Exploring Open Source Options

The RedMonk Going Open Source Series, Part 1

Michael Coté

James Governor

Stephen O'Grady

Overview

Now more than ever, one of the top questions for a software company is: to what degree should we participate in open source? Open source is indisputably the most significant trend affecting the software world at the current time. More than any other single approach, strategy, or tactic, open source has compelled vendors once thought to be unassailable in their respective markets - such as IBM, Microsoft and Oracle – to significantly alter their strategies.

While substantial attention is paid to the source aspects of the code –the fact that it is visible, and in some cases alterable and distributable – the distribution advantages may be the most transformative, at least in the commercial world. From application servers to operating systems to programming languages to relational databases to web servers, very few categories of infrastructure software have been immune to open source competition. The packaged application market appears to be the next archipelago in the path of the tsunami, and vertical applications are undoubtedly close behind.

None of this is to say that open source is necessarily the single path forward for software vendors; while that may or may not be true in the distant future, it is being actively refuted by the continuing ability of vendors such as Microsoft and Oracle to sell products that are less freely available and higher priced than their open source competition. In the case of many, if not most, of these established vendors, however, success rests more on the basis of significant market barriers to entry than on pure technical grounds. Despite these advantages, entrenched businesses are finding themselves significantly threatened by open source.

If entrenched and established business are struggling with open source, then, it is reasonable to ask whether or not it's possible for projects or products with non-specialized aims to compete effectively without being open source. More specifically, can would-be platforms compete with both free and open source alternatives on the one hand, and monopoly-fueled closed source alternatives on the other side – despite significant technical differentiation?

Open Source Definitions and Trends

Thus far, open source has been referred to as a singular entity, as if it were a consistent and simply defined set of behaviors. As even those who follow the space cursorily will know, however, this belies the complexity of the space considerably. There are a great many licenses, models, governance structures, and so on within the greater open source superset. Each of these elements has profound implications for the type and size of the resulting community, the market penetration and distribution, the ability to recombine with other open source projects, the resistance to unintended or anticipated forks, and more. Although we cannot review all of the possible permutations within the confines of this report, we describe below some common models and approaches.

Open Source Definitions

Open Source Licensing

The Open Source Initiative (OSI) is considered the official arbiter of which licenses can and cannot be called "open source." To bear the term "open source," a given license must be approved by the OSI and meet their conditions. Importantly, the OSI has a stated preference for vendors selecting from one of the 50-plus currently approved licenses, as it sees little benefit to users, developers, and vendors from the license proliferation that occurs when individual vendors construct "vanity" licenses.

Liberal/Permissive Licensing

Liberal or permissive licenses impose few restrictions on the usage or consumption of the governed codebase. Permissively licensed code can be easily recombined and consumed in a variety of contexts, both closed and open source, commercial and free software. Microsoft Windows, as an example, includes networking code governed by a permissive BSD license. Common examples of this license style are the Apache (various Apache projects, e.g., Tomcat), the aforementioned BSD (BSD), and MIT (Mono) licenses.

Reciprocal Licensing

Reciprocal licensing can be seen in one respect as the antithesis of permissive licensing. The canonical example of a reciprocal license is the GPL (Linux, MySQL). While a permissive license allows for essentially the free usage and recombination of the code it governs, reciprocally licensed code typically requires that any derivative works (definitions of which vary) be issued under the same terms and license as the original codebase. Thus if you combine your code with a GPL codebase, the resulting work must carry the GPL license if it's to be distributed. The proponents of the license call this premise "copyleft," as it acts to preserve the initial rights implied. Detractors of the GPL often label its reciprocal nature "viral."

Reciprocal licenses do offer one important advantage to commercial vendors: they act to inhibit forks, both "free" and commercial. They do not prevent them, as Oracle's release of Red Hat's code proves, but they do inhibit them by requiring that any individual or group seeking to fork the code make available all of their modifications and updates under the same terms.

Vanity Licensing

A vanity license – which may or may not be considered open source as defined by the OSI – is uniquely designed and constructed by a single corporate or other entity. Traditionally, these licenses are named for the originating corporation or organization, hence the "vanity" designation. These licenses, which once were common, are considered problematic today because they impose in aggregate an unacceptable burden on end-user legal counsel and users of open source.

Open Source Trends

Attribution License Approach

One increasingly common tactic among commercial open source vendors is to release open source code under a modified OSI approved license (usually an MPL derivative) that includes an attribution provision. The license essentially requires those who would deploy the code to display a vendor trademarked badge or image in the resulting application. Examples of vendors using this approach are Scalix, SugarCRM, and Yahoo!'s Zimbra. The justification for this approach from a vendor perspective is that it protects OEM sales by requiring those who would prefer to run the software without the required badge to secure a commercial relationship with the vendor. The downside is that this approach is highly controversial, and the OSI is strongly opposed to the practice. Thus the benefit is not achieved without cost, as the negative PR that can follow this approach may be damaging.

Dual (or Tri/*)Licensing

Dual licensing is not a license, but a licensing model that describes the practice of licensing code under more than one license. Usually dual licensing refers to some combination of an open source license and a private commercial license. This is the fundamental business tactic that vendors such as MySQL exploit; it allows for the free distribution of the code in question, but under a license that is unlikely to prove acceptable to a variety of commercial partners, who can then engage with MySQL for a private commercial license.

It can also refer to projects, such as Mozilla Firefox, that multi-license their code to ensure that the widest possible audience is able to recombine, build off of, and distribute their software.

