Open Source Software from Commercial Firms – Tools, Complements, and Collective Invention

to appear in: Zeitschrift für Betriebswirtschaft, 2004

this version: May 2003

Joachim Henkel[*]

Abstract

In recent years, open source software such as GNU/Linux, Apache, or Perl has received considerable attention. Commonly, the term is understood to imply that the software is developed by geographically dispersed volunteers. While this is true in many cases, corporate interest in and contributions to open source software have recently increased enormously, IBM being the most prominent example. The purpose of this paper is to analyze firms' benefits and risks from contributing to software that does not allow to charge licensing fees, and that can be freely used and even modified by anyone. Corporate contributors to open source software are classified into users of the software and sellers of complements. For both groups, one of the biggest potential benefits from contributing to open source software is that outside parties, individuals as well as other firms, join in the collaborative development of the software. The central question is if this benefit does indeed materialize. Case studies, literature analysis, and a theoretical discussion show that an "open source community of commercial firms" is indeed possible, provided certain conditions are fulfilled. Finally, a parallel to the phenomenon of "collective invention" is drawn.

Key words: open source software, complements, collective invention

JEL classification: D23, L86, M21

---

[*] I thank Eric von Hippel for his hospitality at MIT's Sloan School of Management. I am indebted to him, to Egon Franck, Dietmar Harhoff, Francis Hunt, Karim Lakhani, Sonali Shah, workshop participants at the III. GEABA Symposium (Berlin, October 2002), and two anonymous referees for helpful discussions and comments. I thank Justin Erenkrantz, Steve Howe, Damian Ivereigh, and Ben Woodard for valuable information relating to the case studies presented in this paper. Of course, all remaining errors are mine. Financial support from Deutsche Forschungsgemeinschaft is gratefully acknowledged.

Address: Institute for Innovation Research, Technology Management, and Entrepreneurship, University of Munich, Kaulbachstr. 45, 80539 Munich, Germany, tel.: +49 – (0)89 – 2180 2986, email: henkel@bwl.uni-muenchen.de.

1. Introduction

For several years now, open source software such as GNU/Linux[1], Apache, or Perl has enjoyed wide popularity, and made large headlines. What received particular attention is the fact that these software programs are largely developed by volunteers, which are in most cases even geographically dispersed. However, in recent years corporate interest in and contributions to open source software have increased enormously. The incentives for commercial companies to develop or contribute to open source software are the focus of this paper.

The most prominent example of a corporate contributor to open source software is IBM, who announced in September 2000 that it would spend 1 bn US$ in 2001 supporting the GNU/Linux open source operating system.[2] Early 2001, investment bank Dresdner Kleinwort Wasserstein released under an open source license its integration tool "openadaptor", that it had paid 5 mio US$ to develop.[3] In July 2001, Sun Microsystems added "Sun Grid Engine", a piece of software for distributed computing, to its already large offer of open source software (OSS).[4]

The above examples, and many others, relate to large firms, but contributions to OSS do come from firms of any size. Ximian Inc., e.g., was founded in 1999 by the initiator of the OSS "GNOME", a desktop for GNU/Linux and Unix. Ximian sells proprietary closed source software as well as conveniently packaged OSS related to GNOME, and continues to contribute to this OSS.[5] CodeWeavers Inc. runs a similar business model, selling "CrossOver Office", a proprietary add-on to the OSS WINE.[6] All changes and improvements made to WINE are returned to the open source community. Innominate Security Technologies AG, as a final

[1] As to the term "GNU/Linux", GNU is a recursive acronym standing for "GNU's Not Unix". See www.fsf.org for details. What is commonly referred to as the "Linux operating system" is in fact a combination of the Linux kernel (the development of which was initiated by Linus Torvalds in 1991) with (GNU-) software from the Free Software Foundation. Hence, it should more properly be called "GNU/Linux", as several authors do (e.g., Moody 2001).

[2] See vnunet.com, 2000/09/01 (www.vnunet.com/News/1118735). Later announcements even increased this sum.

[3] See ITworld.com, 2001/02/03 (www.itworld.com/AppDev/344/CW_2-3-01_opensourceinbanking/).

[4] See InformationWeek.com, 2001/07/30 (www.informationweek.com/story/IWK20010726S0009).

[5] See www.ximian.com.

[6] WINE is a free implementation of Windows APIs (application programming interfaces) on Linux, with the purpose to allow Windows applications to run on Linux without modification. See www.winehq.com

example, sells dedicated communication and pay servers running on GNU/Linux.[7] Like the other firms, Innominate releases the improvements it makes to GNU/Linux as OSS.

There are many more examples of commercial firms contributing to OSS (see Table 2 in the Appendix). As the two incidents that heralded – and precipitated – this shift one can safely identify Netscape Inc.'s release of Mozilla and IBM's adoption of Apache. In early 1998, Netscape Inc. put its browser software Netscape Communicator 5.0 into the open source domain, under the name of Mozilla (Moody 2001, p. 195). In June of the same year, IBM announced that it would abandon its own web server in favor of the OSS Apache (Moody 2001, p. 206).

The riddle why highly-qualified programmers would contribute to software development as unpaid volunteers has received considerable attention. Surveys (Hars and Ou 2002, Hertel et al. 2001, Lakhani et al. 2002, International Institute of Infonomics and Berlecon Research 2002) identified the following motives as the most important ones: learning and developing new skills; improving software for one's own use; sharing knowledge and skills; a general belief that code should be open; the feeling of obligation to contribute back to the open source community; and the joy and intellectual stimulus of writing code. Other authors stressed the effect on the programmer's reputation among peers (Raymond 1999) or on the job market (Lerner and Tirole 2002a, Hann et al. 2002), and the relevance of norms and trust inside the developer community (Osterloh and Rota 2001). Franck and Jungwirth (2002), in their discussion of contributors' motives, distinguish between the roles of maintainer, programmer, and bug fixer. Von Hippel (2002) puts the phenomenon into the broader context of "user innovation networks".

An even larger riddle than why hobbyists develop OSS one could see in the fact that also commercial firms contribute to OSS. They forego the opportunity to license the software they produce against a fee, and its use may provide an advantage to competitors. It is true that many of these open source activities are complements to the respective firm's commercial (i.e., non-free) offering, just as, e.g., Adobe's Acrobat Reader is to the Acrobat Distiller, both of which are proprietary and closed source. However, releasing software as OSS goes far beyond giving it away for free: by open-sourcing it, the releasing company gives the general public – in particular, its competitors – the rights to read the source code, to modify it, and to distribute

[7] See www.innominate.com/.

modified versions. Furthermore, complementarity to commercial offerings falls short of explaining a large number of cases, such as Dresdner's openadaptor.

