PERSONAL VERSION

# The Business of Open Source Software: A Primer

Michael "Monty" Widenius and Linus Nyman

> *"Ideology isn't what has sold the open source model. It started gaining attention when it was obvious that open source was the best method of developing and improving the highest quality technology.*
>
> Linus Torvalds
> Software Engineer and creator of the Linux kernel

This article is meant as a primer for those interested in gaining a basic understanding of the business of open source software. Thus, we cover four main areas: i) what motivates businesses to get involved in open source; ii) common open source licenses and how they relate to community and corporate interests; iii) issues regarding the monetization of an open source program; and iv) open source business models currently employed. This article is particularly suitable for people who want a general understanding of the business of open source software; people who want to understand the significant issues regarding an open source program's potential to generate income; and entrepreneurs who want to create a company around open source code.

## Introduction

In a world built on openness, in which licensing dictates that the product is not only free of charge, but can be freely copied, modified, and redistributed by enthusiasts and competitors alike, how *can* anyone possibly make money on open source? The question of how one can monetize open source software is a significant one. The quest for, and dissemination of, its answer was the spark that started what was to become the *Technology Innovation Management Review* (Lavigne, 2007: timreview.ca/article/92; McPhee, 2011: timreview.ca/article/465).

Although much has been learned during the years since the emergence of open source and the business that grew to surround it, there are still few articles that attempt to summarize its dynamics. Perhaps the most well known of those efforts is Hecker's "Setting up Shop" (1998; tinyurl.com/28n7o3), which largely focused on what strategies could be employed utilizing open source. Now that open source is a much more mature field than it was back then, we can focus on documenting what entrepreneurs have done rather than could do.

The goal of this article is to concisely explain the nuts and bolts of how the business of open source works, including sufficient detail to serve as a useful primer on the topic – a springboard for further reading. Our focus is on approaches that generate income based on open source software and its development (e.g., not hardware manufacturers with an open source involvement).

The article is structured as follows. First, we offer a brief look at some of the main corporate motivations in open source. Second, we cover the most common types of open source licenses and the main aspects and concerns for businesses and programmers regarding licensing. Third, we outline the most significant points in a piece of software's earning potential. Finally, we briefly describe the more common business models in use today, and we examine their pros and cons from the standpoints of both the developers and entrepreneurs. Included at the end of the article is a list of recommendations for further reading.

## Background: Corporate Motivations

The adoption of open source code allows businesses to harness the creativity and labour of both their employees and their customers in a way that is not available to firms employing only proprietary software licenses. Indeed, where developer motivations include many social motivations, firms have tended to emphasize economic and technological reasons for entering and contribut-

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

ing to open source (Bonaccorsi and Rossi, 2003; tinyurl.com/lfx847l). In addition to the possibility of a shortened development time (e.g., Dahlander, 2007; tinyurl.com/kg8wdd6), open source projects commonly report a wider adoption of their code (e.g., West, 2003; tinyurl.com/6s68jno) and receive more high-quality feedback and bug reports than closed source projects (see Schindler [2007; tinyurl.com/mv8eea9] for a comparison). Open source licensing also enables a faster average time from discovery to solution (Schindler, 2007; tinyurl.com/mv8eea9). Indeed, open source products have been often shown to be superior to their proprietary counterparts (e.g., Wheeler, 2007; tinyurl.com/r1yk). Furthermore, companies can see development of their product in directions they did not realize was significant to their users, as well as the development of features that are too far from the firm's core business to receive in-house funding for development. As an example, only two of the more than 20 language connectors for MySQL were programmed in house; the rest were developed and submitted by the community.

By joining an open source development effort, corporations can also influence the direction of its development. Furthermore, open source has been identified as a strategy for implementing long-term sustainable software systems (e.g., Lundell and Gamalielsson, 2011; tinyurl.com/n24dw4u). Open source can also be adopted as a competitive strategy, for example through making the functionality of a competitor's product freely available (Fitzgerald, 2006; tinyurl.com/al995aj). Open source can also be of value to companies that offer products other than software, for example by promoting open source in areas that facilitate the deployment of their hardware (Fitzgerald, 2006; tinyurl.com/al995aj).

