Inf Syst E-Bus Manage (2012) 10:5–17
DOI 10.1007/s10257-010-0149-x

ORIGINAL ARTICLE

The single-vendor commercial open course business model

Dirk Riehle

Received: 11 December 2009 / Revised: 19 April 2010 / Accepted: 6 May 2010 /
Published online: 24 November 2010

Abstract Single-vendor commercial open source software projects are open source software projects that are owned by a single firm that derives a direct and significant revenue stream from the software. Single-vendor commercial open source at first glance represents an economic paradox: How can a firm earn money if it is making its product available for free as open source? This paper presents the core properties of single-vendor open source business models and discusses how they work. Using a single-vendor open source approach, firms can get to market faster with a superior product at lower cost than possible for traditional competitors. The paper shows how these benefits accrue from an engaged and self-supporting user community. Lacking any prior comprehensive reference, this paper is based on an analysis of public statements by practitioners of single-vendor open source. It forges the various anecdotes into a coherent description of revenue generation strategies and relevant business functions.

Keywords Open source · Commercial open source · Single-vendor open source · Commercial open source business model · Dual-licensing strategy · Open core business model · Business model · Go-to-market strategy · Open source sales · Open source marketing · Open source product management · Open source licensing · Software engineering · Collaborative development

1 Introduction

Open source software is software that is available in source code form, can be modified by users, and can be redistributed even in modified form without paying the original owners.

D. Riehle (✉)
Friedrich-Alexander-University of Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany
e-mail: dirk.riehle@cs.fau.de

Open source is changing how software is built and how money is made. In 2006, open source software had a market share of 0.7% of the total packaged software market in terms of revenue (Software and Information Industry Association 2006); (IDC 2007). The prediction for 2008 was a market share of 1.1%. This data is underestimating the usage of open source software as it accounts only for paid-for open source software. According to a 2008 IDC report, less than 1% of all installations had third-party attendant services (IDC 2007), demonstrating that open source is being used significantly more widely than it is being paid for.

The total amount of work invested into open source software projects is growing at an exponential rate and can be expected to continue growing at this rate for a while before slowing down (Deshpande and Riehle 2008). In general, the size of individual open source projects tends to grow at a linear or quadratic pace (Koch 2005); (Godfrey and Tu 2001). The driver behind the overall exponential growth of open source is the exponential growth in the number of viable projects. Viable open source software is now available for any domain including business software, not just infrastructure software.

In many ways, the economic success of open source appears to be a paradox. How can companies make money of software they are making available for free? There are many answers to this question, as discussed in the next section. This paper focuses on one particular category of firms, called single-vendor commercial open source firms (Riehle 2007); (Capra and Wasserman 2008). Single-vendor commercial open source firms are firms that are the sole owner of a product they generate revenue from. Examples are MySQL, SugarCRM, Jaspersoft, and Alfresco. According to a recent Gartner report, by 2012 more than 50% of all revenue generated from open source software projects will come from projects under a single vendor's patronage, that is, from commercial open source (Gartner, Inc. Predicts 2009).

The benefits of single-vendor commercial open source stem from the creation of an active and engaged user community around the product while at the same time preventing the emergence of competitors from that community. In a nutshell, this community helps the company get to market faster, create a superior product, and sell more easily, all at a lower cost than possible for traditional competitors. In exchange, the company offers a professionally developed product of compelling value to the community that this community is free to use under an open source license.

The contribution of this paper is to comprehensively present the core properties of the business models underlying single-vendor commercial open source companies. Prior work typically addressed open source in general without special consideration for commercial open source. This paper reviews every relevant business function and how it works in a single-vendor commercial open source business model. Methodologically, the paper is based on the reception of interviews and presentations by practitioners of single-vendor commercial open source as well as the author's review of the behavior of commercial open source firms in the marketplace.