The exploration of commercial firms' benefits and risks of contributing to OSS is the goal of this paper. Here, "contributions from commercial firms" is meant to imply that these open source activities are based on deliberate management decisions, as opposed to contributions from individual employed developers without (official) knowledge of management. I will not go into details of various business models related to OSS; on this topic, see Raymond 1999, Rosenberg 1999, and Hecker 1999. Instead, the focus lies on the economic reasoning behind freely revealing one's intellectual property (Harhoff et al. 2003). This, after all, is the startling thing: in stark contrast to firms' behavior in the field of OSS, mainstream economic theory holds that innovators should keep their knowledge secret, or protect it by patents or other means. Where this is not possible, innovators' ability to derive rents from an innovation should be diminished, which in turn should lead to an erosion of incentives to innovate. This logic obviously does not apply to the many individual programmers that have contributed to the development of GNU/Linux. Between them, largely free information exchange takes place. The question is to what degree this type of free exchange also happens between commercial firms involved in OSS, and if contributions from different firms build on each other in the way contributions from hobbyists do in other open source projects. GNU/Linux became the award-winning[8] software it is today by free and cooperative development between individual volunteers. I discuss whether comparable co-operation in the sense of "collective invention", as introduced by Allen (1983), can take place between corporate contributors to OSS.

The paper is organized as follows. Section 2 gives some background on OSS and clarifies the terminology. Section 3 discusses the benefits and risks for a commercial firm associated with releasing software into the open source domain. While several aspects are relevant to all contributors, some are specific to users of the software (hence the term "tools" in the title), others to sellers of complements. In section 4, I present three case studies where formerly proprietary software has been released as OSS by firms that are users of the respective software. This focus has been chosen for two reasons. First, excluding the, somewhat more obvious, benefits from selling complements allows to focus more clearly on the benefits derived from the open source development process. Second, success or failure of a firm's

[8] See Wheeler (2002) for a compilation of tests and comparisons between OSS, in particular Linux, and competing commercial offerings.

contributing to OSS are more easily observable when a formerly proprietary project is open sourced than when the firm contributes to existing OSS. Section 5 concludes.

2. Some Background on Open Source Software

Much of the activity that today is summed up under "OSS" predates this term considerably, in particular most GNU software (such as the GNU Emacs editor) and the GNU/Linux operating system. The term "open source software" was coined only in 1998, for one reason to replace the somewhat misleading term "Free Software" used until then. OSS is formally defined in the Open Source Definition, which in particular implies the following:[9] The license under which the software is distributed must allow re-distribution of the software by any party, without the requirement of licensing fees; the software's source code must be available; the license must allow modifications and derived works of the software, and the distribution of those; and the license must not discriminate against persons, groups, or fields of endeavor.

The most widely used open source license is the General Public License (GPL), crafted by Richard M. Stallman in 1985. Among all open source licenses, it goes farthest in protecting the openness of the software: modifications of GPL-ed software are required to be GPL-ed again, and combining GPL-ed software with proprietary code forces the whole combination under the GPL.[10] These provisions are intended to protect the users' rights and the freedom of the software ("free as in speech, not as in beer"), in line with the overall goals of the Free Software Foundation founded by Stallman in 1985.[11] BSD-style licenses, in contrast, are more open in allowing commercial use.[12] Software under a BSD-style license can be turned into proprietary software (and then be licensed out against a fee) by anyone, as long as the developers of the underlying OSS are credited. The most prominent examples of OSS under a BSD-style license

[9] See the Open Source Initiative's web site for the full definition: www.opensource.org/docs/definition.html (version 1.9).

[10] See Moody (2001, pp. 26-30) for a detailed account on the GPL.

[11] See the Free Software Foundation's homepage at www.fsf.org for details.

[12] "BSD" stands for "Berkeley Systems Distribution". For an overview over the different licenses see, e.g., Behlendorf (1999, pp. 164-165), Hecker (2000), Nüttgens and Tesei (2000), and Moody (2001). Categories different from OSS are public domain software, freeware, and shareware. See Nüttgens and Tesei (2000) or Feller and Fitzgerald (2000) for a discussion.

are the Apache web server and BSD-based operating systems (e.g., FreeBSD). Other open source licenses, such as the “Mozilla Public License” (MPL), combine different features of the GPL and BSD-style licenses and add some new ones.

OSS has made large headlines. Apart from the highly visible commitment of firms such as IBM, Netscape, and Sun, also governmental bodies are turning their attention to OSS. In January 2001, the United States’ National Security Agency unveiled a prototype for a “Security-Enhanced GNU/Linux”.[13] In December 2001, the British government published “Use of OSS within UK Government”, with an obvious strong tendency towards OSS.[14] In March 2002, the German parliament decided to replace Microsoft’s Windows NT with GNU/Linux as the operating system for its 150 servers.[15] Other facts about OSS are less extensively covered by the general press, but are very impressive nonetheless: The Apache web server had a market share of 63 percent of active sites in May 2003[16], and Sendmail, a piece of software that manages the delivery of email over the internet, is estimated to own around 75 percent of its market (Moody 2001, p. 243).

The high popularity of OSS has led to a slight confusion about what “open source” really means. While it is true that some aspects of OSS often occur jointly, this is by no means necessarily so. There are at least three aspects that can and should be distinguished:

(a) *OSS as software that comes under an open source license*. This implies that every recipient of the software can freely study its source code, modify it, and distribute it. This interpretation of OSS is the one implied by the Open Source Definition, and it is the one that is referred to in this paper.

(b) *OSS as software from the “open source community”*. This assumes that the software is developed by a large number of volunteers who do not get paid for their contributions (Raymond 1999), and who even perform mundane tasks such as user support (Lakhani and von Hippel 2000). While this is true for many open source projects, in particular for GNU/Linux, others are initiated and driven by commercial firms. And even for GNU/Linux,

[13] See www.infoworld.com/articles/op/xml/01/02/05/010205opswatch.xml.

[14] See www.govtalk.gov.uk/documents/OSS%20Policy%20draft%20for%20public%20consultation.pdf.

[15] See www.heise.de/newsticker/data/anw-14.03.02-012/.

[16] See Netcraft Web Server Survey at www.netcraft.com/survey/.

contributions from corporates, IBM and Red Hat in particular, are gaining importance. As to the number of contributors, most projects seem to have a rather limited number of developers (Krishnamurthy 2002), and even in the prominent large projects, most contributions tend to come from a relatively small core group (Dempsey et al. 1999, Koch and Schneider 2000, Hissam et al. 2001, pp. 18-22).

(c) *OSS as the result of the "OSS development process"*, meaning the collaboration of geographically dispersed developers without a formal hierarchy. This is an intriguing organizational aspect in particular of large OSS projects such as GNU/Linux: that developers who interact nearly exclusively over the internet can come forth with complex software of high quality. Several researchers address this aspect (Achtenhagen et al. 2003, Feller and Fitzgerald 2000, Godfrey and Tu 2000, Mockus et al. 2000, Franck and Jungwirth 2003, Metiu and Kogut 2001, von Hippel and von Krogh 2002, Garzarelli 2002), and attempts have been made to implement similar processes in commercial software development (Dinkelacker et al. 2001, Mockus and Herbsleb 2002).

Another aspect that merits mentioning is that several authors claim that OSS must be of high quality (in particular Raymond 1999), since it is open to review by anyone interested. While this logic seems to hold when interest is high – the quality of GNU/Linux, Apache, or the file server software Samba is proof of this fact (Wheeler 2002, Shankland 2003) – , it will obviously fail when interest is low (Hissam et al. 2001, pp. 14-16 and 22-30).

