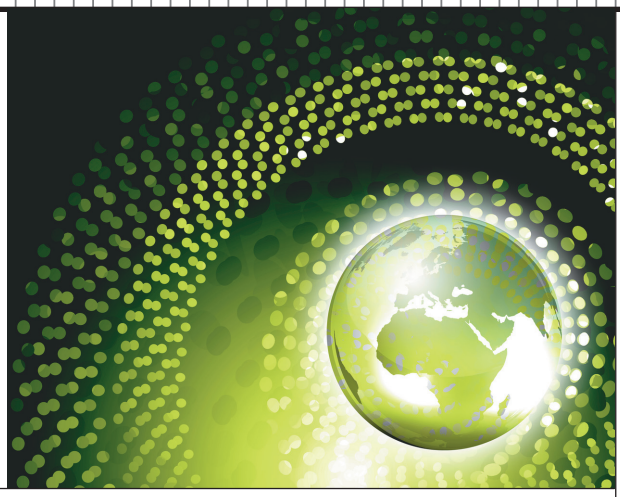
The Economic Case for Open Source Foundations



Dirk Riehle, *Friedrich-Alexander-University of Erlangen-Nürnberg*

By establishing a successful open source platform, software firms can compete more effectively across technology stacks and thereby increase their addressable market.

An open source foundation is a group of people and companies that has come together to jointly develop community open source software. Examples include the Apache Software Foundation, the Eclipse Foundation, and the Gnome Foundation.

There are many reasons why software development firms join and support a foundation. One common economic motivation is to save costs in the development of the software by spreading them over the participating parties. However, this is just the beginning. Beyond sharing costs, participating firms can increase their revenue through the provision and increased sale of complementary products. Also, by establishing a successful open source platform, software firms can compete more effectively across technology stacks and thereby increase their addressable market. Not to be neglected, community open source software is a common good, creating increased general welfare and hence goodwill for the involved companies.

OPEN SOURCE FOUNDATIONS

The Linux operating system and the Apache webserver are popular examples of open source projects that are in widespread industry use. They started out as volunteer projects without any commercial backing. When the industrial significance of these projects became apparent during the 1990s, interested software developers and firms decided to put the future of the software on more solid ground by creating nonprofit organizations.

Such an organization, commonly called a foundation, serves as the steward of the projects under its responsibility. It provides financial backing and legal certainty, making the survival of the software less dependent on the individuals who initially started it. There are many variants of foundations like trade associations and consortia. Each of them has its own matching legal structure, depending on the specific goals of the founders. This article uses the term foundation to denote all of them.

The foundation represents the community of developers, which is also why the software is called *community open source* (D. Riehle, "The Economic Motivation of Open Source: Stakeholder Perspectives," *Computer*, Apr. 2007, pp. 25-32; E. Capra and A. Wasserman, "A Framework for Evaluating Managerial Styles in Open Source Projects," *Proc. 4th Int'l Conf. Open Source Systems* [OSS 2008], Springer, 2008, pp. 1-14).

Community open source is different from single-vendor open source, which is open source software that is being developed by a single firm. Firms behind single-vendor commercial open source expect direct revenue from selling the software and services for it (D. Riehle, "The Commercial Open Source Business Model," *Proc. 15th Americas Conf. Information Systems* [AMCIS 2009], AIS Electronic Library, 2009). This is typically not the case with communally owned open source, as competition is likely to keep revenues down.

However, there are several economic reasons why software firms join and support foundations to develop community open source: Some members expect cost savings for products built on the community open source software, others expect increased revenue and sales from complementary products, and yet others want to grow their addressable market.

ORGANIZATIONAL RESPONSIBILITIES

The main purpose of a foundation is to act as the steward of the software being developed and to ensure

Published by the IEEE Computer Society
0018-9162/10/$26.00 © 2010 IEEE

its long-term survival. A foundation has various responsibilities, including the following:

- organize the project community;
- actively market the software;
- clarify and manage intellectual property rights;
- set strategic directions for the software;
- respond and remain accountable to its members; and
- run all relevant back-office processes.

Open source foundations are usually open to everyone to join; however, a membership fee may apply. Many of their processes are similar to those of traditional software associations and will not come as a surprise. What is different, however, is the provision of the main product as open source and the resulting intellectual property implications.

INTELLECTUAL PROPERTY MECHANISMS

Some eschew open source out of fear of a loss of intellectual property. Open source foundations solve this problem by providing well-defined processes that clarify any intellectual property rights issues associated with the software. In the end, the open source project becomes just like every other software and is provided under an open source license that spells out its usage conditions.