2 Prior and related work

Like the author of this paper, Capra and Wasserman make a fundamental distinction between commercial and community open source (Capra and Wasserman 2008). *Community open source* is open source software that is owned by a community or a legal entity representing the community. The community members typically don't derive direct revenues from the software but subsidize it from ancillary products and services. *Single-vendor commercial open source*, in contrast, is open source software that is owned by a single legal entity with the purpose of deriving revenues from the software. The next section discusses this distinction in more detail.

Various authors have provided summaries of how companies generate revenue from open source software. Watson et al. distinguish five models of software production and distribution (Watson et al. 2008). Three of these they call open source business models. The "corporate distribution" model encompasses the providers of software distributions, for example, Red Hat or SpikeSource. "Sponsored open source" is open source that does not generate revenue for the contributing companies, for example, Apache Software Foundation or Eclipse Foundation projects. Finally, "second-generation open source" is open source where supporting companies generate revenue from complementary services. This last category puts all revenue generating strategies into one basket without drawing distinctions between such different models as consulting and implementation services, e.g., JBoss, or license sales, e.g., MySQL.

Brian Fitzgerald introduces what he calls "OSS 2.0" (Fitzgerald 2006). He argues that prior to OSS 2.0 there were only two revenue models: "Value-added service-enabling," which created revenue from services around successful open source projects, and "loss-leader market-creating," which created revenue by upgrading users of a free open source project to a commercial more feature-rich version of the same software. OSS 2.0 now provides a more differentiated approach to the loss-leader strategy and adds two new strategies, "leveraging community software development" and "leveraging the open source brand."

Many more classifications of open source business models have been made. For example, the European Union's FLOSSmetrics project analyzed 120 firms which derive their main revenue stream from open source, and clustered these firms into six main categories (FLOSSmetrics 2007); (Daffara 2007).

Open source has been discussed from an economic perspective before, for example, by (Valimaki 2005), and others (DiBona et al. 2005). However, there is quite a gap between a general discussion of the economic properties of open source software and the specifics of commercial open source.

Perhaps the clearest account of commercial open source has been provided by Michael Olson in his discussion of the "dual-licensing strategy" of commercial open source firms (Olson 2005). Olson focuses on intellectual property ownership and the business strategies resulting from such ownership, most notably the right to provide the product under both a (free) open source license and a (paid-for) commercial license.

With the exception of Olson's work, none of the prior works focus on single-vendor commercial open source, and Olson mostly addresses its intellectual property aspects. In contrast, this paper comprehensively discusses the key properties of single-vendor commercial open source firms across all business functions.

3 Commercial versus community open source

Open source projects can be categorized into either commercial or community open source projects (Riehle 2007); (Capra and Wasserman 2008). Community open source projects represent by far the majority of projects. These two types of projects are distinguished by their different control and ownership structures.

- *Community open source* is open source that is controlled by a community of stakeholders;
- *Single-vendor commercial open source* is controlled by exactly one stakeholder with the purpose of commercially exploiting it.

3.1 Community open source

Examples of community open source projects with a diverse set of stakeholders are the Linux operating system, the Apache web server, and the PostgreSQL database. The source code of these projects is available under one and only one license, and anyone can enter the market and generate revenue from the project without being disadvantaged.

The contributors to community open source projects used to be the group of volunteer software programmers who developed the open source project. In this case, control is determined by ownership of copyright to the code in the project and related intellectual property as well as social structures such as having the commit (write) rights to the code repository.

Today, the volunteer communities of economically relevant projects are increasingly being represented or replaced by non-profit foundations such as the Apache Software Foundation or the Eclipse Foundation. Legally, many of the foundations have become the sole owner of the project; however, since the foundations are being controlled by their members, they still represent a community of stakeholders that run the foundations' projects.

As the previous section showed, there are many ways of generating revenue from open source software, including community open source. The three dominant ones are.

- consulting and support services around the software,
- derivative products built on the community project, and
- increased revenue in ancillary layers of the software stack.

More details are described in a related paper (Riehle 2007).