While (b) and (c) are fascinating aspects of some or even many OSS projects, they are not defining features. OSS may well have been developed inside a commercial firm, by a hierarchically organized team that works at a single location (see the case studies in section 4). After the software's release into the open source domain, features (b) and (c) may gain importance, but this need not be the case.

3. Benefits and risks for commercial firms contributing to OSS

3.1. Aspects relevant for all contributors

A commercial firm contributing to the development of some OSS can expect several benefits, but also faces some downsides. In the present section 3.1, I consider aspects which concern any company, be it a user of that software or a seller of complements. The following sections are dedicated to aspects specific to users (3.2) and sellers of complements (3.3), respectively. I start with a discussion of two central features of OSS. From these features, I derive the implications – benefits and risks – that contributing to OSS has.

Central features of OSS are that it is free for use, modification, and re-distribution, and that it is generally available for free or for a nominal fee. It should be noted that free (gratis) availability is a *common*, but not a *defining* feature of OSS. It is completely legal to modify OSS and keep the modified version secret, or to sell it to only one buyer. Commissioned custom development of OSS is an example for this.[17] However, "contributing" to OSS is meant to imply that the contribution is made public, and freely available to anyone.

The considerable impact that OSS has is caused by the interaction of these two aspects of being "free" – gratis and open. Figure 1 illustrates this. First, the zero price tag can be expected to increase adoption, other things equal. It furthermore implies dramatically reduced transaction costs, since no contracts, payments, counting site licenses etc. are required (which, more precisely, is not only due to the price, but also to some aspects of the open source licenses). Also, competing offers of OSS can conveniently be browsed and tested before one is adopted. Second, the far-reaching rights that a user of OSS is given both influence how he or she uses the software and, again, boost diffusion: The provision of the source code together with the right to adapt it to one's specific needs makes the software potentially much more valuable than a closed source, proprietary software of comparable functionality (Franke and von Hippel 2002). The lack of restrictions to distribution, finally, allows for an extremely low-cost way of distribution, namely, via free download over the internet.

The wider the diffusion, the stronger the effect which open-sourcing the software has on the contributor ("size of effect" in Figure 1). The "type of effect" is determined by others' use of the software (which may be desirable or not for the contributor), which in turn depends on the rights they are given and their characteristics (e.g., if they themselves are users or sellers of the software, competitors or complementors of the contributor etc.).

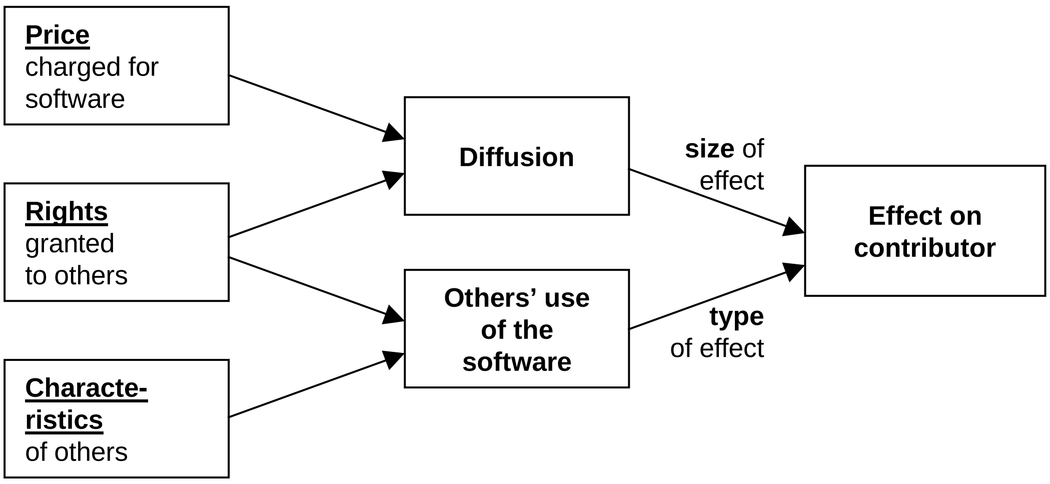

Figure 1: Illustration of how different features of OSS interact

Wide diffusion plus the fact that adopters can modify the software lead to the most cited potential benefit of OSS, namely, that other users work on the software, debug, improve, and extend it (Raymond 1999, Kuan 2000). Apart from qualitative improvements of the software, this reduces the maintenance burden for the contributing firm. For example, both Hewlett Packard and Sun named the objective "to increase collaboration in the development process" as the motivation behind putting software into the OS domain.[18] Obvious conditions for these benefits to materialize are that the software is of sufficient general interest, and that among the potential adopters there are users skilled enough to work on the software and improve it. This condition will more likely be fulfilled for software that requires professional users, such as server or infrastructure software. However, even if public interest in a certain software turns out to be low, the mere fact that outsiders *might* look at the code should have a disciplining effect on programmers which enhances the software's quality (cleaned-up code, improved structure, better documentation).[19]

Setting a standard is another possible benefit that can accrue both to users of a certain software and to sellers of complements (Harhoff et al., 2003). This is particularly important for software that relates to infrastructure and tools (Raymond 1999, p. 192). The main conditions for this benefit to materialize are that the software is of interest to a large share of the market, and that

[17] An interesting case is the European Union's feasibility study of pooling and releasing as OSS commissioned software that has been developed for governmental use. The project aims at both cost reduction, quality improvement, and improved co-operation between different countries (Schmitz and Castiaux 2002).

[18] ITworld.com, 2001/07/23 (www.itworld.com/AppDev/344/IDG010723hp_sun/pfindex.html).

[19] It is true that software from the "typical" OSS community is often not well documented. However, the argument refers to *commercial firms* releasing software as OSS, which will care more for the potential negative effects of insufficient documentation. See Hamerly et al. (1999, p. 199) for the case of Mozilla.

it exerts network externalities. There are two aspects to standard setting: first, setting any standard at all; second, setting a standard which is advantageous to oneself. On the first point, OSS has the strong advantage of offering a level playing-ground to all players, in particular if its origins do not lie with a particular firm.[20] On the second point, the more the proposed standard favors the would-be standard-setter, the more difficult it will be to achieve general adoption. In particular, the nature of OSS implies the risk that other players modify the intended standard and attempt to diffuse their own, modified version. However, so far this right seems to be rarely exercised.[21]

A more technical aspect is that, when a company distributes software that builds upon GPL-ed OSS, then it is required by the license to also release its own changes and improvements, under the GPL, to the recipient.[22] This applies, e.g., to firms employing embedded GNU/Linux in devices such as cellular phones or PDAs (Henkel 2003), or to distributors such as Red Hat or SuSE. Phrased differently, the commitment to release one's developments implies the advantage of being able to use GPL-ed software. This can mean considerable savings in licensing cost, transaction cost, and time, compared to writing the required software from scratch or licensing it from commercial suppliers. Of course, the benefits might not be great enough, such that firms prefer not to use OSS in the first place. In case the OSS to build upon comes under a less restrictive license (e.g. BSD), open sourcing one's own developments will not be required. This would be the ideal case for a developer; however, most OSS is in fact licensed under the GPL (Lerner and Tirole 2002b).

