September 2003

Software Development in Embedded Linux - Informal Collaboration of Competing Firms

Joachim Henkel
Ludwig-Maximilians-Universität München, henkel@bwl.uni-muenchen.de

In: Uhr, Wolfgang, Esswein, Werner & Schoop, Eric (Hg.) 2003. *Wirtschaftsinformatik 2003: Medien - Märkte - Mobilität*, 2 Bde. Heidelberg: Physica-Verlag

ISBN: 3-7908-0111-9 (Band 1)
ISBN: 3-7908-0116-X (Band 2)

Software Development in Embedded Linux - Informal Collaboration of Competing Firms

Joachim Henkel
Ludwig-Maximilians-Universität München

Abstract: The "open source development process" has received considerable attention. It means that loosely co-ordinated, geographically dispersed developers collaborate. While in prototypical open source projects developers are unpaid volunteers, the involvement of commercial firms has recently increased enormously. There are some areas of open source software where indeed most contributions come from commercial firms, and even from firms which consider the development of open source software their core business. It is particularly surprising that these firms take part in the open source development process, as it implies informal collaboration with competitors and the revealing of own developments. The present paper analyzes this phenomenon. It presents an empirical analysis of the embedded Linux industry, based on in-depth interviews with embedded Linux companies and industry experts. It is found that firms in this industry do indeed reveal a considerable share of their developments, and benefit in turn from what their competitors make public.

Key words: open source software, embedded Linux, software development

1 Introduction

Open source software (OSS) such as Linux, Apache, or Samba has made large headlines. Of particular interest is the development process employed by these projects. As Feller and Fitzgerald note concerning the "open source development paradigm", "the case can be made that OSS addresses many aspects of the software crisis, in that reliable, high quality software may be produced quickly and inexpensively" [FeFi00, p. 58]. The main reason why OSS seems to hold this promise is that it enables any user to share in the collaborative development of the software [MoHe02].

The present paper analyzes collaborative development in a somewhat unusual field of open source software, namely, embedded Linux. In contrast to a "prototypical" open source project, most contributions in this field do not come from volunteers but from commercial firms, many of which are dedicated embedded Linux firms. The purpose of this paper is to explore a surprising

finding: that commercial software firms, in many cases direct competitors, reveal (parts of) their code, and perform collaborative software develpment without any contractual base. The question is, hence, if and to what degree the open source development paradigm is applicable when the actors are not volunteers but commercial firms, and what the perceived benefits and risks for these firms are.

The research presented is based on in-depth interviews with embedded Linux companies, other embedded software firms, and industry experts, as well as an analysis of relevant literature. The paper is organized as follows. In section 2, the role of the open source community in OSS development is contrasted to that of commercial firms. Section 3 gives some background on embedded software and embedded Linux in particular. Section 4 presents evidence on revealing of developments in the field of embedded Linux. These empirical findings are interpreted in section 5. Section 6 concludes.

2 OSS development – community versus firms

The term "OSS" is sometimes used synonymous to software developed by the open source community. That is, by volunteers who are not, or at least not directly, paid for their work [Raym99; NüTe00]. While this interpretation is not warranted by the Open Source Definition [Open03], it is correct in many cases. As to the volunteers' motives, surveys [HaOu02; Lak+02; Inte02] identified 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 [Raym99] or on the job market [LeTi02; Han+02], and the relevance of norms and trust inside the developer community [Ost+01]. A "community only" project is illustrated in figure 1a.

For a commercial firm, it may make sense to participate in a (community based) open source project, or to instigate a development community around its own software (figure 1b). "Participate" is meant to imply here that management consciously allocates significant resources to the respective OSS project. One potentially important motive to do so is to increase sales of a complement. The most prominent example for this is IBM, which complements its WebSphere application server with the open source webserver Apache and its UNIX based hardware with the operating system Linux[1] [Mood01, p. 205]. Another motive

[1] More precisely, one should refer to "GNU/Linux", since the operating system is made up of the Linux kernel and complementing software which was largely developed by

might be to obtain outside development support, as, e.g., in the case of Netscape and Mozilla [Mood01, p. 197-204]. In a "restricted" open source approach, this outside support may be restricted to a firm and its customers [Din+01; ORei99].

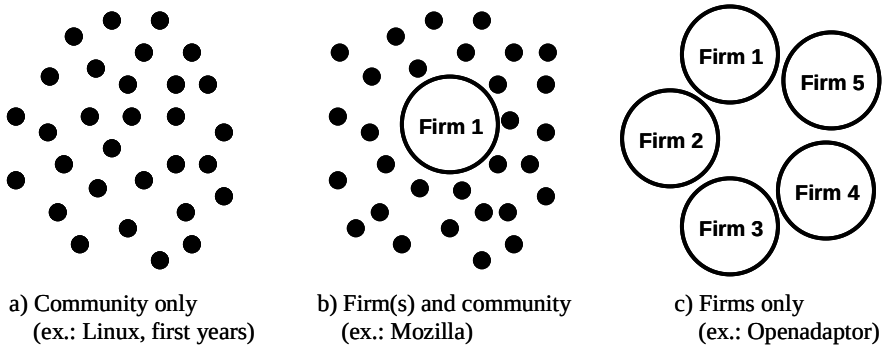

Figure 1: Types of open source projects by their participants

For firms that are users of a particular piece of OSS, it may make sense to contribute to its further development, or to release the software as OSS in the first place. There is no loss of sales value, and a loss of competitive advantage can also often be excluded [Raym99, pp. 139-162]. Since much of the software that firms use is of little value to hobby developers, contributors to this kind of open source projects tend to be mainly firms, as illustrated in figure 1c. A good example is the case of "Openadaptor", a software developed and released as OSS by the investment bank Dresdner Kleinwort Wasserstein [Henk02]. Obviously, projects of this type will hardly muster as large a number of developers as the Linux kernel. However, the same holds true for most community-based OSS projects [Kris02].