## Open Source Licenses

A basic understanding of licensing is important for entrepreneurs and programmers alike. License choice decides what can be done with a program and what other programs (or, rather, licenses) it can and cannot be combined with. All open source licenses guarantee users the rights to use the program, access the source code, modify the source code, and redistribute the program in its original or modified form. However, beyond these basic rights, licenses differ in significant ways. Based on these differences, open source licenses are commonly divided into three main categories: i) permissive licenses, ii) weak copyleft licenses, and iii) strong copyleft licenses. The licensing requirements of copyleft licenses are only triggered upon distribution. This means that, for personal use, one can do largely whatever one wants with open source code, but if and when one distributes a program the stipulations of the license are triggered and must then be complied with. Note, however, that the AGPL license has some minor restrictions, which will be discussed later.

One of the most important elements of, and differences between, open source license types relates to a concept called *license compatibility*. License compatibility is a term used to describe the issue of which licenses can be combined. Particularly, from a business perspective, license compatibility considers which licenses can be combined with proprietary software. A further issue, though one of lesser interest, is that of the right to change the license, in particular whether one is allowed to change an open source license to a proprietary one. For businesses, this may be of interest as a source of free code. The issue of changing to a proprietary license splits the developer community into two camps. Those who are for it generally want to ensure (or at the very least do not mind) that their code is as valuable to corporate interests as possible. Those who are against it generally want to ensure that the open source project remains a freely available community good in perpetuity. The issue of license combining (including embedding) and license change is summarized in Table 1.

*Permissive licenses*

Permissive licenses allow a high degree of freedom to use and reuse (or fork) the code. It is not an extreme oversimplification to distil the permissive licenses down to the message: "here's the code, do whatever you want with it". (Commonly, one needs to distribute a copy of the copyright with the code, but in practice,

**Table 1.** Post-distribution rights of open source license types

| Rights | Permissive | LGPL | GPL |
|---|---|---|---|
| Can it be combined with proprietary programs? | Yes | Yes | No |
| Does it allow the license to be changed to a proprietary license? | Yes | No | No |
| Does it guarantee access to source code? | No | Yes | Yes |

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

this need not be more complicated than including a readme file.) In other words, it is possible to fork a permissively licensed program and make it closed source. (As an example, both Apple's OS X and iOS operating systems contain code that was copied from permissively licensed open source projects, most notably BSD: tinyurl.com/kffrf.) An issue which sets the permissive licenses apart from the copyleft licenses is that, once the source code is compiled, one does not need to distribute the original source code with the compiled (i.e., binary) version of the program. Among the more common permissive licenses are the Apache (tinyurl.com/kmenxch), MIT (tinyurl.com/3vfsyal), and BSD (tinyurl.com/lejoxn7) licenses.

*Weak copyleft licenses (LGPL)*
Weak copyleft licenses, such as the GNU Lesser General Public License (LGPL; tinyurl.com/mp4w4lw), can be combined with proprietary code, but cannot be relicensed under a proprietary license. So, although a firm's proprietary program can remain proprietary, even when combined with the LGPL, the LGPL-licensed program cannot be made proprietary. Furthermore, any modifications to an LGPL program must also be licensed under the LGPL. The Mozilla Public License (MPL; mozilla.org/MPL/) is also a weak copyleft license.

*Strong copyleft licenses (GPL)*
Much like the LGPL is synonymous with *weak* copyleft, the GNU General Public License (GPL; tinyurl.com/2459b5) is synonymous with *strong* copyleft. Hence, we will focus our discussion of strong copyleft licenses on the GPL. Although use of the GPL is in decline (Aslett, 2011; tinyurl.com/7ujq7sj), as of the writing of this article, it is still the most common open source license overall (Black Duck Knowledgebase; tinyurl.com/nl4z94t). The GPL requires any modifications to the code to also be licensed under the GPL. From a business perspective, the key issue to be aware of is that combining or embedding a program with the GPL necessitates the (re)licensing of all connected software so that it is also under the GPL. In practice, this means open sourcing any proprietary programs connected to a GPL-licensed program, and is therefore something many firms seek to avoid. Importantly, programs licensed under a GPL license cannot be re-licensed under a more permissive license (i.e., neither as LGPL or permissive).

