



**Dynamics of Institutions and Markets in Europe** is a network of excellence of social scientists in Europe, working on the economic and social consequences of increasing globalization and the rise of the knowledge economy.  
<http://www.dime-eu.org/>

# DIME Working Papers on INTELLECTUAL PROPERTY RIGHTS



Sponsored by the  
6<sup>th</sup> Framework Programme  
of the European Union

<http://www.dime-eu.org/working-papers/wp14>

Emerging out of DIME Working Pack:

**'The Rules, Norms and Standards on Knowledge Exchange'**

Further information on the DIME IPR research and activities:

<http://www.dime-eu.org/wp14>

**This working paper is submitted by:**

**Georg von Krogh, Sebastian Spaeth, Stefan Haefliger and  
Martin Wallin**

Department of Management, Technology, and Economics, ETH Zurich  
Email: [gvkrogh@ethz.ch](mailto:gvkrogh@ethz.ch)

*Open Source Software: What we know (and do not know) about motives to contribute*

**This is Working Paper  
No 38 (April 2008)**

*The Intellectual Property Rights (IPR) elements of the DIME Network currently focus on research in the area of patents, copyrights and related rights. DIME's IPR research is at the forefront as it addresses and debates current political and controversial IPR issues that affect businesses, nations and societies today. These issues challenge state of the art thinking and the existing analytical frameworks that dominate theoretical IPR literature in the fields of economics, management, politics, law and regulation-theory.*

Open Source Software:

What we know (and do not know) about motives to contribute

5

Georg von Krogh

Sebastian Spaeth

Stefan Haefliger

Martin Wallin

Department of Management, Technology, and Economics

10

ETH Zurich

Switzerland

## **Abstract**

15        Open source software is a major social and economic phenomenon that raises important  
questions about incentives and motivations in information systems development. For  
example, some software developers are unpaid volunteers who seek to solve their own  
technical problems, while others create open source software as part of their employment  
contract. For the past ten years, a substantial amount of academic work has theorized about  
20        and empirically examined developer motivations. We critically examine this work and show  
that it can be divided into two phases. In Phase One, scholars argued why developers  
contribute to open source software, while in Phase Two studies aimed at explaining the  
relationship between motivations of developers and institutional context characteristics such  
as community sponsorship or license restrictiveness. Based on research gaps and weaknesses  
25        in this work, we argue for entering a third phase of research that systematically integrates the  
social practice of software development and the institutional setting, in which open source  
software is embedded, with the types of motivations of developers. We apply the sociology of  
Alasdair MacIntyre to analyze the open source phenomenon and contribute a set of open  
research questions in Phase Three that we hope will sustain future research on open source  
30        software and allow it to flourish as it so far has.

## 1. Introduction

Open source software has two distinct features. First, such software comes equipped with licenses that make it a public good. Usually, these licenses provide existing and future users the right to download, inspect, use, modify, and distribute modified and unmodified software to others for free (Raymond, 1999). The resulting software products aim at many market segments, covering operating systems, middle-ware, and end-user applications including media-players, office suites, and games. Over the last 15 years, open source software products have made their successful inroads in these segments attracting many millions of users. For example, in 2007, Apache achieved 59% market share for web server software (Netcraft survey, 2007). In 2005, Linux product sales amounted to more than 7 Bn USD, according to figures from IBD. Firefox, the web browser, achieved a market share of 28% in Europe (and 18% in the US) according to a study by XiTi Monitor (2007). In 2006, Firefox provided USD 67m of revenues to the Mozilla foundation that markets and coordinates its development.

Second, while software can be termed as “open source” independent of how it was developed (it is sufficient for the software to be released with an open source license affixed), years of development have given rise to a new practice of information systems innovation. Open source software projects are often initiated by a project entrepreneur. Next, volunteer developers join the project and contribute to designing, writing, testing, debugging, distributing, and documenting the software. Depending on their knowledge and interest, these voluntary contributors perform tasks ranging from cheering, via administration and coordination, to technical development (Lakhani and von Hippel, 2003). Popular projects such as Linux or Azureus may receive support from several thousand voluntary developers, who emerge from a much larger group of users (Raymond, 1999; Lerner and Tirole, 2002;

von Hippel and von Krogh, 2003). Currently, Sourceforge, a platform for open source software development, hosts more than 162,000 projects with more than 1.7 million registered users.

A relatively recent phenomenon, open source software has attracted huge interest from the information systems academic community. Google Scholar lists approximately 1,500 papers using the keywords “open source software” and “social science”. In the ISI Database/social science, out of 190 papers that have “open source software” registered as keyword, roughly 75% have been published within the last three years. These studies apply theories and research methods from many fields including sociology, psychology, economics, information systems, organization and management studies, operations research, mathematics, as well as industrial- and software engineering. Researchers have investigated various aspects of the phenomenon, including motivations of open source software contributors; governance, organization, and the process of innovation in open source software projects; and competitive dynamics enforced by open source software (von Krogh and von Hippel, 2006).