Embedded Linux is also best described by figure 1c. Unlike programs that are meant to run on a PC, embedded software can sensibly be used only within the respective device. This implies that few contributions in this field come from hobby programmers, who usually only have a PC at their disposal. Still, embedded Linux is a special case. Dedicated embedded Linux firms consider the development of OSS their core business. They are neither users of this software, nor sellers of complements (even if some of them charge for development toolkits, e.g.). This makes it more surprising than in the cases mentioned above that they share in the open source development process of embedded Linux.

---

the Free Software Foundation ("GNU" software). However, since the term "Linux" has become common, it is used in this paper.

3 Specifics of embedded Linux

3.1 Embedded software

The term “embedded software” generally refers to software in a device that has been built for a specific purpose – it has a “limited mission” [Lomb01, p. xvi].[2] This is in contrast to general-purpose devices such as PCs, which are designed to be extremely flexible and to run a wide variety of software. Examples for “embedded devices” range from very small (mobile phone, wristwatch) to very large (power plant, airplane). Embedded software is steadily gaining importance: the market research firm VDC estimates worldwide shipments of embedded devices for 2002 to be over 1.7 billion units [BaLa02]. Everyday experience confirms this fact, since the share of electronics in many products – cars, household appliances, vending machines – is steadily rising.

The embedded software market is characterized by a large diversity. The variety of different CPUs that are used in the embedded area is considerably larger than that for general-purpose computers. Variations in the board further increase heterogeneity. In addition to hardware diversity, also a huge variety in functional requirements drives heterogeneity in embedded software. Extremely important in most embedded applications is stability. It is vital in applications such as airplanes and power plants; in other, less critical applications such as public payphones and vending machines stability reduces servicing and cost. Real-time capability is another potentially important requirement. It means that the processor responds to signals in a short and deterministic period of time. Real-time capability is paramount in some cases, such as in process controls; in contrast, a response time of hundred milliseconds will hardly be noticed in a PDA. Finally, a small memory footprint is important in small and/or low cost devices. Especially where (expensive) Flash memory chips are needed, an economical use of memory by the embedded software can allow for considerable savings in production. In other situations, such as internet routers, software developers do not have to care at all about memory restrictions.

In the field of embedded operating systems, the aforementioned diversity in hardware as well as use requirements has led to relatively high industry fragmentation. The market leader is generally considered to be WindRiver, with its main product VxWorks. Further players are, among others, Microsoft, QNX, Green Hills Software, Accelerated Technology, and, increasingly, Linux. Of high, though decreasing importance is in-house development: since embedded operating systems are often both relatively simple and rather specific, development by the

---

[2] Unless noted otherwise, statements in this and the following sections are based on interviews with industry insiders (see section 4).

device manufacturer's IT staff is an option. However, increasing complexity of embedded devices – with 32-bit processors, networking connections, and graphics capability – makes complete in-house development less and less attractive [Webb02]. For this reasons, embedded Linux is an attractive option: it offers embedded developers a stable, maintained operating system with full openness and flexibility.

3.2 Developing embedded Linux

This section describes how standard Linux is turned into an embedded operating system, and to what degree embedded versions of Linux remain linked to the standard distribution. As figure 2 illustrates, an embedded Linux firm (Firm 1, 2, 3, …) would create its version of embedded Linux by downloading standard Linux code, as well as, in most cases, code that is specific to embedded Linux. The border between "standard code" and "embedded-specific code" is somewhat blurred, since developments made for embedded applications may later find their way into the standard distribution. Nonetheless, at any given moment it is pretty clear which projects in Linux are specific to embedded Linux. The arrows pointing from the firms to the freely available code base indicate developments by these firms that are made public. How significant these contributions are is the topic of section 4.