In a similar vein, the need to appear a "good citizen" in the OS community may be another reason why employing OSS requires the respective firm to contribute back changes and

[20] Moody (2001, p. 292) quotes IBM's Irving Wladawsky-Berger on the prospects of turning Linux into an industry standard: "The fact that it's not owned by any one company, and that it's open source, is a huge part of what enables us to do that. If the answer had been, well, IBM has invented a new operating system, let's get everybody in the world to adopt it, you can imagine how far that would go with our competitors." But even if a would-be standard comes from a particular firm, a credible commitment to keep it open source will have a reassuring effect on adopters, in a similar manner as licensing reassures adopters of a network good of not being exploited by a monopolist (Economides 1996).

[21] Programmers would rather try and convince the program's maintainer to include a change they made into the program's next "official" release. Spreading one's own modified version, though allowed by the open source licenses, is strongly suppressed, since most contributors want to avoid the risk of "forking" of a project into different, incompatible versions (Raymond 1999, p. 87).

[22] "Each time you redistribute the Program [...], the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions." (GNU General Public

improvements. If the software under consideration is of interest only to a small number of firms, then “being a good citizen” translates into reciprocity between these firms. In addition to building reputation as a good OS player, releasing high quality OSS will also improve the company’s technical reputation. Finally, OSS development may make the job description for developers in the in-house IT-department more attractive. Open source is by many programmers regarded as “cool”, and this was in fact one of the main motives for Dresdner Kleinwort Wasserstein to release openadaptor as OSS (see section 4.2).

An obvious downside of putting software into open source is that one loses most intellectual property (IP) rights to it. This drawback is most relevant for “permissive” licenses such as the BSD, since these allow a competitor to make the software proprietary and commercialize it. In contrast, the GPL protects the originator of the software against such exploitation. Another downside is that one can not make software OSS and at the same time license it out (for a fee), apart from cases where a different license is applied to a proprietary and an open source version of the software (Behlendorf 1999). If selling is an option (which is often not the case for a user-innovator, see 3.2), then the foregone profits from selling must be traded-off against the benefits of open-sourcing. Finally, just “throwing out” software into the open source domain is unlikely to yield strong benefits to the originator, and might even lead to bad publicity. Rather, the release should be well prepared – code has to be cleaned up and documented, modules licensed from third parties might have to be replaced, and it must be checked that the code does not infringe third party IP (Hamerly et al. 1999, p. 199) – and the software should be maintained afterwards. Both causes cost to the originator, which must be taken into account.

The relative importance of the pros and cons listed above and in the following subsections varies strongly between projects, and in particular between firms that open source formerly proprietary software and those that contribute to existing projects. While a detailed comparison is beyond the scope of this paper, it is obvious from the above that the originator of an OSS project faces greater risks, but also greater potential benefits than a contributor.

License, Version 2, June 1991, www.gnu.org/licenses/gpl.html.) It is sometimes said that the GPL requires users of GPL-ed software to release all modifications they made, also those made for internal use. This is not correct.

3.2. Aspects of contributing to OSS specific to users

Raymond (1999, p. 142) argues that 90 percent or more of all software code is written for use value, not for sale value. Hence, making such software OSS would not threaten any revenue streams. However, competitors might use the software, or might obtain insights into the developing firm's business processes.

As to the first point, selling software that originally had been developed for internal use is at least a theoretical possibility. However, the required change of functional role – from user to seller and manufacturer – is in most cases hard to achieve (von Hippel 1988). Even if the company is active in the software sector, it will still usually not be in the business of selling the kind of software it uses itself. It will most likely lack complementary assets such as a service organization, sales channels, and a complete product line (Teece 1986). Hence, transaction cost of finding a market for the software etc. will in most cases be too high to make selling it worthwhile.

The second potentially negative aspect of giving software into the open source domain is the risk of losing competitive advantage. How relevant this point really is depends on four factors: (a) the degree of competition; (b) relation of the software to the domain of competition; (c) availability of alternative software; and (d) specificity of the software to the firm (Harhoff et al. 2003). Obviously, the stronger competition (a), the more damaging an open source strategy will be, other things equal. However (b), if competition takes place mainly in the domain of, say, product quality, brand, or customer service, while the OSS under consideration leads to cost savings in development, then the loss of competitive advantage from open-sourcing is rather limited (Hamel et al. 1989). This point will in many instances apply to tools and infrastructure software, in accordance with the point raised in section 3.1. If the software is not even specific to the industry, but to some other feature of the company (e.g., its location, its workforce, or its technical infrastructure), then aspects of competition become even less relevant. Finally, when alternatives to the software under consideration are easily available (c), then there is little to be gained from keeping it secret. The points (a) to (c) are adapted from Schrader (1991), and follow the same logic as the results he obtained in his analysis of information trading in the U.S. steel industry.

The third aspect, insight into business processes, is strongly related to specificity of the software to the respective firm's needs (d). On the one hand, no competitor will gain a competitive advantage from very specific software (which also implies, however, that no-one will join in the development). On the other hand, competitors, suppliers and customers might learn about the firm's business processes, which usually makes CEOs uncomfortable. However, these fears may be exaggerated (Raymond 1999, pp. 154-155): First, well-written software should be separated in such a way from the data it manages that from the software alone one can not conclude too much about the processes. Second, as for security concerns, the most secure processes are generally considered to be open ones.[23] Furthermore, the nature of the software's firm-specificity matters: as in IT outsourcing decisions (Picot and Maier 1992), idiosyncratic legacy software has to be distinguished from strategically important systems tailored to the firm's needs.

3.3. Aspects of contributing to OSS specific to sellers of complements

The GNOME desktop is a piece of software that runs on GNU/Linux and several other Unix operating systems.[24] The project, started in 1997, was partly a reaction to the widespread criticism that GNU/Linux was not sufficiently user-friendly to reach a wide diffusion. Today, many distributors of GNU/Linux as well as other Unix variants ship their software and/or hardware with GNOME. Among them are the distributors Red Hat, SuSE, and TurboLinux, and also the hardware manufacturers Compaq, IBM, and Sun. For these firms, contributing to GNOME makes sense because they need it as a complement to their (paid) commercial offering. Since they require GNOME to a certain degree customized to their specific needs, free-riding is not fully possible. In addition, especially the pure distributors have to make sure they adhere to the norms of the open source community (Osterloh and Rota 2001), since they depend on it to a much larger degree than hardware sellers. For Ximian, the company that stands at the center of GNOME development, its contributions to GNOME are complementary to its offer of proprietary software, conveniently packaged OSS, and services.

IBM is active in many open source projects. Apart from GNU/Linux, its commitment to the Apache web server is the most prominent one. It was also its first one – the announcement that

[23] Of course, when a firm uses software that does have security holes in it, then taking it into the open source domain is risky (Hissam et al. 2001, pp. 37-43). On the other hand, it implies the opportunity that someone with good intentions points out the security flaws and helps fixing them.

IBM would ship the Apache web server together with its WebSphere Application Server, and offer "commercial, enterprise-level support" for it was made in June 1998.[25] The rationale behind the move was that IBM needed a web server to complement its offerings. Since its own web server software had gone down to a market share of less than one percent (Moody 2001, p. 206), it seemed like a natural choice for IBM to switch to the widely adopted, open source alternative Apache, and to support its further development.

