

Promoting the Penguin: Who is Advocating Open Source Software in Commercial Settings?

December 2006

COMMENTS WELCOME

Oliver Alexy¹, Joachim Henkel^{1,2}

¹Schölller Chair in Technology and Innovation Management
Technische Universität München
Arcisstr. 21, D – 80333 Munich, Germany
phone: +49-89-28925741, fax: +49-89-28925742, email: alexy|henkel@wi.tum.de

²Center for Economic Policy Research (CEPR), London

Abstract

Today, all firms using or developing software are facing the question if and how to engage in open source software (OSS). Yet, little is known about the process of OSS adoption and diffusion within corporations. We address this issue. In light of qualitative evidence on the role of programmers in promoting OSS, we focus on the attitudes of employed IT professionals towards corporate OSS engagement. Our analysis is based on 25 interviews and a large-scale survey with 404 participants, performed at the telecommunications department of a multinational company and guided by the theoretical frameworks of Rogers (innovation diffusion) and Davis (Technology Acceptance Model) Our main findings concern the impact of the respondent's job function. They are consistent with theoretical considerations—and still rather surprising. Distinguishing between developers, testers, architects, project managers, and managers, we find that software testers are generally most favorable towards heightening OSS activities, followed by software architects and managers. Apart from project managers, developers are actually *least favorable* towards increased corporate OSS engagement, despite their large experience in working on OSS. Since developers constitute both the basic level of software development and the largest of the named groups, this finding leads us to question that OSS is a generally supported grassroots movement. Introducing OSS and open innovation processes to a corporation thus requires taking each individual's incentives based on her job function into account, and a close look at organizational aspects.

Key words: open source software; open innovation; adoption

1 Introduction

The commercial use of open source software (OSS) has gone through phases of curiosity, hype, and disillusion before reaching pragmatism (Driver et al. 2005). The once dazzling stock-market valuations of companies such as Red Hat and VA Linux have reached sobering levels. Nonetheless, OSS has gained a strong foothold in commercial settings—removing it would lead to a break-down of many firms' IT infrastructure (Driver and Weiss 2005; Goulde 2006) and, in the electronics industry, even of numerous products. Accordingly, many researchers have addressed the issues of OSS business models¹, of collaboration between firms and the OSS community², and of OSS-based collaborative innovation processes between firms³. It thus seems fair to say that the pros and cons of commercial OSS engagement are quite well understood.

However, little is known about how firms *become* OSS adopters. In particular, it is largely an open question who promotes OSS inside the firm, and what job functions and personal characteristics make an individual disposed towards OSS—both towards adopting it, and towards lobbying for it. Anecdotal evidence (Henkel 2004; Moody 2001; Raymond 2001a; Raymond 2001b) suggests the importance of grassroots initiatives by developers. This view is supported by survey results indicating that software professionals tend to champion the revealing of software as OSS (Henkel 2006a). Systematic evidence on this issue, however, is lacking—the adoption and diffusion of OSS within firms is still largely a black box. Yet, understanding this process is of high importance. Active OSS engagement may be vital to a firm in terms of efficiency gains, standard setting, defining the rules of competition, and responding to competitive threats from OSS. Thus, both the tendency of IT professionals to promote OSS and their willingness to adopt it are highly relevant.

The following questions thus arise: who advocates OSS in commercial settings? More precisely, distinguishing three levels of OSS engagement: who advocates *using* existing OSS, *contributing* to OSS projects, and *revealing* proprietary software as OSS? Note that the levels of contributing and revealing amount to a process innovation in software development, which

¹ Bonaccorsi et al. (2006), Dahlander (2005), Feller and Fitzgerald (2001), Grand et al. (2004), Hecker (1999), Lerner and Tirole (2002), Raymond (2001a), Raymond (2001b), West (2003), Wichmann (2002).

² Bessen (2001), Bonaccorsi and Rossi (2004), Dahlander and Magnusson (2005), Dahlander and Wallin (2006), Henkel (2006a), Osterloh and Rota (2005), Shah (2006).

³ Henkel (2006b), Nuvolari (2001), West and Gallagher (2006).

would imply that employees influence organizational structure and processes. Finally, focusing on the individual: what job functions and what personal characteristics make an employee inclined towards OSS?

In this paper, we tackle the above questions. In doing so, we explicitly focus on the level of IT professionals within the firm, in order to analyze the grassroots aspect of corporate OSS adoption. We thus exclude OSS initiatives by top management. Guided by a framework based on the Technology Acceptance Model by Davis (Davis 1989; Davis et al. 1989) and the concept of innovation diffusion by Rogers (2003), we examine what influences commercial OSS adoption and diffusion on the level of software professionals. We studied these issues in the telecommunications department of a multinational corporation over one and a half years. The firm does not belong to the early and outspoken corporate proponents of OSS (such as, e.g., IBM), and so far has no company-wide OSS policy. It is thus well-suited for studying the role of employees in a relatively early phase of commercial OSS adoption and diffusion, without them being prejudiced by existing corporate strategy on OSS. Our results are derived from 25 interviews and a survey with 404 participants.

Our main findings concern the impact of the respondent's job function. They are consistent with theoretical considerations—and still rather surprising. Distinguishing between developers, testers, architects, project managers, and managers, we find that software testers are generally most favorable towards heightening OSS activities, followed by software architects and managers. Apart from the (small) group of project managers, developers are actually *least favorable* towards increased corporate OSS engagement, despite having the largest experience in working on OSS. They also suggest significantly *less* programs generated by the firm as candidates for release as OSS.

Since developers constitute both the basic level of software development and the largest of the named groups, this finding leads us to question that OSS is a generally supported grassroots movement, at least when it goes beyond using existing OSS. For companies, this has consequences for the management of their OSS engagement. Since with probability of 46% a randomly picked developer from our sample is neutral (26%) or even negative (20%) towards revealing proprietary software as OSS, developers should be given the opportunity to self-select into OSS-related projects. The same recommendation can be made for other job functions, in particular project managers. Thus, self-organization of IT professionals for

corporate OSS engagement may work—but each individual’s incentives based on his or her job function need to be considered.

Despite our findings described above, IT professionals *on average* are “somewhat” positive towards OSS activities of their firm. Differentiating by type of OSS activity, we find that “using existing OSS more often” receives the strongest support, followed by “contributing to OSS projects” and, finally, “releasing proprietary software as OSS.” An individual’s agreement to all types of OSS activities is strongly dependent on the perceived usefulness for the corporation and the perceived relative advantage of OSS versus proprietary software. Also prior exposure to OSS and perceived personal usefulness exhibit significant positive effects.

The remainder of this paper is organized as follows. Chapter two describes the peculiarities of OSS development compared to that of proprietary closed source software (PCSS). Chapter three provides the theoretical background on corporate adoption of OSS and OSS development, and derives hypotheses. In chapter four data and methods are presented. Results will be shown in chapter five. Finally, implications for theory and practice will be derived and limitations of the findings discussed in chapter six.

2 Background: Proprietary versus OSS Development

In PCSS development, the firm builds nearly exclusively on the knowledge of its own employees. As displayed in Figure 1, developers write the source code according to use cases specified by software architects, without any significant interaction with the outside. In the end, product testing ascertains that the final software adheres to the requirements set initially as well as to the company quality standards before the product is released (Jones 2003; Lehman 1980; Royce 1987; Senyard and Michlmayr 2004). Potential influence from the outside is restricted to customer requirements at the beginning of the development process (where “customers” may be internal to the firm), licensed-in commercial third-party software, and beta testing towards its end.

--- Insert Figure 1 about here ---

With OSS development, the boundaries between the firm and its environment become more permeable, as illustrated in Figure 2. Only the beginning and the end of the product development process are set by the company: on the one end, product specifications will

mainly still be defined within the company. On the other end, it is the company who decides when a version of the software will be released as a product.

--- Insert Figure 2 about here ---

The above is a simplification insofar as software development can embrace OSS and OSS practices to varying degrees. Thus, for the purpose of this paper we distinguish between three levels of corporate OSS activities: using existing OSS, contributing to OSS projects, and releasing proprietary software under an OSS license.⁴ The use of existing OSS is widespread both within and outside the IT industry. OSS has often become an integral part of companies' software architectures and in some cases even commercial offerings. Prominent examples of this are the operating system Linux, the web server Apache, or the programming environment Eclipse (Driver and Weiss 2005; Goulde 2006; Grand et al. 2004; Varian and Shapiro 2003). In these cases, OSS is basically treated just as any other third-party software, mostly without or with only limited modifications.⁵ Only one-way interaction between the company and the environment takes place, such that clear boundaries between the two exist.

Yet, there are instances in which corporations fix existing bugs or adapt the software to their needs, and contribute the modified source code back to the OSS project. The latter typically happens selectively (Henkel 2006b). The option of giving back conforms to the idea of reciprocity promoted by proponents of OSS, and of Free Software in particular (Bonaccorsi and Rossi 2006; Raymond 2001a; Shah 2006). However, it typically contradicts established corporate policies. According to them, innovation takes place within the firm's boundaries, and no intellectual property is to leave the corporation—if at all, then only under a licensing contract. It is thus a significant step, in particular for established firms, to move from just using OSS to contributing to OSS projects. Depending on the extent of such activities, the boundaries between the company and its environment have become somewhat blurred (Chesbrough 2003; West and Gallagher 2006).

⁴ Grand et al. (2004) use a related classification, distinguishing between the four levels of using existing OSS, adapting and extending OSS, acting as core OSS developers, and providing development and distribution services related to OSS projects (thus running an OSS compatible business model). The first levels in this and our classification are identical. The second and third level in this classification largely correspond to “contributing to existing OSS” in ours. The top levels differ since Grand et al. (2004) focus on “pure play” OSS firms, while our focus lies on established corporations.

⁵ Franke and von Hippel (2003), e.g., find in a sample of 131 Apache webmasters that only 19% of them had modified the source code, despite the fact that the respondents likely were expert users. In general, this share will be even lower.

Releasing own proprietary software under an OSS license constitutes a still bigger departure from established practice. Doing so in order to initiate co-development with the OSS community completes the transition from closed to open, collective, or private-collective innovation (Allen 1983; Chesbrough 2003; Osterloh and Rota 2005; von Hippel and von Krogh 2003). As the term co-development suggests, this is both a *process and a product* innovation: team organization, project management and coding have to be set up and executed differently than in the conventional mode of development (Grand et al. 2004; von Hippel and von Krogh 2003; West and Gallagher 2006). The result of the process is a new product. In this case, full bidirectional interaction between the company and its environment takes place. With respect to the project and the corresponding knowledge, the boundaries may even have disappeared.

3 Research Questions and Hypotheses

In this section, we first develop a theoretical framework describing the drivers of IT professionals' attitudes towards corporate OSS engagement. Using this framework, we then derive hypotheses regarding the effect of the respondent's job function in Section 3.2. The effects of perceived usefulness of OSS, of individual and of environmental factors, derived from Davis's TAM (Davis 1989; Davis et al. 1989) and Rogers' (2003) innovation diffusion model, are discussed in Sections 3.3 to 3.5. While these variables are rather interesting in their own right we refrain from formulating hypotheses regarding their effects, in order to keep a clear focus.