Dual Licensing is a valuable tactic for vendors who are open sourcing previously closed source software because it allows existing customers to maintain the same licensing terms. Not all existing customers will want to switch to a new, open source licensing scheme.

Exceptions Licensing Approach

Many projects use licenses with exception provisions. Exceptions may be employed to preserve or protect an existing model, work around technology encumbrances, and so on. Alfresco, as an example, licenses its content management software under the GPL with a FLOSS (Free/Libre Open Source Software) exception. This is used to allow open source projects that may be using licenses incompatible with the GPL to embrace Alfresco's software. Sun, on the other hand, has licensed its Java asset under the GPL with a Classpath exception. This permits non-GPL licensed code to use and distribute the GPL licensed Java libraries.

Hybrid Open Source Approaches

Hybrid open source typically blends free and open source components with closed or proprietary components. If anything, hybrid open source approaches are generally considered more controversial than attribution licensed software. While the majority of vendors that embrace this approach do offer free and open source versions of their products, open source advocates decry the practice, as the free versions are often feature-limited. A few examples of this approach at work:

- **EnterpriseDB** - EnterpriseDB contributes to and supports the PostgreSQL relational database project, which is a fully functioning alternative to open source competitors such as MySQL and commercial alternatives such as DB2, Oracle, and SQL Server. In addition to contributing to the project, EnterpriseDB offers commercial-level support for the free open source database. Its flagship product, however, is EnterpriseDB - an Oracle compatible database that includes PostgreSQL at its core with proprietary, non-open source components layered on top.

- **MySQL** - Until the fall of 2006, MySQL had always maintained a single MySQL database offering. In October of 2006, however, MySQL announced that its single codebase was being divided into two separate offerings, MySQL Community and MySQL Enterprise. Although the two editions share a common database codebase, the MySQL Enterprise edition includes some closed source, database management software that is not freely distributable.
- **SugarCRM** - SugarCRM is perhaps the most notable purveyor of packaged applications that follow the hybrid open source approach. SugarCRM produces two distinct offerings – an open source version and a non-open source version. John Roberts, SugarCRM CEO, has described the ratio of open source to non-open source as approximately 80 percent open, 20 percent closed. In addition to pursuing this hybrid strategy, SugarCRM is notable for having pioneered the attribution license strategy.

Patent Handling

Despite widespread criticism of the handling of software patents in the United States, they remain an important concern when selecting an open source license. Older licenses such as the GPLv2 make little provision for the handling of patents, but more recent licenses such as the CDDL and the GPLv3 are more sophisticated, due in large part to the increasing involvement of commercial entities in open source projects.

Platform Licensing

Of particular interest in this case are the strategies employed by platforms. Although the GPL remains the overwhelming license of choice for applications in general, and is also the choice of Linux – probably the most popular open source platform – platform technologies trend towards permissive rather than reciprocal style licenses. The BSD distributions are one obvious example; the licensing for PHP – perhaps the most ubiquitous dynamic language at the current time –is another.

At one time, PHP was dual-licensed under the GPL and the PHP license, but dropped the GPL as of PHP version 4 because of the reciprocal restrictions. Python's license is similarly permissive, as it employs its own custom BSD style license. Additionally, as previously mentioned, Mozilla's Firefox was tri-licensed specifically to address the limitations imposed by its original reciprocal-style license.

Generally speaking, platform technologies prefer permissive licenses because they impose the fewest restrictions on users and developers, thus offering significant advantages if ubiquity and adoption are important goals. These advantages, however, come with a price: permissively licensed technologies can be easily and legally forked, or incorporated into proprietary code, or repurposed. The lack of restrictions is both its biggest strength and biggest weakness.

For a commercial entity, then, permissive licensing is best applied when the vendor wants to grow the market around the platform, monetizing other parts of the market rather than the core platform. For example, the platform may be "free" but tools to interact with and create "content" in the resulting ecosystem may cost. Adobe follows this model, in part, by open sourcing the Flex SDK, but charging for the commercial FlexBuilder and other tools.

Open Source Licensing Matrix

The matrix below compares 4 key features of 4 popular open source licenses.

These features are:

Sublicensable - when combined with other code, can the resulting combination of code be provided under a different license? –There may still be stipulations, such as attribution, but the license is not reciprocal.

Reach - to what level does the license "have power to make demands" outside of its own code base?

Patent Retaliation - what are the possible consequences of patent related litigation?

Patent license - does the license make stipulations about patents and their use?

	Apache	BSD	CDDL/MPL	GPLv2	GPLv3
Sublicensable	Yes	Yes	Yes	No	No
Reach	N/A	N/A	File	Project	Project
Patent Retaliation	Termination	None	Termination	None	Abdication of Rights
Patent license?	Yes	No	Yes	No	(Yes)

About the Creative Commons License

About RedMonk

RedMonk is a research and advisory services firm that assists enterprises, vendors, systems integrators and corporate finance analysts in the decision making process around today's enterprise software stacks. We cover the industry by looking at integrated software stacks, focusing on business and operational context rather than speeds and feeds and feature tick-lists.

Founded by James Governor and Stephen O'Grady, and headquartered in Denver, Colorado, RedMonk is on the web at www.redmonk.com.

If you would like to discuss this report email Michael Coté (cote@redmonk.com), James Governor (jgovernor@redmonk.com) or Stephen O'Grady (sogrady@redmonk.com).