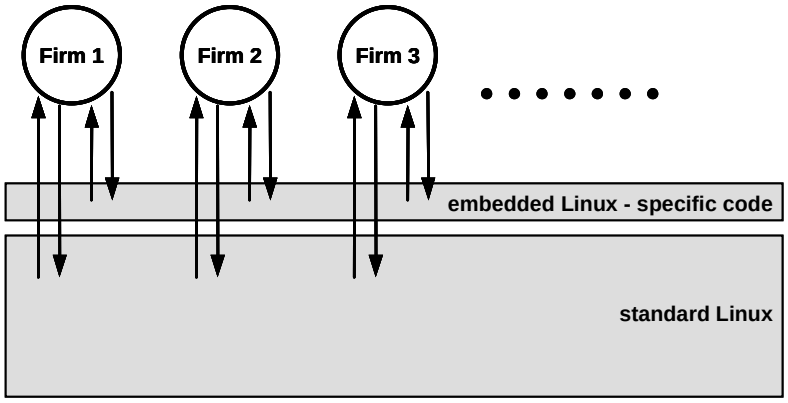

Figure 2: Adoption of and contribution to free code by embedded Linux firms

Since an embedded operating system is tailored to a specific device and purpose, much of the code in standard Linux is not needed. In the (frequent) case of storage restrictions, an important step is thus to strip down Linux to as small a code base as possible. For example, the size of the C library is reduced by either removing unneeded functions, or by replacing the standard library with a smaller one, e.g., uclibc [Lomb01, chap. 2].

This library is one example of code that is specific to embedded Linux. Another important example are modules that improve Linux' real-time capability, which, as discussed in section 2, can be of high importance in embedded applications. Standard Linux is not a real-time operating system – the kernel's response time is not deterministic, and it can reach up to more than hundred milliseconds. There are basically two ways of dealing with this issue [Dank02]. The first is to run Linux on top of a second, much smaller operating system with real-time capability. This approach was chosen by FSMLabs in their development of RTLinux, and is also employed by LynuxWorks, Red Hat, and Lineo. A technically similar project is RTAI, which circumvents the use of technology patented by FSMLabs. The second approach is to make the Linux kernel "preemptible". This means that currently running tasks can be preempted by newly started tasks with higher priority [AnGa00; Wein01]. Preemptible kernels are provided, among others, by MontaVista and TimeSys. Both approaches are well established in the embedded Linux industry, although they sparked considerable controversy [YoSh02]. Obviously, what an "embedded Linux" operating system looks like depends strongly on the respective application.

The above might suggest that every development of an embedded Linux version constitutes a "forking", i.e., a development of differing versions that, over time, grow more and more incompatible. However, this is not the case: an embedded Linux firm would create the next version of its distribution not by updating its present version, but rather by starting from a new version of standard Linux. Hence, compatibility between embedded and standard Linux is largely preserved, and progress in the embedded field can build upon developments in the standard version. It also implies that embedded Linux developers can obtain support from the "Linux community", at least on topics that relate to the standard code base. A survey by the market research firm VDC shows that embedded developers indeed make use of this possibility: asked how often they would consult the open source community, the distribution of answers was: never: 0%; rarely: 39%; monthly: 34%; weekly: 27%; daily: 0% [LaBa02].

Linux is an attractive option for embedded systems developers for several reasons. It comes under a royalty-free license, which, according to a recent survey, is the second most important consideration in selecting an embedded operating system [EDC02]. It has been ported to a very wide range of processor architectures, and the choice of available device drivers is huge. The openness and modularity of the code simplify modifications, and its being OSS decreases dependency on suppliers. Interest is correspondingly high: About 30% of respondents in the EDC survey were planning to use embedded Linux in their next project, neck and neck with embedded Windows and far ahead of VxWorks, which came in third. Of course, there are also drawbacks of Linux [Fidd02]. In itself, it is not a real-time operating system. Since it is OSS, and since many different firms offer embedded Linux distributions, there is a certain risk of forking, i.e., of different, incompatible versions of embedded-specific modules developing. Finally,

longevity of most embedded Linux firms has yet to be proven. This leads to the issue of industry structure in embedded Linux.

3.3 The embedded Linux industry

There are a number of firms offering embedded Linux distributions, such as FSMLabs, Lineo, LynuxWorks, MontaVista, Red Hat, REDSonic, and TimeSys. To give an example of a popular embedded device, Sharp's PDA Zaurus runs on Lineo's Embedix distribution. In addition to these larger firms, there are quite a number of other companies active in this field. Industry dynamic is high, however, with many especially smaller firms exiting completely (e.g., RidgeRun) or changing their business model. This is partly due to external factors – the burst of the dot-com bubble, the economic down-turn in general, and necessary consolidation in a young industry. In addition, competition for embedded Linux firms is perceived as strong, at least by embedded Linux firms in the US. They compete on one side against proprietary embedded operating systems (e.g., VxWorks), on the other side against freely available embedded Linux distributions.[3]

The latter fact may seem surprising – after all, any embedded Linux distribution is OSS, so why should some of them not be freely available? The explanation lies in a widespread misunderstanding. The General Public License (GPL), under which Linux stands, stipulates that if someone distributes the software or modified versions of it, then this must be done under the conditions of the GPL. In particular, the recipient of the software must obtain the source code as well as the rights to modify and redistribute the software [Free01]. However, the GPL does not require to make modified software publicly available. Further, it does not exclude selling the software (for a one-time price, not per-unit royalties). This means that embedded Linux firms are free to sell their distributions to their paying customers only, which is what most of them do.

Still, two important differences to proprietary software exist. First, when the device is finally sold to the market, then each buyer must be able to obtain the source code of the OSS inside the device. Hence, after a while – approximately one year – the complete distribution does become publicly available. Second, the embedded Linux vendor is blocked, by the GPL, from charging per-unit license fees. This leads to the question of what to charge for, and thus to the issue of business models.