A general comment regarding license change is that one can commonly change a license to a more restrictive license type but not to a more permissive one. Furthermore, only the permissive licenses can be changed to proprietary.

With the rise of cloud computing, a variation of the GPL license worth special mention is the Affero General Public License (AGPL; tinyurl.com/lzmmq8n). The AGPL differs from the GPL in that online use of a program is considered distribution, thus triggering the requirement for license compliance (i.e., source code access is required) even though a physical copy of the program has not been distributed. In other words, using an AGPL-licensed program in the cloud necessitates distribution of source code.

*Choosing a license*
Open source licensing is a more complex topic than can be covered in detail here. Furthermore, because legal precedent is rather limited, there are issues regarding licensing that are still subject to interpretation and that are coloured, among other things, by pragmatic versus ideological concerns. Thus, what may and may not be done under certain conditions is to some extent a matter of opinion. We recommend a close study of licensing before any final licensing decisions are made. For further reading, please refer to the links at the end of this article.

## On the Business of Open Source

Establishing a sufficient, steady income is a significant challenge in creating a company around open source software. Thus, although open source is a superior development model, there is no guarantee that one's program will make enough money to fund its continued development. Of particular significance to the business of open source are the questions of program ownership and location in the software stack, because these factors affect what business models one can choose from. In particular, the answers to these questions help decide whether one can employ what is arguably the most lucrative open source business model: dual licensing.

*Ownership of code*
A company or person that owns the rights to the code they develop can sell closed source copies of the code, which is a standard practice with proprietary programs. The dual licensing, business source, and (to a lesser extent) open core business models, which will be described in further detail later, require ownership of the code.

*Location in the software stack (and "embedded" programs)*
Most software relies on other software to run. This concept of software codependence is most apparent in the so-called software stack. On the top of the stack is the application: a word processing program, a photo ed-

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

itor, a game, etc. Digging deeper, one can find elements such as databases, middleware, and an operating system. It is not important for the purposes of this article to understand the layers or functions of a software stack; it is merely enough to know that such layers exist and that a program's location in the stack is significant to its overall importance to the stack. Programs higher up in the stack rely on programs lower down to function, but not the other way around. Whereas a word processor needs an operating system to be able to run, an operating system does not need a word processor for it to function. One way for an open source program to gain potential value is having other programs rely on it: by being embedded in the software stack and by being a required component for applications and other programs to function properly – or even run at all.

## Business Models

Although a business model can usefully be seen as something much more complex than merely a revenue source (e.g., West, 2007: tinyurl.com/dxsemd; Bailetti, 2009: timreview.ca/article/226), at its essence is the question of how the firm can create value for the customer while simultaneously extracting some of that value for itself (West, 2005; tinyurl.com/ov69jb8). For the purposes of this article, we make use of very broad brush strokes in our interpretation, using the term "business model" to indicate the way in which a company delivers value to a set of customers at a profit (e.g., Johnson, 2010; tinyurl.com/m9uf6xe). Recommended reading for more in-depth analyses of questions related to business models are offered at the end of the article.

The business models of open source can be divided in two main categories: those that require complete (or at least partial) ownership of the code and those that do not. Table 2 outlines the criteria for selecting an open source business model; however, it should be noted that these business models need not be mutually exclusive.

*Support contracts and services*

Support and services are closely related approaches; in fact, companies that provide one commonly also provide the other. Thus, although they could be separated, we have chosen to group them under one heading. The services business model is one in which income is generated by offering services in the form of, for example, training, consulting, or extensions development around an open source product. Companies that offer services will commonly also offer long-term support contracts, thereby achieving a more stable income than by merely focusing on one-off services. Two of the main challenges with the support and services approach are the lack of scalability and that the typical profit margin of 20–30% is not enough to pay for full-time developers for the project.

