Patterns of Free Revealing – Balancing Code Sharing and Protection in Commercial Open Source Development

Joachim Henkel[*], August 2004

Abstract: Commercial firms increasingly contribute to the development of open source software (OSS). However, a conflict often arises between the requirements of the General Public License to make "derived work" available, and firms' interest to protect their intellectual property embodied in the code. If there are ways to mitigate or solve this conflict, the conditions under which OSS will be an appealing solution to firms become much more general. This paper is the first to provide a quantitative empirical study of this conflict and the ways firms deal with it. I present a study of embedded Linux, based on an online-survey that yielded 268 valid responses. It turns out that firms routinely use various means to protect their developments, while keeping the GPL. Still, they do reveal a considerable share of their code—on average, 49%. Heterogeneity between firms is analyzed using multivariate analysis. I show how the relative importance of various benefits and downsides of revealing determines a firm's pattern of revealing. An analysis of reported reasons for revealing and of the type of code that is revealed provides further insights into these patterns. Putting the different dimensions of revealing behavior together, I find that consistent patterns of revealing can be identified for different types of firms.

Key words: open source software, embedded Linux, free revealing, appropriation, IP protection

JEL classification: L86, M11, O31

---

[*] Institute for Innovation Research, Technology Management and Entrepreneurship, University of Munich, Kaulbachstr. 45, D – 80539 Munich, Germany, and CEPR, London. henkel@bwl.uni-muenchen.de.

I am grateful to Marc Gruber, Dietmar Harhoff, Georg von Graevenitz, Eric von Hippel and to participants at the Workshop on User Innovation and Open Source Software in Munich and the OWLS Workshop in Oxford for helpful comments and discussions. I thank Mark Tins for valuable help in data collection.

1 Introduction

Commercial firms increasingly contribute to the development of open source software (OSS). They do so out of various motives[1], most of which are furthered by, or even require, wide diffusion of the respective software. Examples are the motives to increase demand for complements, to establish the software as a standard, or to garner external development support. In these cases, the originator of the software willingly accepts—or even benefits from the fact—that the intellectual property it has put into the source code becomes available to the public.

In other cases the motives behind commercial OSS development do not require diffusion of the software. Its originator might then well prefer to keep the software protected. However, when the development in question is based on OSS under the General Public License (GPL), this choice is excluded: the development must also be licensed as OSS, and under the GPL. In such a situation—typically arising when OSS is used as a convenient input to some development process—a clear conflict between OSS development and the protection of intellectual property (IP) exists. Be it perceived or real, it can constitute a considerable obstacle to commercial OSS adoption and development. If there are ways to mitigate or solve this conflict, the conditions under which OSS will be an appealing solution to firms become much more general.

This paper is the first to provide a quantitative empirical study of this conflict and the ways firms deal with it. Going beyond the argument of complementary assets, this study also analyzes how protection is obtained within and for the software code itself.

The particular instance I study is embedded Linux, in itself of high relevance because of its wide and increasing deployment (Webb 2002, VDC 2004). The term denotes variants of the operating system that are adapted to the use in embedded devices such as VCRs, machine controls or mobile phones. The study is based on an online-survey targeting embedded Linux developers. The survey was carried out in 2003/2004 and yielded 268 valid responses.

Central results are the following. First, even for “derived work” in the sense of the GPL there exist means for firms to protect their code. This implies that firms can participate in OSS

[1] See, e.g., Behlendorf (1999), Hecker (1999), Raymond (1999), and Feller and Fitzgerald (2002).

projects and still have a real choice regarding the protection of their software code from wide and immediate diffusion. I describe the ways firms can do this, and the frequency with which the various protection mechanisms are used in the sample.

Considering what share of their developments firms make public and what share they protect, I find that firms reveal on average about half of the code they develop for embedded Linux. That is, firms make ample use of the various means to protect their developments. The degree to which they do so turns out to be strongly heterogeneous. Exploring this heterogeneity using multivariate analysis, I find that the extent of code a firm reveals is far from random. Instead, the analysis indicates rational cost/benefit considerations. For example, small firms *ceteris paribus* reveal significantly more, likely because, due to resource scarcity, they expect to benefit more from external development support.

Also the type of code revealed is found to be compatible with protection of competitively important IP. This code is in most cases generic, the rationale being that revealing it entails little loss of competitive advantage. However, also code specific to the firm's hardware or application software is often revealed. Here, the rational is rather that protection is provided by a complementary asset. If, however, this specificity becomes too high then code is typically not revealed, since it would not be useful to anyone else.

The motives behind revealing are further explored by survey questions addressing explicitly the reasons to reveal. It turns out that acquiring external development support is indeed the most important motive behind revealing. This finding is not obvious, given that the large majority of respondents develop embedded Linux as part of their job; hobbyists play a minor role. *Ex ante* it was not clear if open source development processes would work in such a mainly commercial environment, moreover one in which firms are concerned about protecting their IP. As it turns out, OSS processes do work nonetheless.

Putting the various dimensions of revealing behavior together, I find that consistent patterns of revealing can be identified for different types of firms. To illustrate, I sketch the specific pattern of revealing for two rather typical firm profiles: a small, young software vendor with relatively long experience in embedded Linux, and a large, established component manufacturer that only recently started using the open source operating system.

To summarize, I find that firms use various means to protect their developments where necessary. At the same time, they benefit from the OSS development process for those developments that do not require protection. They practice "selective revealing" so as to

minimize competitive losses—and are able to do so within the GPL. The patterns of free revealing I find are consistent with profit-maximizing behavior. It thus seems conceivable that OSS, even OSS under the GPL, becomes a standard part of industrial firms' innovative activity. Key is to decide what to reveal and what to protect—i.e., how to repartition innovative activities surrounding OSS into an open and a protected part.

The remainder of the paper is organized as follows. In Section 2, background information is given on firms' benefits and downsides of developing OSS and on embedded Linux. Section 3 presents research design and data. In Section 4, analysis and results are presented. Section 5 concludes with a summary and a discussion. All tables and figures are in the appendix. Further statistics are available as an online supplement.

2 Background

2.1 Benefits and downsides of commercial OSS development

The question what benefits commercial firms could derive from contributing to OSS development has been addressed by various authors.[2] The motives that have been identified can roughly be grouped into five categories: setting a standard and enabling compatibility; increasing demand for complementary goods and services; benefiting from external development support; signaling technical excellence and/or good OSS citizenship; and adapting existing OSS to the firm's needs .