---

[3] An overview of freely available embedded Linux implementations is provided by LinuxDevices.com at www.linuxdevices.com/articles/AT4525882120.html (accessed 2002-10-24).

Given that each user of embedded Linux needs a somewhat customized version, selling the service of adapting the software to individual needs is the most obvious business model [Heck99; Raym99, pp. 165-167; Rose99]. It is, to some degree, practiced by all embedded Linux firms. Selling the distribution itself is also common, often bundled with development support for a limited time. In 2002, licensing complementary proprietary software has gained considerable importance. Many firms now offer proprietary development toolkits, and in some cases Linux-compatible proprietary operating systems. Examples are the "ELinOS" development environment by Sysgo AG, or the real-time operating system "LynxOS" by LynuxWorks.[4]

4 Empirical study

4.1 Data collection

During the period from June to December 2002, twenty interviews have been conducted with industry insiders (see table 1). Eleven of them work with embedded Linux firms, four with other embedded software firms, and five are industry experts related to, but not directly working with embedded Linux. Interviews were conducted in a semi-structured fashion, in order to combine comparability with openness for unexpected statements. Questions, including the following examples, were asked under four headings:

(a) *Background*: What differentiates embedded Linux from standard Linux? What are the most important developments in embedded Linux?

(b) *Revealing of innovations*: Do embedded Linux firms make significant parts of their development publicly available? What are the benefits and downsides of doing so? When a developer asks a question on a web forum, would employees from other firms answer it?

(c) *Building upon others' work*: In developing an embedded Linux distribution, does a firm build upon work done by competitors? If so – how?

(d) *Improvements*: Is the present quality of embedded Linux the result of contributions from different sources? Where did the more important contributions come from initially? Have they been improved upon and added to by other firms?

[4] See http://www.elinos.com/pdfs/el22.pdf and http://www.lynuxworks.com, respectively (accessed 2003-02-12).

The following two subsections correspond to questions (b) and (c); answers to (d) are given in either 4.2 or 4.3, as appropriate. Answers to (a), even though they were obtained in the course of the empirical study, do not directly refer to the research question at hand. For this reason, they were used in section 3 to complement background information on embedded Linux obtained from other sources.

Firms contributing to embedded Linux		**Other embedded SW firms**	**Industry experts**
USA	**Europe**		
FSMLabs Independent developer Lineo LynuxWorks MontaVista	Convergence Denx Software Innominate Mind Pengutronix SSV Embedded Systems	Microsoft QNX Sleepycat Wind River	Author “Embed-ded Linux” Free Software Foundation LinuxDevices.com LinuxJournal Maintainer of Debian Linux

Table 1: Firms interviewed

4.2 Results: Contributing to publicly available code

In 1999 and 2000, when enthusiasm for Linux and open source in general peaked at Wall Street, embedded Linux firms were strongly open source oriented. Distributions were freely available for download, and even toolkits which were not based on pre-existing GPL-ed software (and, hence, could have been kept proprietary) were released as OSS. The typical business model was to give away the software and to charge for services.

Two years later, the picture has changed. Only few companies make their full distributions freely available, and the majority makes at least part of their profit by selling some non-free software. This change was driven by tough market conditions, a difficult economy in general, and in some cases by demands from the venture capitalist backing the embedded Linux firm.

However, this does not mean that developments in embedded Linux are not made public any longer. Embedded Linux is a nascent industry whose members, so it seems, have to find the right balance between revealing and not revealing their developments. Also in 2002, intellectual property is given away. This happens on

five levels: Full embedded Linux distributions; single modules; co-operation in source trees; bug fixes; support for other developers. Below, each of these points is discussed in turn. Unless noted otherwise, quotes, even if taken from a particular interview, reflect the opinion of the majority of interviewees.

Full embedded Linux distributions: As of November 2002, TimeSys and Denx Software Engineering are among the few companies to offer a full embedded Linux distribution for free download.[5] However, Lineo's Embedix version of Linux can be obtained by anyone purchasing Sharp's Zaurus PDA, since the GPL obliges Sharp to deliver the source code with the embedded software (it is irrelevant if the compiled version of the software is distributed on a disk or in a device). Hence, if the device is sold to the general market, then with a lag of on average one year embedded OSS leaks out to the public.

Single modules: Quite a number of important modules in embedded Linux go back to individual companies. MontaVista developed a patch to make the Linux kernel preemptable, and it actively pushed this module out to the OSS community. It was accepted by Linus Torvalds as part of standard Linux (version 2.5). Other contributions are the fixed-overhead scheduler, simple graphical user-interfaces (GUIs) such as MicroWindows and NanoX and the toolset Busybox. Firms do not always make their developments freely available when they are not obliged to do so. E.g., software written for a specific application that provides the device manufacturer a competitive advantage is kept inhouse, and only given out as late as possible. In contrast, "generic pieces of code" are shared more openly. In cases where the GPL requires to make code publicly available, compliance seems to be good:

> "I'm not aware of anyone who hasn't returned code to Linux when they were supposed to."

