Single-Vendor Open Source Firms

Dirk Riehle, Friedrich Alexander-University of Erlangan Nüremberg

The single-vendor open source model dominates venture-capital funding for open source software firms. It is of high economic relevance and an excellent example of how open source licensing and related strategies are simply tools in the design of business models, not philosophies.

A business model describes how a company operates and achieves its goals. Open source itself is not a business model, but it can be an important strategy to help a company reach its goals. While each firm has its own distinct plan, there are naturally distinguishable types of business models.[1–3] In my work, I have found three distinct coarse-grain types of open source business models, based on their value proposition and the intellectual property (IP) that supports it.[5] The three core models are

1. open source service and support firms
2. open source software distributors
3. single-vendor open source firms.

Service and support firms do not necessarily own specific IP but, rather, service users of existing open source projects. Open source distributors provide a complex assembly of open source components packaged as one well-working product but typically don't own the software they distribute. Between the two types of companies, only distributors can earn the returns on investment that draw the interest of venture capitalists.

SINGLE-VENDOR OPEN SOURCE

Single-vendor open source firms own some piece of software that they provide under an open source license. Typically, they develop the software themselves and earn money through complementary products and services. The companies have many options to design their business models. To avoid confusion, I'd like clarify two strategies they often use, called *dual licensing* and the *open core model*.

- *Licensing strategy*: In addition to providing an open source project, the vendor sells the software to customers under a proprietary license, together with the set of services and warranties that customers usually ask for. Providing the software under

Digital Object Identifier 10.1109/MC.2020.2969672
Date of current version: 9 April 2020

EDITOR **DIRK RIEHLE**
Friedrich Alexander-University of Erlangen Nürnberg;
dirk.riehle@fau.de

FROM THE EDITOR

Welcome back to "Open Source Expanded." This column takes a broad view of open source, even though recent articles focused on the theme of using open source. With this article, we turn to a hot issue: commercial open source and its licensing strategies. We will digress from the column's themes at times to examine controversial issues, such as commercial open source, ethical licensing, meritocracy, and so forth. After this issue's article, we plan to return to the topic of using open source and finish with a discussion of how to manage the supply chain, what tooling to use, and so on before starting the next theme on open source project communities. Happy hacking! — *Dirk Riehle*

two different licenses, one open source and the other proprietary, is known as the *dual* (or *multi*) *licensing strategy*.

- *IP modularity strategy*: Sometimes, the vendor withholds some functionality from the open source software and provides the feature as part of a paid version. Drawing a distinction between an open source core and nonopen source extensions that customers pay for has been called the *open core model*. In more general terms, it is named *IP modularity*.[4]

There are at least three generations of companies that utilize the single-vendor open source model.

1. *The pioneers*: The idea of open source software that is owned and exploited by a single vendor dates back to the 1990s. Products such as MySQL and companies including Sleepycat Software and Trolltech (now Qt) fall under this category.
2. *The second wave*: During the early 2000s, entrepreneurs and venture capitalists realized that open source was an effective strategy for disrupting existing enterprise-software markets. In 2004, newly incorporated single-vendor open source firm SugarCRM coined the term "commercial open source" to alleviate potential customers' fears about open source software. Other examples include Jaspersoft and MuleSoft, both of which have been acquired by larger companies, providing their investors the desired return on investment.
3. *The current breed*: The idea of single-vendor open source is alive and well in the current generation, which roughly dates to the 2008 recession. Example firms are MongoDB, Redis Labs, and Neo4j. The focus has shifted from enterprise applications to DevOps tooling and infrastructure, usually with a cloud component.

Some single-vendor open source firms have had sizable exits that included initial public offerings, and many of the current breed are considered to be worth more than US$1 billion. No other type of open source–based business model has supported so many companies that generated such high returns on investment for venture capitalists.

Revenue streams

Venture-capital funding flows only if a firm can believably promise significant returns on its investors' money. Single-vendor open source firms achieve this by promising the same revenue streams that traditional software vendors do. Revenue streams must be based on some complement to the open source code; otherwise, it makes no sense for investors to commit their money. These revenue streams consist of, but are not limited to, the following:

1. commercial licenses for the core software and possible extensions
2. guarantees, such as warranties, indemnification, and certification
3. early and preferential access to bug fixes and new features
4. support services, such as hotlines and on-site assistance
5. operational services, including hosting the software in the vendor's cloud
6. complementary materials for documentation, training, and so forth
7. access to self-help services, such as forums and chat bots.

None of these revenue streams should surprise practitioners; they are familiar to traditional vendors and single-vendor firms alike. The main difference between single-vendor and traditional firms is that single-vendor companies often forego the initial license fee and start with what is traditionally known as a *maintenance fee* for the product and service. Charging maintenance fees is often called the *subscription model*. Increasingly, companies emphasize the cloud. Vendors focus the previously listed feature array on their cloud service as the primary customer incentive.