Against these possible benefits a number of potential downsides must be weighed. The central one is obviously that, by publishing some piece of software as OSS, the originator gives up most property rights to that software. This has three main implications. First, software that is freely available to anyone can no longer be sold—customers will at best be willing to pay for convenient packaging. Second, software revealed as OSS can be used also by competitors, which may imply a loss of competitive advantage. Third, the firm that originated the software might lose control over its further development, even when it acts as the official maintainer of the respective public OSS project.

[2] See Behlendorf (1999), Hecker (1999), Raymond (1999), Feller and Fitzgerald (2002), Lerner and Tirole (2002a), Wichmann (2002), Bonaccorsi and Rossi (2004), and Dahlander (2004).

These downsides must be accepted if—in order to set a standard, to garner external development support, or to increase demand for complements—wide diffusion of the software and its source code is desired by the originator. If not, it might be preferable to protect the software—provided, of course, this choice exists. It does not if the piece of software is based on existing OSS under the GPL, so-called "derived work". In this case, it must be licensed under the GPL due to the GPL's "copyleft" characteristic. This lack of choice constitutes a further potential downside of (GPL'ed) OSS. In order to avoid it, the firm would need to abstain from using GPL'ed software in the first place.

So, how can firms appropriate and maximize profits from OSS development despite these downsides? A number of authors have addressed this issue. A recommendation for firms planning to turn proprietary software into OSS is to choose "non-copyleft" OSS licenses or to use an OSS and a proprietary license in parallel (Behlendorf 1999, Hecker 1999, Raymond 1999).

The use of "standard" means of protection—legal mechanisms, secrecy, lead time, and complementary assets—in the case of OSS has been analyzed by Dahlander (2004) using case studies. He finds that four out of five firms in his sample make sure to keep the copyright to the entire software program, thus maintaining a higher level of control over the software. Three of the firms close at least parts of the software, thus using secrecy and legal protection mechanisms. The remaining two firms keep their software open, relying instead on a committed developer community as a complementary asset. Note, however, that only the last of these solutions is viable in the case of derived work based on GPL'ed software.

A protection and appropriation means independent of the type of licensing is provided by complementary assets (Teece 1986). In the present context, this may be a brand name, as for distributors such as Red Hat and SuSE (e.g., Feller and Fitzgerald 2002); a developer community committed to the respective firm and its software (Dahlander 2004); proprietary software specific to the OSS in question; or hardware to which the respective OSS is tailored.

Despite these results on protection mechanisms available for OSS, further research is needed. First of all, several of the mechanisms described above are not available when a firm develops software based on existing OSS under the GPL. Given the wide diffusion of Linux and the dominance of the GPL among open source licenses (Lerner and Tirole 2002b), this case obviously merits particular attention. Second, there is no study that analyzes the decision to reveal code or not on the level of individual code contributions—acknowledging the fact that firms might reveal selectively only some of their developments. Third and finally, there is

no quantitative empirical study that relates a firm's revealing behavior to its characteristics, thus providing a deeper understanding of the motives and reasons that drive revealing. This paper is the first to address these issues. It does so by studying "embedded Linux".

2.2 Embedded Linux

The term "embedded Linux" denotes versions of Linux used in embedded devices such as mobile phones, VCRs, and machine controls. Embedded Linux has experienced rapid development over the last years (Webb 2002). According to a recent survey, it has become one of the top choices for embedded systems, the other top contenders being Microsoft's Windows and Wind River's VxWorks (VDC 2004). Due to high heterogeneity of embedded devices, there is no standard version of embedded Linux. Correspondingly, "developing embedded Linux" refers to the development of modules or extensions that make Linux suitable for embedded systems. Examples are the *RTAI* real-time module ("Real-Time Application Interface"), the toolkit *busybox*, the shrinked C library *uclibc*, and architecture-specific code for processors used in embedded devices.

Such code is to be regarded as "derived work" in the sense of the GPL, which governs the use of Linux. This implies that, by the time a device containing embedded Linux comes onto the market, the source code of the version of Linux it contains must be made available to all buyers. Unless the number of buyers is very small, this implies that the code is all but publicly available, in particular to competitors. Still, considerable leeway exists with respect to revealing or protecting one's developments for embedded Linux, as will be laid out in detail in Section 4.1. This leeway makes embedded Linux well suited for the questions at hand, since variations in the decision to reveal—patterns of free revealing—can be analyzed.

Further characteristics add to making embedded Linux a suitable object of study for my purpose. Nearly all developments come from firms—device manufacturers, component manufacturers, and dedicated software firms—while hobbyists play only a minor role. Hence, garnering support from hobby developers can be ruled out as a major motivator for firms to engage in this field of OSS. Furthermore, these firms benefit from contributing to embedded Linux not by increasing demand for complements, nor by pursuing strategic interests as, e.g., in the case of IBM supporting Linux as a server operating system. Instead, embedded Linux constitutes, for device manufacturers, a part of their products. For that purpose, diffusion of the respective developments is not required. Hence, within the boundaries defined by the GPL, firms weigh the expected benefits that diffusion might nonetheless bring against the

downside of giving away their IP. As a result, they typically neither reveal all nor nothing, but reveal or protect their developments selectively.

Embedded Linux thus offers a perfect opportunity to explore patterns of free revealing. Given the industrial context in which it is used, it also allows to study the (partial) opening-up of traditional proprietary innovation processes—in other words, the diffusion of OSS practices into firms' R&D labs. This opening-up concerns the operating system, a product component which is of vital importance but typically not an important differentiator (even though this may be the case). The case of embedded Linux thus demonstrates a repartitioning of innovative activity into an open and a protected part. The decision that firms face is where to draw the line—how to shape, in accordance with the GPL, their individual pattern of free revealing.

3 Research Design and Data

Between November 2003 and March 2004 a web-based questionnaire on the development of embedded Linux was online. The questionnaire was developed based on a series of 30 interviews with industry-participants in the field of embedded Linux (Henkel 2003). It targeted developers, since they actually perform the act of revealing code and should thus be best informed about it. The survey was advertised on web portals and mailing lists dedicated to embedded Linux development. It contained 115 questions and yielded 268 valid responses. A comprehensive description of the data is given by Henkel and Tins (2004).

Participants were asked to indicate what type of organization they work for, with five possible alternative answers. 22.4% of respondents described their employer as a "Software company specializing on embedded Linux", 42.5% as a "Device manufacturer", and 8.6% as a "Manufacturer of components like chips and boards". The remaining 26.5% of respondents ticked "I am working as a hobbyist" (15.3%) or "University or other non-profit research organization" (11.2%).

Hence, only a minority of respondents conform to the cliché of the OSS developer as a hobbyist. Weighted by the number of hours per week spent on embedded Linux, this result becomes even more pronounced. In addition, hobbyists are likely over-represented in the sample, due to lower time pressure and higher emotional involvement than employed programmers (which is likely also true for university programmers). The notion that hobbyists play a minor role in embedded Linux development is thus supported. On the other hand, 60%

of employed programmers continue to work on embedded Linux in their spare time, even if only for a few hours per week. Hence, motivations similar to those of hobbyists also do play a certain role.