Co-operation in source trees: Linux is a huge project, or rather a huge collection of single projects. An individual project may be dedicated, e.g., to porting Linux to a new processor architecture, to improving the file system, or to developing a USB port. A lot of co-operation between embedded Linux firms happens on this level. As an interviewee from MontaVista put it:

> "We contribute by doing our core original development out in open source on the trees themselves. [...] We actually are deeply involved in many projects [...] are even gatekeepers in a number of projects [...]. For example, the Linux PowerPC tree. We have six people on that tree. Some of our competitors [...] are also participants in that tree. [...] folks from Lineo, one guy from Red Hat, [...] some people from FSMLabs, people from Terasoft [...]."

---

5 See www.timesys.com/index.cfm?hdr=linux_header.cfm&bdy=linux_bdy_downloads.cfm and www.denx.de (accessed 2002-11-20).

An exploratory analysis of the "Linux PowerPC embedded" mailing list confirms the above statement.[6] The two most active firms in this mailing list are MontaVista and Denx Software Engineering, and other embedded Linux firms – FSMLabs, Sysgo, and TimeSys – are also frequent contributors. Another interviewee brought the issue of collaboration between competitors to the point:

> "...in general there are, on the public lists, different competing companies contributing to improve the public version. That's very obvious."

Social norms ensure that collaboration on the technical level remains remote from market competition:

> "There is a non-written policy that on these lists you just stick to your technical statements, and you try to improve the quality of the code, and you don't explicitly market your own company."

Bug fixes: When an embedded Linux developer discovers a bug in the standard code base, he would report it to the maintainer of the respective module. In case he fixed the bug he would usually submit the patch. In this respect, programmers working with embedded Linux firms do not behave much different from hobby programmers in the prototypical open source community.

Support for other developers: When someone asks a question in a web forum that is dedicated to an embedded Linux project, other developers would usually answer:

> "Developers would also exchange information directly. E.g., one is programming for an ARM processor and runs into a problem. There are, maybe, 5 to 15 people worldwide working on similar problems, and knowing of each other (because of trade shows etc. it's no secret who's working on what). They meet on the internet to discuss their problems. It is normal to ask a question about a particular programming problem and to get an answer, even though the other programmers work for competitors."

4.3 Results: Benefiting from publicly available code

Embedded Linux firms potentially benefit in a variety of ways from work done by others: by the fact that the standard version of Linux improves; by code re-use; by learning from code written by others; and by direct support from other developers.

Improved standard version of Linux: Since, as was laid out in section 3.2, a new embedded Linux distribution is derived from a new version of standard Linux (and not from an older embedded Linux distribution), improvements in the standard code base benefit all embedded Linux firms. This concerns modules, ports to new processors, as well as bug fixes.

---

6 See lists.linuxppc.org/linuxppc-embedded/index.html (accessed 2003-01-05).

Code re-use: Publicly available code serves as a platform on top of which embedded Linux firms add their developments:

> "I'm porting Linux to a new family of single board computers, but 90% of the code to actually support the host processor was written by a bunch of other people. I'm only doing the bits that make my board unique. And yes, my enhancements will go back to the community."

Drivers constitute the largest part of the Linux code base.[7] Since, apart from some exceptions, their source code is available, drivers offer a good opportunity for code re-use. This is common practice in the field of embedded Linux:

> "In device drivers the typical approach is to read and copy interfaces from an existing driver."

Learning from code written by others: Most interviewees agreed that one can learn a lot from studying code, even if it was written for a different CPU:

> "Processors are not that different at the bottom level."

Getting support from other developers: This corresponds to the point "support for other developers" in section 4.2.

4.4 Differences US – Europe

In Europe, and in particular in Germany, the use of Linux in embedded systems is widespread. However, while in the US there are quite a number of embedded Linux firms that offer complete, branded distributions which are maintained in the long term, this appears to be less frequent in Europe. Here, the focus seems to be more on adapting Linux to individual industry applications. In Germany, e.g., one focus in embedded Linux is on employing it in machine controls and measurement devices. These are traditionally strong industries in this country, with a correspondingly high need for embedded software [Fri+02]. Firms such as Denx Software Engineering, Pengutronix, and Sysgo are active in this field. In such an environment, the advantage for an embedded Linux firm of having its own branded distribution may be smaller than in the US. However, this tentative interpretation needs to be corroborated by a more thorough international comparison.

---

[7] For a visualization of the Linux kernel code, see http://kernelmapper.osdn.com/map.php (accessed 2002-10-25). The outermost ring consists of device drivers.

5 Motives to freely reveal software developments

The above section has shown that embedded Linux firms reveal a considerable share of their developments to the public. They do so in 2002 to a lesser degree than in 2000, but still to a large extent. How can this pattern of behavior be understood? Explanations can be categorized in three groups: economic benefits, obligations due to OSS licenses, and sociological and psychological factors.

5.1 Economic benefits

Concerning innovation incentives, the standard argument goes that an innovator should seek to protect its innovations in order to use them exclusively and/or to license them out against a fee. Where such protection is imperfect, incentives to innovate should be reduced since the innovator can appropriate only a smaller share of the rent its innovation creates.