In software development, three main categories of intellectual property rights must be considered:

- copyright (to the source code and related texts),
- trademarks, and
- patents.

The contributors to the project provide the relevant intellectual property. Most foundations define the relationship between a contributor and the project using a so-called contributor agreement. Any legal entity that wishes to contribute to the project, whether a member of the foundation or not, must sign this agreement.

A common practice of open source foundations is to own the copyright to all source code and related texts. Thus, the contributor agreement is set up so that the contributor, be it a company like IBM or a volunteer programmer, signs over the copyright of any current or future contributions to the foundation. (In a weaker form, sometimes only a relicensing right is required.)

Open source foundations provide well-defined processes that clarify any intellectual property rights issues associated with the software.

Using this mechanism, the foundation becomes the sole owner of the copyright. It is important that there be only a single owner: Decision making is with the foundation rather than a distributed group of diverse copyright holders. The foundation can now define and enforce the license terms under which the project is made available to the public and can defend the software in court.

The choice of the license depends on the foundation's goals. Most foundations choose a liberal license to allow for the widest variety of use circumstances of the software by its members. Such a liberal license typically allows embedding the software in other software packages without requiring the open sourcing of these other packages.

An important practice of a foundation is to ensure that no source code is contributed from another open source project with an incompatible license. The specific fear is that the contribution of incompatible code would require an undesired change of license, as might happen, for example, with the contribution of GPL-licensed code to Apache-licensed code. The GPL license is the original reciprocal ("viral") open source license that requires all derived code to have the same license as well. "Keeping the code clean" is a prime directive at many foundations.

Naturally, the foundation also becomes the owner of the software trademarks and acts to enforce them. Thus, the foundation becomes the trustee of both the source code and its trademarks.

Finally, the contributor agreement clarifies the use of software patents. Source code implements software patents. Even if the foundation owns the copyright to the source code, without further measures, users of the software may still have to pay royalties to the holders of the patents implemented by the software. This can become particularly nasty if royalty requests surface only after the software has been put to use in a user organization. For this reason, the contributor agreement typically requires contributors to provide a general (perpetual, unrestricted, royalty-free) usage grant of the patent to all users of the open source software. This protects users from unanticipated patent royalty requests.

SHARING DEVELOPMENT EXPENSES

There are many economic reasons to start, join, or support an open source foundation. The original and still most widely known reason is cost savings realized by standardizing on one platform and sharing its development expenses.

Consider the situation of Unix and Linux desktops in the late 1990s: Several competing windowing systems existed, each with incompatible desktop applications, and all of them configured and deployed differently, depending on the Unix or Linux distribution they came with. By then, Unix had lost the competition for

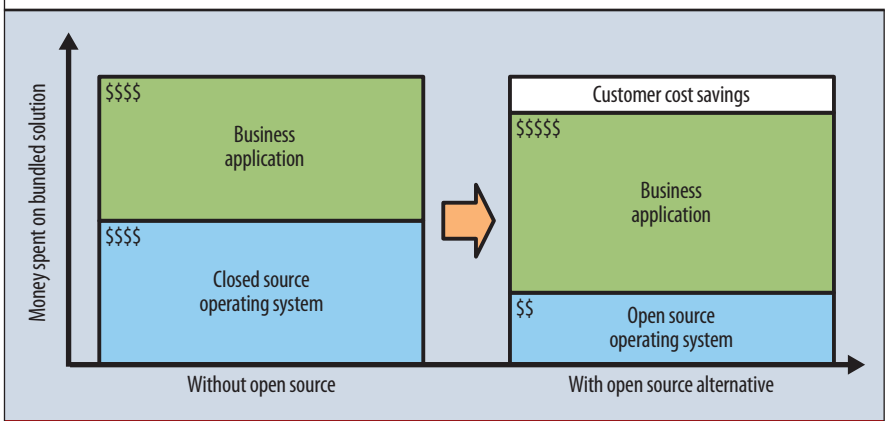


Figure 1. The support of open source software lets vendors sell at a higher price.