3.1 Theoretical Framework

Employed IT professionals can drive corporate OSS engagement in two ways, reflecting the grassroots nature OSS is attributed with (Henkel 2004; Moody 2001; Raymond 2001a). First, they can opt for OSS in decisions that are in their own discretion; second, they can lobby for OSS engagement more broadly. In both cases, the professional's individual attitude towards adopting OSS and the OSS development style matters. This is obviously so in the first case. In the second case, consider that broad OSS adoption by the corporation will, for many employed IT professionals, imply that they themselves become more strongly exposed to OSS. This means that, also when they lobby for OSS engagement by the *corporation*, their

attitudes towards OSS engagement by *themselves* matter. Consequently, we also look at the factors which influence employees' attitude towards engaging in OSS.

According to the Technology Acceptance Model (TAM) by Davis (Davis 1989; Davis et al. 1989) two main factors drive an individual's attitude towards a new a technology: perceived usefulness and perceived ease of use (Figure 3). Attitude towards using a new technology positively affects the behavioral intention to use it. One's intention to use a new technology then determines whether one becomes an actual user.

--- Insert Figure 3 about here ---

Especially perceived usefulness is highly influential on both attitude towards using and behavioral intention to use. Perceived ease of use has a more indirect effect on both of the latter factors through perceived usefulness: an individual perceives a new technology as more useful if it has been previously identified as easy to use (Davis 1989; Davis et al. 1989). Intrinsic motivation, ideology, and control have been shown to have a significant influence on ease of use (Davis et al. 1992; Venkatesh 2000).

Following Rogers, the diffusion of an innovation depends on the innovation itself, communication and communication channels, time, and social systems. Rogers' measures for the innovation dimension include relative advantage, compatibility, complexity, trialability, and observability; the most importance usually rests with the first two factors (Rogers 2003).

Combining the most important elements of both models will enable us to address the effects on diffusion that relate to the respondent's job function as well as those that do not.

3.2 Job Functions

Individuals with different job functions will, in general, have different attitudes towards the adoption of new information technology (Agarwal and Prasad 1999), and in particular towards engagement in OSS. Depending on how their daily routines are affected, individuals will have different levels of perceived usefulness and perceived ease of use.

For the purpose of this paper, we distinguish five different job functions in the software development process. Ordered (roughly) by the sequence of their activities, these are software architects, developers, testers, project managers, and (general) managers. In the following, we describe each of these roles in turn and discuss how they would be affected by an increased corporate OSS engagement.

Software architects translate user needs into a set of (high-level) software requirements and subdivide the system into subsystems that are worked on by developers (Bass et al. 2003; Royce 1987). Even in the development of PCSS they have outside interaction through receiving customer input, and through selecting and integrating third-party modules into the design of the software. Thus, the interaction with the outside world that OSS brings with it is already familiar to software architects, such that their perceived ease of use of OSS should be high. In addition, also their perceived usefulness should be high since, first, the availability of OSS broadens the design choices and, second, the suitability of OSS for a certain purpose can more easily be checked, due to availability of its source code, than that of PCSS. We thus expect software architects to have a rather positive attitude towards corporate OSS engagement.

Developers' job is to build the program based on the system design. For them, OSS entails various changes, which can be described by the key words of Not-Invented-Here, OSS development style, and coding for re-use. First, the share of external software that a developer works with will typically be much larger in OSS than in PCSS development, and re-using existing OSS instead of writing one's own code may be subject to the Not-Invented-Here (NIH) syndrome (DiBona 2005; Katz and Allen 1982). There is, however, also a potential positive aspect for developers since re-using OSS may simplify and speed up their work. Second, many developers will be new to OSS development and consequently unfamiliar with its development method, its design principles, and its meritocratic style (Scacchi 2004). Even if developers have used external components in PCSS development, they were usually not communicating with its originators nor modifying its source code. Adapting to these changes requires effort; on the other hand, being able to communicate with outside experts may help to solve problems. Third, the importance of modularity and coding for reusability are far greater in OSS than in PCSS development (Bonaccorsi and Rossi 2003; Goldman and Gabriel 2005;

Raymond 2001a).⁶ Adapting to the changes named above requires effort, reducing the developer's perceived ease of use of OSS. On the other hand, some aspects should increase perceived usefulness. Overall, hence, developers should have a less positive attitude towards OSS than architects.

Testers scan programs for bugs or logical flaws according to specifications set by software architects and programmers (Royce 1987). With OSS development, their situation should improve: on the one hand, proven OSS components contained in the software should have been scrutinized by the OSS community previously, thereby reducing the number of bugs (Raymond 2001a). With actual OSS development, the tester can now make use of the community for the company's software, thereby increasing the number of tests and test designs significantly. Already with PCSS development, external actors are used for so-called beta testing to catch some of these benefits. Here, selected users receive a copy of a release candidate of a software program for testing purposes. Hence, for testers (as for architects) OSS brings no fundamentally new activities, which should imply high perceived ease of use. On the other hand, some of the tester's tasks are simplified by corporate OSS engagement, entailing high perceived usefulness. We thus expect testers to be rather positive towards OSS.

Project managers run software projects, coordinate tasks and personnel, allocate resources within the project, and set milestones. In OSS development, project managers are now faced with an additional amount of uncertainty in the coding realm (Goldman and Gabriel 2005): how to set milestones and allocate resources in a project when one does not even know who is working on it? A second aspect takes effect towards the end of the project. According to our interviews, contributing source code to existing OSS projects or revealing proprietary software as OSS means additional work for project managers which does not yield immediate and clearly visible benefits to them and their project. Thus, for project managers both perceived usefulness and perceived ease of use of OSS should thus be rather low. Accordingly, we expect them to be rather negative towards OSS.

Managers define and enact corporate strategy. They decide on which projects to follow and allocate the respective resources (Burgelman 1983). While they should perceive the same

⁶ Some more modern software development methods—the spiral model, extreme programming, and agile methodologies—require, by their nature, stronger modularization and coding for re-usability than classical approaches such as the waterfall or V-model. The former are thus more “compatible” with OSS development (Feller and Fitzgerald 2001a, Goldman and Gabriel 2005, Hang et al. 2004).

downsides of OSS development as project managers (albeit less strongly since they are not directly responsible for keeping deadlines), managers need to think beyond projects. Long-term benefits resulting from OSS engagement should thus offset the negative short-term effects. Quite generally, the benefits of revealing proprietary software as OSS are strategic in nature, and will thus be perceived by managers (Dahlander 2005; Feller and Fitzgerald 2001; Grand et al. 2004; Hecker 1999; Raymond 2001a; Raymond 2001b; West 2003). We thus expect managers to be more positive than project managers towards OSS, but less positive than architects and testers.

Overall, we arrive at the following predictions regarding the impact of an individual's job function on his or her attitude towards OSS: architects and testers should be most favorable towards OSS, followed by developers and managers. The least favorable group should be project managers. We thus formulate the following hypotheses (the letters in square brackets indicating the groups to which the respective hypothesis refers):

- H1[TM]: Testers are more favorable towards OSS than managers.*
- H2[TD]: Testers are more favorable towards OSS than developers.*
- H3[TP]: Testers are more favorable towards OSS than project managers.*
- H4[AM]: Architects are more favorable towards OSS than managers.*
- H5[AD]: Architects are more favorable towards OSS than developers.*
- H6[AP]: Architects are more favorable towards OSS than project managers.*
- H7[MP]: Managers are more favorable towards OSS than project managers.*
- H8[DP]: Developers are more favorable towards OSS than project managers.*

In order to keep the presentation clear and not too burdensome, we refrain from deriving detailed hypotheses for each of the three levels of OSS engagement. We will comment on specific aspects of the different levels in discussing our results.

3.3 Advantages and Disadvantages of Corporate OSS Engagement

Advantages and disadvantages of corporate OSS engagement need to be evaluated both from a personal and a corporate point of view, as both the employee and her employer are affected by corporate engagement in OSS. This analysis needs to consider the perceived usefulness and the perceived ease of use of corporate OSS engagement, and the relative advantage OSS

might have over PCSS and OSS development over PCSS development. As the TAM focuses on the adoption of IT product innovations by individuals for personal reasons, we have chosen to modify the constructs “perceived usefulness” and “perceived ease of use” for *contributing to existing OSS projects* and *revealing proprietary software*, as both of these are process innovations that may have a significant effect on the corporation as a whole. While including peer influence, the TAM includes no measurement for the reasons of system adoption for the corporation, which again will be important for contributing to existing OSS projects and revealing proprietary software. The respective constructs⁷, which will be described in more details in the following paragraphs, were assembled by collecting items from existing studies on motives for corporate OSS engagement. For *using OSS* a single-item construct to measure perceived usefulness was employed.⁸

Usefulness of using and corporate advantages of contributing and revealing (CACR).

Increased engagement in OSS can have several advantages for the corporation. For *using OSS*, this can be savings on licensing fees or development effort, reduced lock-in, shorter time-to-market, and higher quality and performance (Goldman and Gabriel 2005; Hecker 1999; Raymond 2001a). *Contributing code to existing projects* may bring benefits to the company from both a technical and a marketing point of view. Technically, the company may influence the standard version of the software, thereby eliminating the need to redo the changes for each update of the OSS and take into account possible incompatibilities, new security issues, etc. Furthermore, this may lead to others making improvements to the company’s contribution. From a marketing point of view, contributing to the OSS world will increase the company’s reputation and visibility as a software developing company (Behlendorf 1999; Gruber and Henkel 2006; Hecker 1999; Henkel 2004; Koenig 2004; Raymond 2001a; Raymond 2001b; Seel 2006). *Revealing proprietary software as OSS* can yield the same advantages already attainable by contributing (Goldman and Gabriel 2005; Lakhani and von Hippel 2003; Shah 2006). Furthermore, new business models such as the sale of complementary goods or services might become viable, and commoditizing a certain layer of the software architecture may help to shift competition to an area in which the corporation has competitive strengths (Raymond 2001b; West 2003).

⁷ The four constructs are are potential advantages or benefits (perceived usefulness) and disadvantages or costs (perceived ease of use) of corporate OSS engagement– each from a personal and a corporate point of view.

⁸ No item for perceived ease of use was included, as much of its effect should have already been shifted to perceived usefulness due to previous OSS exposure (Davis, 1989; Davis et al., 1989).

Corporate disadvantages of contributing and revealing (CDCR) encompass the costs that need to be borne in order to be able to use the innovation. For OSS, this includes a potential loss of competitive advantage and additional resources that need to be allocated to support (Hecker 1999; Henkel 2004; Raymond 2001b).

Personal advantages of contributing and revealing (PACR). People should have a more positive view of engaging in OSS when they feel that this will improve their work performance (Davis 1989). Also the promise of external rewards may motivate individuals to contribute to OSS. By writing good source code, they could advertise themselves on the job market. Their ego might also play a role: by showing their superior programming solutions to the public they might hope for fame and peer recognition in the OSS community. Many individual contributors will also be looking for feedback on their source code, thereby improving their programming skills (Hars and Ou 2002; Lakhani and Wolf 2005; Lerner and Tirole 2002).