The availability of support and services is an important factor for customers (e.g., Shanker, 2012; timreview.ca/article/635) and can be considered a necessary element for software to become truly successful. Bear in mind that, although support should be offered, it need not be provided by the same company that develops the software. Examples of a support and services providers are Red Hat (redhat.com) and SkySQL (skysql.com). For more information on Red Hat's approach, see Suehle (2012; timreview.ca/article/635).

*Open core or commercial extensions*

Open core is a business model in which the core of a program is open source, with additional closed source features provided for a fee. Open core has gained much momentum over the past few years. However, it is an approach primarily focused on appealing to the venture capitalist rather than the end user (Prentice, 2010; tinyurl.com/pqpmptk). The economic rationale is clear-cut, but the reaction of the community and customers may not be as easy to estimate. Although pragmatic firm motivations are accepted by the community provided they comply with the rules of the community (Bonaccorsi and Rossi, 2003; tinyurl.com/lfx847l), some developers see

**Table 2.** Criteria for business model selection

| Criteria | Support and Services | Open Core | Business Source | Dual Licensing |
|---|---|---|---|---|
| Can I choose this business model even if I do **not** own the code? | Yes | No* | No | No |
| Can I choose this business model even if my program is **not** embedded? | Yes | Yes | Yes | No |

*Ownership is required for the closed source extensions

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

the open core approach a breach of those rules. The proponents of free software criticize it on ideological grounds and proponents of open source software criticize it on technical grounds, due to the restrictions to the development model caused by limited access to the code. From the perspective of the end user, open core forces vendor lock-in and is furthermore faulted with not delivering and sustaining the cost savings and flexibility of open source software (e.g., Phipps, 2012; tinyurl.com/9tjv8c9). Potential outcomes of adopting this model may include problems in attracting and maintaining developers (see Dahlander and Magnusson, 2005: tinyurl.com/88djuec; 2008: tinyurl.com/6w6k95q), or even the emergence of a competing fork (Nyman, 2013; tinyurl.com/mahze3o).

However, it should be noted that there are successful open core projects, which show that the approach can work. If considering an open core approach, it is worth bearing in mind that the more useful the core product is, the greater the potential community interest will be. Thus, making non-critical parts of the program closed will lessen the potential negative effect on developer interest in the project. A time-limited hybrid licensing (Sprewell, 2010; tinyurl.com/n8zeoqr), in which the closed source components of open core become open source after a 1–5 year delay, has been proposed to help meet the demands of both users and developers. However, we posit that the business source approach explained below may be a more mutually beneficial means to the same end. Examples of open core are not as easy to come by as the frequent discussion of the topic over the past few years would imply. Perhaps the best-known example is MySQL (mysql.com), which offered dual licensing of an identical product (a closed source and a GPL version) under its previous owners, but has changed to an open core approach for its free version after it was purchased by Oracle (Young, 2011; tinyurl.com/3hyxttc).

*Business source*

Business source is a business model that employs two different licenses with a time delay. In this business model, the source code is openly distributed and freely editable. However, for a set amount of time, a predefined segment of users (0.1–1% is suggested) have to pay to be allowed to use it. After this initial time period (3 years is suggested), the license automatically changes to an open source license. Business source is a new entrant in the field of open source licensing, which we first detailed in the June 2013 issue of the *Technology Innovation Management Review* (Widenius and Nyman, 2013; timreview.ca/article/691). It was created to help simultaneously meet the needs of both the open source community and the open source entrepreneur; being too restrictive in one's licensing can harm community growth, whereas being too permissive can harm business growth. Though a newly introduced concept, there are already reports of companies switching to business source, with both developers and owners pleased with the results (Widenius, 2013; tinyurl.com/mkurs58). For a more in-depth presentation of the business source approach, with a sample license, see Widenius and Nyman (2013; timreview.ca/article/691).