Participants are highly experienced programmers, with on average 14.2 years of experience in developing software. Average experience in developing OSS and embedded software, resp., is 4.9 and 7.1 years.

Those respondents working for commercial firms were asked some information on their employer. It turns out that software firms in the sample are comparatively young and small, with a median founding year of 1997 and 64% of respondents working for smaller firms (max. 50 employees). Device manufacturers have a median founding year of 1988, and 54% of respondents work for smaller firms. Component manufacturers, finally, are on average the oldest and largest firms, with median founding year of 1986 and only 26% of participants working for smaller firms. As to experience in developing embedded Linux, the distribution is strongly left-skewed, with very few firms (12.5%) having started between 1994 (the earliest date) and 1999. The mean starting year is 0.75 years earlier for software than for hardware firms ($p < 0.01$ in one-sided t-test).

4 Analysis and Results

In order to understand firms' patterns of revealing, one has to analyze both their revealing behavior and the reasons behind it. The following two subsections provide a general picture, describing protection and revealing behavior on an aggregate level. 4.1 addresses the question by what means firms protect their OSS code, and how frequently these means are used. 4.2 then looks at the share of code that is revealed, and its development over time. The next two subsections are descriptive on a more detailed level, and allow to draw first conclusions on the reasons and motives behind revealing. In 4.3 I explore, using multivariate analysis, how firm characteristics can explain heterogeneity in the amount of revealed code. The subject of 4.4 is the type of code that firms reveal. Finally, 4.5 presents the reported reasons of why firms reveal code. In the concluding subsection 4.6, I put the various perspectives together to establish the patterns of revealing for typical types of firms.

4.1 Ways to Protect Derived Work

The GPL stipulates that the source code of derived work based on GPL'ed software must be made available to all receivers of the software. In the case of embedded Linux this means that when a device comes to market, any buyer is entitled to obtain the source code. This regulation, strict as it appears, nonetheless leaves room for protection.

First, derived work does not need to be made public—it must only be supplied, including its source code, to customers. If these are few—or even only one as in the case of commissioned OSS development—and if the customers themselves are not interested in revealing the code (and not obliged to by the GPL), then the software can effectively be kept secret. As Table 1 shows, 46.6% of respondents working for software firms stated their firm reveals code "sometimes" or more often only to its customers.

Second, even if a device is sold to the mass market, the developing firm can nonetheless delay and restrict diffusion by providing the source code on demand only, and without active support. Since, on average, about 18 months pass between development of the code and market launch of the device (Henkel 2003), considerable protection can be attained this way. Also this means is frequently used: 45.0% of all respondents working for commercial firms[3] stated that "sometimes" or more often their firm reveals code only when the device containing it comes onto the market, and only when buyers request it.

Third, the period of time between code development and market launch can be used to optimize the timing of active revealing. When at some point—after a year, say—the advantages of increased lead time are outweighed by the benefits of revealing (standard setting, external development support etc.), then the code is submitted to the corresponding public OSS project. This means of protection—i.e., to actively reveal code, but only after a certain delay—was said to be used at least "sometimes" by 35.7% of respondents.

Fourth and finally, it is accepted (if disputed) practice to make drivers available only as loadable binary modules, not as source code (Marti 2002). This turns out to be the most commonly used means, with 53.1% of respondents stating that their firm uses it at least sometimes; 16.1% even indicated "always".

---

[3] Unless noted otherwise, all further percentages refer only to respondents working for commercial firms (N=197).

Further means of protection, which were not surveyed, must be applied before code is written as derived work of GPL'ed code. Often, re-designing the code architecture allows to shift to the (proprietary) application layer those functionalities that require protection (Henkel 2003).

Hence, despite the GPL various means to obtain a certain protection of one's code exist, and are routinely used. Still, firms do reveal considerable amounts of code nonetheless, as will be reported in the following.

4.2 Revealing of Code—Extent and Change over Time

When quantifying the amount of code for embedded Linux that a firm reveals, a distinction is in place. Code that is so specific to a particular device that it is of no value for other purposes will in most cases not be revealed: public OSS projects would not accept the contribution, and offering it for download on the firm's website would remain without consequences.[4] For such code, protection is not an issue. Hence, the corresponding survey question excludes overly specific code: "*Please consider those embedded Linux developments by your firm that are potentially useful for others. That is, they are not too specific to your firm, and others would benefit (use them, develop them further) if they were made public. What share of this code is freely revealed? [0 – 100%]*" "Revealing" was defined as follows: "*the code is actively made public, be it for downloading on a website, as a posting on a mailing list, or as a submission to the maintainer of a module.*"

Table 2 provides descriptive statistics of the answers to the above question. Hobbyists and developers in universities exhibit what one could call a "typical" open source behavior, revealing nearly all of the code in question (mean = 92%, median = 100%). In contrast, the mean across the three commercial categories is 49.3% (median = 50%). This value lies right in the middle between "typical" proprietary behavior and "typical" open source behavior. So, firms on average do make a considerable share of their developments public—49% is very much for an organization used to proprietary processes. However, they do not blindly follow the popular myth that derived work of GPL'ed code has to be made public. Instead, they protect, on average, about half of their developments for embedded Linux.

[4] Morrison et al. (2000) report similar findings in their study on user innovation on library computer systems.

A closer look at Table 2 shows that revealing behavior varies strongly between firms. For all three categories, the standard deviation of the share of revealed code is above 35%, the minimum at 1%, and the maximum at 100%. It will be the subject of the following section to explain some of this heterogeneity by observed characteristics of the firms.

One might conjecture that revealing in the field of embedded Linux is a left-over from the hype that surrounded OSS in 1999 and 2000, and that it decreases further over time. However, the survey yielded exactly the opposite result. Only 10.3% of "commercial" respondents (total responses: 145) stated that their company reveals less now than in 2000. 40.7% did not see a change, while nearly half of all respondents (49%) said their company reveals "somewhat more" or "much more" than in 2000. Hence, code sharing by commercial firms in the field of embedded Linux is anything but a dying-out left-over from a hype; quite the contrary, it seems to be on the way to becoming mainstream behavior.

4.3 Firm Characteristics as Determinants of Revealing

The descriptive analysis of the amount of revealed code showed high heterogeneity between firms (Table 2). In the following, I present a multivariate analysis that aims at explaining some of this heterogeneity by firm characteristics. Since the share of revealed code is restricted to the interval [0%,100%], a Tobit regression model is the natural choice.[5] The choice of explanatory variables was guided by hypotheses. However, since the main purpose of this analysis is exploratory, not hypothesis-testing, I do not explicitly introduce hypotheses.