The challenge that makes or breaks a single-vendor company concerns turning nonpaying users into paying customers. Behind each restricted feature is a motivation for a nonpaying

user to become a paying customer. For example, some users may not like a copyleft-based open source license or need 24-h support and decide to upgrade to the paid version or service.

Business functions

Given that single-vendor open source products and revenues don't look much different from those of traditional software vendors, one might wonder why the software is provided as open source in the first place. What about the downside, such as not turning a user into a customer because the open source version is sufficient and the services are not that important? The short answer is that going to market with an open source strategy drives adoption faster, better, and cheaper than any comparable approach. The longer answer relates to improving core business functions so that they become superior to traditional competitors. Marketing, sales, product management, engineering, and so on all are better with a single-vendor strategy. Different business functions utilize the community of nonpaying users for the company's gain.

Marketing. High-quality software for free is a great value proposition. As a consequence, word-of-mouth marketing is particularly effective for single-vendor firms. Happy users like to talk about the software that makes their work life easier. They also make good reference material if they are willing to talk about their experience. This helps build a large nonpaying user base, which feeds sales and supports more efficient and effective product and engineering management and support functions.

Sales. By open sourcing some or all of its product, a single-vendor firm enables potential customers to use its product with minimal friction. Unlike trials, open source licenses permit users to employ the software as long as they want to, for whatever purpose. As a consequence, users will happily keep the software as long as it fulfills their needs.

From a sales perspective, having an installed base of nonpaying users is not a problem but an opportunity. A single-vendor firm often tracks the organizations that download its software and gathers email addresses, which can indicate the potential for a follow-up. Having email addresses for multiple users at the same company signals that a sales call may be fruitful. This way, the open source strategy enables the sales organization to prioritize where to direct its effort.

The sales process also becomes more effective. The initial adopters of the free version are often line-of-business (LoB) users, particularly for hosted software that can be subscribed to with a credit card. If there are multiple LoB users in one company, the IT department may want to rein in the service and purchase it centrally. In a comparative evaluation of possible vendors, the single-vendor open source firm has a leg up on its competitors. Its product is already in use and has champions inside an organization, while its competitors do not. You can ask yourself, "What would you rather buy, a product from a vendor that you have yet to test and evaluate or one that is already in use in your organization where you can ask for feedback about it?"

Product management. A large user base, even one including nonpaying users, is a great resource for product managers to draw on for new product-feature discovery. Product managers can do so by monitoring user problems and requests in product forums. They can make an issue tracker publicly available so that users can report bugs, and they can create user polls to prioritize upcoming features. Due to the large user base, this takes place on a scale that traditional vendors cannot match.

Engineering management. A large user base finds problems faster than a small one. Users that file bug reports might also provide a patch that fixes the problem. In general, single-vendor open source providers do not expect nonpaying users to help develop their software, but they will not reject code contributions. Patch submitters, however, usually have to sign over their copyright, as discussed in the "Intellectual Property" section. At the same time, contributions are an excellent indicator of various developers' capabilities and potential interest in a position at a company. Single-vendor open source firms use their community edition as a recruiting mechanism to identify and acquire engineering talent.

Product support. Nonpaying users typically understand that open source software does not come with a right to support. As a consequence, and in the spirit of open source, many users are willing to help each other. If the single-vendor firm provides appropriate tools, such as forums and wikis, users may create documentation and self-help materials. Product support can benefit from understanding user problems and incorporating the materials that users develop.

Community management. Most of the benefits of open sourcing result from having a large but nonpaying user base. Creating that user base comes at a cost: The company needs to manage the community, and that requires labor. Community managers must engage with users and should provide, for example, a website, forums and wikis, and a software forge. Most of these costs are variable and scale with the number of users. Community managers' primary efficiency consideration concerns growing the user base as much as possible with minimal effort. They achieve this through various best practices. Mostly, they try to establish a community that helps itself and to which company resources are allocated as a last resort. If, for example, a nonpaying user asks a question in a forum, community managers will wait for another user to answer. If no answer is forthcoming, they may nudge users to help out, and if that doesn't work, they may provide the answer themselves.

INTELLECTUAL PROPERTY

Single-vendor open source firms face two unique business concerns.

1. How do we motivate a nonpaying user to buy the commercial version of their product?
2. What if a competitor takes the product and competes with the single-vendor open source firm?

Both problems can be addressed by the same IP strategy. I previously called it the *IP-rights imperative of single-vendor open source firms*:

> *Always act in such a way that you, and only you, possess the right to provide the open source project under a license of your choice.*[6]

By remaining the sole proprietor of the IP, a single-vendor open source firm can multilicense, that is, provide the product under different licenses to different parties. Usually, there are two licenses: an open source version to build the user community and a commercial one for paying customers.