A key research question initiating IS-related research on open source software was formulated by Josh Lerner and Jean Tirole (see Lerner and Tirole, 2002): *Why would thousands of top-notch software developers contribute for free to the creation of a public good?* The question of why people contribute to a public good through acting collectively is established and heavily researched in economics, sociology and psychology. However formulated in this particular way, the research question posed huge challenges for scholars who study information systems development within firms that use traditional incentive systems and careers to motivate developers. The question was also largely attractive because answering it would mean explaining a major social and economic phenomenon of high

80 academic and practical interest. Theory and research on motivations and incentives shed light  
on critical issues such as the emergence and growth of distinct open source software projects,  
their organization, and sustainability (Ulhoi, 2004; West and O'Mahoney, 2005; Markus,  
2007). This research was of high relevance to IS practice since firms are both heavy users of,  
and contributors to, open source software. For example, if a firm decides to invest millions of  
85 USD in order to migrate its servers to a Linux-IBM system, managers will be interested to  
know to what extent Linux will continue to receive contributions by volunteers, how the  
software will evolve, and if Linux projects manage to regularly release new and improved  
versions of the software. Research on motivation and incentives directly targets this interest.

Theory building and research on the topic of motivations and incentives in open source  
90 software is vast, and the objective of this paper is to review major contributions. We seek to  
provide interested scholars with an inroad to the literature, and thus provide a platform for  
new research on the topic. Through a critical examination of previous work, we also seek to  
identify research gaps and contribute a set of new research questions. Since contributions on  
motivations and incentives in open source software originate in different disciplines,  
95 including anthropology, economics, psychology, and sociology, they apply a great variety of  
qualitative and quantitative research designs. Hence, the method for this review is a synthetic,  
qualitative, and critical analysis.

We show that theory and research on motivations and incentives can be organized in two  
phases. In Phase One, theoretical and empirical studies attempted to explain *why* voluntary,  
100 unpaid developers contribute to open source software, - a direct response to the original  
question by Lerner and Tirole. In their pioneering work Lerner and Tirole (2002) used  
economic theory to propose that signaling incentives were available and strong in the labor  
market for software developers and that developers contribute to open source software in

order to increase their human capital. By writing and sharing software in the public domain  
105 developers signal to current and prospective employers their level of skills and thereby  
increase their salaries and advance their careers. Partly dissatisfied with the lack of empirical  
grounding of this proposition in open source software projects, von Hippel and von Krogh  
(2003) combined economic and sociological theory to develop a “private-collective” model  
of innovation incentives. The authors suggested that developers contribute to a public goods  
110 innovation because they garner private benefits related to the innovation process. These  
benefits, including fun, reputation, learning, and peer recognition, are not supplied in the  
same amount to people who use but do not contribute to software development. This view  
was supported by an early survey study by Hertel et al. (2003) who found that Linux  
developers were motivated to contribute by the need to improve the product for their own use  
115 of it. These developers also identified with the larger “community of Linux developers”, and  
thus were motivated by group-related factors, such as their “perceived indispensability” for  
the group in which they work. Taking these ideas further, Bagozzi and Dolhakia (2006), using  
a survey design, found that participants in Linux User Groups (technical support groups)  
were motivated to contribute at different levels, by a combination of social and psychological  
120 factors. “We-intentions” were important for predicting individual actions that contribute to  
group-level actions.

In Phase Two, researchers ventured beyond the straightforward connections between  
motivation and contribution by studying interrelations between and among motivations,  
contributions, and institutional arrangements. Inseparable from the individuals' decisions to  
125 contribute (Shah, 2006; Roberts et al., 2006), institutional arrangements such as sponsorships  
and license characteristics enable or inhibit individual motivation. This view was supported  
in the work by Stewart, Ammeter and Maruping (2006) who found that the profit orientation

of an organization that sponsored an open source project was related to the development activity the project could attract.

130       Based on a critical analysis of work in the two phases, we show that research starts to grapple with a larger and thus far hidden issue: the relationship between *institutions* that provide traditional incentives such as careers and pay for developers, and *practices* that incent developers through social norms, ideology, virtue ethics, and peer recognition. In order to shed light on this issue, we argue for entering into a third phase of research where IS  
135 scholars begin to theorize and study the nature of contributions to open source software and the motivations to contribute as they emerge in the tension between institutions and practices. Thus, we argue for a context dependent understanding of motivation, as well as a more fine grained analysis of intrinsic motivation because intrinsic motivation and a large number of volunteers could be identified as a standard characteristic of open source software  
140 development (von Hippel and von Krogh, 2003). In this Phase Three, scholarship should be directed towards the knowledge and expertise people contribute to open source software and their impact on the software's quality and the institutions it generates.

