the open source platform. Reciprocal patent peace is a valuable benefit of open source software, but the cost may be your ability to enforce some of your own patents against other companies. Understand not only the line between your commercial product and the open source platform, but acknowledge that you'll probably not be able to sue open source authors for patent infringement even if your patent claims read on their software on the platform side of your commercial products.

- Open source software implies a shared risk. Such software is usually distributed without warranties, without indemnity in the event of infringement lawsuits, and without any promises whatsoever that the software will perform or will continue to be supported. Even though open source software may be free, expect to pay for its continued maintenance and support. Engage in the development community even when you are not required to do so. Where appropriate and available, negotiate commercial licenses for open source software that include the risk mitigation strategies your products demand, or that relieve you from reciprocity obligations that your business model cannot live with. It is ethical in most parts of the open source community (although not in all quarters) that you be able to buy your way out of reciprocity and other risks, as long as you are not perceived to be a free rider on someone else's free software.

Expressing Open Source Values in Corporate Policies and Practices

Most software companies remain cathedral-style software development organizations. Open source development, on the other hand, is very different. For example, it always warms me to witness high-quality software being created under the meritocratic and democratic "rules" of the Apache Software Foundation. Members of ASF, although often the employees of companies with differing commercial interests in creating and supporting web- and Java-based open source platforms for their use with commercial products, are with respect to ASF expected to work toward a common vision of their software rather than their own companies' private needs.

The result, remarkably often, is high-quality software that can be a platform for exceptional but competing commercial products in many fields of use.

One challenge you will face in developing commercial products on open source platforms is to accommodate to the new development philosophies and values of open source in appropriate places in your corporate policies and practices. Expect that your engineers will participate in open source platform development even as they build your commercial products. Encourage it.

The architectural diagrams you draw during your initial product analyses are inevitably going to become blurred over time. The interactions of your employees with the open source world, the incorporation of additional open source software into your commercial products, and

the opportunities for your own contributions to open source platforms, can overtake your original plans. You may need to move your architectural lines—the lines that separate your commercial software from the open source commons—to accommodate your evolving open source interactions.

I suggest that managing at least the legal aspects of open source is in many ways far less complicated than managing the commercial interactions of hundreds of different proprietary software licenses, each probably a long and complex legal document with arcane definitions and provisions. Unlike proprietary software, the vast majority of open source software is available under a dozen or fewer licenses, each a few pages in length. Establishing clear rules for acceptable and unacceptable licenses based upon your company's commercial product objectives can be done in a straightforward way by merely reading and understanding a few licenses.

Making those licensing policies clearly known to your engineers will solve most licensing issues before they become problems.

I am occasionally bemused by companies that believe that they are small enough or well-enough managed to know what every engineer is doing and where he or she is sharing code. Expect, however, that there will be a somewhat free-flowing exchange of software by your employees to and from the vast library of open source software already available worldwide. The open source community encourages it, and engineers learn how to do it in school and on-the-job. In fact, as I have already suggested, such exchanges might be a good idea, resulting in better products than you would be able to create behind your own four walls. At the very least, tell your employees what code sharing policies and practices are acceptable. If participation in open source projects requires management pre-approval, try to find every reasonable opportunity to approve.

While published policies are essential, it is also valuable to follow a signature rule of the Cold War, “trust, but verify” ["doveriai, no proveriai" in Russian]. Specifically when developing software on the commercial product side of the line, verify that your product contains only open source components you've authorized under licenses you've approved. There are commercial software tools (such as Black Duck and Palomida) to help you with this, but an intelligent analysis by an open source specialist will help you to conform your software to your ethical and business policies.

Conclusions

The old models of commercial software product development are no longer sustainable. No company can afford today to start with a blank slate for its software-based products, and more and more companies are discovering that it is far cheaper to build on top of low-cost open source platforms than on proprietary ones.

Among the costs of that free open source software, however, are: (1) diligent analysis to understand how your commercial products interact with the open source platform you use; (2) commitment to reciprocate whenever it is ethical or required by license to do so; and (3) effective policies and practices that allow your company to both take from and give back to the open source software commons.