RedMonk

Working with Open Source Companies

Overview

Michael Coté

Stephen O'Grady

For all of the rhetoric, all of the debate, it is indisputable that open source software is playing an increasingly important and often mission critical role within successful IT organizations all over the globe. While no panacea, open source can provide compelling economic and technical benefits to businesses and other organizations constantly under pressure to eke more performance out of their technology operations for less money.

While "vendor relationship management" models for traditional, non-open source software providers are well understood and tested, far less so are the techniques for successfully managing relationships with open source communities and vendors. This paper discusses several key differences between closed and open source software vendor management, providing advice for managing your deployments and investments in open source technologies.

The Role of the Open Source Community

Open source software may be produced by an individual, a group, a company, or a combination of all of the above. This ecosystem of people, users, and commercial entities with a common interest in the software are collectively called "the community."

One of the most important distinctions between closed and open source software is the importance of community. In the closed source world, a software buyer is purchasing not only the right to use the software, but also a relationship with the company backing the software. While this type of relationship can exist in the open source world, as with single entity developed open source projects such as MySQL, open source software is

more typically the product of individuals spread across disparate organizations and geographies.

This is an important consideration for commercial buyers for two reasons.

Because assessing the long term viability of a commercial vendor can at times be more straightforward than determining the future of a community.

Because the relationship between a community is governed by different expectations than the relationship would be with a vendor.

How should buyers, then, identify vibrant, open source communities? The metrics will vary from project to project, but generally buyers should look for the following:

- **Commercial Investments:** The more commercial vendors are invested in a given project, generally speaking, the greater its long term sustainability. The Eclipse project, for example, is the platform of choice for literally dozens of vendors and benefits from the resulting network effect.

- **Governance:** There is no single governance structure that is appropriate for every project, but it is generally true that openness is a metric of relevance. Insular, isolationist project teams that eschew outside contributions -either in code or other contributions -tend to suffer in comparison with peers that are more open.

- **Integration:** Projects that are either well integrated or function smoothly alongside of other existing open and closed source infrastructure products are more likely to be successful in the longer term. Application platforms, for example, that can run easily with either the open source MySQL database or the closed source DB2 and Oracle products will be more successful than projects that favor a single product.

- **Speed:** While project size means that release frequency will vary widely -smaller projects can release more frequently than larger ones -trending is important. Look for release timing over the projects history; if releases

are fewer and less frequent, it may be an indication of poor project health.

- **Traffic:** Strong communities typically have a high degree of "traffic." This traffic will be expressed chiefly in the communications around the community: email lists, online forums, chat channels like IRC, bug tracking systems blog postings, books, magazine articles, podcasts, and other artifacts discussing the open source project.

In addition to determining whether or not a given community is viable in the long term, it's useful to consider how a commercial entity might best interface with it. Often, commercial organizations turn to other commercial organizations related to a particular community as a practical interface.

The Role of the Open Source Company

In recent years many companies have emerged whose role is to explicitly sell service and support to users of open source software. These "open source companies" are typically a large part of their respective communities. Indeed, if an open source company is not "active" in a community, be suspicious of their claims to fully support the software.

Though it is common for sizable open source communities to offer substantial options for informal support in the form of community maintained and generated forums and wikis, communities are under no obligation to support users of the software, let alone respond rapidly or in-depth. For non-production deployments of the technology, this lack of a service level guarantee is often acceptable, but for technologies deployed to production, support equivalent to that available from commercial vendors is often a requirement.

The demand for this type of support has in turn given rise to open source companies, which meet this need by offering support and services around a particular open source project or projects. The importance of this trend cannot be overstated, as it delays the point at which potential customers are required to invest in technology.

Evaluating Open Source Software

From an open source customer's perspective, one of the major benefit of open source is that users of open source can perform self-service evaluations and Request for Proposals, or RFPs: running and testing the software, talking freely with other users in the community about their experience, and discussing your project's needs with the community. Because of this, most open source companies are not geared towards working on long RFPs and evaluations.

They also typically invest less on a relative basis in high cost marketing and sales personnel activities. This lack of investment may at times make them appear less credible, but it also reduces the cost the vendor must pass along to the customer.