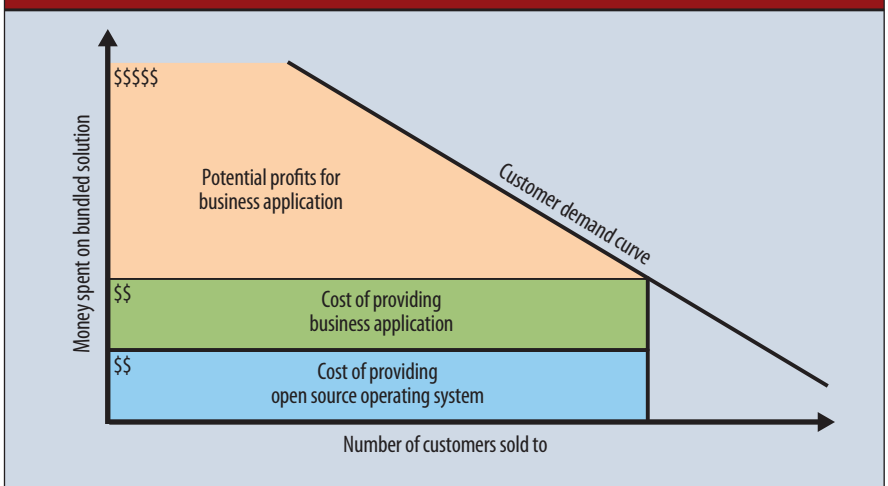


Figure 2. The support of open source software lets software firms sell to more customers.

the users' desktop to Microsoft Windows. Graphical user interfaces for Linux were a pure cost position for the distributors.

In this situation, any good-enough desktop software would do for the involved software firms. Distributors like Red Hat and SUSE (Novell) as well as IBM and HP decided not to compete on the merits of their desktop configuration but rather to support a common desktop environment. This led to the continued development and consolidation of the GNOME and KDE desktop environments, formally supported by the GNOME foundation and the KDE e.V., respectively. These two foundations remain volunteer efforts, however, with strong corporate support.

It is not always the software firms that start or grow a software foundation. Cost savings through community open source can have multiple roots. Sometimes, customers join forces to create a foundation and require that any software development work by a contractor utilize and develop the open source software further. Currently, the US healthcare industry is undertaking steps in this direction.

PROFITS PER SALE IN A GIVEN MARKET

Beyond cost savings in research and development and in user organizations, original software development firms can use open source foundations to their competitive (economic) advantage.

The initial thrust behind company contributions to Linux and the Apache Software Foundation projects focused on supporting an alternative to more expensive closed source solutions. If, for example, a company is selling a business application that also needs an operating system to run, more money will be available for the business application vendor if no money is spent on the operating system license and its maintenance fees. Hence, for the customer, an open source alternative saves money, while for the business application vendor more money becomes available, at the expense of the closed source operating system vendor, who misses a sale. Figure 1 illustrates this economic situation.

An early example of this mechanism is IBM's support of Linux. Realizing that OS/2, IBM's then-competitor to Microsoft's Windows, was losing in the marketplace, IBM threw its weight behind Linux and related open source projects. Having an alternative to Windows meant that IBM could keep Microsoft's license fees in check when selling to customers.

In general terms, replacing a high-cost closed source component of the technology stack with a lower-priced open source component increases pricing flexibility for the vendors of the other components in the stack. It also reduces costs for customers and makes more money available for other purchases.

INCREASED SALES IN A GIVEN MARKET

A second consequence of the increased pricing flexibility is that a software developer can sell to more customers than before. Some customers are more price-sensitive than others. Figure 2 illustrates this as the customer demand curve. Going down the demand curve from left to right, the price for the business application plus operating system bundle goes down, and more customers are willing to buy. In simplified terms (a nontransparent market), the developer stops selling only if the price has come down to its own total cost.

Thus, replacing the more expensive closed source operating system with a lower-cost open source alternative reduces the lowest possible price point for the bundle. This lets a vendor sell to more customers, which leads to more profits.

Higher profits on a given sale and more profits by selling to more customers are two important reasons why a software firm may support open source software that is complementary to its own product line. From the firm's perspective, supporting the open source software is a subsidy, paid out of increased profits from its own product. Basically, the open source alternative lets the firm shift revenues from a complementary product, owned by someone else, to its own product.

GROWING THE ADDRESSABLE MARKET

The size of the market a software firm can sell into depends on the platforms on which it is based. If a vendor builds on a platform that customers aren't willing to operate, the firm's products will not be considered. Thus, the choice of the platforms a firm's product runs on is crucial. As indicated, an open source platform is economically more beneficial than a closed source platform. Thus, the software development firm should support an appropriate platform and encourage other vendors to do the same. More and better applications will grow the value of the platform to customers. With growing acceptance of the platform, more customers will be operating it, first increasing the total size of the market and then the size of the market that the software firm's products can address.

Figure 3 illustrates the dynamics of shrinking a closed source platform to the advantage of an open source platform. The money leaving the market around the closed source platform enters the market for products built on top of the open source platform. As customers review the choice of products available, they prioritize purchases anew in accordance with what's available and how big their IT budget is. This dynamic is particularly attractive to the providers of mission-critical applications, which typically get higher purchasing priority than less important, more incremental applications.