Using GNU/Linux in embedded devices such as machine controls has become an attractive option (EDC 2002). A number of dedicated embedded Linux firms are active in this field, e.g., MontaVista, TimeSys, and Sysgo. They typically sell support, customized development, or complementary proprietary tools and applications. For a variety of reasons, they release changes they make to the code, thus collaborating informally in the further development of embedded Linux (Henkel 2003).

The above examples are among a number of business models that are or could be built around OSS (Raymond 1999, Rosenberg 1999, Hecker 1999). Most of them imply that the respective firm develops OSS, while making money selling complements to it. These can be hardware (e.g., in the case of IBM or Sun), proprietary software (e.g., Netscape's server software, CodeWeaver's CrossOver Office), support service (e.g., from IBM or Red Hat), or the distribution service of existing OSS (e.g., Red Hat, SuSE). Effectively, these strategies amount to cross-subsidizing the production and distribution of OSS with the proceeds from the sale of a complement.

The condition for such a strategy to work is that the complement is tied closely enough to the OSS, such that improvements to the latter have the desired positive effect on sales of the former. Obviously, the tie is the closer the more specific the OSS is to the complement. In cases where a certain piece of OSS is bundled with a proprietary offer (e.g., Apache with WebSphere, or GNOME with Sun's Solaris operating system), the contributor benefits from improving the OSS when the proprietary offer has sufficiently large sales figures. On the other hand, when a firm's contribution to OSS is intended to induce adopters to purchase the complement

---

[24] See www.gnome.org/faqs/users-faq/index.html#WHATIS. KDE is an alternative OSS desktop software that also enjoys wide popularity, in particular in Germany and Europe (see, e.g., www.kde.de). I focus on GNOME because there is a commercial firm at the center of its development.

[25] See Moody (2001, p. 205-212). IBM's active role is underlined by the fact that four of the 38 members of the Apache Software Foundation are IBM employees (see www.apache.org/foundation/members.html).

separately, care has to be taken that the complement is not provided cheaper or in a better quality by competitors (see Behlendorf 1999).

There is another possible benefit that merits mentioning. The motive to weaken a competitor or to decrease dependency on a supplier (in particular, Microsoft) certainly plays a role for some firms supporting GNU/Linux, as well as for Sun in its decision to release OpenOffice as OSS and to support its further development. Table 1 summarizes benefits and risks of contributing to OSS.

	all contributors	specific to users	specific to sellers of compl.
bene-fits	improvement by others reduced maintenance cost programming discipline standard setting GPL-ed OSS can be used attraction for programmers reputational gains - technical skills - “good OS citizen”	decreased dependency on proprietary software	increased sales of complementary goods pricing pressure on competitors
risks/ down-sides	loss of intellectual property cost of maintaining OS project (for maintainer) forking	loss of competitive advantage insights for competitors into business processes	loss of sales opportunities

Table 1: Possible benefits and risks for commercial firms of contributing to OSS

4. Case Studies

Benefits that commercial firms derive from contributing to OSS are the focus of this article, and in particular those benefits that relate to improvement of the respective software by other parties. For this reason, the following case studies look at software that firms had developed for internal use, and then made open source. While cases in which contributions to OSS are made in order to promote sales of a complement are more frequent, the possible benefit of joint development is more clearly visible when the contributor is solely a user of the software.

4.1. CEPS – Cisco Enterprise Print System[26]

CEPS is software for the management of large printer networks. It has been developed at Cisco Systems Inc. for internal use, and is described on the web site as “a collection of tools and utilities designed to work together to create a highly scalable, robust printing environment for a medium to large corporation.” Two programmers inside Cisco advocated the release of CEPS as OSS. They checked with management and the legal department, and the release was authorized. Finally, in 1997, one of the programmers offered it for download on his private website. However, even though the move had received management’s blessing, the programmer got upset mails from other managers in the following days, for whom it was “completely unthinkable to take software outside” and to “give away” the company’s intellectual property. Still, CEPS remained OSS. The project is hosted by the widely used OSS repository Sourceforge.net since March 2000. CEPS is licensed under the GPL.

It was never intended to sell the software, and neither was this a viable option at any time. Hence, Cisco did not lose sales value by turning CEPS into OSS. A loss of competitive edge vis-à-vis a competitor could also safely be excluded. First, CEPS is typical infrastructure software that does not contribute to a firm’s differentiation. Second, the software is not even specific to Cisco’s industry, but to its printer network environment. Initially, a certain risk was perceived that Cisco might be held liable for damages that a possible malfunctioning of CEPS might cause other adopters. However, the GPL clearly excludes this type of liability.[27]

Hence, there were nearly no downsides of releasing CEPS as OSS, apart from the programmers’ opportunity cost. On the other hand, the benefit of external development support was limited. The mailing list shows a relatively low level of activity, with a total of 214 messages over 39 months, and 26 messages in the first three months of 2003.[28] The maintainer does receive emails from companies interested in CEPS “every couple of months”. A few have

[26] The study of this example has been instigated by its mentioning by Raymond (1999, pp. 157-159). It draws upon interviews with Damian Ivereigh and Ben Woodard in June 2002 (both from Cisco), to whom I am very grateful, and an analysis of the web site at ceps.sourceforge.net. Quotes, unless otherwise stated, are taken from interviews with Damian Ivereigh.

[27] See Metzger and Jaeger (2001) and www.gnu.org/licenses/gpl.html.

[28] See sourceforge.net/mailarchive/forum.php?forum_id=6805.

tried to install CEPS, and at least two companies adopted it. Contributions to the code base from outside Cisco are relatively rare. Those that do come are from other companies, not from the prototypical open source hobby developer. It seems, though, as if activity around CEPS has picked up since the time of the interviews: the CVS (concurrent version system), a commonly used tool in open source development, shows a relatively high level of activity since September 2002, with 334 messages in nine months.

Still, even though the maintainer found outside contributions helpful, the external side of CEPS has been a somewhat limited success. One reason for this certainly was that "CEPS fills a very specific niche and there may not be other people who have the same niche to be filled." Another likely reason is that, initially, the documentation was not very elaborate, so that "it is really hard to install". Documentation, web site, and installation procedure were improved around the end of 2002. Hope is that the re-launch will re-invigorate the external community.

Internally, however, the move of CEPS to open source is considered a big success by its maintainers. The fact that CEPS is OSS works as a commitment vis-à-vis suppliers of related software, who could be convinced more easily to make their source code available to Cisco. Since developers value the availability of source code very highly[29], this was an important, even though unexpected, benefit of going open source. Another indication that the move to open source was also more generally considered a success is that by now a formal process within Cisco has been established for releasing software as OSS.

4.2. Openadaptor[30]

The software openadaptor has been developed internally at the investment bank Dresdner Kleinwort Wasserstein (DrKW) in London. It offers "an infrastructure for expediting the

[29] In the words of one of the maintainers, when using OSS, it is possible for developers to guarantee "we will get it working", "we know we can do it". A colleague is working on interfaces between Windows 2000 and CEPS. "The amount of work required is two to three times as much as for an equivalent open source environment. One can't trace into libraries [...] runs against a black wall when debugging under Win 2000."