Personal disadvantages of contributing revealing (PDCR). People who associate corporate engagement in OSS with high personal cost should see less of a reason to do so. This includes the additional work that comes with programming OSS, such as designing for modularity or new tasks such as support (Bonaccorsi and Rossi 2003; Goldman and Gabriel 2005; Raymond 2001a), or the fear of negative feedback (Bénabou and Tirole 2003) .

Relative advantage. When directly compared to PCSS and PCSS development, OSS has often been claimed to be more aesthetic and OSS development to be more efficient (Bonaccorsi and Rossi 2003). While we will not debate this issue here, we think that individuals who agree to this point of view will also have a more positive view on the usefulness of OSS in general. Also, the possibility given by OSS to modify the source code should positively affect this view. Believing in these relative advantages of OSS versus proprietary software should then positively influence an individual's attitude towards engaging in OSS.

3.4 Individual Factors

Time. Increased exposure to OSS should increase the individual's level of understanding of the usefulness of the technology. Studies have shown that exposure to the technology is a mediator between perceived usefulness and ease of use: after having being exposed to the

technology, the direct effect of ease of use on attitude towards using a technology is reduced, but transferred to an indirect effect via perceived usefulness (Agarwal and Prasad 1999; Davis 1989; Davis et al. 1989). However, cause and effect are not clearly separable here: people who perceive a higher ease of use and consequently a higher perceived usefulness, i.e. innovators and early adopters, should have engaged in OSS earlier than their peers (Rogers 2003). Still, their perceived usefulness of OSS *now* should be significantly higher than of those people who have not yet engaged in OSS.

Similar to Rogers' adopter categories, Kirton proposes his Adoption-Innovation index (KAI) as a means to measure a person's innovativeness. Kirton identifies personal creative style or innovativeness as an important factor in coping with change. The lower a person scores, the more likely she is to be an adopter, i.e. a person that prefers no or at most very little change to the status quo (Kirton 1976; Kirton 2003). As, again, using OSS, contributing to existing OSS projects, and revealing proprietary software as OSS involves considerable change for all people involved, more innovative people should be more favorable towards OSS.

Compatibility. Compatibility in the context of innovation diffusion may but does not necessarily have to mean compatibility in a technical sense. More importantly, compatibility encompasses the coherence of the innovation with existing norms and premises (Rogers 2003). Compatibility thus has an important influence on the perceived ease of use of a new technology (Davis 1989; Venkatesh 2000). While identification with the goals of (part of) the OSS community such as freedom or reciprocity is not *the* main driving force behind most individual's OSS engagement (Hars and Ou 2002; Lakhani and Wolf 2005), it may still be so for some individuals. For them, these values represent unifying aspects as they provide a basic philosophy for the OSS movement (Stewart and Gosain 2006). Agreeing with this philosophy may positively influence the perceived ease of use of contributing to an OSS project and of "giving back" to the community by revealing formerly proprietary software as OSS (Venkatesh 2000).

3.5 Environmental Factors

Social systems. In addition to raising issues of compatibility, embeddedness in social systems will also lead to innovation diffusion being influenced by the individual's peer group (Granovetter 1973; Rogers 2003). Belonging to a group, people are likely to take actions

based on a frame of reference created by this group (Merton and Rossi 1949), a concept which has been validated empirically (see, e.g. Falk and Ichino 2006; Katz et al. 2001). Depending on her immediate environment, peer influence on a company employee may be exerted either in favor or against OSS.

Communication is probably the best way to bridge gaps of compatibility, and, in addition, can also be a valuable source of innovation (Chakrabarti and O'Keefe 1977; Ebadi and Utterback 1984; Katz and Allen 1985). In the context of interaction and knowledge exchange with people inside and outside the organization, peer influence is again highly important (Granovetter 1973; Katz and Allen 1982; Katz and Allen 1985; Laursen and Salter 2006; Rogers 2003).

4 Data and Methods

The study was conducted at the telecommunications department of a multinational electronics company, in which software development plays a key role. Thus, most to all employees get in touch with software development during their everyday work. During the time of the study, the department had no officially communicated strategy with respect to OSS. Rules on and initiatives surrounding the optimal use of OSS existed, but only a minority of employees were aware of those. No general policy existed on contributing to existing OSS projects or revealing proprietary software as OSS, yet both had actually happened in individual cases in the past.

4.1 Sample

As a first step, current involvement in OSS and current practices when dealing with OSS were analyzed in 25 interviews, with an average duration of 45 minutes. The interviews were conducted with company employees in different countries and on different hierarchical levels, and with experts from outside the company.

Building on these interviews, we launched a large-scale online survey for the department's IT employees in early 2006. Survey participants were asked to share their general opinion on OSS and to describe their current experience with and exposure to OSS, and how they thought of their peers with respect to OSS. Afterwards, participants could name current projects or

programs to be revealed as OSS as well as their reasons why this might be a good idea. Additionally, a measure for innovativeness was included.

The survey was distributed to several international sites of the corporation in both English and German. Pre-tests had confirmed validity of the survey and consistency of the translation. Addressees were invited to participate during a three-week time span. People at the different sites received invitation emails from their respective superiors, containing a text composed by the authors of the survey. This invitation contained a general user-password combination valid for all employees to protect the survey against unauthorized participation. Individual participants answered the questions anonymously. At the middle of the three-week time span, a reminder was sent out to encourage additional participation in the survey.

In total, approximately 1350 people in seven different countries were contacted. 404 valid replies were received, yielding a response rate of 30%. Between the sites, response rates varied from 80% to 24%. Due to legal restrictions, demographic information could not be collected in all countries, leading to a total of 249 replies usable for the current analysis. By job function, we received usable replies from 37 testers, 23 architects, 27 managers, 153 developers, and 9 project managers.

--- Insert Table 1 about here ---

4.2 *Dependent Variables*

Attitude towards engaging in OSS. In assessing an employee's attitude towards corporate OSS engagement, we distinguish three levels: using existing OSS, contributing to OSS projects, and revealing proprietary software as OSS.⁹ Respondents were asked to indicate their agreement, on a 5-point Likert scale ranging from 1 = "strongly disagree" to 5 = "strongly agree", to the following statements:

I think that [company department] could benefit from...

⁹ We argued in Section 3.1 that a respondent's attitude towards OSS engagement by *herself* should be closely related to her attitude towards *corporate* OSS engagement. This is confirmed empirically: Agreement to the statement "I would like to use more OSS in my job" is highly correlated with the agreement to "[Company] could benefit from using OSS more often" (Spearman rank correlation: 0.51), and agreement to "I would like to develop more OSS in my job" is highly correlated with the agreement to "[Company] could benefit from contributing modified OSS back to the public" (0.44) and "... from making some of its own proprietary software public under an OSS license" (0.52).

- ... *using existing OSS more often*
- ... *contributing modified OSS back to the public*
- ... *making some of its own proprietary software public under an OSS license*

Complementing the last of the above questions, participants were prompted to name up to three current products or projects of the company as candidates for a release as OSS, specify their level of familiarity with the product, and indicate why and how they thought that revealing would bring value to the company. Since respondents were aware that these suggestions were of real interest to the corporation, naming release candidates constitutes an act of lobbying for OSS engagement. Even if it is a small step, it goes beyond indicating one's attitude. Overall, 74 respondents (18.3%) made suggestions. Descriptive statistics of the dependent variables (see Table 3) will be discussed in Section 5.1.

4.3 Explanatory Variables

Most explanatory variables are indices made up of several items as derived from theory (see sections 3.3 to 3.5). A complete list is given in Tables A.1 to A.4 in the Appendix. Descriptive statistics and correlations between explanatory variables are provided in Tables A.5 to A.8 in the Appendix.

Job functions are coded by dummy variables. In doing so, we use testers as the reference group, for two reasons. First, this group is reasonably large (14.9% of all respondents, see Table 1), and second, it is most positive towards OSS both according to our theoretical considerations and to the descriptive statistics provided in Table 4.

Usefulness of using, CACR, CDCR, PACR, PDCR, and relative advantage. As explained in Section 3.3, we employed separate constructs to capture the pros and cons of the various types of OSS engagement. Usefulness of *using OSS* was measured by a single-item construct.¹⁰ Items measuring the perceived advantages and disadvantages of contributing to existing OSS projects and revealing proprietary software as OSS from both a personal and a corporate point of view were taken from several existing studies and slightly modified to ensure compatibility (Henkel 2006b; Lakhani and Wolf 2005). As for all following constructs (unless noted

¹⁰ The respective question in the survey asked for the participant's level of agreement on the statement "Using existing OSS makes my work easier and/or better". Detailed statistics are given in the Appendix.

otherwise), items were scored on Likert scales ranging from 1 = “strongly disagree” to 5 = “strongly agree.” The items measuring the perceived advantages and disadvantages of OSS load on five factors explaining usefulness and ease of use engaging in OSS from both a personal and corporate perspective, and relative advantage of OSS over PCSS. Cronbach’s α for the resulting five scales is good to satisfactory: personal advantages of contributing and revealing (PACR): 0.83; personal disadvantages of contributing and revealing (PDCR): 0.62; corporate advantages of contributing and revealing (CACR): 0.86; corporate disadvantages of contributing and revealing (CDCR): 0.66; relative advantage of OSS: 0.68.

Time. In order to measure the effects on perceived usefulness created by previous OSS exposure, we included a dummy variable (“DidOSS”) for whether a participant herself had worked on OSS code before.

As for innovativeness, the individual items of the Kirton-Adoption-Innovation index (KAI) are scored on a Likert scale ranging from 1 = “very adopter-like” to 5 = “very innovator-like.” The KAI items load on three factors. The first factor, originality, describes how well a person can deal with new ideas. The second one, efficiency, describes a person’s need for efficient processes and her desire to carry out her task in great detail. The last one, conformity, describes a person’s adherence to rules and authorities. The items loading on the efficiency and conformity scales have to be scored in reverse, so that the resulting indices actually correspond to inefficiency and non-conformity and higher scores in fact indicate higher innovativeness (Kirton 1976; Kirton 2003). Because the factor structure of the original 32-item scale has been disputed, the 13-item version of the KAI was selected (Foxall and Hackett 1992b; Taylor 1989a; Taylor 1989b).

Several studies have already proven the relevance of the KAI inventory for IT and IT adoption. Gallivan (2003) found out that people who scored higher on the originality factor showed higher overall job performance and people scoring higher on the (in-)efficiency scale had better communication skills. Foxall and Hackett (1992a) show that managers with higher levels of computer use as indicated by number of software applications used are more innovative with respect to the originality and conformity scales, but more adopter-like with respect to efficiency. Principal component analysis retains three factors with eigenvalues larger than 1, explaining 58% of total variance. After varimax rotation, all items load higher than 0.64 on the factors predicted by the KAI model, and lower than .21 on any other.

Cronbach's α for the factors originality, (in-)efficiency, and (non-)conformity are 0.82, 0.79, and 0.68, respectively.