3.2 Single-vendor commercial open source

Single-vendor commercial open source firms build their business around an open source software project that they fully control, typically by having developed the software and never having shared control with third parties. This is done by owning the full copyright to the code and related intellectual property such as patents and trademarks.

According to Olson, the maintenance of full control over the project is crucial to the functioning of commercial open source (Olson 2005). One consequence is that single-vendor open source firms do not accept outside contributions to the code base. Or, if they accept them, they require a transfer of copyright from the creator to the firm to not dilute the firm's rights to the project. Augustin, however, argues that full ownership transfer is not needed and that receiving relicensing rights is sufficient (Augustin 2009).

Single-vendor open source firms differ from traditional software vendors by not only providing the product for free as an easily installable binary but also by providing it in source code form. By providing the source code under an open source license, such firms qualify as open source firms. However, because these firms own the copyright to the product, they are not constrained to only one license but rather they can relicense the software to customers as they see fit.

Typically, the free open source form is provided under a reciprocal license like the GPL to drive adoption but stall possible competitors. Paid-for versions of the software are then provided under a commercial license like traditional software vendors do. This is also known as the dual-license strategy of commercial open source (Lampitt 2008); (Olson 2005).

4 The Single-vendor commercial open source business model

In this paper, the term business model is defined as the combination of revenue generation strategies and supporting business practices and functions. This definition is a simplification over recent work defining electronic business models, for example, Timmers or Clarke (Timmers 1998); (Clarke 2004). The focus on traditional business functions, however, lets this paper stay close to the structure and behavior of real firms and leaves the creation of a more general abstraction to future work.

Practices and functions include sales and marketing processes, software production processes, and customer support processes. Thus, this paper first discusses what customers pay for and then how it is being produced and sold. It is understood that there is not just one but many commercial open source business models. Hence, this section focuses on those key properties that are shared across all or most commercial open source firms.

4.1 Revenue sources

Generally speaking, the products and services that customers pay for are not new. Bearden identified several categories of products and services that customers pay for

(Bearden 2008). Paraphrased by the author of this paper these are the four categories:

- *Core product.* Some customers pay for the software, simply because they cannot accept the open source license. Mostly, this is for legal reasons. For example, companies may pay for a commercial license to receive certification or indemnification or to embed the software into their products without having their own code touch open source code.
- *Whole product.* Commercial users pay for the utility derived from using the software. Increasingly, the free open source product does not provide the full utility, only a more comprehensive non-free commercial version does, as summarized by Asay (2009). To meet all requirements, commercial users have to upgrade from the free to the non-free version.
- *Operational comfort.* Customers also want to ensure that the software reliably fulfills its duty. Thus, they may be buying hot-line and technical support, subscription services to bug fixes, or real-time systems monitoring. There are many such non-functional requirements that companies may want to buy, many of which are specific to the software at hand.
- *Consulting services.* Finally, customers may want to pay for training, documentation, and implementation services.

Different names have been given to different aspects of single-vendor commercial open source. The term "dual-license strategy" refers to selling a commercial license to the project separate from the open source license (Olson 2005). The term "freemium model," a word play on "free" and "premium," refers to withholding features from a free version and making them available only in a commercial version. Lampitt coined the term "open core model" which combines the dual-license strategy with a freemium approach (Lampitt 2008). Asay puts it together in what he calls a "phased approach" to creating commercial open source businesses.

Selling a comprehensive product and providing operational support for it is not really novel. What is novel is how the software is being built and sold.

4.2 Business functions

Releasing a product's source code as open source can create an engaged user community which can impact the various functions of the commercial open source firm in multiple positive ways. This impact can create a significant competitive advantage over traditional (non-open-source) competitors. Thus, we first need to discuss.

- *Community management:* How to create and sustain an engaged community.

From the community then, the following benefits accrue, listed by business function:

- *Sales:* More and easier sales due to customer-side champions.
- *Marketing:* More believable and cheaper marketing through engaged community.