*Dual licensing*

Dual licensing is a business model in which a program is offered under two separate licenses, commonly one version under a copyleft, GPL-style license and another under a commercial, closed source license allowing for proprietary use (and combining with other proprietary software). Traditionally, the source for both versions is identical, except for changes in the copyright. Dual licensing is the best option for programs that are embedded, and for which one owns the code. The primary customers are companies who want to include software in their own packages, but who do not want to release their code under open source, as is required by the GPL. Its excellent scalability makes dual licensing the most potentially lucrative of the business models presented herein. The first ever program to adopt a dual licensing approach was Ghostscript (tinyurl.com/2p6zmt); MySQL (msql.com) – (before and during its ownership by Sun – was the second program to utilize this approach, and the first to use GPL as the open source license.

*Software as a service*

Software as a service (SaaS) is a fairly new business model in which connectors and application programming interfaces are open source but the server code they connect to is not accessible to the end user. For instance, one may use an application that can access certain data on a server, but not be able to access the actual source code (of, for example, the database management system) on the server one accesses. Although SaaS is not directly related to open source, it is included here because it can incorporate open source components. Examples of SaaS businesses are Salesforce (sales force.com) and Web of Trust (mywot.com); in building their service, they may use open source software on their servers, but this software is not distributed to their users.

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

## Managerial Implications

When deciding whether or not to start an open source project, the following managerial implications should be considered:

1. Before starting a new open source project, check if a similar project already exists. Participating in an active program is preferable to starting a new fork. If there are similar programs that have been abandoned, do some research to find out why they were abandoned. Repositories such as GitHub (github.com) and SourceForge (sourceforge.net) have a myriad of abandoned programs.

2. Find a company or a group of users that want to work with you to define the scope of the project. From the start, you will want to have users using the product while it is still in development.

3. Two of the most important decisions will be business model and license. If you are planning on relying on community participation, be mindful of their reactions to both business model and license choices. See the end of this article for further reading on community.

4. In choosing a business model, consider these questions: Do you want to concentrate on services or development? Do you plan to have a big community or work with a few big companies? Do you plan to take in investors? And, if so, what is your exit plan?

5. In choosing a license, consider these questions: What will your business model be? How much control do you want to have over the use (and potential forks) of your code? What kind of community do you want to attract around the product?

6. If you plan to rely on community participation, remember to use community-creating tools to reach and communicate with them: web pages, a forum or knowledge-base, email lists, bug system, build systems, source code repository, etc. You can start by hosting your project on GitHub, SourceForge, or another repository, but you will eventually want to host it yourself.

7. Significant enabling factors for creating a successful business around open source are ownership of code and embeddedness (a program's location in the software stack). These same factors also largely determine what business models one can choose from. Figure 1 provides a flowchart to help choose a business model based on ownership, embeddedness, and intentions for further development. If the flowchart recommends against starting a business, consider either partnering, or releasing the code (e.g., under an Apache or BSD license) for someone else to continue developing the software.