Compatibility was measured with two items. First, identification with the OSS community was measured with a single-item construct ("ID_with_OSS").¹¹ Second, reciprocity was measured by the level of agreement to three statements reflecting the give-and-take philosophy of the OSS community. The items were taken from existing studies and slightly adjusted (Henkel 2006b; Lakhani and Wolf 2005). The three items measuring reciprocity load higher than 0.8 on a single factor, explaining 71% of total variance. Cronbach's α of the index is 0.79.

Environmental factors potentially influencing the respondent's attitude were measured using seven questions based on statements collected in our interviews. These items mainly capture how peers and supervisors of the respondent think about OSS, and how familiar they are with it. The items were spread out over the questionnaire in order to minimize social desirability bias. All items load higher than 0.5 on one factor with eigenvalue larger than one, explaining 44% of total variance. Cronbach's α of the index (EnviInfluence) is 0.77.

Further explanatory variables. Each participant's age was asked for; this variable is included in the regressions. In line with the findings of Agarwal and Prasad (1999) we also checked for the highest level of education attained, the country in which this degree was received, and the respective major. We also collected information at which site an individual was working. However, in no specification did any of the latter control variables show significance, so we only report specifications without these variables.

5 Results

5.1 Job Functions – Descriptive Analysis

Respondents' attitudes towards corporate OSS engagement have to be viewed in light of their OSS experience. Table 2 thus displays the means of various OSS-related characteristics.¹² Taking both use and development of OSS into account, we find that architects, developers,

¹¹ The respective question in the survey asked for the participant's level of agreement on the statement "I identify with the OSS community". Detailed statistics are given in the Appendix.

¹² For comparison, also variables describing the respondent's general programming activity are displayed.

and testers qualify as most experienced in OSS.¹³ In particular, the share of respondents having worked on OSS code is largest among developers. These facts are important for the following analysis. As we will show below, developers are significantly less positive towards corporate OSS engagement than testers and architects. These facts also allow to rule out an explanation stipulating a simple positive relationship between a person's OSS experience and her attitude towards corporate OSS engagement.

Respondents' willingness to use or develop more OSS themselves mirrors developers' not-so-enthusiastic stance towards OSS engagement by the corporation. They, together with architects, are clearly lagging behind testers in their agreement to "I would like to use/develop more OSS at [Company]" (last two lines of Table 2). Developers in particular only reach an average score slightly below "I somewhat agree."

--- Insert Table 2 about here ---

We now turn to our main question. As Table 3 shows, respondents on average show a "somewhat" positive attitude towards increased corporate OSS engagement. However, a distinction by the *type* of OSS engagement is required. While the mean value on a scale from 1 ("strongly disagree") to 5 ("strongly agree") is 4.25 for *using OSS*, it goes down to 3.90 for *contributing to OSS projects* and to 3.53 for *revealing proprietary software as OSS*. The share of respondents ticking "strongly agree" or "somewhat agree" reflects this finding, decreasing from 85.1% (*using*) over 69.9% (*contributing*) to 56.2% (*revealing*).

Regarding the influence of respondents' job function, both the descriptive analysis presented here and the multivariate analysis in the following Section 5.2 matter. On the one hand, we need to know how testers, architects, etc. as *groups* behave and think about OSS, which is analyzed using univariate analysis and comparison of medians. On the other hand, we want to isolate the effect of the *job function*, net of other characteristics of the respondent that may be correlated with it.

-- Insert Table 3, Table 4 about here ---

¹³ The most experienced OSS users are software architects, both in terms of the number of applications used and the years of working on OSS code. They are followed by developers and, on rank three, managers. As to the share of respondents who have worked on OSS code, developers are clearly leading (48%), before architects (39%) and projects managers (33%). In terms of hours spent per week writing and testing OSS code (at work and at home), testers (5.9h/w) are leading before architects and developers (each 4.9h/w).

As predicted in Section 3.2, we find (see Table 4) that the level of support for OSS engagement is highest for the group consisting of testers and architects, followed by the group made up of managers and developers. Project managers are least positive towards OSS. This finding is consistent across all three levels of OSS engagement, including the number of suggestions made for projects to be released as OSS. In more detail, the five job functions can be ranked by their attitude towards OSS as follows: testers, architects, managers, developers, and project managers. Again, this ranking holds, with only two exceptions¹⁴, for all three types of OSS engagement as well as for the number of suggestions made.

Given that only for developers the number of respondents is really large (153), it comes as no surprise that a Mann-Whitney test on the equality of medians fails to reject the Null hypothesis in a number of cases. Table 5 shows the results of this test. Note that we conservatively report tests of *undirected* hypotheses, even though our hypotheses are in fact directed. For testing a directed hypothesis, the displayed significance level (p) is to be reduced by a factor of 0.5.

--- Insert Table 5 about here ---

Using OSS receives quite similar (and high) levels of agreement from all job functions, such that only H2[TD], H3[TP] and, using a directed hypothesis, H6[AP] receive support. For *contributing to public OSS projects*, we again find significant differences between testers and developers (H2[TD]) and testers and project managers (H3[TP]). In addition, H6[AP] and, using a directed hypothesis, H5[AD] receive support. The largest and most significant differences between job functions exist with respect to their attitude towards *revealing proprietary software as OSS*. We find support for H1[TM], H2[TD], H3[TP], H4[AM], H6[AP] and, using a directed hypothesis, for H8[DP]. Finally, the number of suggestions made for OSS-projects launched by the corporation is significantly different between testers and developers, architects and developers, and architects and project managers, supporting H2[TD], H5[AD], and H6[AP]. Using a directed hypothesis, also H4[AM] receives support. We thus find quite a number of our hypotheses supported. In particular, the difference

¹⁴ The exceptions are that developers are more positive than managers with respect to revealing software as OSS, and that architects made more suggestions than testers.

between developers and the two “leading” groups of testers and architects is significant five times (resp. six times, using a directed hypothesis) out of eight.

5.2 Job Functions – Multivariate Analysis

The results presented above inform us about attitudes of the five groups we consider. However, they might be due not so much to the respondent’s job function but rather to other characteristics which, for whatever reasons, are correlated with the task a person fulfills. In order to separate these intertwined effects, we now turn to multivariate analysis. Table 6 shows the result of ordered probit regressions, using the level of agreement to the three statements specified in Section 4.2 plus the binary variable if a person has made a suggestion for OSS release candidates as dependent variables. The first four lines show the estimation coefficients of the dummy variables coding the respondents’ job functions, with testers as the reference group (that is, their coefficient is implicitly set to zero). Significant coefficients thus indicate differences between the attitudes of testers and the respective other group. In order to compare the displayed coefficients also among each other, we ran post-estimation analyses, yielding the results shown in Table 7.

--- Insert Table 6 and Table 7 about here ---

The first column of Table 6, referring to attitudes towards *using OSS*, shows no significant differences between the groups. This means that the (few) differences which were found significant in Table 7 are accounted for by other characteristics of the respondent than her job function. When it comes to *contributing to public OSS projects*, we do find significant differences between the job functions of tester/developer and architect/developer. This is consistent with the univariate analysis presented above, which also lent support to H2[TD] and H5[AD]. In addition, the coefficient describing the difference to testers is for developers larger than for all other groups. Hence, performing the job function of developer has, *ceteris paribus* (i.e., after correcting for characteristics potentially correlated with it) an even more negative effect on a person’s attitude towards contributing than in the univariate analysis.

Also for *revealing software as OSS* we find significant differences between testers on the one hand, and managers, developers, and project managers on the other hand, supporting H1[TM], H2[TD], and H3[TD]. Finally, being a developer leads, *ceteris paribus*, to a significantly

lower likelihood that this person makes *suggestions for revealing* proprietary software than being an architect or a tester (directed hypothesis), supporting H2[TD] and H5[AD].

We thus find that the differences between job functions in attitudes towards corporate OSS engagement can only partly be explained by other characteristics of the respondents. In particular, in four (five, with a directed hypothesis) out of eight pairwise comparisons the job function of developer implies a significantly lower support for corporate OSS engagement than the job functions of architects or tester.¹⁵

Since a full discussion of all control variables is beyond the scope of this paper, we briefly comment only on a few salient points. To summarize, all significant coefficients carry the expected sign. The perceived usefulness of using OSS has a highly significant and large effect on the respondent's attitude towards corporate use of OSS. Also, perceived personal advantages of contributing and revealing exhibit a positive effect on the attitude towards revealing, and perceived corporate advantages of contributing and revealing exert a highly significant effect on the probability that a person makes a suggestion. All other coefficients related to the TAM factors perceived usefulness and ease of use, however, are insignificant. This finding is likely due to the fact that the pros and cons of OSS a person perceives are strongly correlated with the relative advantage of OSS she perceives, her opinion on reciprocity, and her identification with OSS (see Table A.7), all of which turn out significant.

Regarding the remaining variables, a respondent's experience with OSS ("DidOSS") has a significant positive influence on her attitude towards using OSS, and a highly significant effect on her attitude towards contributing. The relative advantage of OSS she perceives ("RelativeAdvOfOSS") has a significant to highly significant effect on all dependent variables. The same is true also for a person's identification with OSS ("ID_with_OSS"), with the exception of "Suggestions made." A respondent's endorsement of "Reciprocity" has the expected positive effect on her attitude towards contributing to OSS projects. Somewhat surprisingly, the constructs derived from Kirton's Adoption Innovation Index (KAI) matter little; even joint insignificance can not be rejected for the first three regressions. The sole

¹⁵ Given that developers are one of the two most OSS-experienced groups, one might also interpret our finding as an indication that corporate OSS engagement *is* in fact less beneficial for the firm than testers (the most enthusiastic group) think. However, architects are both highly experienced and significantly more positive towards OSS than developers. This fact lends support to our interpretation that the way in which OSS affects a person's work routine matters.

exception constitutes “KAI(non)-Conformity” having a positive effect, significant on the 5% level, on the number of suggestions made.

As a robustness check, Table A.8 displays specifications in which the TAM-related variables have been omitted, since one might object that they are too close to the dependent variables (see correlations in Table A.7). Doing so yields larger and more highly significant effects of “RelativeAdvOfOSS”, “Reciprocity”, and “ID_with_OSS.” Also the differences between job functions become more pronounced and more highly significant. The overall picture, however, remains unchanged.

6 Discussion and implications

6.1 Job Functions and OSS adoption

Corporate OSS engagement is viewed “somewhat” positively by most people in the company we have studied. However, in order to understand how OSS fits into corporate software development processes, a much more differentiated view is required. Apart from distinguishing between different levels of OSS engagement, one needs to take the respondent’s job function into account.

Regarding the type or intensity of OSS engagement we find that *using existing OSS more often* is quite generally seen as potentially beneficial to the company—85% agreed “somewhat” or “strongly” to this statement. For more intense types of OSS engagement, agreement decreases: *contributing to OSS projects* is seen as advantageous by 70% of respondents, *revealing proprietary software as OSS* by only 56%.

Our main contribution concerns the interplay between a person’s job function and her support for corporate OSS engagement. As hypothesized, testers are most favorable towards increased OSS engagement, followed by software architects and, for two out of three types of OSS engagement, managers. Developers are still on average positive towards OSS, but much less so than anecdotal evidence suggests. Project managers are least favorable towards OSS. These differences are relatively small for *using existing OSS*, larger for *contributing to OSS projects*, and most pronounced for *revealing proprietary software as OSS*.