- *Product management:* Superior product thanks to broad and deep user innovation.
- *Engineering:* Superior product that is developed faster thanks to fast and immediate community feedback.
- *Support:* Lower support costs thanks to self-supporting user community.

Open sourcing also has its downside, for example, increased risk of getting sued for patent violations or of leaking important intellectual property. Also, catering to a non-paying user community and providing the public infrastructure for the community increases costs. The biggest danger, however, is that the firm's commercial product ends up competing with its own free open source project. This challenges product management as discussed below.

4.2.1 Community management

An engaged community is at the core of any working open source software project (Walker 2008). In community open source projects, this community comprises both users and developers, as the development work is carried out by the community itself. In single-vendor commercial open source, almost all of the core product development work is carried out by the commercial firm, with occasional contributions from the community (MIT 2008).

Commercial open source firms are interested in creating an active and self-supporting user community. Such a user community is key to achieving the desired business benefits. Commercial open source firms are also interested in creating an ecosystem of developers and service companies that extend the core product to increase its overall value proposition.

The main problem with seeding and growing a user community is the support cost. With closed source software, only the firm developing the software can provide the support. With a rapidly growing user base, the support cost can quickly outgrow any existing revenue or cash reserves.

Commercial open source firms address this problem by leading the community to become self-supporting. For this, they provide not only an easily available product, they also provide the source code to the product under an open source license. From a user perspective, this has the following benefits:

- *Free use.* Providing the product under an open source license grants free irrevocable usage rights; thus, users do not have to worry about having to pay down the road if they don't want to.
- *No lock-in.* Because the source code is available under an open source license, users can become independent of the commercial firm and hence (sometimes naively) think are not locked into the firm's future decisions.
- *Self-support.* Because the source code is open source, users can solve their problems themselves without having to resort to asking the firm, which might not want to provide that support to non-paying users in the first place.

From the firm's perspective, providing the product as open source accelerates adoption without increasing support costs. Specifically, it reduces hurdles to

adoption as potential users perceive no or little lock-in, and it makes it possible that the community becomes self-supporting once it reaches critical mass.

Walker as well as Capobianco provide some insights into how commercial open source firms seed and grow such communities (Walker 2008) (Capobianco 2008). On the most basic level, communities need a place to gather, and they need tools of communication. For this reason, most commercial open source firms host a software forge with integrated or ancillary tools like wikis, forums, and mailing lists. Much of the general advice on community building on the web applies, like aiding the construction of explicit social structures and rewarding members for good behavior (Kim 2000).

More specific to single-vendor commercial open source is the application of traditional marketing techniques: Firms need to understand the different sub-communities and their significance and target and support them accordingly. Specific programs aimed at different segments may become necessary. In general, community managers try to create win/win situations, which are easy to achieve as each constructive contribution by a community member not only benefits the product and the firm but increases that member's buy-in and his or her reputation within the community.

Each of the following business functions (sales, marketing, product management, engineering, support) has its own requirements and best practices of engaging with the community, and they are discussed in turn.

4.2.2 Sales

Augustin provides an account of the commercial open source sales funnel, as depicted in Fig. 1 (Augustin 2008). An eventual customer goes through a process of downloading, installing, and using the software, before they are recognized as a lead, become a prospect, and finally are converted from user to customer.

Compared with the traditional sales funnel,

- commercial open source has a different lead generation model, and
- it replaces the traditional pre-sales-to-sale activities with a user-to-customer conversion process.

Because the open source product is available for free, potential customers can download, install, and use the product without ever getting in touch with the commercial firm behind the product. At the same time, the firm can track via (typically voluntary) download registration and community forum activities who is actually using the product. Some products also provide usage information back to the firm.

Fig. 1 Commercial open source sales funnel according to Augustin (2008)

A lead analysis can then determine which of these users might be potential customers. More often than not, however, the firm will wait until a non-paying user steps forward and asks for a sales contact to purchase any of the services outlined in the revenue generation section. Thus, leads emerge from the existing user community, either voluntarily or by analysis. Of course, the commercial firm can still engage in a traditional sales cycle with non-using prospects as well.