Table 3 shows the regression outcome. Specification (1) contains the full model, while specification (2) is obtained by successive elimination of insignificant variables. A test on the hypothesis that the three eliminated variables are also jointly equal to zero can not be rejected (Likelihood Ratio test, $p = 0.49$).[6]

---

[5] In contrast to an OLS regression, a Tobit model accounts for the censoring of the dependent variable. In my case this means that the share of revealed code can not be less than 0%, nor larger than 100%. To check robustness of results, I also used an Ordered Probit model with the dependent variable indicating the quintile of the interval [0%,100%] in which the share of revealed code lies. This model allows for a non-linear dependence on the explanatory variables inside the interval. Results are qualitatively the same as in the Tobit model.

[6] The pseudo R^2 value seems rather low for both specifications. However, a pseudo R^2 can not be interpreted as an R^2 in an OLS regression. To give an idea of the size of the explained variance, a standard OLS regression

Firm size: Active participation in the open source development process can provide firms with external development support as well as a low-cost marketing channel (Gruber and Henkel 2004). Due to resource constraints, both should be relatively more important for small than for large firms. And indeed, the coefficient for the dummy variable "SizeLarge" indicating that the firm is large (more than 200 employees) is significantly negative. This finding is consistent with the high level of agreement (average: 1.42 on a scale from -2 to +2) that the statement "Open source software allows small enterprises to afford innovation" received (only commercial respondents, N = 185 of 197). The dummy variable "SizeMedium" coding medium sized firms (11 – 200 employees) is insignificant; obviously, this group is not different enough from the reference group (1 – 10 employees).

Familiarity with embedded Linux: Embracing the open source development process—and hence, freely revealing code—requires a change from the traditional proprietary attitude to an "open" one. This change will rarely be accomplished quickly, such that firms new to embedded Linux development should reveal less than more experienced firms. In addition, novices might not yet have high-quality code to reveal. The regression analysis confirms this expectation, yielding a positive and significant (5% level) coefficient of the firm's experience in developing embedded Linux ("CompYearsEL", measured in years).

Firm policy towards revealing: Respondents were asked about their firms' official policy towards revealing OSS developments. Two dummy variables capture the effects such policies might have on revealing: "PolEncourages" equals 1 if the respondent ticked the statement "*My company encourages me to contribute source code that is not critical for competition*" (agreed to by 22.8%) and "PolRestrictive" equals 1 if the person agreed to "*My company is very restrictive in revealing code*" (agreed to by 16.8%). While "PolEncourages" does not exhibit a positive effect, "PolRestrictive", as expected, carries a significantly negative coefficient.

Complementary assets: Device and component manufacturers develop code as a complement to their hardware, which can be expected to act as a—specialized—complementary asset. Hence, *ceteris paribus* hardware manufacturers should stand less to lose from revealing than software firms. Regression analysis partly confirms this expectation: the dummy variable "TypeCompMan" carries a positive and significant coefficient in the final

---

(leading to roughly the same coefficients and significances as the Tobit model) yields an R^2 of 0.14. This still leaves much variance unexplained—as expected—but is a much more reasonable value than 0.02.

specification, while “TypeDevMan” is insignificant (reference group: software firms). The most likely interpretation of this finding is that OSS developed by component manufacturers is, in most cases, driver software highly specific to the components that the firm sells. In the case of device manufacturers, the specificity of the code to their hardware is probably lower, such that the hardware as a complementary asset affords only a lower level of protection.

To summarize, the observed patterns of free revealing can sensibly be related to firm characteristics. The results show that a firm’s size, its experience with the OSS process, its policy towards revealing, and the level of protection provided by complementary assets are determinants of revealing behavior. They also suggest that external development support is an important reason to reveal.

4.4 Type of Code that is Revealed

Having established how code is protected, how much of it is revealed, and how the amount of revealed code relates to firm characteristics, the next question is what *type of code* firms typically reveal. The questionnaire offered five statements relating to the type of code, and participants were asked to indicate their agreement on a 5-point ordinal scale. Figure 1 shows the share of agreeing and disagreeing responses, separately for software and hardware firms.

The highest agreement, for both types of firms, received the statement that the revealed code is “generic”—with 63% agreement from hardware firms and even 85% from software firms. This is a very plausible finding, since revealing generic code should not harm the competitive position of the company (cf. Schrader 1991, Fauchart 2003).

Yet, this is not the whole story. Also the statements “is important for our competitive position” and “helps to differentiate our product from others” received more agreement than disagreement. While the net agreement (i.e., share agreement minus share disagreement) is small in the case of hardware manufacturers, it is considerable in the case of software firms (27.1% and 30%, resp.). A likely interpretation of this finding is that revealing for these firms is not at odds with, or may even support, differentiation and competitive advantage. This may be the case when revealing signals the firm’s competences or when it creates demand for related services (for which the respective firm is best qualified) or related proprietary software.

The logic of complementary assets as a protection mechanism applies when the code is specific to some hardware or application software. Net agreement to the statement “is specific

to our hardware" is relatively large (34.5%) from hardware firms; similarly, respondents working for software firms show a relatively high net agreement (28.6%) to the statement "is specific to our application software".

Differences in the distribution of responses between software and hardware firms are highly significant for the statement "is generic" ($p = 0.001$, Mann-Whitney test), as the figure suggests. They are significant on the 5-percent level for "helps to differentiate" ($p = 0.037$) and "is specific to our hardware" ($p = 0.01$), and insignificant for the remaining two statements. These differences confirm what became apparent in Section 4.3, namely, that the observed patterns of revealing—and most plausibly also the *optimal* patterns—depend on the type of firm.

Since some of the statements contradict each other, consistency requires that their respective levels of agreement be negatively correlated. This is indeed the case. In particular, there exists a highly significant negative correlation between agreement to "is generic" and both statements relating to specificity of the code. A strong positive correlation exists, as one would expect, between "competitively important" and "differentiating", and also between the two types of specificity.[7]

Two responses to the open question "*What types of development would your company typically make public?*" add a further perspective to the above statistical findings. The trade-off between costs and revenues linked to proprietary software is behind the first quote: "*[We reveal code] where the cost of support is more then the profit made on sales. We then give it out without support.*" The second quote in a way summarizes the basic idea behind finding the optimal pattern of revealing: "*Anything that will not give our competitors the possibility to copy our products one to one in a short time.*" Against this (and other) downside, firms weigh a number of potential benefits, which are dealt with in the following.