[30] This case study is based on an interview with Steve Howe from Dresdner Kleinwort Wasserstein, one of the maintainers of openadaptor, to whom I am very grateful (2002/08/28; unless stated otherwise, quotes are taken from this interview); an analysis of the web site www.openadaptor.org; a presentation given by Dresdner Kleinwort Wasserstein at the O'Reilly Open Source Convention in San Diego, 2001/07/26 (www.openadaptor.org/docs/oreilly_strategy_talk.ppt); and the project's description on Collab.net's web site (www.collab.net/media/pdfs/drkw_success.pdf).

connection of disparate e-business systems."[31] DrKW, as other banks, uses a wide array of software systems dedicated to different tasks. Among them are systems which price equities or derivatives; others amalgamate information from the various trading systems and business lines; and finally, there is the settlement system. Openadaptor interconnects these different IT systems, internally as well as to customers.

DrKW released the software as OSS in January 2001, under a license similar to a BSD license. The initiative came from two programmers, who argued the release would serve as an advert for DrKW. The company had been to some degree familiar with the open source process, since openadaptor had followed open source practice even before it was officially released as OSS: "We shipped source code to our customers all along, inviting them to fix bugs and contribute enhancements, and they did."[32] In early 2002, the copyright and related IP rights were donated to "The Software Conservancy", a non-profit organization to support OSS.[33] Adoption by other banks was considerable. "Definitely at least ten" have adopted it, among them Deutsche Bank and JP Morgan, and "about another 20" have likely done so.

Selling of the software was never envisioned: "[There is] no threatened revenue stream [...] We aren't primarily a technology company."[34] Furthermore, the competitive effect from others' use of openadaptor was considered less important than the benefits from open-sourcing it: "We knew we were helping our competition as well, but we didn't mind that since we were improving the lot of everyone involved, including our customers and ourselves."[35] And even though openadaptor is considered high quality software, all competitors to DrKW are basically using similar packages. Hence, it gave only very limited competitive advantage to keep it proprietary. The maintainer put it this way: "If everyone is doing the same sort of development work in the City, and obviously within New York and Tokio, then why not pool it, because it makes very little sense to waste time and effort on doing something which is basically bread and butter for most banks."

[31] See www.collab.net/media/pdfs/drkw_success.pdf.

[32] See www.collab.net/media/pdfs/drkw_success.pdf.

[33] See www.collab.net/news/press/2002/softwareconservancy.html.

[34] See www.openadaptor.org/docs/oreilly_strategy_talk.ppt.

[35] See www.collab.net/media/pdfs/drkw_success.pdf.

Outside contributions to the development of openadaptor are frequent. Between January 2001 and August 2002, about 100 bugs have been reported; roughly 100 more have been reported and identified within the code; and about 10 to 20 have been reported and fixed. In addition, there have been 20 to 30 cases where people made improvements to satisfy their own specific needs, and returned them back to the openadaptor foundation. While these numbers appear small compared to those of popular projects such as GNU/Linux, one has to bear in mind that openadaptor is, for obvious reasons, much less widely deployed. The prototypical hobby programmer who contributes to popular projects is unlikely to work on openadaptor, because he or she can not *use* the software (which is an important motivator, see Lakhani et al., 2002). Instead, the majority of contributors work for either banks or enterprise integration consultancies.

Interestingly, the standard opinion that intellectual property must not be given away does not only concern the initiator of the OSS, but also the contributors. Usually, banks in the UK are automatically assigned the copyright to all the code their employees write, whether they do it in their spare time or at work. For that reason, openadaptor receives "a lot of anonymous contributions."

DrKW benefited also from wider adoption of openadaptor because of network effects. The software not only links different internal systems, but also systems between different companies. A wider adoption of openadaptor, driven by the fact that it is open source and available for free (see Figure 1), benefits all firms that employ the software, and in particular DrKW. This was rather important in early 2001 because of the uptake of electronic commerce and electronic trading.

In addition to the above more general benefits, two other ones were relevant for DrKW. In late 2000, "it was very difficult to employ competent developers, because everybody was interested in joining start-ups and getting their share options." Turning openadaptor into open source "acted as a kind of advert [...] we are a bank but we do really cool stuff [...]" The strategy succeeded, and DrKW received considerable positive headlines. The second benefit relates to security. Customers as well as competitors in a sensitive field such as investment banking need to trust the software they are using. It has to be clear and verifiable that there are no backdoors, and that no sensitive information is passed on to the originator of the software. Open source software is obviously ideally suited for this purpose.

Operationally, the project seems very well done, with a professional looking web site. The DrKW developers use email addresses "...@openadaptor.org", which certainly helps to create a less competitive atmosphere. Despite the already high degree of professionalism, the maintainer still saw room for improvement in aspects such as documentation ("if people could use it more easily, then they could contribute more easily"), and did update the documentation in late 2002.

4.3. Flood

Flood is a tool for web site testing, more precisely, a profile-driven HTTP load tester. It was released into the open source domain by eBuilt, Inc. in mid 2001 and has subsequently become a project of the Apache Software Foundation.[36] As such, it is licensed under the Apache license, a BSD-type license. Flood contains about 5000 lines of code, mostly written in C, and employees at eBuilt spent around five man-months developing it. Hence, compared to the two cases presented above, Flood is a relatively small project. It is used in-house by eBuilt to performance test their own projects.

Flood was planned to be an open source project from the outset, since eBuilt realized it would not make money off the tool by selling it. In contrast to the cases of CEPS and openadaptor discussed above, for Flood there was no initiative by programmers required to release it as OSS. The management's attitude was not characterized by the default position "we don't give away our IP", but was instead very much open towards OSS. This is understandable, given the culture of eBuilt as a young (founded in 1999) and comparably small company (around 100 employees) with a strong focus on technology.[37]

Community contributions to Flood since it has been made OSS include patches that make it work on Microsoft's Win32 architecture, and several other patches. With 90 messages in the developer mailing list in the month of September 2002, the project can be considered relatively active. Hence, eBuilt does benefit from community contributions, and the loss of competitive advantage is most likely small.

[36] See httpd.apache.org/test/flood/. I am indebted to Justin Erenkrantz of the Apache Foundation for details on Flood.

[37] See www.ebuilt.com/who/who.html.

5. Summary and Discussion

In this paper, the reasons why commercial firms would contribute to OSS have been analyzed. The most interesting one is the prospect of collaborative development jointly with other firms or individuals. In addition, several other potential benefits have been identified: reduced maintenance cost, the possibility to set a standard, reputational gains (for technical skills as well as for being a “good citizen” in the open source community), increased job attractiveness for programmers, the possibility to use GPL-ed software for one’s own developments, pricing pressure on competitors, and the chance to increase sales of complements to the respective OSS.

In deciding if to contribute to OSS or not, a firm must weigh the above benefits against the potential downsides (which mainly apply to firms that release their own software as OSS, as opposed to contributing patches to an existing OSS): a loss of intellectual property, the risk of forking, the cost of maintaining the open source project, a potential loss of competitive advantage or sales opportunities, and possible insights for competitors into business processes.