In a traditional setting, a software firm's product is unknown to the potential customer except through marketing material. In the commercial open source setting, the potential customer is sometimes already using the product and hence is familiar with it. Thus, from the buyer's perspective, the open source project has significantly less risk associated with it. In this situation, there is likely to be an inside champion in the buyer's organization who downloaded and installed the product and is using it. These factors make a sale significantly easier than possible if the software firm had no prior relationship with the buyer.

As free open source software, commercial open source can make it into potential customer companies under the radar screen of the CIO. IT organizations may have strict rules in place not to install arbitrary software, however, in practice these rules are frequently circumvented (MIT 2008). Such early footholds in potential customer companies drive customer acquisition cost down significantly (Wittig and Inkinen 2004). Whether a significant percentage of potential customers is already using the product typically depends on the type of product. For some it is the case, for others it is not.

One role of the community is to support the potential buyer during the lead generation phase. For economic reasons, the commercial firm cannot provide this support on a broad scale, since only a small and hard-to-identify percentage of users might actually turn into customers. According to Taylor, conversion rates of 0.5–2% are common for single-vendor commercial open source firms (Taylor 2009). Since the user is not paying at this stage, voluntary community support is typically acceptable. As soon as the user is converted into a paying customer, professional support becomes available.

4.2.3 Marketing

Most single-vendor commercial open source software firms engage in traditional marketing: They advertise, they exhibit at trade shows, and they give talks (Capobianco 2008). What is new is that an engaged user community aids these marketing efforts. More specifically, the community makes marketing more effective and cheaper than possible without this support.

Marketing is more effective because non-paying users are credible sources of good testimonials. Thankful for a good product and the positive engagement in the community, users evangelize and market the product themselves without much support necessary from the commercial firm (Wittig and Inkinen 2004).

Free marketing can significantly reduce the marketing cost of a software firm, and hence create a competitive advantage over a competing traditional firm. According to Augustin, the ratio of sales and marketing (S&M) expenses to research and development (R&D) expenses in traditional software firms is 2.3

(and sometimes much higher), while it can be much lower for commercial open source firms (Augustin 2007). In the CRM space, for example, the S&M/R&D ratio of non-open-source firm Salesforce is 6, while Augustin estimates the S and M/R and D ratio of a hypothetical open source CRM vendor to be 0.6, suggesting significant savings in sales and marketing expenses (Augustin 2005). From a startup perspective, such a reduced cash burn rate increases the likelihood of survival for the commercial open source firm over the traditional firm.

4.2.4 Product management

Von Hippel has shown how user innovation can be a significant source of product innovation for any commercial firm (von Hippel 2005) and Shah has shown how this applies to open source software (Shah 2003). Mickos discusses how user innovation has aided the MySQL database (WSJ 2008); (MIT 2008): By providing the source code, firms encourage volunteers to innovate and contribute to the product for free. As mentioned, no such contributions will be accepted unless the rights are transferred to the commercial firm. Nevertheless, such user innovation can significantly improve the product, and if only through ideas rather than code.

An engaged community actively discusses strengths and weaknesses as well as future prospects of the open source product. Almost every commercial open source software firm provides the means to such discussions in the form of mailing lists, forums, and wikis on a company-run software platform. Thus, product managers can easily observe and engage with the community and discuss current and future features. This in turn brings product managers close to users and customers, aiding the product management process, for example, by helping feature definition and creation of a product roadmap.

In commercial open source, this community does not only include current customers but also current non-paying users and possibly even researchers and students. Thus, compared with a traditional community of customers, the breadth of perspectives in such discussions is much higher. This breadth of perspective in turns helps product managers understand new features and issues that have kept non-users from becoming users as well as existing users from converting to customers.