4.5 Reasons to Reveal

Rounding off the picture of firms' patterns of revealing, reasons to reveal as reported by the survey participants are presented in the following. Based on the interviews I conducted as

[7] Spearman rank correlation equals –0.292 between "generic / specific to hardware"; –0.401 between "generic / specific to application software"; 0.727 between "competitively important / differentiating"; and 0.431 for "specific to hardware / specific to application software". Significance level $p < 0.0002$ in all cases.

well as a literature analysis (see Section 2.1) twelve potential reasons why firms would reveal their code were identified. Participants were offered these reasons and were asked to indicate their agreement to the statements "*My company reveals code because [reason x]*" on a 5-point ordinal scale. Results are shown in Figure 2, for hardware manufacturers only. This differentiation is required because hardware and software firms differ considerably with respect to some of these reasons. The focus is laid on hardware manufacturers since they are more numerous (137 vs. 60) in the sample. I will briefly address the corresponding results for software firms at the end of this section.

The highest agreement received the statement "My company reveals code because the GPL requires it." This was to be expected—despite ways to circumvent it, the GPL obviously does have a strong effect on revealing behavior.[8] However, the four reasons on ranks 2 to 5 received only slightly less net agreement. Three of them—bugfixes by others, further development by others, and reduced maintenance effort—are directly related to informal outside development support, that is, to the technical benefits of the open source process. The remaining one, on rank 2 ("to appear as a good OSS player"), is indirectly related to that same aspect, since receiving support from an open source community (be it commercial or not) requires to be regarded as a good open source player (Osterloh et al. 2001, Franck and Jungwirth 2003). Hence, firms do seem to derive technical benefits from the open source development process in embedded Linux, and are willing to share their code with others in order to realize these benefits.

For the reasons on ranks 6 and higher, the level of agreement drops significantly compared to rank 5. On rank 6, still receiving twice as much agreement as disagreement, a reason related to marketing appears: "revealing good code improves our company's technical reputation." Further details can be obtained from Figure 2.

In order to check consistency of the responses and to identify potential latent constructs behind the various reasons to reveal, an exploratory factor analysis was carried out. With three components, it explains 60.0% of total variance and yields good quality measures (KMO: 0.80, $p < 0.001$). The resulting components can unambiguously be labeled

[8] Given the survey's definition of "revealing" (see 4.2), the GPL does not require revealing at all, strictly speaking. Still, the requirement of the GPL to make the code *eventually* available may constitute a reason to do so *proactively*. In any case, this reason being on rank 1 does not affect the findings on the importance of the other reasons.

"development support" (ranks 3, 4, 5, 7, 9 in Figure 2), "marketing/networking" (ranks 6, 8, 10, 11, 12), and "open source" (ranks 1, 2). Hence, consistency of responses is clearly confirmed, and plausible latent constructs could be identified.

An analogous analysis was carried out for software firms. The main difference is that reasons related to marketing ("revealing good code improves our company's technical reputation" and "visibility on the mailing list is good marketing") rank much higher (ranks 3 and 6, in contrast to ranks 6 and 10 for hardware manufacturers). This can be explained by the fact that these software firms act as suppliers to hardware manufacturers, performing commissioned development and/or selling tools or specific distributions of embedded Linux.

4.6 Typical Patterns of Revealing

In order to illustrate how the above findings can be put together to obtain a comprehensive picture of firms' revealing behavior, I describe in the following two typical patterns of revealing.

The first profile is that of a small and young software vendor specializing in commissioned development, and having started embedded Linux development relatively early. As pointed out in Section 3, software firms in the sample are significantly younger and smaller than hardware firms. Such a firm would reveal relatively large shares of the code it develops: both its small size and its long experience were found to positively influence the extent of revealing (see 4.3).

This software firm would typically reveal *generic* code: all three characteristics (small size, young age, being a software vendor) show a highly significant correlation to the revealed code being generic.[9] This is a plausible finding since such firms typically lack complementary assets that would protect more specific developments; furthermore, as was stated by several participants, customers commissioning software development will usually only allow generic code to be made public. Still, software firms also significantly more often reveal code that "helps to differentiate" them. This need not be a contradiction to the code being generic: differentiation might come from high quality of the code, which thus serves better for marketing purposes.

[9] Details on this correlation and those mentioned in the following are provided in the online supplement.

Marketing is indeed paramount as a reason to reveal code for the type of firm considered here: the level of agreement to the reasons “revealing good code improves our company’s technical reputation” and “visibility on the mailing list is good marketing” is positively correlated with small size, young age, and being a software firm (significance on 1% level in all case). As one would expect from resource considerations, also external development support counts more for such a firm: further development, bugfixes, and added functionality by others (as reasons to reveal) show a positive correlation to young age and, in the case of functionality, also to small size. Finally, that type of firm also showed a higher level of agreement to the reason “we identify potential employees by looking at suggestions on our code.”

The second profile that shall illustrate typical patterns of revealing is that of a large, established component manufacturer that only recently has started to use embedded Linux. As the regression analysis has shown, large size and little experience in using the open source operating system affect the amount of revealed code negatively; on the other hand, being a component manufacturer has a positive effect. Considering the size of the coefficients in Table 3, the effects might more or less cancel out each other compared to the median firm in the sample, such that the firm under consideration reveals an intermediate amount of code.

This firm, as a component manufacturer, will significantly more often than other types of firms reveal code that is important for its competitive position, differentiating, and specific to its hardware. As was argued above, this code will mostly be drivers specific to the firm’s components. Hence, revealing it does not harm the company’s competitive position due to protection by a complementary asset.

These particular circumstances of revealing are also reflected in the reasons to reveal. The one reason to which component manufacturers’ agreement differs significantly from that of other firms is “because others develop the code further”—obviously something not very useful when the code is tailored to a certain component.

These two examples have illustrated that patterns of revealing—what is revealed and why—differ strongly between different types of firms. But, more importantly, they have also shown that for a particular type of firm the different dimensions of revealing behavior add up to yield a consistent, specific pattern of revealing.

5 Conclusion

There is a long-standing debate on how profits from innovations can best be appropriated.[10] This discussion usually makes an implicit a priori: It presupposes that exclusivity is desirable for the innovator. It thus focuses on the *protection of innovations*, while what actually matters is *appropriation of profits* from innovation. The two often go along with each other—but there are important exceptions. As the rise of OSS has impressively shown, freely revealing one's developments may sometimes be a sensible thing to do, in particular when community-based development is viable (von Hippel 2001). This is favored when, as for embedded Linux, needs are strongly heterogeneous (cf. Bessen 2001, Franke and von Hippel 2003, Henkel 2004) and the underlying technology is highly modular (Baldwin and Clark 2003).