      We introduce the sociology of Alasdair MacIntyre (for an introduction, see Knight, 1998) whose work can shed new light on how developers who establish a new social practice,  
145 such as an open source project, develop norms, values, and commitments *related to excellence of the goods they produce*. On the one hand, MacIntyre theorizes that an institution offering monetary incentives and careers only to a limited extent can motivate practitioners to bring forth excellence in their products (using MacIntyre's theory in a strict sense, institutions tend to destroy these efforts). For example, consider Lerner and Tirole's proposition on the  
150 signaling incentive. On the one hand, firms may tend to reward developers who are productive in terms of coding volume. From the vantage point of MacIntyre's theory, when

such institutional incentives are strong, developers may find it more beneficial to contribute a large volume of software across many projects rather than providing small, targeted, high quality (but obscure) software patches. On the other hand, practices with strong values and norms attached to public goods can motivate people to strive for excellence in products and processes (Hertel et al. 2003). In addition to formulating a set of research questions for Phase Three, we also discuss possible research designs for the study of open source software practices. Promising designs include comparative, longitudinal, case studies of different open source software projects (Markus, 2007).

The paper is organized as follows: The next section presents and critically examines research in Phase One. Subsequently, we show and analyze work in Phase Two that aims at explaining interrelations between motivations to contribute and institutional enablers. The fourth section of the paper introduces the sociology of Alasdair MacIntyre and uses his work to analyze how the tension between institution and social practice shape developer motivations. Based on this analysis, section five develops a set of research questions aimed at sustaining and developing research on open source software development. Section six concludes this review.

## **2. Phase One: The motivation to contribute**

This section provides an overview of motivations to contribute to open source software, based on a comprehensive review of literature, including books, articles, conference proceedings, and, when accessible, working papers. We provide an critical overview of the frameworks used in previous studies of open source projects, and proceed to use the dominant taxonomy to cluster identified motivation factors.

Theoretical and conceptual papers were examined for motivational factors that were used to create theory. Thus articles simply listing motivations based on a literature review

were excluded from the analysis. Motivational factors that proved relevant in empirical papers were also included in our review. Whenever a study used a different terminology, but the motivation seemed sufficiently close to an existing one, they were merged under one category. We start by arguing that in studying contributors to open source software, the dominant framework of human motivation has distinguish between extrinsic and intrinsic motivation. Next, we proceeded to analyze contributions to this framework in open source software development.

## **2.1 Frameworks of Human Motivation**

Studies of individual motivation to contribute to open source software development can be grouped into categories depending, on which stream of the literature that forms the basis for theory building, data-gathering, and -analysis. While various frameworks have been used by authors, the most frequent has been intrinsic/extrinsic motivation. This distinction is based on different reasons that bring about action. An activity is extrinsically motivated when it is done in order to obtain some separable outcome, whereas an activity is intrinsically motivated when the activity is done for the inherent interest or joy of performing said activity (Deci and Ryan, 1985). A number of empirical studies have proposed and found evidence that open source software developers have both intrinsic and extrinsic motivations for contributing to the development (Hars and Ou, 2002; Lakhani and Wolf, 2005; Roberts et al., 2006; Wu et al., 2007). Following the work by Lindenberg (2001), Lakhani and Wolf (2005), and Osterloh and Rota (2007) distinguished between enjoyment-based intrinsic motivation and obligation/community-based intrinsic motivation. The latter paper provided a theoretical overview whereas the former paper by Lakhani and Wolf empirically found that both types of intrinsic motivation as well as extrinsic motivation impacted on work effort. Wu, Gerlach and Young (2007) also used a framework based on intrinsic and extrinsic motivation to explain

200 continuance intention in open source software projects. Hars and Ou (2002) suggested that  
intrinsically motivated contributors will spend more time and effort in open source projects  
but did not examine this empirically. Other empirical studies concentrated on intrinsic  
motivation rather than extrinsic motivation. For example, Lakhani and von Hippel (2003)  
linked feelings of competence and fun to willingness to help other developers. In contrast,  
205 Lerner and Tirole's (2002) mainly explained contributions by means of extrinsic motivation.  
Many authors have also approached motivational aspects from a perspective of reciprocity,  
for example gift giving (Bergquist and Ljungberg 2001; Wu, Gerlach and Young 2007),  
reciprocal helping behavior (i.e. helping because have been helped or expect to be helped)  
(Lakhani and von Hippel 2003), status motivation (Roberts, Hann and Slaughter 2006). Their  
210 work may be said to focus on "internal extrinsic" as it deals with extrinsic motivation which  
has been internalized by individual contributors (Hars and Ou, 2002; Lakhani and Wolf,  
2005; Wu, Gerlach and Young, 2007).

While alternative frameworks to extrinsic/intrinsic have been proposed, they are often  
closely related to Deci and Ryan's original contribution. For example, Bonaccorsi and Rossi  
215 (2006) distinguished between economic, social and technological motivations, building on a  
taxonomy proposed by Feller and Fitzgerald (2002). The economic motivation is similar to  
extrinsic motivation, and the social motivation is close to intrinsic motivation. However, the  
authors also suggest a third type "technological motivation" that includes benefits from  
learning and working with a bleeding-edge technology.