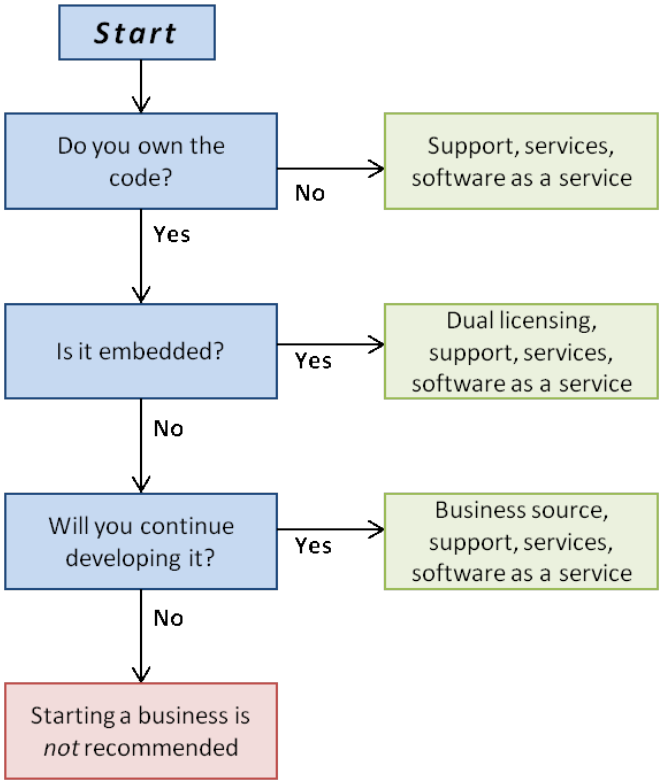


**Figure 1.** Flowchart for choosing an open source business model

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

## Conclusion

Through this primer, we have given a brief answer to the question: "How can one make money on open source?" To the uninitiated, financing a business based solely around the development of open source code may perhaps seem somewhat enigmatic. Although challenging, it is nonetheless possible. Our goal in this article was to clarify this enigma by explaining some of its most significant parts.

The possibilities for monetization of a program are dependent on many factors, and key among them are ownership of code, choice of license (including the issue of license compatibility), and location in the software stack. These factors in turn affect the choice of business model.

As a primer, this article will hopefully provide a useful introduction to the business of open source. It is not intended to cover every aspect of open source businesses in full detail, nor can it provide conclusive recommendations that will apply in every case. However, in Table 3, we have included a list of recommended reading for those that want to dive deeper into the topic.


## About the Authors

**Michael "Monty" Widenius** is the founder and original developer of MySQL and MariaDB. He has been an entrepreneur since 1979 and is the founder of MySQL Ab, Monty Program Ab, SkySQL, the MariaDB Foundation, and Open Ocean capital.

**Linus Nyman** is a doctoral researcher at the Hanken School of Economics in Helsinki, Finland, where he is researching code forking in open source software. A further research interest of his is free-to-play gaming. He also lectures on corporate strategy, open source software, and the new business models of the Internet age. Linus has a Master's degree in economics from the Hanken School of Economics.


**Citation:** Widenius, M. and L. Nyman. 2014. The Business of Open Source Software: A Primer. *Technology Innovation Management Review*. January 2014: 4–11.



**Keywords:** open, open source business models, open source software development, open source licenses, dual licensing, business source, open core, entrepreneurship

# The Business of Open Source Software: A Primer

*Michael "Monty" Widenius and Linus Nyman*

**Table 3.** Recommended reading

| Topic | References |
|---|---|
| Open source licensing | • Välimäki (2005; tinyurl.com/ahljzwu)<br>• International Free and Open Source Software Law Review (ifosslr.org) |
| Open source license selection in relation to business models | • Daffara (2011; timreview.ca/article/416) |
| License compatibility, compliance, and legality concerns | • Wheeler (2007; tinyurl.com/cbc579)<br>• Hammouda et al. (2010; tinyurl.com/bfp82mw)<br>• Lokhman et al. (2013; tinyurl.com/n64q3wm) |
| Popularity of various licenses | • Black Duck Knowledgebase (tinyurl.com/kp25s8s) |
| More on specific licenses | • Open Source Initiative (opensource.org/licenses)<br>• Free Software Foundation (tinyurl.com/4e7wm) |
| Open source and business models | • West (2007; tinyurl.com/dxsemd)<br>• Bailetti (2009; timreview.ca/article/226)<br>• Hecker (1999; tinyurl.com/28n7o3)<br>• Prowse (2010; timreview.ca/article/366)<br>• Suehle (2012; timreview.ca/article/513) |
| Matching licenses with business models | • Lindman et al. (2011; tinyurl.com/na3e6fd) |
| Business source | • Widenius and Nyman (2013; timreview.ca/article/691) |
| Partnership strategies | • Riekki-Odle (2010; timreview.ca/article/364) |
| Business models for companies partnering with an open source vendor | • Groganz (2011; timreview.ca/article/463) |
| Collaboration models between open source projects and their communities | • Noori and Weiss (2013; timreview.ca/article/647)<br>• Weiss (2011; timreview.ca/article/436) |
| Customer value propositions for corporate open source software | • Shanker (2012; timreview.ca/article/635) |
| Open source support and its requirements | • Peters (2007; timreview.ca/article/54) |
| Establishing a community | • Byron (2009; timreview.ca/article/258) |
| Participation architecture in corporate open source | • West and O'Mahoney (2008; tinyurl.com/66fly95) |