The above findings are in line with our hypotheses, which were based on an analysis of the advantages and disadvantages of OSS engagement for each job function. Attitudes towards OSS are influenced by the fact that the engagement in OSS projects and the revealing of proprietary software represent, to varying degrees for different job functions, a significant deviation from business-as-usual. Especially for developers, OSS seems to come close to what Lyytinen and Rose (2003) call a “disruptive IT innovation”.¹⁶ Developers, on average, can thus not be expected to react enthusiastically to an increased commitment to OSS and OSS development because of the inherent organizational change.¹⁷ The more OSS development will differ from the current development model and the less skilled an individual developer considers herself to be¹⁸, the less supportive the person will be.

6.2 Limitations and Possible Extensions

Some limitations of our study need to be mentioned. While the number of survey participants is high enough to provide statistical validity, external validity of our study is not ensured since it has been conducted at a single firm. In other corporations, the level of support for OSS engagement might be higher or lower depending on corporate culture, the firm’s exposure to OSS, its industry (Klevatorick et al. 1995) or its home country. However, the *differences* in attitudes towards OSS between testers, developers, etc. result from the professional activities these groups perform, which should be largely independent of firm or location. Hence, we are quite confident that our findings concerning the impact of job functions hold more generally.

As to country-specific effects, our respondents were located in seven different countries. We did check for national idiosyncrasies without, however, finding any significant differences between countries. Still, conducting a similar study in different countries, in one or more other firms, and possibly in different industries would provide valuable insights.

¹⁶ This study also confirms the findings of Sherif et al. (2006) who show that similar conflicts arise for developers in software reuse. Sherif et al. suggest managerial intervention as a solution to this problem.

¹⁷ This finding must be seen in light of the given corporate environment, where OSS does not play a central role. Henkel (2006a), in contrast, studies firms active in, or even dedicated to, the development of embedded Linux, many of which are rather small and young. In his sample of 197 commercial OSS developers, 49.7% of respondents stated that they either make suggestions to their supervisor as to what code could be revealed, or that this decision is even within their own discretion.

¹⁸ Our interviewees consistently indicated that the share of developers that would have the necessary skills (programming, social, and management) to work in a corporate OSS project was around 25%.

Finally, the firm's software development method might have influenced our results. At the time of our survey, the firm in this study was still mainly using the waterfall model. Agile software development had been introduced in early 2005, but was not yet widely used. Consequently, with OSS, developers of this firm are confronted with an entirely new model of software development. Developers in firms that have experience in extreme programming or agile methodologies, or that are using the spiral model should thus be more positive towards OSS development.

6.3 Recommendations for Practice

Introducing OSS and OSS methods implies changes in particular for developers and project managers—and precisely these groups turned out to be least favorable towards it. Possible steps towards solving this dilemma include training and a step-by-step introduction of corporate OSS engagement. Training programs should be established for new hires and advanced training programs for developers, managers, IP and legal staff. One could also think about brown-bag seminars for developers, putting a copy of the open source policy in employee handbooks, and online seminars or trainings (Fan et al. 2004).

A first step, one which has also been suggested by many of our survey and interview participants, could be the introduction of a “corporate source program” (Dinkelacker et al. 2001). In corporate source initiatives, companies mimic the OSS development style but limit it to happen within corporate boundaries. This might be a good start to make people familiar with the OSS development style while minimizing risks with respect to loss of intellectual property and sociological issues such as the not-invented-here syndrome.

Finally, individuals should be given the opportunity to self-select into OSS-related activities. Even among project managers, the most skeptical group towards OSS, one third of our respondents considered an OSS engagement on all three levels as “somewhat” beneficial for the corporation. Hence, it should well be possible to staff pilot OSS projects, in all job functions that are required, with OSS supporters—provided the staffing is done diligently.

The issue of OSS engagement is relevant for all firms active in software development. The question appears not so much *if* they should engage in OSS, but rather when, how, and to what extent. In answering these questions, the impact of OSS on the firm's software

development processes and on the individuals involved needs to be taken carefully into account. Our study is an attempt to shed light on these issues, and to enable a broader and more informed use of OSS and the OSS development style.

Figures & Tables

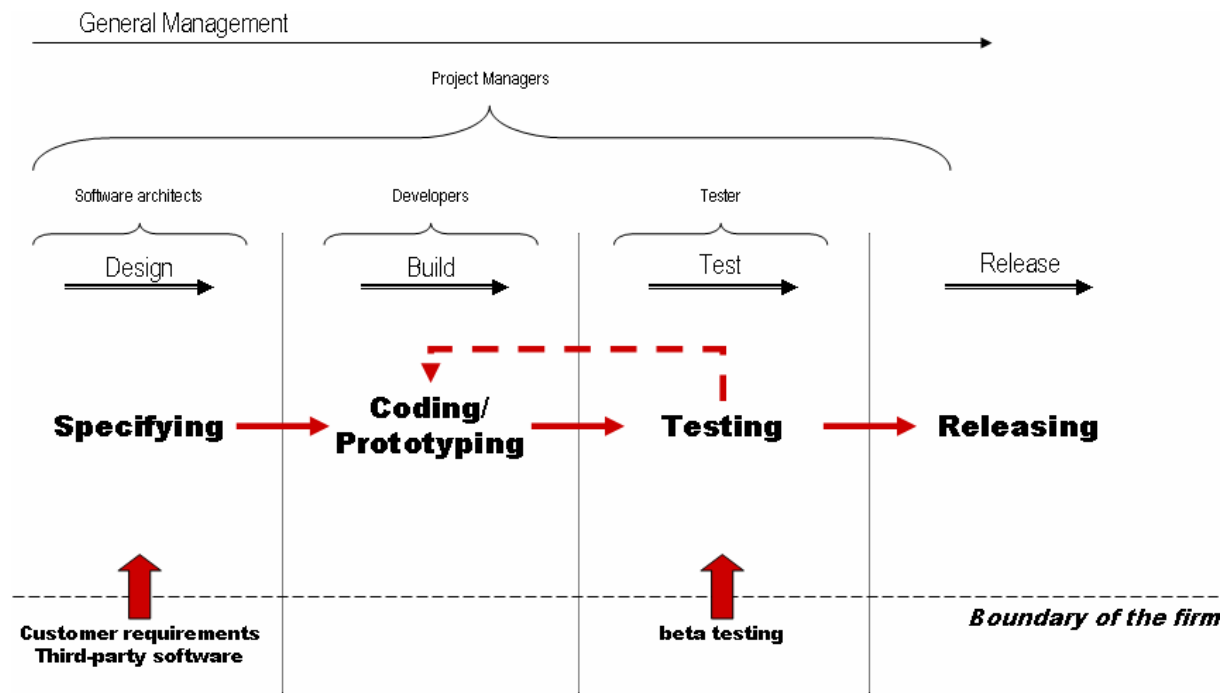


Figure 1: Closed Source Software (CSS) Development

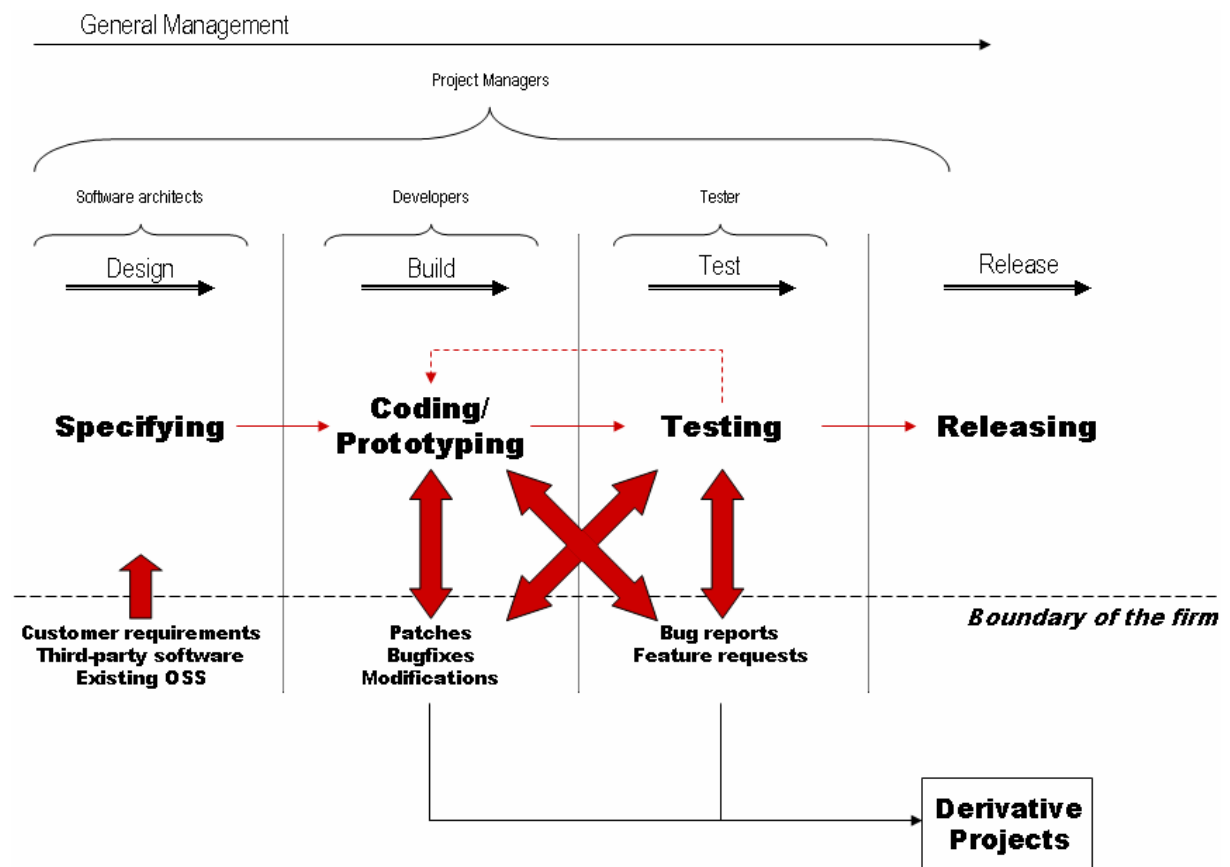


Figure 2: Open Source Software (OSS) Development

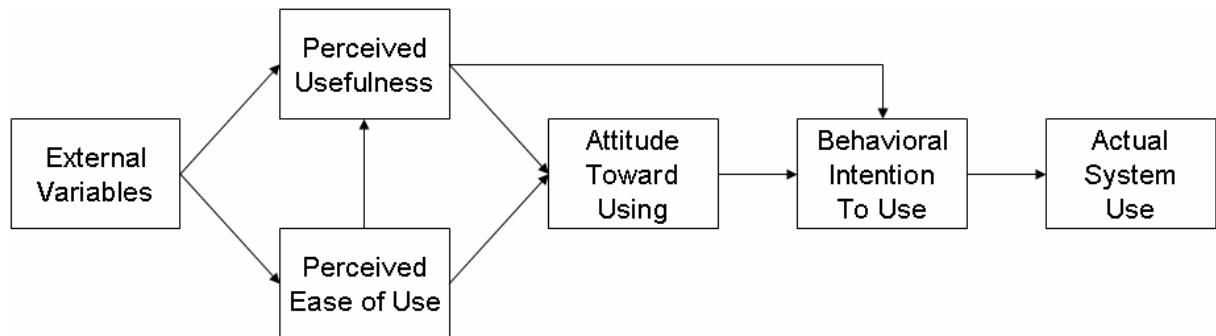


Figure 3: Technology Acceptance Model

Job Function	Frequency	Percent	Cumulative
Tester	37	14.86	14.86
Architect	23	9.24	24.10
Manager	27	10.84	34.94
Developer	153	61.45	96.93
Project Manager	9	3.61	100.00
Total	249		