Basic strategy

To remain the sole proprietor, a company must own 100% of the software, community-developed extensions notwithstanding. For this, it buys or develops the software itself. Community contributions are accepted only if the people who make them surrender their copyright or provide it with a relicensing right. Almost always, the open source license is the most far-reaching reciprocal one available, which at the time of writing is the Affero General Public License, version 3 (AGPLv3). It has two goals:

1. Make the maximum number of the licensed software's use cases trigger the copyleft clause. This clause requires that when the software is passed along (a "distribution"), all source code that has been touched must be laid open. Competitors that distribute a modified product can, therefore, do so only under the AGPLv3 license, preventing the creation of unique IP and damage to the software creator's competitive position.
2. Help to make software patents go away through a patent-retaliation clause. The retaliation clause revokes a user's rights if the user brings a software-patent lawsuit against someone else.

Both commercial users and potential rivals generally dislike such licensed code. Thus, choosing this license 1) motivates nonpaying users to become paying customers and 2) discourages other companies from picking up the product and using it to compete. Until recently, as the venture-capital returns show, this IP strategy worked well, and a single-vendor firm rarely faced serious competition from another company using its product against it. The advent of large public cloud infrastructures, most notably Amazon Web Services (AWS), has changed this.

Cloud strategy

Increasingly, software products are hosted in public clouds. Providing their products as cloud services is a key strategy for many single-vendor open source firms. Users build applications utilizing the service. Companies that do this include MongoDB, Redis Labs, and Confluent, all valued at more than US$1 billion. For this to work, the vendor's cloud service needs to provide a better experience than what the user would have by hosting the software in its own cloud. This can be achieved through additional functionality, as discussed, or simply superior and more cost-effective service. However, providing a secure multi-tenant cloud service at scale is not a trivial undertaking: It increases the engineering challenges of the vendor significantly.

The move to the cloud coincided with a shift from enterprise applications (2G) to infrastructure software (3G). For developers to incorporate open source software into their products, there must not be a copyleft license that touches their own code. Otherwise, they would have to open source their own products, which for most commercial development is not acceptable. As a consequence, some single-vendor open source firms packaged their copyleft-licensed core using permissively licensed shims that shielded user code from the copyleft effect. Large cloud operators, such as AWS, started to compete with the firms by offering the vendors' products as their own by using the open source license. With pure copyleft licensing, this would not have happened, because no cloud operator would like to open source its own infrastructure software.

Some single-vendor open source firms have changed to proprietary "almost open source" licensing. MongoDB's Server-Side Public License (SSPL) is an example. It is similar to the Apache 2.0 license but does not permit companies to use MongoDB software in competing products. After intense discussion, the Open Source Initiative, the provider of the open source definition and arbiter of licenses, decided that SSPL is not an open source license. MongoDB and firms that took similar measures are no longer considered to be open source companies.

The licensing change drew the ire of the open source community at large and generated bad publicity. However, the vendors that took this step were mature, successful organizations whose products could stand on their own without an open source license. Whether this will be possible for less mature vendors remains to be seen.

Single-vendor open source firms have positioned themselves as superior successors to classic proprietary-software vendors by utilizing an open source strategy to drive adoption and enable better business functions. However, moving applications and components to the cloud opened the door to competition from large cloud vendors, leading to changes in licensing strategies, with

unclear consequences for the future viability of the model. C

ACKNOWLEDGMENTS

I would like to thank Ann Barcomb, Andreas Bauer, Maximilian Capraro, Michael Dorner, Nikolay Harutyunyan, Joseph Jacks, Andreas Kaufmann, Heather Meeker, Georg Schwarz, and Mathias Zinnen for feedback on earlier versions of this article.

REFERENCES

1. E. Capra and A. I. Wasserman, "A framework for evaluating managerial styles in open source projects," in *Proc. IFIP Int. Conf. Open Source Systems*, Boston: Springer, 2008, pp. 1–14. doi: 10.1007/978-0-387-09684-1_1.
2. C. Daffara, "Business models in FLOSS-based companies," in *Proc. Workshop Presentation 3rd Conf. Open Source Systems (OSS 2007)*, 2007, pp. 1–8.
3. P. J. Ågerfalk and B. Fitzgerald, "Outsourcing to an unknown workforce: Exploring opensurcing as a global sourcing strategy," *MIS Quart.*, vol. 32, no. 2, pp. 385–409, 2008. doi: 10.2307/25148845.
4. J. Henkel, C. Y. Baldwin, and W. Shih, "IP modularity: Profiting from innovation by aligning product architecture with intellectual property," *Calif. Manage. Rev.*, vol. 55, no. 4, pp. 65–82, 2013. doi: 10.1525/cmr.2013.55.4.65.
5. D. Riehle, "The single-vendor commercial open course business model," *Inform. Syst. e-Bus. Manage.*, vol. 10, no. 1, pp. 5–17, 2012. doi: 10.1007/s10257-010-0149-x.
6. D. Riehle, "The intellectual property rights imperative of single-vendor open source," Software Research and the Industry, July 18, 2009. [Online]. Available: https://dirkriehle.com/2009/07/18/the-intellectual-property-rights-imperative-of-single-vendor-open-source/

DIRK RIEHLE is the professor for open source software at the Friedrich Alexander-University of Erlangen Nürnberg. Contact him at dirk@riehle.org.

Digital Object Identifier 10.1109/MC.2020.2980780