While this logic has its merits, there are cases where the benefits of freely revealing one's innovation outweigh those attainable by protecting it [Har+03]. In the context of OSS, the main benefits are development and maintenance support from outside, i.e., from other firms or from hobby programmers. Due to its openness, OSS allows users to identify and to fix bugs they encounter. Even if they do so purely for their own benefit, the cost of submitting the bug-fix to the maintainer is extremely low. This can increase the software's quality and stability enormously[8] since, as Raymond puts it, "given enough eyeballs, all bugs are shallow" [Raym99, p. 41]. When software is mainly interesting for firms in a certain industry, and less so for hobby programmers, the number of contributors will be lower, but outside contributions to development and maintenance may still be significant [Henk02]. For an embedded Linux firm, these benefits are strongly increased when a certain module becomes part of standard Linux (as, e.g., the preemptable kernel patch). This gives an incentive not only to make software available, but to actively push its wide adoption. Network effects help in establishing a certain solution as a standard.

Reputation is another important aspect. By developing high-quality modules for OSS, a company demonstrates its technical capabilities [Raym99, pp. 166-167]. Acceptance of a module into the standard code base by Linus Torvalds is widely regarded as a proof of quality. A second dimension of reputation is that of being a "good open source player". Since embedded Linux firms do benefit from the open source community, this aspect must not be neglected.

---

[8] See Wheeler for a compilation of comparisons between OSS and commercial, proprietary software [Whee02]. Of course, open-sourcing software is not a panacea: when interest is low, the software will obviously not benefit from outside development contributions [His+01].

When a piece of software is to some degree specific to its originator, revealing it as OSS (and, possibly, setting a standard in doing so) has several positive aspects. Due to its specificity, the software will in general be less useful to competitors than to the originator. This may be caused by different customer requirements or by a better fit with the originator's complementary offer. It also implies that this company is probably best skilled in maintaining and further developing the respective piece of code, such that buyers of customized versions of the software will prefer to buy this service from the originator.

The lower competition, the sooner firms will be willing to reveal their developments [Hipp87; Schr91]. In embedded Linux, competition is relatively strong, at least in the US; however, what fits the picture is that co-operation is stronger in early phases of development (namely, in the source trees) while the final products (the full distributions) are in most cases kept secret. In this respect, collaboration between embedded Linux firms resembles collaborative, pre-competitive R&D in the IT sector as described by Quintas and Guy [QuGu95].

Of course, revealing one's developments also carries risks and downsides. The more code is publicly available, the more difficult differentiation between embedded Linux firms, and the more attractive the option of in-house development for device manufacturers. Embedded Linux firms respond to these threats by a variety of measures. Code that is considered important for differentiation will only be given to paying customers (in strict accordance with the GPL), such that it becomes publicly available only when the device comes to market, with a time lag of about one year. If a newly developed driver constitutes a competitive advantage for the respective firm (e.g., because it contains details of a proprietary protocol), then it can be kept proprietary as a loadable kernel module.

The issue of differentiation is particularly tricky for firms that follow a product business model. Selling an embedded Linux distribution is possible, but, due to the GPL, the firm has to differentiate itself by either complementary service or proprietary software tools. Often, however, code is developed in commission for a device manufacturer who pays for its development. In this (widespread) case of a service business model, the embedded Linux firm only loses little differentiation when the code is revealed. On the contrary, revealing can even strengthen the firm's differentiation as knowledgeable and skilled in the respective field.

5.2 Obligations due to OSS licenses

The GPL forces firms that build upon and improve Linux to distribute their modified versions again under the GPL. In particular, each recipient must receive the source code together with the right to modify and redistribute it. Since customers of an embedded Linux firm, such as Sharp, usually have no incentive to distribute the software, the GPL's effect is mainly that buyers of the final device

can obtain the source code. Thus, since the exchange of information and developments in source trees and mailing lists is considerable, the GPL falls short of explaining much of the co-operation that takes place in embedded Linux. However, it seems rather plausible that the GPL has shaped the culture of firms and individuals working with GPL-ed software, such as Linux.

5.3 Sociological and psychological factors

In their book on human resources, Baron and Kreps note concerning IT professionals: "[...] their loyalties are to their work, not to an organization" [BaKr99, p. 462]. This may contribute to understanding the behavior of software engineers from competing firms helping each other out. Management is usually more restrictive with this kind of information exchange, but is frequently not even informed about it. Von Hippel [Hipp87] and Schrader [Schr91] found a very similar behavior of "information trading" in other industries. This does not mean, though, that it is detrimental to the firms: they might be in a prisoner's dilemma where all fare better with than without information exchange [Hipp87].

The observed pattern of behavior is linked to the sociological concept of "embeddedness" (see Granovetter [Gran85] and, in the context of interfirm networks, Uzzi [Uzzi97]). This concept considers economic action as embedded in structures of social relations. That is, actions are neither completely explainable by the neoclassical assumption of strict utility maximization, nor by that of social groups whose behavior is determined by norms. Employed programmers fit this description rather well: their actions do take their own and their employer's economic objectives into account, but they remain embedded in the programmer's social environment as a "hacker" (in the positive sense of Raymond [Raym99]).