Table 1: Survey participants by job function

Job Function	Tester	Architect	Manager	Developer	Project Manager
N	37	23	27	153	9
Number of OSS Applications used (in total)	2.486	3.652	2.741	3.046	2.444
Number of OSS Applications used (out of 6 suggestions)	2.297	3.478	2.556	2.627	2.444
Years working on OSS source code	1.541	2.391	1.815	2.373	0.889
Has worked on OSS code (1: Yes, 0:No)	0.297	0.391	0.259	0.484	0.333
Hours per week spent on programming at work (incl. testing, documentation)	16.784	7.043	5.852	27.193	13.389
Hours per week spent on programming at home	2.514	0.696	0.593	2.601	5.611
Hours per week spent on programming OSS at work (incl. testing, documentation)	5.270	4.130	3.296	4.412	2.222
Hours per week spent on programming OSS at home	0.622	0.783	0.074	0.471	0.000
I would like to use more OSS at [FIRM]	4.344	3.909	4.043	3.979	3.333
I would like to develop more OSS at [FIRM]	4.069	3.864	3.632	3.806	2.833

Table 2: Means of OSS- and programming related characteristics, by Job Function
Bold: largest and second largest value in each line.

Variable	N	Median	Mean	Std. Dev.	Min	Max
Corporation could benefit from using OSS	249	4	4.253	0.854	1	5
Corporation could benefit from contributing to existing OSS projects	249	4	3.9	0.981	1	5
Corporation could benefit from revealing proprietary software as OSS	249	4	3.53	1.136	1	5
Suggested projects to be revealed as OSS (Yes:1, No:0)	249	0	0.209		0	1

Table 3: Descriptive statistics of dependent variables

Variable	Tester	Architect	Manager	Developer	Project Manager
Corporation could benefit from using OSS	4.459 (0.73)	4.304 (0.822)	4.296 (0.953)	4.203 (0.884)	4 (0.5)
Corporation could benefit from contributing to existing OSS projects	4.216 (0.75)	4.174 (0.834)	3.889 (1.013)	3.804 (1.045)	3.556 (0.527)
Corporation could benefit from revealing proprietary software as OSS	4.135 (0.918)	3.739 (0.915)	3.148 (1.292)	3.458 (1.135)	2.889 (1.054)
Suggested projects to be revealed as OSS (Yes:1, No:0)	0.270	0.478	0.259	0.150	0.111

Table 4: Mean values of the dependent variables by job function. Bold: largest and second largest value in each line. Standard deviation in parenthesis.

<i>Using</i>	Tester	Architect	Manager	Developer	Project Manager
Mean	4.459	4.304	4.296	4.203	4
Median	5	4	5	4	4
Tester	---				
Architect	0.454	---			
Manager	0.663	0.805	---		
Developer	0.098*	0.625	0.419	---	
Project Mgr	0.028**	0.145	0.123	0.205	---
<i>Contributing</i>	Tester	Architect	Manager	Developer	Project Manager
Mean	4.216	4.174	3.889	3.804	3.556
Median	4	4	4	4	4
Tester	---				
Architect	0.921	---			
Manager	0.222	0.338	---		
Developer	0.036**	0.121	0.739	---	
Project Mgr	0.011**	0.029**	0.249	0.271	---
<i>Revealing</i>	Tester	Architect	Manager	Developer	Project Manager
Mean	4.135	3.739	3.148	3.458	2.889
Median	4	4	3	4	5
Tester	---				
Architect	0.078*	---			
Manager	0.002***	0.098*	---		
Developer	0.001***	0.344	0.223	---	
Project Mgr	0.002***	0.051*	0.613	0.132	---
<i>Suggestions made (1: Yes, 0: No)</i>	Tester	Architect	Manager	Developer	Project Manager
Mean	.270	.478	.259	.150	.111
Median	0	0	0	0	0
Tester	---				
Architect	0.103	---			
Manager	0.922	0.111	---		
Developer	0.085*	0.000***	0.163	---	
Project Mgr	0.321	0.058*	0.361	0.748	---

Table 5: Mann-Whitney Test on differences in medians (p)
* significant at 10%; ** significant at 5%; *** significant at 1%.

	Corporation could benefit from... (1-5 scale, ordered probit)			Suggestions made (Probit)
	... using	... contrib.	... revealing	
Job_Architect	-0.329 (0.369)	0.067 (0.282)	-0.571 (0.358)	0.349 (0.400)
Job_Manager	-0.020 (0.353)	-0.098 (0.282)	-0.874** (0.347)	-0.012 (0.373)
Job_Developer	-0.357 (0.253)	-0.438** (0.212)	-0.661*** (0.234)	-0.413 (0.285)
Job_ProjectMgr	-0.275 (0.375)	-0.364 (0.349)	-0.733* (0.445)	-0.175 (0.722)
Usefulness of Using	0.541*** (0.117)			
PACR (Pers. Adv. of Revealing)		0.029 (0.110)	0.254** (0.124)	0.162 (0.150)
PDCR (Pers. Disadv. of Revealing)		0.155 (0.119)	0.068 (0.116)	0.085 (0.153)
CACR (Corp. Adv. of Revealing)		0.244 (0.157)	0.204 (0.155)	0.627*** (0.209)
CDCR (Corp Disadv. of Revealing)		0.094 (0.133)	0.116 (0.121)	-0.056 (0.155)
RelativeAdvOfOSS	0.389*** (0.120)	0.212* (0.118)	0.328*** (0.127)	0.259* (0.154)
DidOSS	0.301* (0.162)	0.401*** (0.149)	0.112 (0.140)	-0.084 (0.208)
KAIOriginality	0.171 (0.145)	0.063 (0.128)	0.172 (0.124)	0.242 (0.176)
KAIEfficiency	-0.063 (0.142)	-0.112 (0.128)	-0.135 (0.114)	-0.043 (0.175)
KAIConformity	-0.186 (0.150)	-0.075 (0.136)	0.128 (0.128)	0.453** (0.190)
ID_with_OSS	0.282*** (0.101)	0.176* (0.099)	0.183* (0.094)	0.044 (0.146)
Reciprocity		0.269** (0.124)	0.128 (0.135)	0.090 (0.182)
EnviInfluence	-0.069 (0.140)	0.021 (0.119)	0.085 (0.121)	0.093 (0.156)
Age	0.003 (0.009)	-0.015* (0.008)	-0.011 (0.009)	0.020* (0.011)
Constant				-8.197*** (1.580)

Observations	249	249	249	249
Pseudo R-squared	0.22	0.16	0.17	0.22
Pseudo Likelihood	-216.890	-276.778	-305.292	-99.881
Wald's chi-squared	111.151	91.984	135.548	45.327
Degrees of freedom	13	17	17	17

Robust standard errors in parentheses

* significant at 10%; ** significant at 5%; *** significant at 1%

Table 6: Regression (including TAM variables)

<i>Using</i>	Testers	Architects	Managers	Developers	Project Managers
Coefficient Value	0	-0.320	-0.020	-0.357	-0.275
Testers	---				
Architects	0.471	---			
Managers	0.761	0.265	---		
Developers	0.256	0.933	0.367	---	
Project managers	0.561	0.936	0.367	0.864	---
<i>Contributing</i>	Testers	Architects	Managers	Developers	Project Managers
Coefficient Value	0	0.067	-0.098	-0.438**	-0.364
Testers	---				
Architects	0.812	---			
Managers	0.728	0.567	---		
Developers	0.039**	0.037**	0.133	---	
Project managers	0.297	0.227	0.452	0.807	---
<i>Revealing</i>	Testers	Architects	Managers	Developers	Project Managers
Coefficient Value	0	-0.571	-0.874**	-0.661***	-0.733*
Testers	---				
Architects	0.111	---			
Managers	0.012**	0.38	---		
Developers	0.005***	0.742	0.404	---	
Project managers	0.099*	0.729	0.758	0.858	---
<i>Suggestions made (1: Yes, 0: No)</i>	Testers	Architects	Managers	Developers	Project Managers
Coefficient Value	0	0.349	-0.012	-0.413	-0.175
Testers	---				
Architects	0.383	---			
Managers	0.974	0.357	---		
Developers	0.147	0.029**	0.201	---	
Project managers	0.808	0.502	0.827	0.728	---

Table 7: Ordered Probit and Probit Post-estimation: Test of Equality of Coefficients (p-values). * significant at 10%; ** significant at 5%; * significant at 1%**

Appendix

Personal Advantages of Contributing and Revealing		
	Factor pers. Advantage	Factor pers. Disadvantage
I would reveal code because ...		
... I get feedback on my code, which improves my personal skills	0.827	0.192
... I get feedback on my code, which improves my performance at my current job	0.826	0.15
... I get recognition in the open source community for my contributions	0.836	-0.04
... it demonstrates my skills	0.779	-0.168
Personal Disadvantages of Contributing and Revealing		
	Factor pers. Advantage	Factor pers. Disadvantage
What arguments do you see that would speak against revealing code?		
I am making myself obsolete by giving away my knowledge openly	0.112	0.613
It would be embarrassing if others found errors in my source code	0.051	0.681
Designing and writing a program so that others could re-use it requires too much work	0.03	0.722
Revealing code creates new tasks (such as support) that need too much time	0.026	0.683
Corporate Advantages of Contributing and Revealing		
	Factor corp. Advantage	Factor corp. Disadvantage
I think [FIRM] could benefit from revealing because ...		
... others develop the code further and reveal their developments in turn	0.654	0.144
... other developers make bugfixes and reveal them	0.71	0.05
... our own development should fast become the standard (i.e. be widely adopted)	0.681	0.007
... we want to appear as a good player in the open source community	0.623	0.111
... this way, our products stay compatible to other products	0.664	0.109
... it reduces our maintenance effort when the code becomes part of the standard distribution (e.g. Tomcat)	0.669	0.158
... revealing good code improves our company's technical reputation	0.632	0.135
... we often do not have sufficient resources to make developments on our own	0.395	0.119
... we could identify potential employees by looking at suggestions on our code	0.586	0.165
... others add functionality that we did not anticipate	0.606	0.044

Table A.1: Questions Underlying Factor Constructs (Part One)