220 Attempts towards a broader and integrative framework can be found in Hemetsberger  
(2004) and Hertel et al. (2003). Hemetsberger developed two main categories of motivation  
"self-interest" and "others-orientation." Self-interest was further divided into task- and  
product-related motivation (corresponding to intrinsic motivation) and others-orientation

including long-term utilitarian goals and social significance (resembling extrinsic motivation)  
225 was divided into internalized group goals and values, and socio-emotional relationship.  
Hertel, Nieder and Herrmann (2003) extended a model of voluntary action in social  
movements proposed by Bert Klandermans (1997), to include four factors: collective  
motives, norm-oriented motives, reward motives, and identification motives. The authors also  
also built on the VIST-model of Hertel (2002) to explain individual motivation to work in a  
230 virtual team. The model included valence (the subjective evaluation of goals), instrumentality  
(the perceived importance of one's own contributions), self-efficacy (the team members'  
perceived capability showing the required activities for the team task.

There are three reasons why the extrinsic/intrinsic framework became dominant in the  
study of motivations to contribute to open source software. First, the framework has been  
235 widely used in psychology and economics to explain why and how much people contribute to  
economic activity (for overviews see Deci and Ryan, 1985; Benabou and Tirole, 2003; Frey,  
1997). Thus, the framework is backed up strong empirical research findings, and as well  
guided by many years of experience from various empirical research designs. Second, using  
and adopting one dominant framework allowed for the comparison of results across project  
240 samples. Third, as observed by Lerner and Tirole (2002), open source software is a successful  
outcome of collective action by people who may or may not be paid for their contributions.  
Thus, while information systems development normally hinges on the work by paid  
developers, the context of open source software represents an interesting phenomenon for the  
study of complimentary intrinsic motives in collective action (e.g., Lakhani and Wolf, 2005;  
245 Hars and Ou, 2002). This triggered the interest of social scientists from many fields.

The large work on motivations based on the extrinsic/intrinsic framework can be  
grouped into four categories of intrinsic motivation (ideology, altruism, kinship amity, and

enjoyment/fun), and four categories of extrinsic motivation (reputation, reciprocity/gift economy, learning, own-use value, career concerns, and pay). In the following, we analyze  
250 each of the categories in further detail.

## **2.2 Intrinsic Motivation**

An activity is intrinsically motivated when the activity is done for the inherent interest or joy of performing said activity (Deci and Ryan, 1985). The literature presents theory and research on four categories of intrinsic motivation; ideology, altruism, kinship amity, and  
255 enjoyment/fun.

### ***Ideology***

From the very beginning of the Free software movement in 1983, ideology played an important role in shaping the dominant GNU General Public License, as well as Richard Stallman's and others' advocacy of the ideas behind Free software. Ideology has been quoted  
260 as a major reason for starting the GNU project, one of the earliest coherent attempts to write software under an explicitly open license (Stallman, 1999). Early observations of communities stated that ideology varies for different developers: "one degree of variation is zealotry; whether open source development is regarded merely as a convenient means to an end [...] or as an end in itself." (Raymond, 1998)

265 Researchers employed models that explain voluntary participation of contributors to open source software development (Hertel et al., 2003). One of these models was developed by Bert Klandermans (Klandermans, 1997) and uses a variety of social and political motives to explain collective action, including ideology. The extent to which contributors adhere to ideology is captured by items such as: "software should be free for all", "free to modify and  
270 redistribute", or that "open source code should replace proprietary software." A number of

other empirical studies found support for ideological motives (Ghosh, 2005; David et al., 2003; Lakhani and Wolf, 2005), even if weak (Hemetsberger, 2004), explaining contributions to open source software development. Most notably, Hertel et al. (2003) found a positive, significant relationship between social and political motives with accepted source code  
275 patches and lines of code contributions. Stewart and Gosain (2006) investigated how ideology shapes the motivation of developers to produce behavior that enhances effectiveness of an open source software project team by examining the link between the level of ideology and the developer's effort spent. Their results confirm that open source developers' adherence to the community ideology (defined as "open source" norms, values, and beliefs) impacts on  
280 team effectiveness. They conclude that "adherence to some ideological components was beneficial to the effectiveness of the team in terms of attracting and retaining input, but detrimental to the output of the team." (p. 291)

While ideology explains level of contributions, it is still unclear in these studies how this construct affects the practice of open source software development and also how it impacts  
285 on the quality of contributions to the development effort. We will return to this point in part 4 of the paper.