From this behavior arises a certain legal problem.[9] Labor contracts of software engineers usually state explicitly that intellectual property rights to all code they develop belong to the employer. There are indications that, for this reason, employed programmers sometimes submit patches to open source projects on an anonymous basis [Henk02]. In embedded Linux, though, this problem should not be too severe: first, the managers interviewed clearly saw the benefits of revealing (parts of) their developments; second, as embedded *Linux* firms, these firms share and support the open source culture, at least to some degree, on all levels of management.

---

9 I am grateful to an anonymous referee who pointed out this issue to me, and made several other helpful comments.

6 Summary and conclusion

The analysis of the embedded Linux industry presented in this paper shows that collaborative software development by competing firms without a contractual base can go surprisingly far. The resulting overall development process should be extremely efficient, since duplication of effort is strongly reduced. Just like standard Linux improves at a rapid pace due to contributions from many different sources, so does embedded Linux. Somewhat surprisingly, there are indeed good theoretical arguments why a regime of weak intellectual property protection, as it is arguably given for GPL-ed software, may speed up rather than hinder innovation [Asay02; BeMa00].

The flip-side of building on a common code basis is that market entry becomes easier and differentiation more difficult, which increases competitive pressure. This might contribute to the economic difficulties which some embedded Linux firms face, although it is hard to distinguish this effect from the influence of the general economic climate. Time will tell which firms and business models survive the unavoidable consolidation.

The results presented here have implications beyond the embedded Linux industry and beyond open source software. It was found that firms benefit from revealing their developments, and that they strike a balance between revealing and secrecy. This finding challenges the rather widespread attitude “we do not give away our IP”, which is more often based on habits than on a conscious strategy. For software firms in general it should make sense to evaluate options for revealing their developments, thus likely arriving at a mix of revealing and protection that is superior to “maximum protection”.

This paper is based on a series of interviews. Future research will seek to quantify the results obtained here both by an analysis of artifacts on the internet – web forums and mailing lists – and by a systematic survey of embedded developers. Despite its qualitative nature this paper should have shed light on an important aspect of innovation: on the appropriation of innovation rents not by protecting one’s developments, but by making them public.

References

[AnGa00] Anzinger, G.; Gamble, N.: Design of a Fully Preemptable Linux Kernel. LinuxDevices.com. http://www.linuxdevices.com/articles/ T4185744181.html, 2000, accessed 2002-08-09.

[Asay02] Asay, M.: A Funny Thing Happened on the Way to the Market: Linux, the General Public License, and a New Model for Software Innovation. Stanford Law

School, Stanford, CA. http://www.linuxdevices.com/files/misc/asay-paper.pdf, 2002, accessed 2002-11-26.

[BaLa02] Balacco, S.; Lanfear, C.: The embedded software strategic market intelligence program 2002/2003, Volume I. White paper, Venture Development Corporation, Natick, MA. http://www.vdccorp.com/embedded/ reports/02/br02-49.html, 2002, accessed 2002-11-29.

[BaKr99] Baron, J. N.; Kreps, D. M.: Strategic Human Resources. Frameworks for General Managers. John Wiley & Sons: New York, 1999.

[BeMa00] Bessen, J.; Maskin, E.: Sequential Innovation, Patents, and Imitation. Working paper 00-01, Massachusetts Institute of Technology, Cambridge. http://www.ftc.gov/ os/comments/intelpropertycomments/jimbessenericmaskin.pdf, 2000, accessed 2002-11-12.

[Dank02] Dankwardt, K.: Real time and Linux. Parts 1 – 3. Embedded Linux J Online. http://www.linuxdevices.com/articles/AT5997007602.html, AT5503476267.html, and AT6320079446.html, 2002, accessed 2002-10-23.

[Din+01] Dinkelacker, J.; Garg, P. K.; Miller, R.; Nelson, D.: Progressive Open Source. Hewlett Packard Laboratories: Palo Alto, http://www.hpl.hp.com/techreports/ 2001/HPL-2001-233.pdf, 2001, accessed 2002-11-12.

[EDC02]: Embedded Systems Developer Survey, Vol. 2. Evans Data Corporation, Santa Cruz, CA. http://www.evansdata.com/embeddedTOC_02-2_xmp2.htm and http://www. linuxdevices.com/articles/AT7342059167.html, 2002, accessed 2002-11-29.

[FeFi00] Feller, J.; Fitzgerald B.: A Framework Analysis of the Open Source Software Development Paradigm. 21st International Conference in Information Systems, 2001.

[Fidd02] Fiddler, J.: Linux In Embedded Systems: Where Are The Benefits? White paper, Wind River Systems, Almeda, CA. http://www.windriver.com/ whitepapers/ wp_linux.html, 2002, accessed 2002-11-25.

[Free01] Free Software Foundation: GNU General Public License. www.fsf.org/licenses/ gpl.html, 2001, accessed 2002-12-14.

[Fri+02] Friedewald, M.; Blind, K.; Edler, J.: Die Innovationstätigkeit der deutschen Softwareindustrie. Wirtschaftsinformatik 44, 2002: pp. 151-161.

[Gran85] Granovetter, M.: Economic action and social structure: The problem of embeddedness. Amer. J. of Sociology 91, 1985: pp. 481-510.

[Han+02] Hann, I.-H.; Roberts, J.; Slaughter S.; Fielding, R.: Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives. Carnegie Mellon University, Pittsburgh, PA. http://opensource.ucc.ie/icse2002/HannRoberts SlaughterFielding.pdf, 2002, accessed 2002-11-12.