Corporate Disadvantages of Contributing and Revealing

	Factor corp. Advantage	Factor corp. Disadvantage
How important do you think are for [FIRM] the following disadvantages of making code public as OSS?		
Competing companies use the code or learn from it, so there is a loss of competitive advantage	0.061	0.61
Our customers see less of a reason to pay for our products when the code is freely available	0.132	0.591
Revealed code requires some support, which is costly and time-consuming	0.193	0.543
We could be held liable for software failures	0.154	0.397
Patent infringements (if they occur) become more visible	0.065	0.427

Table A.2: Questions Underlying Factor Constructs (Part Two)

Relative Advantage of OSS	Factor Loadings
The source code of OSS is usually more aesthetic than that of proprietary software	0.773
Users of software should have the right to see the source code	0.774
Open Source development is the most efficient way to develop software	0.809
Reciprocity	Factor Loadings
[FIRM] has an obligation of giving back to the OSS community	0.807
I would reveal code because I consider it fair to give back to the community, since the company benefits from it	0.882
I would reveal code because in the long run, you only get something when you gave something before	0.823
Popularity of OSS amongst Co-Workers (Environmental Factors)	Factor Loadings
Management promotes the use of existing OSS	0.7
Which of the following factors do you consider supportive or impeding to the wider use of OSS within [FIRM]? My supervisor	0.727
Which of the following factors do you consider supportive or impeding to the wider use of OSS within [FIRM]? My colleagues	0.601
My supervisor is familiar with OSS	0.701
Most programmers at [FIRM] are familiar with OSS	0.638
In case I had questions on OSS, I would know someone at [FIRM] I could turn to	0.577
Management sees the benefit of OSS	0.64

Table A.3: Questions Underlying Factor Constructs (Part Three)

KAI Originality

Would you consider yourself someone who...	Originality	Efficiency	Conformity
... has fresh perspectives on old problems	0.689	-0.216	-0.045
... copes with several new ideas at the same time	0.679	-0.116	-0.002
... is stimulating	0.802	-0.093	-0.016
... has original ideas	0.802	-0.1	0.075
... proliferates ideas	0.77	-0.07	-0.09

KAI Efficiency

Would you consider yourself someone who...	Originality	Efficiency	Conformity
... enjoys detailed work	-0.085	0.706	0.214
... is thorough	-0.117	0.786	0.093
... masters all details painstakingly	-0.159	0.766	0.163
... is methodical and systematic	-0.119	0.745	0.113

KAI Conformity

Would you consider yourself someone who...	Originality	Efficiency	Conformity
... conforms	0.152	0.184	0.636
... is prudent when dealing with authority	-0.092	0.044	0.734
... never acts without proper authority	-0.024	0.206	0.662
... fits readily into "the system"	-0.025	0.221	0.75

Table A.4: Questions Underlying Factor Constructs (Part Four)

Variable	Tester	Architect	Manager	Developer	Project Manager	Overall
Usefulness of Using	3.944 (0.788)	3.918 (1.05)	3.763 (0.985)	3.918 (0.905)	3.573 (1.214)	3.892 (0.92)
PACR	3.676 (0.758)	3.609 (0.723)	3.388 (1.009)	3.49 (0.863)	2.909 (0.882)	3.496 (0.859)
PDCR	3.583 (0.808)	3.49 (0.664)	3.346 (0.641)	3.592 (0.798)	3.583 (0.375)	3.554 (0.76)
CACR	3.915 (0.621)	3.831 (0.549)	3.58 (0.777)	3.694 (0.664)	3.46 (0.529)	3.719 (0.661)
CDCR	2.77 (0.673)	2.727 (0.678)	2.626 (0.727)	2.682 (0.782)	2.436 (0.652)	2.685 (0.745)
RelativeAdvOfOSS	3.571 (0.656)	3.467 (0.737)	3.191 (0.728)	3.311 (0.836)	2.983 (0.816)	3.339 (0.796)
DidOSS	0.297	0.391	0.259	0.484	0.333	0.418
KAIOriginality	3.745 (0.633)	4.239 (0.558)	4.03 (0.604)	3.767 (0.606)	3.754 (0.615)	3.836 (0.62)
KAIEfficiency	2.17 (0.655)	2.205 (0.757)	1.926 (0.857)	1.986 (0.625)	1.944 (0.541)	2.026 (0.669)
KAIConformity	2.336 (0.597)	2.482 (0.812)	2.254 (0.642)	2.346 (0.66)	2.3 (0.465)	2.345 (0.656)
ID_with_OSS	3.403 (0.856)	3.683 (0.819)	3.316 (0.952)	3.248 (1.085)	2.444 (0.726)	3.289 (1.021)
Reciprocity	3.739 (0.54)	3.815 (0.638)	3.777 (0.597)	3.738 (0.85)	3.556 (0.882)	3.743 (0.765)
EnviInfluence	3.24 (0.704)	3.676 (0.562)	3.416 (0.721)	3.29 (0.691)	3.285 (0.697)	3.332 (0.69)
Age	35.757 (8.163)	44.783 (8.49)	45.556 (8.469)	38.621 (10.316)	44 (7.159)	39.711 (10.015)

Table A.5: Mean values of independent variables by job function. Standard deviation in parenthesis.

Variable	Tester	Architect	Manager	Developer	Project Manager	Overall
Usefulness of Using	4	4	3.809	4	3.795	4
PACR	3.75	3.75	3.5	3.75	2.75	3.5
PDCR	3.5	3.5	3.308	3.5	3.75	3.5
CACR	3.994	3.8	3.6	3.8	3.5	3.8
CDCR	2.683	2.7	2.8	2.7	2.4	2.6
RelativeAdvOfOSS	3.667	3.333	3.333	3.333	3	3.333
DidOSS	4	4	3.809	4	3.795	4
KAIOriginality	3.8	4.4	4	4.4	3.6	3.8
KAIEfficiency	2.25	2	2	2	2	2
KAIConformity	2.25	2.5	2	2.5	2.5	2.25
ID with OSS	3	4	3	4	3	3
Reciprocity	3.667	3.977	4	3.977	3.667	3.874
EnviInfluence	3.286	3.714	3.429	3.714	3.286	3.429
Age	3.75	3.75	3.5	3.75	2.75	3.5

Table A.6: Medians of independent variables by job function

	Benefit of Using	Benefit of Contributing	Benefit of Revealing	Suggestions made	Job_Architect	Job_Manager	Job_Developer	Job_ProjectMgr	Usefulness of Using	PACR	PDCR	CACR	CDCR	RelativeAdv OfOSS	DidOSS	KAIOriginality	KAIEfficiency	KAIConformity	ID_with_OSS	Reciprocity	EnviInfluence	Age
Benefit of Using	1																					
Benefit of Contrib.	0.60	1																				
Benefit of Revealing	0.50	0.62	1																			
Suggestions made	0.20	0.19	0.22	1																		
Job_Architect				0.21	1																	
Job_Manager			-0.11		-0.11	1																
Job_Developer		-0.11		-0.18	-0.40	-0.44	1															
Job_ProjectMgr			-0.11				-0.24	1														
Usefulness of Using	0.51	0.43	0.30	0.13					1													
PACR	0.35	0.31	0.40	0.19			-0.14		0.26	1												
PDCR	0.22	0.24	0.22	0.14		-0.11			0.25	0.19	1											
CACR	0.44	0.42	0.43	0.33					0.33	0.49	0.30	1										
CDCR	0.11	0.21	0.24	0.12					0.17		0.29	0.24	1									
RelativeAdvOfOSS	0.42	0.39	0.46	0.21					0.38	0.40	0.16	0.46	0.25	1								
DidOSS	0.20	0.21	0.11			-0.11	0.17		0.31			0.18		0.14	1							
KAIOriginality	0.15	0.13	0.12	0.19	0.23	0.10	-0.14				0.15	0.16				1						
KAIEfficiency												0.16				-0.34	1					
KAIConformity				0.15								0.28				-0.13	0.37	1				
ID_with_OSS	0.45	0.46	0.43	0.20	0.13			-0.17	0.45	0.32	0.17	0.38	0.25	0.47	0.18				1			
Reciprocity	0.38	0.41	0.37	0.21					0.32	0.37	0.18	0.45	0.18	0.42	0.14	0.14			0.50	1		
EnviInfluence					0.17				0.25	-0.12		-0.12						-0.20			1	
Age		-0.19	-0.22		0.16	0.21	-0.14		-0.16	-0.33	-0.13	-0.19		-0.17	-0.11		-0.12		-0.17		0.17	1

Table A.7: Correlations (Spearman rank correlation, displayed only for p<0.1)

	Corporation could benefit from... (1-5 scale, ordered probit)			Suggestions made (Probit)
	... using	... contrib.	... revealing	
Job_Architect	-0.453 (0.355)	0.028 (0.278)	-0.554* (0.324)	0.298 (0.392)
Job_Manager	-0.039 (0.349)	-0.177 (0.286)	-0.901*** (0.333)	-0.066 (0.384)
Job_Developer	-0.310 (0.250)	-0.454** (0.207)	-0.678*** (0.215)	-0.513* (0.274)
Job_ProjectMgr	-0.218 (0.349)	-0.383 (0.343)	-0.834* (0.433)	-0.457 (0.705)
Usefulness of Using				
PACR (Pers. Adv. of Revealing)				
PDCR (Pers. Disadv. of Revealing)				
CACR (Corp. Adv. of Revealing)				
CDCR (Corp Disadv. of Revealing)				
RelativeAdvOfOSS	0.511*** (0.119)	0.285*** (0.110)	0.446*** (0.120)	0.409*** (0.148)
DidOSS	0.433*** (0.158)	0.397*** (0.149)	0.076 (0.142)	-0.041 (0.202)
KAIOriginality	0.190 (0.141)	0.114 (0.131)	0.214* (0.127)	0.322* (0.170)
KAIEfficiency	-0.085 (0.138)	-0.100 (0.126)	-0.123 (0.113)	-0.053 (0.171)
KAIConformity	-0.107 (0.146)	-0.027 (0.127)	0.156 (0.125)	0.437** (0.175)
ID_with_OSS	0.391*** (0.094)	0.210** (0.102)	0.214** (0.095)	0.030 (0.148)
Reciprocity		0.356*** (0.120)	0.263** (0.120)	0.300* (0.172)
EnviInfluence	0.137 (0.125)	0.011 (0.117)	0.043 (0.117)	0.026 (0.149)
Age	-0.002 (0.009)	-0.018** (0.008)	-0.018** (0.008)	0.011 (0.011)
Constant				-5.918*** (1.461)

Observations	249	249	249	249
Pseudo R-squared	0.18	0.14	0.15	0.17
Pseudo Likelihood	-230.698	-281.659	-312.433	-106.444
Wald's chi-squared	99.676	70.872	119.159	35.561
Degrees of freedom	12	13	13	13

Robust standard errors in parentheses

* significant at 10%; ** significant at 5%; *** significant at 1%

Table A.8: Regressions (without TAM variables)

References

- Agarwal, R., and Prasad, J. "Are Individual Differences Germane to the Acceptance of New Information Technology?," *Decision Sciences* (30:2) 1999, pp 361-391.
- Allen, R.C. "Collective invention," *Journal of Economic Behavior & Organization* (4:1) 1983, pp 1-24.
- Bass, L., Clements, P., and Kazman, R. *Software Architecture in Practice*, (2nd ed.) Addison-Wesley, Boston, MA, 2003.
- Behlendorf, B. "Open source as a business strategy," in: *Opensources: Voices from the open source revolution*, C. DiBona, S. Ockman and M. Stone (eds.), O'Reilly, Sebastopol u.a., 1999, pp. 149-170.
- Bénabou, R., and Tirole, J. "Intrinsic and Extrinsic Motivation," *Review of Economic Studies* (70:244) 2003, pp 489-500.
- Bonaccorsi, A., and Rossi, C. "Why Open Source software can succeed," *Research Policy* (32:7) 2003, pp 1243-1258.
- Bonaccorsi, A., and Rossi, C. "Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business," *Knowledge, Technology, and Policy* (18:4) 2006, pp 40-64.
- Burgelman, R.A. "A Model of the Interaction of Strategic Behavior, Corporate Context, and the Concept of Strategy," *Academy of Management Review* (8:1) 1983, p 61.
- Chakrabarti, A.K., and O'Keefe, R.D. "A Study of Key Communicators in Research and Development Laboratories," *Ground & Organization Studies* (2:3) 1977, pp 336-346.
- Chesbrough, H.W. *Open Innovation. The New Imperative for Creating and Profiting from Technology* Harvard Business School Press, Boston, 2003.
- Dahlander, L. "Appropriation and Appropriability in Open Source Software," *International Journal of Innovation Management* (9:3) 2005, pp 259-285.
- Davis, F.D. "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," *MIS Quarterly* (13:3) 1989, pp 318-340.