### *Altruism*

Altruism is the selfless concern for the welfare of others. A typical altruistic act consists of three characteristics: "a) it is an end in itself; it is not directed at gain b) is emitted  
290 voluntarily, and c) does good." (Heider, 1958 in Krebs, 1970, p.259). Due to the self-containment of an altruistic act, it fits well with the category of intrinsic motivation, and several authors have used the concept of altruism to explain code contribution of open source software developers. The reviewed papers are of several types; purely theoretical, mathematical, or empirical investigations. Most papers attempt to explain the decision to

295 contribute, but not the type or level of contributions. However, a few papers explore this  
avenue as well. Most authors argue for a positive relationship between altruism among  
developers and contributions, but one survey of developers finds selfish behavior to such a  
degree as to rule out altruistic behavior as an important characteristic of open source software  
development (Ghosh, 2005). In a theoretical paper Osterloh and Rota (2007) suggested that  
300 altruistic behavior caused by “pro-social motives” influences developers to contribute to open  
source software development. The “pro-social motive” is a type of intrinsic motivation  
(Lindenberg, 2001), which the authors link to open source contributions. The authors argue  
that the good of the community enters into the preferences of the individual through e.g.,  
charitable giving (Frey and Meier, 2004).

305 Mathematical modeling has also been used to examine links between altruism and  
participation. Reviewing literature, Bitzer et al. (2007) argued that pure altruism is an  
important motive for publishing source code for some individuals. Running their model, the  
authors found that an individual who experienced a larger gift benefit (perceived giving as  
more rewarding) was likely to contribute more code. Drawing on ideas of altruism (Becker,  
310 1974) as well as utilitarian ethics (Harsanyi, 1978), Haruvy et al. (2003) develop a  
mathematical model assuming that open source software developers are less motivated to  
contribute if the product is marketed commercially, which in turn leads to a reduction in the  
improvements of the software.

In a recent empirical paper, Wu et al. (2007) investigated the intention of open source  
315 software developers to continue their involvement in future projects. Their structural equation  
model shows that altruism in the form of helping behavior influences the developers  
continuance, only if mediated by the developers' satisfaction. An empirical study by  
Hemetsberger (2004) reported that 22 percent of developers ranked altruism as a motivational

factor to contribute. Hemetsberger also attempted to differentiate between types of developers  
320 and found that the importance of altruism in explaining contributions, is stronger for people  
who contribute much (30.7 percent), compared to medium contributors (23.9 percent) and  
low contributors (6.5 percent). Open source software developers surveyed by Hars and Ou  
(2002) reported that altruism motivate them to contribute. 16.5 % of the participants rate high  
on altruism. Student and hobby programmers rate altruism the highest at 24.2 %, followed by  
325 salaried and contract programmers at 11.1 %, whereas only 7.7 % of the programmers paid  
for open source development was driven by altruistic motivations. Hars and Ou correlated the  
contribution level, measured as effort in hours per week spent on a specific project, with the  
motivational factors and found that altruistic motives had a correlation coefficient of 0.192 to  
effort to contribute<sup>1</sup>. However, in his study for the European Union, Ghosh (2005) did not  
330 find altruism to be a main factor explaining contributions to open source software, with 55  
percent of the respondents reporting “selfish” reasons to participate.

Researchers have not consistently distinguished between “altruism” and the “gift  
economy” arguments; and often these terms are used synonymously. However, the concept of  
“gift economy” relies on a social norm of reciprocity (Mauss, 1959; Bergquist and Ljungberg,  
335 2001) while altruism-motivated action is self-contained (Krebs, 1970). Since some research  
focuses on either the one or the other aspect, we have decided to keep them separate. In the  
future, research should clearly distinguish between altruism and gift-economy related  
motives. While the two may produce the same effect (open source software contributions) the  
underlying explanations are quite different.

---

<sup>1</sup>The correlation coefficient for students and hobby programmers was 0.356, for salaried and contract  
programmers 0.061, and for programmers paid for open source development -0.163 (N.B. negative).

340 ***Kinship Amity***

The concept of *kinship amity* (Fortes, 1969) has been related to the concept of the gift economy (see Zeitlyn, 2003). However, kinship amity differs from the gift economy since the former does not assume reciprocity in social relations. For example, in families, - kin, there is no calculated economic relationship. Kinship amity thus also differs from altruism as a  
345 motive to contribute, because it is restricted to the group to which one belongs, such as the open source software community.

In our review, some equivalent constructs are subsumed under kinship amity, for example, the frequently cited motivation for open source software contributors, *community identification* (e.g., Hars and Ou, 2002). Community identification instills a feeling of  
350 belonging to a certain group, and urges people to help others in that group.

Kinship amity as a motive in open source software was first suggested by anthropologist David Zeitlyn (2003) as an explanation for why people contribute to open source software. Zeitlyn explicitly criticized the view that the gift economy could explain the fundamental motives of open source developers, suggested in early work by Eric Raymond (1999).  
355 Hemetsberger (2004), using concepts such as “group boundaries” and “group bonds,” found a weak relationship between kinship amity and contribution level of developers. Lahani and Wolf’s survey (2005) identify kinship amity as an important motive for contributing, and show it is an important determinant of the effort invested (hours per week). Studying “community identification”, Hars and Ou (2002) find a correlation between kinship amity  
360 and the spent number of hours per week. Hertel et al. (2003) tested the relationship between kinship amity and the number of accepted patches and lines of code, and found it to be positive and significant. Thus developers with a higher sense of kinship are also more productive in terms of tangible source code output.