With open source software, rather than invest up front in technologies that might address a business need, organizations can experiment with open source technologies often with no up front licensing, investing in support only at the point in which it enters production. Sun Microsystems' Simon Phipps has in the past likened this to "paying at the point of value."

Budgeting for Open Source Software

While open source software may be free to download and use, this does not typically equate to free in terms of usage and deployment. Production deployments are very rarely free, and typically involve expenditures for support, services, training, and even project management.

Because there are typically no up front license fees, the cost of open source software is typically amortized over the lifetime of the software. Though this change in fee structure can be difficult for traditional procurement organizations to adapt to, many open source buyers have success by simply treating the support and service fees as the license in contractual terms.

Supporting Open Source Software

Support can broadly be defined as assisting a user of software with achieving their goals. Often this means trouble-shooting error conditions or unexpected results. Support can occur during all phases of the software life-cycle: development, testing, production, and maintenance.

The open source community may provide an operable level of support, but can not be depended on to support the use of the software on your exact terms. An open source company, however, under Service Level Agreements (SLAs) with customers can be contractually trusted to respond rapidly and in-depth. As with any support relationship, the more familiar the supporters are with the system they're supporting, the quicker and more accurately they'll be able to address problems. In particular, purchasers of open source support should question providers on how many "committers" -developers with the permissions to actually commit changes to the codebase -they have for the project in question.

As is typically the case, support is traditionally offered in tiers, in which the level of service is commensurate with the cost. Buyers can expect pricing for the highest levels of 24x7 support to be reflect the costs of delivering that support; the open source nature of the software has little impact on the cost of maintaining staff around the clock, for example. Buyers should question vendors closely as to their specific support capabilities; the ability to provide true, enterprise level support after all is not universal.

One potential advantage of open vs. closed source software comes at the end of a software's life. While the availability of commercial software is under the control of the company that backs it, the availability of the source code allows open source software to "live" as long as there's someone to support it. While "forking" the life of software like this is not easy, it's at least possible to stave off the end-of-life process for open source software.

Contributing Back to Open Source Software

In addition to free access to source code, open source is unique in its ability for users to contribute to the project itself. While that process is rarely trivial and is by no means required, as a user you may be in the position to contribute back to the project in the form of bug fixes, documentation, testing, or even code for entirely new features.

Clearly, business differentiation and legal ramifications must be considered before contributing code. Many organizations find this concept counterintuitive, believing that any improvements they make represent a competitive advantage over other users -some of whom may be their direct competitors.

However, there is an important advantage to contributing back fixes and updates.

If an organization creates a fix for the code but keeps it purely internal, that fix must be maintained and reapplied indefinitely. This can prove unnecessarily expensive, and may invalidate existing support contracts, mandating as it does alterations to the existing package. Contributing such fixes is effectively outsourcing the ongoing maintenance of that code to the community and/or vendor, at the minimal cost of sharing a bug fix or feature.

In addition to funding developers, you may also consider influencing the open source project via the company if you cannot be involved yourself. That is, you may hire the open source company to be your proxy for involvement in the open source community, even sponsoring additional developer to accelerate the release of key features.

Open Source Licensing and Indemnification

Licensing is undoubtedly one of the more controversial aspects to open source software, but the actual risks are often poorly understood. For example, in many if not most cases, open source customers have no

intentions of distributing the software itself, which minimizes concerns around even the most restrictive open source licenses they might encounter. Likewise, some vendors may make noises about the patent risks in the software itself, but historical precedent indicates that patent issues are resolved amongst vendors as opposed to customers.

Buyers should consult their legal counsel, as always, for more detailed examinations of potential liabilities, particularly if they're contributing code back. Many customers choose to restrict their purchasing to software released under an OSI approved license, which provides some non-legally binding assurance of license quality but more importantly ensures that legal isn't dealing with license to product ratios of 1:1.

As for indemnification, given the low probability of it coming into play it should be viewed as an asset, but likely not an asset worth significant additional cost or the justification for selecting an inferior product.

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) or Stephen O'Grady (sogrady@redmonk.com).