Many commercial open source firms distinguish between a free community version of the product and a paid-for enterprise edition. Product management faces the challenge of motivating enterprises to purchase a commercial license without annoying the non-paying community by withholding important features. Smart product managers address this problem by determining which enterprise features are irrelevant to the open source community and by taking a time-phased approach to making features available that are needed by both communities.

Product management benefits greatly from the immediate connection with the community, which provides ideas and feedback and keeps the product focused on its needs. Thus, the community helps the firm create a superior product.

4.2.5 Engineering

Obviously, volunteer contributions can speed up development. Also, an engaged technical community represents a potent pool of possible future employees that proved themselves before being hired, taking risk out of the hiring process.

More importantly, however, and similar to product management, are the benefits of direct and immediate feedback from the community. A single-vendor commercial open source company is likely to provide the latest release, sometimes a daily release, to the community, including potential bugs. An engaged (and fearless) community picks up the latest release and provides feedback to the company about bugs and issues they found, sometimes together with a bug fix. While such community behavior may appear as counterintuitive, it is nevertheless what practitioners experience (MIT 2008); (WSJ 2008).

The distinction between an experimental community edition and slower-paced but more stable enterprise edition in turn lets the commercial open source firm sell operational comfort, that is, the stable enterprise edition, more easily. Still, engineering management may not want these two versions to become too different from each other to avoid (re-)integration problems with outside contributions as well as unnecessarily redundant development on both versions.

4.2.6 Support

An engaged community supports itself by and large. Users who are not customers typically don't expect professional support from the commercial firm and are willing to utilize (and contribute to) community support. The commercial firm needs to aid in the support, but does not have to perform the bulk of the work. It would be prohibitively expensive for the commercial firm to provide support to all users, including those that don't pay. Thus, a self-supporting community is necessary to grow a large (non-paying) user base that might be converted into paying customers later. Paying customers can then receive full support from the commercial firm as part of their maintenance contracts.

The self-support activities of the community benefit the support activities of the commercial firm as well, reducing its cost. Specifically, engaged communities frequently develop and manage their own documentation, or at least contribute to and expand company documentation. User-maintained wikis and knowledge bases have become common. Thanks to the power of Internet search, many users, including paying customers, find it easier and faster to browse for problem solutions before turning to paid support in the form of phone calls or emails. Thus, the community takes some of the support burden of the commercial firm's shoulders, reducing the overall support expenses.

5 Conclusion

Open source is changing how software is built and how money is made. Industry analysts predict that by 2012 more than half of all open source revenue will accrue

to single-vendor dominated open source projects, called commercial open source. This paper comprehensively presents the core properties of commercial open source firms as well as their main business functions. Through a review of interviews and presentations by practitioners of commercial open source as well as other sources, this paper shows how at the core of the successful commercial open source firm is an engaged and self-supporting user community. From this user community, many benefits accrue, touching almost every business function of the firm: Sales are eased and increased through inside champions and reduced customer risk, marketing becomes more effective through better testimonials and active community support, product management more easily meets customer needs and benefits from user innovation, engineering creates a superior product faster and cheaper, and support costs are reduced. Thus, first order of business for a commercial open source firm is to create and sustain this community, a business function frequently non-existent or neglected in traditional software firms.

Acknowledgments I would like to thank Eliot Miranda, Wesley Mukai, Martin Stein, Jens Strücker and Christof Wittig for feedback on an early version of the paper. Then, I would like to acknowledge and particularly thank Jacob Taylor for helping me refine this paper further. Last but not least, I would like to thank Larry Augustin, Craig Hughes, Andrew Lampitt, Mike Moody, Mike Olson, Julio Toffoli as well as the anonymous reviewers for feedback in the final stages of the paper.