Hars and Ou (2002) further distinguished between unpaid and paid developers in their  
365 survey, and find that unpaid developers exhibit somewhat higher levels of kinship amity. This  
indicates that institutions, such a software firms paying developers, relate to the extent  
developers are willing to help other volunteers and paid developers in open source software  
communities. Hars and Ou's findings have not been further investigated, however. We return  
to this issue in section 4 of this paper.

### 370 ***Enjoyment and Fun***

Enjoyment and fun has been suggested to motivate contributors to open source projects.  
One of the main drivers of the so called “hacker culture” emerging during the 1980s was for  
developers to enjoy the playfulness and experimentation with hard- and software (Levy,  
1984; Torvalds and Diamond, 2001). Often connected to motives of fun and enjoyment as an  
375 intrinsic motive, is the concept of “flow” (Csíkszentmihályi, 1975, 1990). Flow results from  
an intense focus on a challenging task and leads to a state in which participants lose track of  
time. Benkler (2002) and Osterloh and Rota (2007) suggested that enjoyment plays an  
important role in open source software. Lakhani and von Hippel (2003) showed in their  
survey that developers considered enjoyment and fun as important when conducting  
380 technically challenging tasks, whereas mundane tasks, such helping users to install software,  
requires different motives.

Several empirical studies confirm enjoyment and fun as motivating participation.  
Luthiger and Jungwirth (2007) conducted the most comprehensive study focusing exclusively  
on fun and enjoyment motivations. Their survey of 1,330 open source developers revealed  
385 that the factor *fun* had a significantly positive effect on both the number of hours spent on a  
project as well as on their intention to participate in the future. *Fun* accounted for 28 percent  
of the effort in term of number of hours dedicated to projects. In a survey of comparable size,

Lakhani and Wolf (2005) found that enjoyment-based motivation was deemed by developers as an important source of motivation. In their research, high levels of enjoyment also  
390 increased the hours per week that developers spent on a project. Hertel, Hermann, and Niedner (2003), measuring the number of accepted patches and lines of code in the Linux project, also found a significant positive impact of fun and enjoyment.

Interestingly, Hemetsberger's (2004) survey identified modest positive impact of enjoyment on contributions to projects, and a study by Roberts, Hann, and Slaughter (2006)  
395 could not identify a significant link between intrinsic enjoyment and the number of accepted patches and lines of code. Thus, it seems intrinsic motivation such as enjoyment and fun, may not be universally applicable. In section 4, we discuss how fun and enjoyment possibly relate more directly to social practice and other motivations, such as virtues and excellence.

### **2.3. Extrinsic Motivation**

400 An activity is extrinsically motivated when it is done in order to obtain some separable outcome. Extrinsic motivations can be divided in two groups. Some motivations are extrinsic by definition but contributors could internalize these motivations so that they are perceived as self-regulating behavior rather than externally imposed (Roberts et al., 2006; Deci & Ryan, 1987). Internalized extrinsic motivations include reputation, reciprocity, learning, and own-  
405 use value. Pure extrinsic motivations include careers and pay.

#### ***Reputation***

Based on the literature, reputation can be classified in “peer reputation” and “outside reputation.” First, one of the earliest recognized and frequently cited motives for contribution to open source software is increased reputation among peers. Raymond's (1998) essay  
410 “Homesteading the Noosphere”, links reputation to reciprocity in the gift economy and

describes it as the “major motivation” for developers: “prestige is a good way (and in a pure gift economy, the only way) to attract attention and cooperation from others. If one is well known for generosity, intelligence, fair dealing, leadership ability, or other good qualities, it becomes much easier to persuade other people that they will gain by association with you.”

415 Moreover, “...you do not become a hacker by calling yourself a hacker - you become a hacker when other hackers call you a hacker.” (Raymond, 1998)

In a similar vein, Stewart (2005) and Lerner and Tirole (2002) propose peer reputation as a fundamental motivation. Osterloh and Rota (2007) term this motivation *ego gratification*, which could easily be confused with intrinsically motivated self-determination (the happiness  
420 of having/being able to achieve something, see Deci and Ryan, 1987). However, the authors classify it as a extrinsic signaling incentive, aimed at increasing one's own labor market value. Lakhani and von Hippel (2003) differentiate peer reputation further. They propose peer reputation motivates “gratifying” technical tasks, while it fails to motivate the “necessary but mundane tasks” that are an inherent part of each software project. This becomes clear as such  
425 tasks in their survey of the Apache project is often conducted by anonymous contributors. Furthermore, Lattemann and Stieglitz (2005) propose the contributor's role are related to motivations. In their view, programmers (rather than bug fixers, or managers) are motivated through peer reputation. Rather than examining contributor roles, Spaeth et al. (2008) argue that some motives are formed as by-products of contributions. In their empirical study of the  
430 Freenet project, the authors find that higher levels of contributions provide more peer reputation, such as positive mentioning in e-mail lists. The surveys by Lakhani and Wolf (2005), Hemetsberger (2004), Hars and Ou (2002), Ghosh (2005) report peer reputation as an driver for participation.