On the other hand, developing OSS is often considered to *imply* free revealing. Also this view is not correct: Commercial OSS development, even if based on GPL'ed software, perfectly well accommodates a combination of free revealing and various means of protecting one's code. Firms thus have the chance to practice *selective revealing*.

A case where such combination of revealing and protection is common has been analyzed in this paper, namely, embedded Linux. Firms in the sample routinely use various means of protection that are consistent with the GPL. Among them are restricted or delayed revealing and the use of binary loadable modules for drivers. As a result firms reveal, on average, about half of the code they have developed for embedded Linux, while protecting the other half.

Between firms, revealing behavior is strongly heterogeneous, which can partly be explained by firm characteristics. Among other things I find that the amount of revealed code *ceteris paribus* is larger for smaller firms, which likely benefit more from external development support. It is also higher for component manufacturers, likely because these firms enjoy a good protection by complementary assets (the components they sell). Finally, experience with OSS matters: the longer a firm has been using embedded Linux and the higher hence its familiarity with the OSS development process, the more it reveals.

[10] See, e.g., Horstmann et al. (1985), Levin et al. (1987), Harabi (1995), Cohen et al. (2000), Arundel (2001), Cohen et al. (2002), Gallini and Scotchmer (2002), and Sattler (2003).

Heterogeneity between firms was also found with respect to the type of code they reveal. While this code was described as "generic" most frequently, also the statements "is specific to our hardware / application software" received relatively high agreement. Respondents working for software firms even showed a relatively high net agreement of about 30% to the statements that the code "is important for our competitive position" and that it "helps to differentiate our product". Thus, a different logic applies to different instances of free revealing.

Among the reported reasons to reveal, informal outside development support figures prominently—firms indeed seem to derive technical benefits from the open source development process. Thus, even in this mainly commercial environment, the private-collective model of innovation described by von Hippel and von Krogh (2003) seems to work. For software firms, also motives related to marketing rank relatively high.

Putting the different dimensions of revealing behavior together—the amount and type of revealed code as well as the reasons to reveal—allows to identify various typical types of firms that each are characterized by a specific pattern of free revealing. The cases of a small, young software vendor and that of a large, established component manufacturer served to illustrate these specific patterns.

A point that merits discussion and further research is the role played by the individual developer. One might even conjecture that revealing by firms is in fact driven by OSS enthusiasm of the programmer—possibly against his employer's interests. However, neither the survey nor my interviews support this view. Employed programmers *are* somewhat enthusiastic about OSS, but significantly less so than hobbyists and university programmers. While a detailed discussion is beyond the scope of this paper, it seems save to conclude that programmers do often act as "champions of revealing", but not as uncontrollable OSS enthusiasts disregarding their firm's interest.

The question then arises what effect selective revealing has on the OSS process and the OSS community in this field. One might conjecture that the chance to protect *some* developments leads, over time, to a more and more restrictive attitude, and a gradual dying-out of public OSS projects in this field. However, my results suggest otherwise. It seems in fact more plausible that the chance to reveal selectively does not harm the OSS process, but makes it possible in the first place: otherwise, firms might prefer to stick to proprietary software altogether. Selective revealing thus facilitates that OSS, even OSS under the GPL, becomes a standard part of industrial firms' innovative activity.

Open innovation processes as observed in the field of embedded Linux carry a considerable potential for efficiency gains (Foray 2004, Henkel and von Hippel 2005). They allow to reduce duplication of effort and to avoid transaction costs of commercial licensing. Their downside is that they invite free riding and will disadvantage firms that reveal too much. But even for OSS under the GPL, and much more so in other instances of innovation, openness is not an all-or-nothing decision. Firms can and should balance revealing and protection in such a way as to optimize their pattern of free revealing.

References

Arundel, A. 2001. The relative effectiveness of patents and secrecy for appropriation. *Research Policy* **30**(4) 611–624.

Baldwin, C. Y., K. B. Clark. 2003. The architecture of cooperation: how code architecture mitigates free riding in the open source development model. *Working Paper, Harvard Business School.*

Behlendorf, B. 1999. Open source as a business strategy. C. Dibona, S. Ockman, M. Stone, eds. *Open-sources: Voices from the open source revolution*. O'Reilly, Sebastopol, CA. 149-170.

Bessen, J. 2001. Open source software: free provision of complex public goods. *Working Paper 5/01*. http://www.researchoninnovation.org/opensrc.pdf.

Bonaccorsi, A., C. Rossi. 2004. Comparing motivations of individual programmers and firms to take part in the open source movement. From community to business. *Working paper* Sant'Anna School of Advanced Studies Pisa.

Cohen, W. M., A. Goto, A. Nagata, R. R. Nelson, J. P. Walsh. 2002. R&D spillovers, patents and the incentives to innovate in Japan and the U.S. *Research Policy* **31**(8-9) 1349–1367.

Cohen, W. M., R. R. Nelson, J. P. Walsh. 2000. Protecting their intellectual assets: appropriability conditions and why U.S. manufacturing firms patent (or not). *NBER Working Paper* No. w7552.

Dahlander, L. 2004 Appropriating returns from open innovation processes: A multiple case study of small firms in open source software. *Working Paper* Chalmers University of Technology, Gothenburg http://opensource.mit.edu/papers/dahlander.pdf.

Fauchart, E. 2003. On knowledge sharing patterns among rival firms: The case of knowledge on safety. *Working Paper*. http://userinnovation.mit.edu/papers/safety3.pdf.

Feller, J., B. Fitzgerald. 2002. *Understanding Open Source software development*. Addison Wesley, Boston, MA

Foray, D. 2004. *Economics of Knowledge*. MIT Press, Cambridge, MA.

Franck, E., C. Jungwirth. 2003. Reconciling investors and donators—The governance structure of open source. *Journal of Management and Governance* **7** 401-421.

Franke, N., E. von Hippel. 2003. Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software. *Research Policy* **32**(7) 1199-1215.

Gallini, N. T., S. Scotchmer. 2002. Intellectual Property: When is it the best incentive system?, in: A. Jaffe, J. Lerner, S. Stern (eds.), *Innovation policy and the economy*, vol. 2 , MIT Press, Cambridge, MA.

Gruber, M., J. Henkel. 2004. New ventures based on open innovation—an empirical analysis of start-up firms in embedded Linux. *Working Paper*, University of Munich. http://opensource.mit.edu/papers/gruberhenkel.pdf.

Harabi, N. 1995. Appropriability of technical innovations – an empirical analysis. *Research Policy* **24**(6) 981–992.

Hecker, F. 1999. Setting up shop: The business of open-source software. *IEEE Software* **16**(1) 45-51.