References

Asay M (2009) Building an open source business? some tips. CNet News (Feb 27 2009), http://news.cnet.com/8301-13505_3-10173773-16.html

Augustin L (2005) The next wave of open source: applications. Presentation given at GOSCON 2005

Augustin L (2007) A new bread of P&L: the open source business financial model. Presentation given at the 2007 open source business conference

Augustin L (2009) Ownership vs. relicensing rights. Personal communication

Augustin L Smoothing the on-ramp to commercial. Presentation given at the 2008 open source business conference (2008), http://www.infoworld.com/event/osbc/08/

Bearden R Tailoring an open source business model. presentation given at the 2008 Open Source Business Conference (2008), http://www.infoworld.com/event/osbc/08/

Capobianco F (2008) Building vibrant and sustainable communities. Presentation at the Open Source SIG of SDForum

Capra E, Wasserman A (2008) A framework for evaluating managerial styles in open source projects. In: Proceedings of the 4th international conference on open source systems (OSS 2008), pp 1–14. Springer, Heidelberg

Clarke R Open source software and open content as models for eBusiness. Presentation given at the 17th international eCommerce conference (Jun 2004), http://www.rogerclarke.com/EC/Bled04.html

Daffara C (2007) Business models in FLOSS-based companies. In: Workshop presentatioon at the 3rd conference on open source systems (OSS 2007), http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf

Deshpande A, Riehle D (2008) The total growth of open source. In: Proceedings of the fourth conference on open source systems (OSS 2008), pp 197–209. Springer, Heidelberg

DiBona C, Cooper D, Stone M (2005) Open sources 2.0. O'Reilly, Sebastopol

Fitzgerald B (2006) The transformation of open source software. MIS Q 30(3):587–598

FLOSSmetrics Open Source Business Models, http://robertogaloppini.net/2007/04/06/open-source-business-models-a-taxonomy-of-open-source-firms-business-models/

Gartner, Inc. Predicts 2009: The Evolving Open Source Model. Gartner, Inc. (2008)

Godfrey M, Tu Q (2001) Growth, evolution, and structural change in open source software. In: Proceedings of the 4th international workshop on principles of software evolution, pp 103–106. ACM Press, New York

IDC (2007) Industry Adoption of open source software, Part 2: project adoption. IDC
IDC (2007) Worldwide open source software business models 2007–2011 Forecast: A preliminary view. IDC
Kim AJ (2000) Community building on the web. Peachpit Press
Koch S (2005) Evolution of open source software systems—a large-scale investigation. In: Proceedings of the 1st international conference on open source systems (OSS 2005)
Lampitt A Open-Core Licensing (OCL): is this version of the dual license open source business model the new standard? http://alampitt.typepad.com/lampitt_or_leave_it/2008/08/open-core-licen.html
MIT (2008) An interview with Marten Mickos: the Oh-So-Practical magic of open-source innovation. MIT Sloan Manage Rev 50(1), 15
Olson M (2005) Dual Licensing. In: open sources 2.0, ch. 5. O'Reilly, Sebastopol
Perens B (2005) The emerging economic paradigm of open source, http://perens.com/Articles/Economic.html
Riehle D (2007) The economic motivation of open source: stakeholder perspectives. IEEE Comput 40(4):25–32
Shah S (2003) Community-based innovation and product development: findings from open source software and consumer sporting goods. MIT, Cambridge
Software and Information Industry Association (2006) Packaged Software Industry Revenue and Growth. SIIA
Taylor J (2009) User to customer conversion rates. Pers commun 4(7):213–229
The wall street journal online. Software firm is open for innovation. WSJ (Jul 7 2008), http://online.wsj.com/article/SB121494378874020445.html
Timmers P (1998) Business models for electronic markets. Electronic markets 8(2):3–8
Valimaki M (2005) The rise of open source licensing. Turre publishing
von Hippel E (2005) Democratizing innovation. MIT Press, Cambridge
Walker J (2008) Building vibrant and sustainable communities. Presentation at the open source SIG of SDForum
Watson RT et al (2008) The business of open source. Commun ACM 51(4):41–46
Wittig C, Inkinen S (2004) MySQL Open source database in 2004. Stanford graduate school of business, case: SM-124. Stanford