Three studies have examined the linkage between peer reputation and the level of

435 invested effort: Hars and Ou (2002) identify a weak but existing relationship between  
reputation and the number of hours invested. Lakhani and Wolf (2005) find peer reputation to  
be the fourth biggest determinant of invested effort. Roberts et al. (2006) measure the  
accepted lines of code. They identify a significant positive relationship between peer  
reputation motives and accepted code in the Apache project. Stewart (2005) who examined  
440 an online community dedicated to free software developers, found that a newcomer's  
reputation tends to stabilize some time after joining. To date, this is the only study that  
examines the dynamics of developer's reputation in open source software development  
projects.

Second, peer reputation is usually targeted to community insiders (peers, or kin) and  
445 potential employers who perceive peer reputation to signal potential talent. Only very few  
studies have considered reputation directed to the outside of the community and not targeted  
towards potential future employers. *Outside reputation* is concerned with the expected  
reactions and prestige awarded to the contributors, by significant others such as friends or  
relatives. Hemetsberger (2004) finds a weak relationship between outside reputation and  
450 participation. Hertel et al. (2003) test the impact of outside reputation on the number of  
accepted patches and lines of code. *Ceteris paribus* they find a significant positive impact of  
outside motivation on accepted code. Lattemann and Stieglitz (2005) also argue that  
programmers are motivated by peer and outside reputation, although they do not elaborate  
further on this. Outside reputation can be an important motivation because open source  
455 software developers contribute goods to external users who are not in any way affiliated with  
developer communities or firms. We will return to this point in part 4.

### ***Gift Economy/Reciprocity***

Originally a concept from anthropology (Mauss, 1959), several authors discussed the  
460 logic of gift giving in the context of open source software development (Raymond, 1999;  
Bergquist and Ljungberg, 2001; Zeitlyn, 2003). Viewing open source software development  
as a gift economy asserts that developers give code to others expecting gifts in return. The  
corresponding internalized, extrinsic motivation can be termed “reciprocity.” As previously  
discussed a few authors mistake the implications of a gift economy with altruism (e.g., Wu et  
465 al., 2007). Reciprocity as a motivation for contributions to open source software, was first  
suggested by Bergquist and Ljungberg (2001), and later confirmed by empirical studies that  
found moderate (Hemetsberger, 2004; Lakhani and Wolf, 2005) to strong support (David et  
al., 2003). Interestingly, Lakhani and von Hippel (2003) in their survey found that reciprocity  
motivated developers to perform mundane tasks. It seems if people have been helped by other  
470 contributors in the past, as they gain experience and knowledge, they are more inclined to  
reciprocate. This speaks to a certain social practice, and we return to this point in section 4.

### ***Learning***

The motive to acquire new skills or to learn in open source software development (von  
Hippel and von Krogh, 2003) appears in almost every contribution to this review article.  
475 However, the definition of learning was often vague and referred to survey items such as  
“improve programming skills” (the possibility to learn from the experience of writing  
software and the feedback provided by the peers who tested, integrated, and commented upon  
the software published). As such, as a single motivation learning without a specification of  
subject and object runs danger of oversimplifying or making empirical results non-  
480 comparable. Some studies categorize learning as a “human capital” motive (e.g., Hars and

Ou) which does not contribute further to a sharp definition of the construct.

In spite of conceptual fuzzyness, survey studies confirmed that learning motivated individuals to participate in open source software development (Ghosh, 2005; Hemetsberger, 2004; Lakhani and Wolf, 2005). This link appeared as particularly strong in two studies  
485 (David et al., 2003; Hars and Ou, 2002). Wu et al. (2007) found that learning motives led to the intention to participate. In the survey by Robert et al. (2006) the effort spent on developing open source software could be explained by learning and the accepted patches and lines of code written were (significantly) positively impacted by learning. Ye and Kishida (2003) suggested to consider legitimate peripheral learning based on the work by Lave and  
490 Wenger (1991) to explain increasing levels of participation over time (see also Rullani, 2007). In this framework, newcomers start learning about the project by initially participating in peripheral activities and then get “dragged” to the core of the activity set over time, provided the developers become increasingly familiar with the core. Thus, learning is a dynamic motive that related to social practices that afford those learning opportunities for contributors.  
495 We return to this in section 4.