The case studies presented, in particular that of openadaptor, showed that open source projects started by commercial firms, and of interest only to other firms, not to hobbyists, can be successful. There are several levers by which the originator of such an open source project can actively influence its success. First, OSS is more attractive for potential adopters if the software is well structured and documented, the project’s web site up-to-date, and the maintainer responsive to requests. The example of CEPS showed that simply offering the software for download, in particular complex software, misses out on some of the potential benefits of open-sourcing. Second, it is helpful when the originator and maintainer of an open source projects commits to neutrality. The move of transferring the rights to openadaptor to a non-profit organization was certainly reassuring for DrKW’s competitors. Third, reputational benefits, as well as diffusion, can be considerably increased by marketing efforts.

A final learning is that firms should rethink the habit of keeping private as much of their IP as possible – giving it into the open source domain may be more profitable, despite the unusually weak protection of intellectual property. That this need not be disadvantageous for innovation was demonstrated by Allen (1983), in a field far remote from software. He describes a phenomenon he calls “collective invention” in the nineteenth-century English iron industry. During 1850-1875, two important attributes of iron furnaces were subject to steady improvement. Allen found that in many cases innovators publicly revealed data on their furnace design and performance in meetings of professional societies and in published material. The explanations he offers for this pattern of behavior are quite similar to those proposed here for contributions by commercial firms to OSS.

The regime of weak IP protection described by Allen proved extremely beneficial for the firms involved. A similar regime is, in the area of software, institutionalized by open source licenses. This is true in particular for the GNU General Public License (GPL), which goes farthest in restricting commercial exploitation of OSS and might, for this very reason, boost innovation (Asay 2002). Also results by Bessen and Maskin (2000) suggest that, under such a regime of collective invention, firms active in certain fields of software might fare collectively better than with strong IP protection. The authors argue that innovation in software is “sequential”, that is, every innovator builds upon earlier innovations by others. Their model analysis shows that a regime of very strong intellectual property protection (i.e., software patents) actually stifles innovation, leading to a reduction in social welfare. If the value of innovation is large compared to its cost, then patent protection is also detrimental to the firms involved

A firm that releases proprietary software as OSS is free to choose the license. Lerner and Tirole (2002b) conjecture that in this case restrictive licenses, in particular the GPL, are more likely to be chosen. Otherwise, they argue, it might not be possible for firms to attract open source developers. The projects presented here, of which only one is licensed under the GPL, suggest a different reasoning. First, “community appeal”, as stressed by Lerner and Tirole, does not really matter for the type of projects analyzed here, since most contributors do not come from the typical open source community. Second, path dependence and culture may play a strong role: for example, the main developer of Flood is a member of the Apache foundation, which could well explain Flood’s licensing under the Apache license. Third, the GPL might indeed be attractive for firms, but for a different reason, namely, because it blocks competitors from commercializing the software. How important the latter argument is depends on the role of the releasing firm – users of the software will typically object less to a commercialization by others than sellers of complements (Harhoff et al. 2003). These issues certainly merit further investigation.

To summarize, the analysis in this article has shown that OSS is not confined to hobby programmers. Commercial firms such as IBM or Sun contribute to open source projects in order to promote sales of complements. Furthermore, firms who are users of a certain piece of software may find it profitable to release it into the open source domain, and may indeed receive outside development support from other firms, in particular their competitors. Hence, an “open source community” is not necessarily a community of hobby programmers – it may well be made up of investment banks.

References[38]

Achtenhagen, L.; Müller-Lietzkow, J.; zu Knyphausen-Aufseß, D. (2003): “Das Open Source-Dilemma: Open Source-Software-Entwicklung zwischen organisatorischer Optimierung und notwendiger Kommerzialisierung.” In: Zeitschrift für betriebswirtschaftliche Forschung, 55.

Allen, R. C. (1983): “Collective Invention.” In: Journal of Economic Behaviour and Organization, 4, pp. 1-24.

Asay, M. (2002): “A Funny Thing Happened on the Way to the Market: Linux, the General Public License, and a New Model for Software Innovation.” Discussion paper, Stanford Law School. URL: www.linuxdevices.com/files/misc/asay-paper.pdf.

Behlendorf, B. (1999): “Open Source as a Business Strategy.” In: C. DiBona, S. Ockman, M. Stone, Open Sources - Voices from the Open Source Revolution. Cambridge MA et al: O'Reilly, pp. 149-170.

Bessen, J.; Maskin, E. (2000): “Sequential Innovation, Patents, and Imitation.” Working paper 00-01, MIT, Cambridge, MA. URL: www.ftc.gov/os/comments/intelpropertycomments/ jimbessenericmaskin.pdf.

Dempsey, B. J.; Weiss, D.; Jones, P.; Greenberg, J. (1999): “A Quantitative Profile of a Community of Open Source Linux Developers.” In: SILS Tech. Rep. TR-1999-05, School of Information and Library Science, University of North Carolina at Chapel Hill. URL: ils.unc.edu/ils/research/reports/TR-1999-05.pdf.

Dinkelacker, J.; Garg, P. K.; Miller, R.; Nelson, D. (2001): “Progressive Open Source.” Hewlett Packard Laboratories, Palo Alto, CA. URL: home.earthlink.net/~gargp/data/progressive.pdf.

[38] All internet sources were checked and found available in May 2003, unless noted otherwise.

Economides, N. (1996): “Network Externalities, Complementarities, and Invitations to Enter.” In: European Journal of Political Economy, 12, pp. 211-232.

EDC (2002): “Embedded Systems Developer Survey Vol. 2.” Evans Data Corporation, Santa Cruz, CA. URL: www.evansdata.com/embeddedTOC_02-2_xmp2.htm (accessed 2002/11/29).

Feller, J.; Fitzgerald, B. (2000): “A Framework Analysis of the Open Source Software Development Paradigm.” Proceedings of the 21st International Conference in Information Systems, pp. 58-69.

Franck, E.; Jungwirth, C. (2003): “Open versus Closed Source.” In: Zeitschrift für Betriebswirtschaft, forthcoming.

Franck, E.; Jungwirth, C. (2002): “Die Governance von Open-Source-Projekten.” University of Zürich. URL: www.unizh.ch/ifbf/webFuehrung/Dokumente/WorkingPaper/ FranckJungwirthGovernanceofOpenSourceWP9.pdf.

Franke, N.; von Hippel, E. (2002): “Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software.” Working paper #4341-02, MIT Sloan School of Management, Cambridge, MA. URL: opensource.mit.edu/papers/frankevonhippel.pdf.

Garzarelli, G. (2002): “Open Source Software and the Economics of Organization.” University of Rome, La Sapienza. URL: opensource.mit.edu/papers/garzarelli.pdf.

Godfrey, M. W.; Tu, Q. (2000): “Evolution in Open Source Software: A Case Study.” International Conference on Software Maintenance, San Jose, CA, USA.

Hamel, G.; Doz, Y. L.; Prahalad, C. K. (1989): “Collaborate with Your Competitors - and Win.” In: Harvard Business Review, 67 (1), pp. 133-139.

Hamerly, J.; Paquin, T.; Walton, S. (1999): “Freeing the Source - the Story of Mozilla.” In: C. DiBona, S. Ockman and M. Stone, Open Sources - Voices from the Open Source Revolution. Cambridge MA et al.: O'Reilly, pp. 197-206.