Henkel, J. 2003. *Open-Source-Aktivitäten von Unternehmen—Innovation in kollektiven Prozessen*. Unpublished Habilitation thesis, University of Munich.

Henkel, J. 2004. The jukebox mode of innovation – a model of commercial open source Development. *CEPR Discussion Paper* 4507.

Henkel, J., M. Tins. 2004. Munich/MIT Survey: Development of embedded Linux. *Working paper*. http://opensource.mit.edu/papers/henkeltins.pdf.

Henkel, J., E. von Hippel. 2005. Welfare implications of user innovation. *Journal of Technology Transfer*, forthcoming.

Horstmann, I., G. M. MacDonald, A. Slivinski. 1985. Patents as information transfer mechanisms: To patent or (maybe) not to patent. *Journal of Political Economy* **93**(5) 837-858.

Lerner, J., J. Tirole. 2002a. Some simple economics of open source. *Journal of Industrial Economics* **50**(2) 197–234.

Lerner, J., J. Tirole. 2002b. The scope of open source licensing. *NBER Working paper* W9363.

Levin, R. C., A. Klevorick, R. R. Nelson, S. G. Winter. 1987. Appropriating the returns from industrial research and development. *Brookings Papers on Economic Activity* (3) 783–820.

Marti, D. 2002. Use binary-only kernel modules, hate life. *LinuxJournal.com*, 06/19/2002. http://www.linuxjournal.com/article.php?sid=6152.

Morrison, P. D., J. H. Roberts, E. von Hippel. 2000. Determinants of user innovation and innovation sharing in a local market. *Management Science* **46**(12) 1513-1527.

Osterloh, M., S. Rota, M. von Wartburg. 2001. Open source – New rules in software development. *Working Paper* Institute for Research in Business Administration, University of Zurich. http://www.ifbf.unizh.ch/orga/downloads/OpenSourceAoM.pdf.

Raymond, E. S. 1999. *The cathedral and the bazaar: Musings on Linux and open source by an accidental revolutionary*. O'Reilly, Sebastopol, CA.

Sattler, H. 2003. Appropriability of product innovations: An empirical analysis for Germany. *International Journal of Technology Management* **26**(5-6) 502–516.

Schrader, S. 1991. Informal technology transfer between firms: Cooperation through information trading. *Research Policy* **20**(2) 153–170.

Teece, D. J. 1986. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy* **15**(6) 285–305.

VDC. 2004. White paper, Venture Development Corporation. Referenced in: Linux now top choice of embedded developers. *LinuxDevices.com*. http://www.linuxdevices.com/news/NS2744182736.html.

von Hippel, E. 2001. Innovation by user communities: Learning from open source software. *MIT Sloan Management Review* **42** (Summer) 82–86.

von Hippel, E., G. von Krogh. 2003. Open source software and the "Private-Collective" innovation model: Issues for organization science. *Organization Science* **14**(2) 209–223.

Webb, W. 2002. Pick and Place: Linux grabs the embedded market. *edn.com.* http://www.reed-electronics.com/ednmag/contents/images/253780.pdf.

Wichmann, T. 2002. FLOSS final report – part 2: Free/Libre open source software: Survey and study – firms' open source activities: Motivations and policy implications. *Berlecon Research.* http://www.berlecon.de/studien/downloads/200207FLOSS Activities.pdf.

Appendix

Table 1: Frequency of use of various means to protect code

Means of protection	al-ways	often	some-times	rare-ly	never	N	mis-sing
Revealing only to customers*	7.0%	14.0%	25.6%	11.6%	41.9%	43	17
Revealing only on request of device buyers	5.8%	22.5%	16.7%	16.7%	38.4%	138	59
Revealing only after delay	2.8%	11.4%	21.3%	19.9%	44.7%	141	56
Loadable binary modules	16.1%	19.5%	17.5%	10.7%	36.2%	149	48

* This means of protection applies to software vendors only. Observations with missing data or with a reply "don't know" are not included in the calculation of percentages.