- Davis, F.D., Bagozzi, R.P., and Warshaw, P.R. "User Acceptance of Computer Technology: a Comparison of Two Theoretical Models," *Management Science* (35:8) 1989, pp 982-1003.
- Davis, F.D., Bagozzi, R.P., and Warshaw, P.R. "Extrinsic and Intrinsic Motivation to Use Computers in the Workplace," *Journal of Applied Social Psychology* (22:14) 1992, pp 1111-1132.
- DiBona, C. "Open Source and Proprietary Software Development," in: *Open Sources 2.0: The Continuing Evolution*, C. DiBona, D. Cooper and M. Stone (eds.), O'Reilly Media, Sebastopol, CA, 2005, pp. 21-36.
- Dinkelacker, J., Garg, P.K., Miller, R., and Nelson, D. "Progressive Open Source," Hewlett Packard Laboratories Palo Alto, 2001.
- Driver, M., Drakos, N., Weiss, G.J., Claunch, C., Govekar, M., Feinberg, D., Maio, A.D., Hostmann, B., Igou, B., Cantara, M., Phifer, G., Enck, J., Pescatore, J., Latham, L., Gilliland, M., Silver, M.A., Haight, C., Valdes, R., Girard, J., Perkins, E.L., Lee, M., Hafner, B., Natis, Y.V., and Cain, M.W. "Hype Cycle for Open-Source Software, 2005," Gartner, p. 22.
- Driver, M., and Weiss, G.J. "Predicts 2006: The Effects of Open-Source Software on the IT Software Industry," Gartner, p. 5.
- Ebadi, Y.M., and Utterback, J.M. "The effects of communication on technological innovation," *Management Science* (30:5) 1984, pp 572-585.
- Falk, A., and Ichino, A. "Clean Evidence on Peer Effects," *Journal of Labor Economics* (24:1) 2006, pp 39-57.
- Fan, B., Aitken, A., and Koenig, J. "Open Source Intellectual Property and Licensing Compliance: A Survey and Analysis of Industry Best Practices," 2004.
- Feller, J., and Fitzgerald, B. *Understanding Open Source Software Development* Addison-Wesley, Boston, MA, 2001.
- Foxall, G.R., and Hackett, P.M.W. "Cognitive Style and Extent of Computer Use in Organizations: Relevance of Sufficiency of Originality, Efficiency and Rule-Conformity," *Perceptual and Motor Skills* (74:2) 1992a, pp 491-497.

- Foxall, G.R., and Hackett, P.M.W. "The factor structure and construct validity of the Kirton adaption-innovation inventory," *Personality and Individual Differences* (13:9) 1992b, pp 967-975.
- Gallivan, M.J. "The influence of software developers' creative style on their attitudes to and assimilation of a software process innovation," *Information & Management* (40:443-465) 2003.
- Goldman, R., and Gabriel, R.P. *Open Source as Business Strategy: Innovation Happens Elsewhere* Morgan Kaufmann, San Francisco, CA, 2005.
- Goulde, M. "Open Source Becoming Mission-Critical In North America And Europe," Forrester, p. 14.
- Grand, S., Von Krogh, G., Leonard, D., and Swap, W. "Resource Allocation Beyond Firm Boundaries: A Multi-Level Model for Open Source Innovation," *Long Range Planning* (37:6) 2004, pp 591-610.
- Granovetter, M.S. "The strength of Weak Ties," *American Journal of Sociology* (78:6) 1973, pp 1360-1380
- Gruber, M., and Henkel, J. "New ventures based on open innovation – an empirical analysis of start-up firms in embedded Linux," *International Journal of Technology Management* (33:4) 2006, pp 356-372.
- Hars, A., and Ou, S. "Working for Free? Motivations for Participating in Open-Source Projects," *International Journal of Electronic Commerce* (6:3) 2002, pp 25-39.
- Hecker, F. "Setting Up Shop: The Business of Open-Source Software," *IEEE Software* (16:1) 1999, pp 45-51.
- Henkel, J. "Open Source Software from Commercial Firms – Tools, Complements, and Collective Invention," *ZfB-Ergänzungsheft* (2004:4) 2004.
- Henkel, J. "Champions of Revealing - The Role of Open Source in Commercial Firms," Technische Universität München, 2006a, p. 35.
- Henkel, J. "Selective revealing in open innovation processes: the case of embedded Linux," *Research Policy* (35:7) 2006b, pp 953-969.

- Jones, C. "Variations in software development practices," *Software, IEEE* (20:6) 2003, pp 22-27.
- Katz, L.F., Kling, J.R., and Liebman, J.B. "Moving to opportunity in Boston: early results of a randomized mobility experiment," *Quarterly Journal of Economics* (116:2) 2001, pp 607-654.
- Katz, R., and Allen, T.J. "Investigating the Not Invented Here (NIH) syndrome: A look at the performance, tenure, and communication patterns of 50 R&D project groups," *R&D Management* (12:1) 1982, pp 7-19.
- Katz, R., and Allen, T.J. "Organizational Issues in the Introduction of New Technologies," in: *The Management of Productivity and Technology in Manufacturing*, P.R. Kleindorfer (ed.), Plenum, New York, 1985, pp. 275-300.
- Kirton, M.J. "Adaptors and Innovators: A Description and Measure," *Journal of Applied Psychology* (61:5) 1976, pp 622-629.
- Kirton, M.J. *Adaption-innovation: in the context of diversity and change* Routledge, London and New York, 2003.
- Klevorick, A.K., Levin, R.C., Nelson, R.R., and Winter, S.G. "On the sources and significance of interindustry differences in technological opportunities," *Research Policy* (24:2) 1995, pp 185-205.
- Koenig, J. "Seven open source business strategies for competitive advantage," in: *IT Manager's Journal*, 2004.
- Lakhani, K., and von Hippel, E. "How open source software works: "free" user-to-user assistance," *Research Policy* (32:7) 2003, pp 923-943.
- Lakhani, K., and Wolf, B. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," in: *Perspectives on Free and Open Source Software*, J. Feller, B. Fitzgerald and S. Hissam (eds.), MIT Press, 2005.
- Laursen, K., and Salter, A. "Open for innovation: the role of openness in explaining innovation performance among U.K. manufacturing firms," *Strategic Management Journal* (27:2) 2006, pp 131-150.

- Lehman, M.M. "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE* (68:9) 1980, pp 1060-1076.
- Lerner, J., and Tirole, J. "Some Simple Economics of Open Source," *Journal of Industrial Economics* (52) 2002.
- Lyytinen, K., and Rose, G.M. "The disruptive Nature of Information Technology Innovations: The Case of Internet Computing in Systems Development Organizations," *MIS Quarterly* (27:4) 2003, pp 557-595.
- Merton, R.K., and Rossi, A.K. "Contributions to the Theory of Reference Group Behavior," in: *Social Theory and Social Structure*, R.K. Merton (ed.), Free Press, New York, 1949, pp. 225-275.
- Moody, G. *Rebel Code - Inside Linux and the Open Source Revolution*, (1st ed.) Perseus Publishing, Cambridge, MA, 2001.
- Osterloh, M., and Rota, S. "Open Source Software Development—Just Another Case of Collective Invention?," 2005.
- Raymond, E.S. "The Cathedral and the Bazaar," in: *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, E.S. Raymond (ed.), O'Reilly, Sebastopol, 2001a, pp. 19-63.
- Raymond, E.S. "The Magic Cauldron," in: *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, E.S. Raymond (ed.), O'Reilly, Sebastopol, 2001b, pp. 113-191.
- Rogers, E.M. *Diffusion of Innovations*, (5 ed.) Free Press, New York, NY, 2003.
- Royce, W.W. "Managing the development of large software systems: concepts and techniques," in: *Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, Monterey, California, United States, 1987.
- Scacchi, W. "Free and open source development practices in the game community," *Software, IEEE* (21:1) 2004, pp 59-66.
- Seel, C. "Der Lego-Roboter geht jetzt auf seine Kunden zu," in: *Die Welt*, 2006.

- Senyard, A., and Michlmayr, M. "How to Have A Successful Free Software Project," 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004.
- Shah, S. "Motivation, Governance & the Viability of Hybrid Forms in Open Source Software Development," *Management Science* (52:7) 2006, pp 1000-1014.
- Sherif, K., Zmud, R.W., and Browne, G.J. "Managing Peer-to-Peer Conflicts in Disruptive Information Technology Innovations: The Case of Software Reuse," *MIS Quarterly* (30:2) 2006, pp 339-356.
- Stewart, K.J., and Gosain, S. "The Impact of Ideology on Effectiveness in Open Source Software Teams," *MIS Quarterly* (30:2) 2006, pp 291-314.
- Taylor, W.G.K. "The Kirton adaption-innovation inventory: Should the sub-scales be orthogonal?," *Personality and Individual Differences* (10:9) 1989a, pp 921-929.
- Taylor, W.G.K. "The Kirton Adaption -Innovation Inventory: a re-examination of the factor structure," *Journal of Organizational Behavior* (10:4) 1989b, pp 297-307.
- Varian, H.R., and Shapiro, C. "Linux Adoption in the Public Sector: An Economic Analysis," 2003.
- Venkatesh, V. "Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model," *Information Systems Research* (11:4) 2000, pp 342-365.
- von Hippel, E., and von Krogh, G. "Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science," *Organization Science* (14:2) 2003, pp 209-233.
- West, J. "How open is open enough? Melding proprietary and open source platform strategies," *Research Policy* (32:7) 2003, pp 1259–1285.
- West, J., and Gallagher, S. "Challenges of Open Innovation: The Paradox of Firm Investment in Open Source Software," *R&D Management* (36:3) 2006, pp 319-331.