Hann, I.-H.; Roberts, J.; Slaughter, S.; Fielding, R. (2002): “Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives.” Carnegie Mellon University. URL: opensource.ucc.ie/icse2002/HannRobertsSlaughterFielding.pdf.

Harhoff, D.; Henkel, J.; von Hippel, E. (20032): "Profiting From Voluntary Information Spillovers: How Users Benefit by Freely Revealing their Innovations." In: Research Policy, in press.

Hars, A.; Ou, S. (2002): "Working for Free? Motivations for Participating in Open-Source Projects." International Journal of Electronic Commerce, 6 (3), pp. 25-39.

Hecker, F. (1999): "Setting Up Shop: The Business of Open-Source Software." IEEE Software 16 (1), pp. 45-51.

Henkel, J. (2003): "Embedded Linux - Informal collaborative software development by commercial firms." In: Proceedings of the 12th International Conference on Management of Technology (IAMOT2003), Nancy, France.

Hertel, G.; Niedner, S.; Herrmann, S. (2001): "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel." In: Research Policy, forthcoming.

Hissam, S.; Weinstock, C. B.; Plakosh, D.; Asundi, J. (2001): "Perspectives on Open Source Software." Software Engineering Institute, Carnegie Mellon University. URL: www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr019.pdf.

International Institut of Infonomics; Berlecon Research (2002): "Free/Libre and Open Source Software: Survey and Study, FLOSS Final Report." University of Maastricht. URL: www.infonomics.nl/FLOSS/report/index.htm.

Koch, S.; Schneider, G. (2000) "Results from Software Engineering Research into Open Source Development Projects Using Public Data." Wirtschaftsuniversität Wien. URL: wwwai.wu-wien.ac.at/~koch/forschung/sw-eng/wp22.pdf.

Krishnamurthy, S. (2002): "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects." University of Washington. URL: opensource.mit.edu/papers/ krishnamurthy.pdf.

Kuan, J. (2000): "Open Source Software As Consumer Integration into Production." University of California at Berkeley. URL: opensource.mit.edu/papers/Jenny%20Kuan%20-%20Open% 20Source%20Software%20As%20Integration%20into%20Production.pdf.

Lakhani, K.; von Hippel, E. (2000): “How Open Source Software Works: “Free” User-to-User Assistance.” In: Research Policy, forthcoming.

Lakhani, K. R.; Wolf, B.; Bates, J. (2002): “The Boston Consulting Group Hacker Survey.” Boston Consulting Group, Boston, URL: www.osdn.com/bcg/BCGHACKERSURVEY.pdf.

Lerner, J.; Tirole, J. (2002a): “Some Simple Economics of Open Source.” In: Journal of Industrial Economics, 502, pp. 197-234.

Lerner, J.; Tirole, J. (2002b): “The Scope of Open Source Licensing.” URL: opensource.mit.edu/ papers/lernertirole2.pdf.

Metiu, A.; Kogut, B. (2001): “Distributed Knowledge and the Global Organization of Software Development.” The Wharton School, University of Pennsylvania, URL: opensource.mit.edu/ papers/kogut1.pdf.

Metzger, A.; Jaeger, T. (2001): “Open Source Software and German Copyright Law.” In: International Review of Industrial Property and Copyright Law, 32, pp. 52-74.

Mockus, A.; Fielding, R. T.; Herbsleb, J. D. (2000): “A Case Study of Open Source Software Development: The Apache Server.” 22nd International Conference on Software Engineering, Limerick, Ireland: Association for Computing Machinery.

Mockus, A.; Herbsleb, J. D. (2002): “Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?” International Conference on Software Engineering (ICSE 2002), Orlando, FL, USA.

Moody, G. (2001): “Rebel Code - Inside Linux and the Open Source Revolution.” Cambridge, MA: Perseus Publishing.

Nüttgens, M.; Tesei, E. (2000): “Open Source - Produktion, Organisation und Lizenzen.” In: Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi) Nr. 157, Universität des Saarlandes. URL: www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/heft157/heft157.pdf.

Osterloh, M.; Rota, S.; von Wartburg, M. (2001): “Open Source - New Rules in Software Development.” University of Zürich, Institute for Research in Business Administration. URL: www.unizh.ch/ifbf/orga/downloads/OpenSourceAoM.pdf.

Picot, A.; Maier, M. (1992): “Analyse- und Gestaltungskonzepte für das Outsourcing.” In: Information Management, 4, pp. 14-27.

Raymond, E. S. (1999): “The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.” Sebastopol: O'Reilly.

Rosenberg, D. K. (1999): “How to Make Money With Open-Source Software.” URL: kuya.vlsm.org/docs/biz/opensource-bisnis.pdf.

Schmitz, P.-E.; Castiaux, S. (2002): “Pooling Open Source Software - An IDA Feasibility Study.” European Commission, DG Enterprise. URL: europa.eu.int/ISPO/ida/export/ files/en/1115.pdf.

Schrader, S. (1991): “Informal Technology Transfer Between Firms: Cooperation Through Information Trading.” In: Research Policy, 20, pp. 153-170.

Shankland, S. (2003): “Study lauds open-source code quality.” CNET news.com, 19.2.2003, URL: news.com.com/2100-1001-985221.html.

Teece, D. J. (1986): “Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy.” In: Research Policy, 15, pp. 285-305.

von Hippel, E. (1988): “The Sources of Innovation.” New York: Oxford University Press.

von Hippel, E. (2002): “Open source projects as horizontal innovation networks - by and for users.” MIT Sloan School of Management working paper 4366-02, Cambridge, MA. URL: opensource.mit.edu/papers/vonhippel3.pdf.

von Hippel, E.; von Krogh, G. (20032): “Open source software and the “private-collective” innovation model: Issues for organization science.” In: Organization Science, 14, pp. 208-223.

Wheeler, D. A. (2002): “Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!” URL: www.dwheeler.com/oss_fs_why.html.

Appendix

Table 2: Examples of companies engaging in OSS projects

Company	**Software**	**Description**
Proprietary software turned into OSS		
Aladdin Enterprises	Ghostscript	interpreter for postscript language
Apple	Darwin	Mac OSX server system
Cisco	CEPS	printer network management tool
Dresdner Kleinwort Wasserstein	OpenAdaptor	integration tool for reconciling global banking and trading accounts
ebuilt	Flood	tool for web site testing
Hewlett Packard	CoolTown	development platform for "pervasive computing"
IBM	Eclipse	development tool
	Jikes	Java compiler
Netscape	Mozilla	web browser
Sun	Sun Grid Engine	tool for distributed computing
	Open Office	office suite, OSS version of Star Office
	JXTA	peer-to-peer computing project
	Net Beans	set of Java tools
id Software	Doom	shooter game
Contributions to OSS projects that were initiated non-commercially		
CodeWeavers	WINE	implementation of Windows APIs on GNU/Linux
Ericsson / Red Hat	embedded Linux	operating system in Ericsson's Cordless Screen Phone HS210
IBM	Apache	web server software
SuSe	KDE	desktop for GNU/Linux and Unix
Ximian	MONO	OSS version of the Microsoft.NET development platform
IBM, Red Hat, SuSE	GNU/Linux	operating system
Red Hat, Sun, Ximian	GNOME	desktop for GNU/Linux and Unix