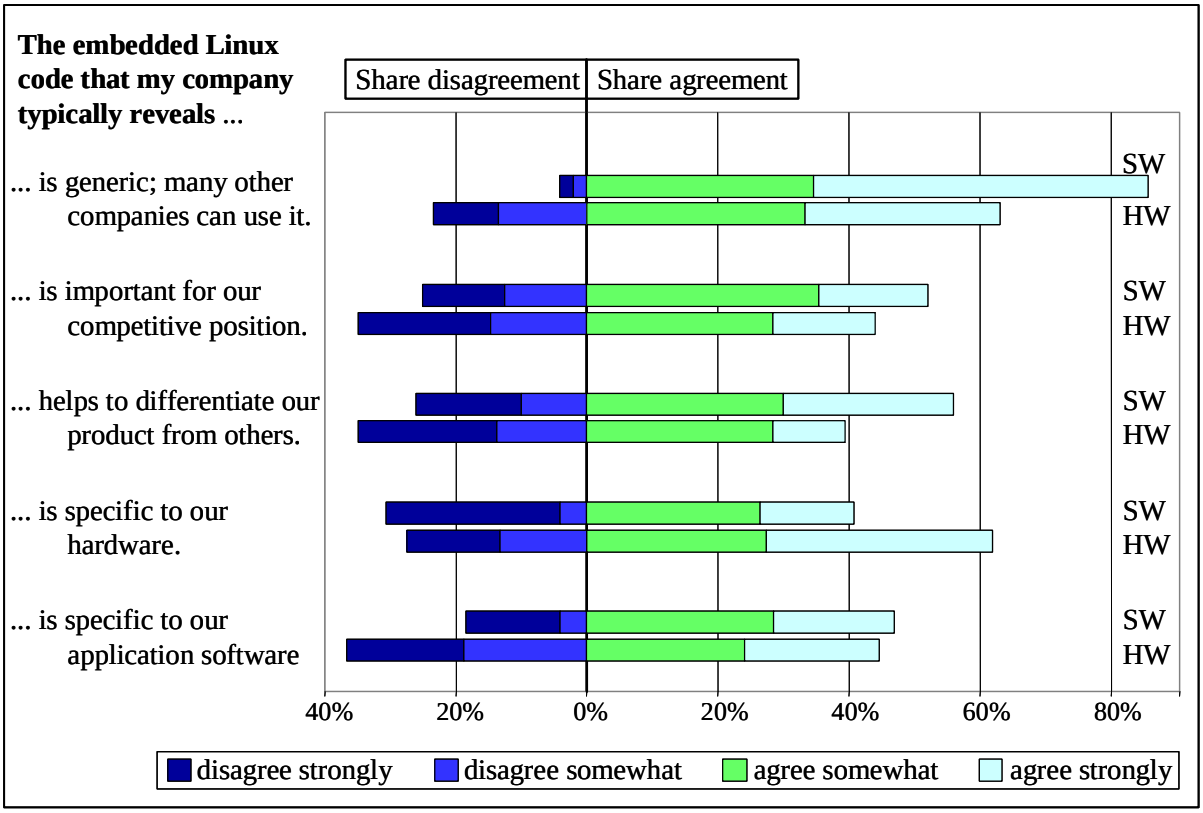
Table 2: Share of code that is revealed

Type of organization	Mean	Median	St. dev.	Min.	Max.	N	mis-sing
Software company	57.5%	60%	35.9%	1%	100%	39	21
Device manufacturer	42.3%	25%	36.3%	1%	100%	69	45
Component manufacturer	58.8%	60%	40.2%	1%	100%	17	6
Commercial, all	*49.3%*	*50%*	*37.3%*	*1%*	*100%*	*125*	*72*
University / non-profit research org.	90.0%	100%	21.0%	30%	100%	20	10
Hobbyist	93.3%	100%	19.8%	25%	100%	31	10
Non-commercial, all	*92.0%*	*100%*	*20.1%*	*25%*	*100%*	*51*	*20*
ALL	*61.7%*	*75%*	*38.4%*	*1%*	*100%*	*176*	*92*

Table 3: Multivariate analysis of share of code that the firm reveals (commercial organizations)

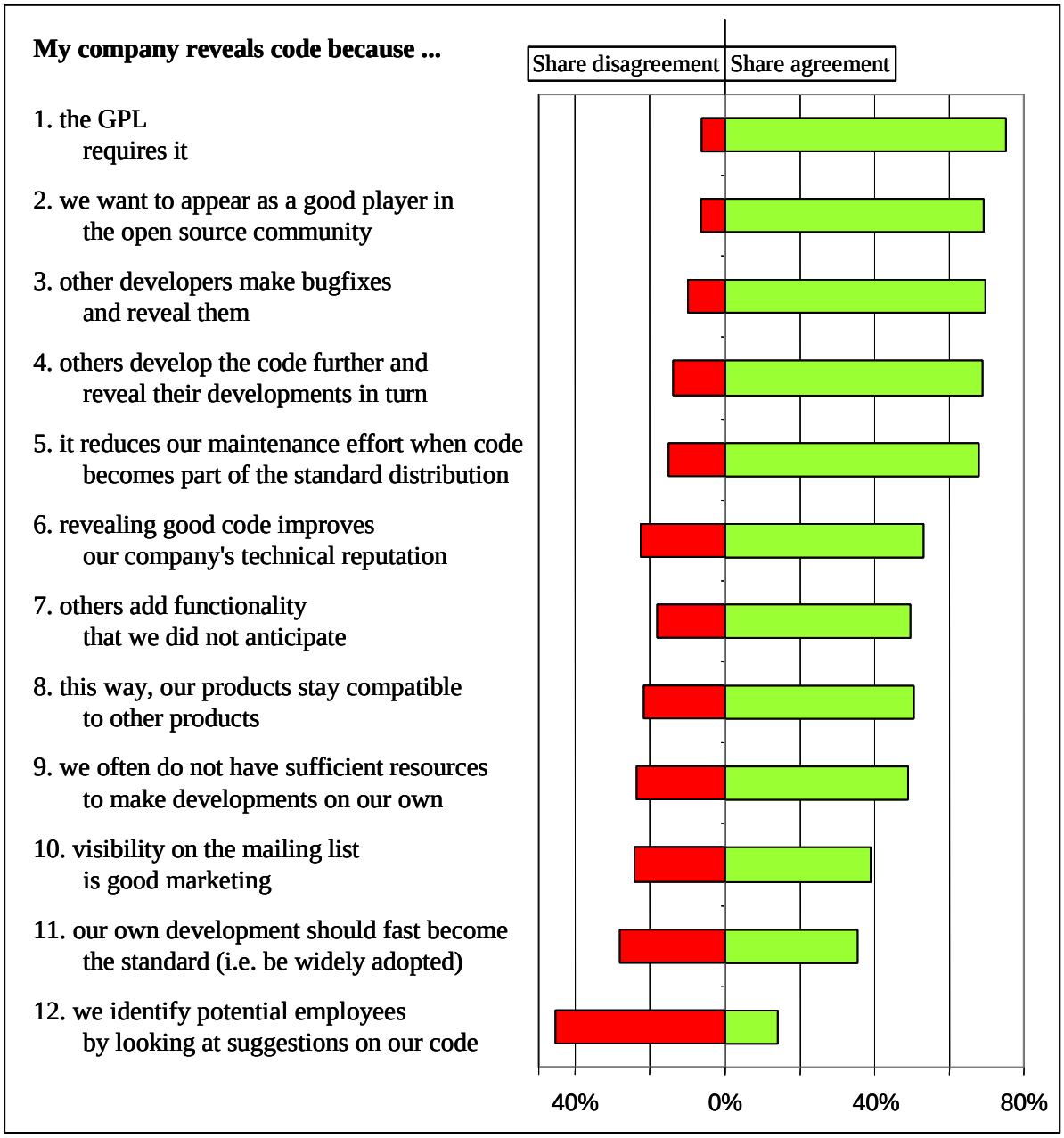
Explanatory variable	**Tobit: ShareRevealed (0 – 100%)**			
	(1)		**(2)**	
SizeMedium	0.152	(9.04)		
SizeLarge	-16.308	(10.29)	-17.329**	(8.53)
CompYearsEL	4.610**	(2.01)	4.990**	(1.96)
PolEncourages	-4.023	(8.08)		
PolRestrictive	-26.021**	(12.55)	-26.758**	(12.50)
TypeDevMan	-12.664	(8.48)		
TypeCompMan	19.649	(13.12)	28.807**	(11.50)
Constant	49.032***	(10.91)	38.634***	(8.27)
Observations	122		122	
Pseudo R^2	0.02		0.02	
Likelihood ratio test	$\chi^2(7)=21.1$, p=0.004		$\chi^2(4)=18.7$, p=0.0009	
* / ** / *** significant at 10% / 5% / 1% level; standard errors in parentheses				

Figure 1: Type of code that is revealed



Note: Shares of agreement and disagreement shown for software and hardware firms (see last column). The number of valid responses varies between 48 and 50 for software firms and between 109 and 113 for hardware firms. Responses “neither agree nor disagree” are not shown.

Figure 2: Share of respondents working for hardware manufacturers (N = 137) that agree / disagree to various reasons to reveal



Note: The number of valid responses varies between 105 and 113. The remainder is either missing or the answer was “don’t know”. The share of respondents answering “neither agree nor disagree” is not shown.