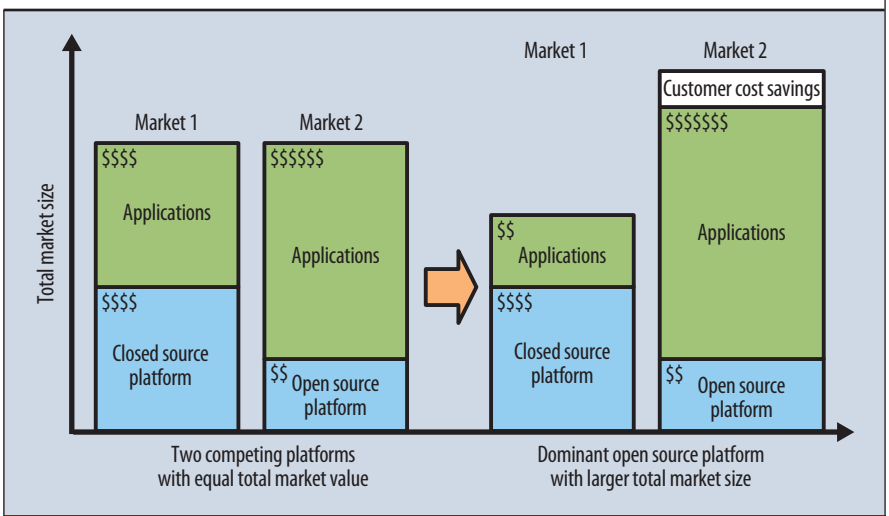


Figure 3. Growing an open source platform increases the total market size.

Participating in the development of the open source platform is of strategic interest to a software firm. It ensures visibility of the firm to potential customers and promises high technical quality of its software products. Gaining a strong position in the foundation and development processes of the software creates a significant positional advantage over later competitors.

There are more platforms or layers in the technology stack than some might think. The obvious platforms are operating systems and middleware solutions. Beyond this, many more potential platforms exist, addressing vertical as much as horizontal slices of the stack. Whether it is platforms for business accounting or medical imaging, automotive software buses or electronic patient records, we can expect a wealth of new domain-specific open source platforms to appear in the coming years.

A firm should consider creating community open source and supporting an open source foundation if it is not only competing within the same stack, but across stacks. Linux, the Apache projects, and the Eclipse platform can all be viewed as software platforms on which revenue-generating applications are built. These platforms compete with closed source alternatives, for example, the Microsoft set of platforms, namely Windows, ASP.NET, and Visual Studio.

A reliable platform attracts other software vendors that might base their own products, whether provided as open source or not, on this platform. The increasing richness of functionality around a given platform benefits everyone: Customers cannot go wrong in deciding for this platform. Moving customers from a not-supported platform to a firm's own platform increases the size of the addressable market, which is likely to lead to more sales.

Every software development firm today should ask which open source foundation to support or, if necessary, to found. The benefits are clear: Done right, the firm can expect cost savings, increased profits per sale, a higher number of sales, and a larger addressable market. The question then becomes one of investment: How much to

invest and what return to expect. At present, we lack economic models and decision processes to answer these questions.

The open source research group at the Friedrich-Alexander-University of Erlangen-Nürnberg and its collaborators are working on this question. In addition, we are looking at the processes and tools used by open source foundations and in open source software development in general. In collaboration with the Open Source Business Foundation, an international nonprofit organization located in Nuremberg, Germany, we are making our research findings available to industry. Finally, we are interested in supporting public policy decisions with economic and technical insight to help increase general welfare through high-quality community open source. C

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

Editor: Sumi Helal, Department of Computer and Information Science and Engineering, University of Florida; helal@cise.ufl.edu

Readers are encouraged to use the message board at www.computer.org/industry_perspective to post comments, offer feedback, or ask questions.

cn **Selected CS articles and columns are available for free at http://ComputingNow.computer.org.**

"All writers are vain, selfish and lazy."

—George Orwell, "Why I Write" (1947)

(except ours!)

The IEEE Computer Society Press is currently seeking authors. The CS Press publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. It offers authors the prestige of the IEEE Computer Society imprint, combined with the worldwide sales and marketing power of our partner, the international scientific and technical publisher Wiley & Sons.

For more information contact Kate Guillemette, Product Development Editor, at kguillemette@computer.org.

IEEE CSPress
www.computer.org/cspress

WILEY
Publishers Since 1807