[Har+03] Harhoff, D.; Henkel, J.; von Hippel, E.: Profiting From Voluntary Information Spillovers: How Users Benefit by Freely Revealing their Innovations. Res. Pol., to appear, 2003.

[HaOu02] Hars, A.; Ou, S. : Working for Free? Motivations for Participating in Open-Source Projects. Int. J. of Electronic Commerce 6, 2002: pp. 25-39.

[Heck99] Hecker, F.: Setting Up Shop: The Business of Open-Source Software. IEEE Software 16, 1999: pp. 45-51.

[Henk02] Henkel, J.: Open source software from commercial firms – tools, complements, and collective invention. Zeitschrift f. betriebswirtschaftliche Forschung, accepted for publication.

[Her+03] Hertel, G.; Niedner, S.; Herrmann, S.: Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux kernel. Res. Pol., to appear, 2003.

[Hipp87] von Hippel, E.: Cooperation Between Rivals: Informal Know-How Trading. Res. Pol. 16, 1987: pp. 291-302.

[His+01] Hissam, S.; Weinstock, C. B.; Plakosh, D.; Asundi, J.: Perspectives on Open Source Software. Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA. http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr019.pdf, 2001, accessed 2002-11-12.

[Inte02] International Institute of Infonomics and Berlecon Research: Free/Libre and Open Source Software: Survey and Study. FLOSS Final Report, University of Maastricht. http://www.infonomics.nl/FLOSS/report/index.htm, 2002, accessed 2002-11-17.

[Kris02] Krishnamurthy, S.: Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. University of Washington. http://opensource. mit.edu/papers/krishnamurthy.pdf, 2002, accessed 2002-06-05.

[Lak+02] Lakhani, K. R.; Wolf, B.; Bates, J.: The Boston Consulting Group Hacker Survey. Boston, MA. http://www.osdn.com/bcg/ BCGHACKERSURVEY.pdf, 2002, accessed 2002-06-05.

[LaBa02] Lanfear, C.; Balacco, S.: Linux's future in the embedded systems market. White paper, Venture Development Corporation, Natick, MA. http://www.vdc-corp.com/ embedded/white/02/02embeddedlinux.pdf, 2002, accessed 2002-10-20.

[LeTi02] Lerner, J.; Tirole J.: Some Simple Economics of Open Source. J. of Ind. Econ. 52, 2002: pp. 197-234.

[Lomb01] Lombardo, J.: Embedded Linux. New Riders: Boston, MA, 2001.

[MoHe02] Mockus, A.; Herbsleb, J. D.: 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. http:// opensource.ucc.ie/icse2002/MockusHerbsleb.pdf, 2002, accessed 2003-01-15.

[Mood01] Moody, G.: Rebel Code - Inside Linux and the Open Source Revolution. Perseus Publishing: Cambridge, MA, 1999.

[NüTe00] Nüttgens, M.; Tesei, E.: Open Source - Konzept, Communities und Institutionen. Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Heft 156, 2000.

[Open03] Open Source Initiative: The Open Source Definition, Version 1.9. http://www. opensource.org/docs/definition.php, 2003, accessed 2003-01-15.

[ORei99] O'Reilly, T.: Schlüsse aus der Open-Source-Software-Entwicklung. http://www.heise.de/tp/deutsch/special/wos/6433/1.html, 1999, accessed 2003-01-17.

[Ost+01] Osterloh, M.; Rota, S.; von Wartburg, M.: Open Source - New Rules in Software Development. Zürich, Institute for Research in Business Administration, 2001.

[QuGu95] Quintas, P.; Guy, K. : Collaborative, Pre-Competitive R&D and the Firm. Res. Pol. 24, 1995: pp. 325-348.

[Raym99] Raymond, E. S.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly: Sebastopol, 1999.

[Rose99] Rosenberg, D. K.: How to Make Money With Open-Source Software. http://www-106.ibm.com/developerworks/linux/library/license.html?dwzone=linux, 1999, accessed 2002-11-25.

[Schr91] Schrader, S.: Informal Technology Transfer Between Firms: Cooperation Through Information Trading. Res. Pol. 20, 1991: pp. 153-170.

[Uzzi97] Uzzi, B.: Social structure and competition in interfirm networks: the paradox of embeddedness. Admin. Sc. Quart. 42(1), 1997: pp. 35-67.

[Webb02] Webb, W.: Pick and place – Linux grabs the embedded market. EDN, http:// www.edn.com, 2002, accessed 2002-11-07.

[Wein01] Weinberg, B.: Embedded Linux – ready for real-time. White paper, MontaVista Software, Sunnyvale, CA. http://www.mvista.com/dswp/ RTReady.pdf, 2001, accessed 2002-10-18.

[Whee02] Wheeler, D. A.: Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! http://www.dwheeler.com/oss_fs_why.html, 2002, accessed 2002-11-27.

[YoSh02] Yodaiken, V.; Sherer, M.: RTLinux is unfair by design. FSMLabs, http://www.fsmlabs.com/developers/white_papers/unfair.html, 2002, accessed 2002-10-17.