### *Own-use value*

Own-use value refers to internalized extrinsic motives to create open source software for contributors personal use. Many authors suggest that developers of open source software “scratch their itch” by developing software they find useful, by fixing bugs, and by adding  
500 features they need (Raymond, 1999; Lakhani and von Hippel, 2003; Osterloh and Rota, 2007). With the important exception of Roberts et al. (2006), other surveys we considered in this review identified own-use value as a motive for participating in the development of open source software including Ghosh (2005), David et al. (2003), Hars and Ou (2002), Lakhani and Wolf (2005), as well as Hemetsberger (2004). Wu et al. (2007) found use value

505 connected to the intention to participate. Lakhani and von Hippel (2003) identified use value as a motive to take on mundane tasks. Regarding actual effort measured in hours spend per week, Hars and Ou (2002) report a high score developers attributed to use value as their motive. Hertel et al. (2003) reported a significant effect of this motive on accepted patches and lines of code contributed. Surprisingly, Roberts et al. (2006) reported a significant  
510 negative impact own-use value exerted on the level of participation in the Apache project, also measured in accepted patches and lines of code. One explanation offered is that developers motivated by own-use value worked “eclectically”: they would fix bugs that annoy them and then leave the development again, rather than remaining as long-term developers. This behavior would result in relatively low total contributions to one project.

515 Lattemann and Stieglitz (2005) propose that own use value may impact on open source software development via the roles individuals assume in communities. Contributors who mainly fix bugs may be particularly motivated by use value, whereas others such as managers (maintainers), might be more motivated by pay. While there has been no empirical work on this distinction, it suggest that social practices and institutions may exert influence on  
520 contributors motives. We return to this in section 4.

### *Careers*

Lerner and Tirole (2002) first suggested to study the signaling behavior of open source software developers. Their proposition derived from economic literature stated that individual developers would be motivated by career concerns when developing open source software.  
525 By publishing software which was free for all to inspect, they could signal their talent to potential employers and thus increase their value in the labor market. Subsequent empirical studies asked survey participants to indicate whether career concerns mattered for their participation in open source software development. While Lakhani and Wolf (2005) and

Hemetsberger (2004) found weak support, Hars and Ou (2002) and Ghosh (2005) found more  
530 substantial support for career concerns as a motivational factor. Wu et al. (2007) found career  
concerns also related to intended participation. When it comes to efforts measured in hours  
per week spent for open source software development, Hars and Ou (2002) found strong  
support, including support for motivation related to the sale of complementary products. For  
participation in general, however, they found complementary sales to play a rather  
535 unimportant role. Most notably, Roberts et al. (2006) and Hertel et al. (2003) documented a  
positive and significant relationship between career concerns and accepted code patches and  
lines of code. According to Hars and Ou (2002) both career concerns and complementary  
sales played a more important role for paid participation in open source software  
development than for unpaid participation.

#### 540 *Pay*

Open source software developers are often characterized as driven by intrinsic  
motivations receiving no financial compensation for their time and effort. For example,  
Bessen (2002) states that “When you use the World Wide Web, it is likely that the web pages  
are sent to you by software that was developed by unpaid volunteers” (there are more  
545 examples by many authors). However, this is not necessarily the case anymore, for example,  
an examination of contributions to the Linux kernel found only 9 percent of the involved  
developers working on their own time (LWN.net, 2007).

In an eclectic conceptual article Benkler (2002) suggested that monetary compensation  
was related to participation. Several empirical studies on the topic exist, albeit much of the  
550 data is purely descriptive. For example, the FLOSS-EU survey undertaken by Ghosh (2005)  
related motivation to wage, but not to effort. Their full report on that survey (Ghosh et al.,  
2002) reports a strikingly small difference between the invested effort (in number of hours

per week) and the employment status (unemployed/unpaid and employed/self-employed) of developers. In a survey of Sourceforge contributors, Lakhani and Wolf (2005) noted that a significant minority of approximately 40 percent are paid to participate in the open source software projects. This finding, however, requires some more indepth investigation. Lakhani and Wolf arrive at their conclusion by combining the responses by those programmers who receives direct financial compensation with those who receives indirect payments. A large majority of 87 percent of all respondents reported that they did not receive any direct payments. However, 55 percent of all respondents contributed code during their work time. By lumping these contributions together Lakhani and Wolf arrive at the figure of 40 percent paid contributors to open source software development. The authors also examined the degree of participation and its link to financial motives. They concluded that the financial subsidy of these projects is substantial. For example, paid contributors dedicated 17.7 hours per week on all FOSS projects they participated in while volunteers contributed 11.7 hours per week. As programmers often participate in several projects simultaneously, Lakhani and Wolf reports the results for the focal project of the programmers as well. These results showed a similar pattern: 10.3 hour per week for the paid contributor and 5.7 hours per week for the volunteer. The differences between the groups were found to be significant.

The results by Lakhani and Wolf are consistent with findings from other FOSS surveys (Hars and Ou, 2002; Hertel et al., 2003; Luthiger and Jungwirth, 2007). Lakhani and Wolf's claim that open source software developers are often paid is supported by the study on contributions to the Linux kernel made by Hertel et al. (2003). In their study it was found that 67 percent of the participants were full-time employees and a small percentage (five) were working half-time. The authors divided their contributors into two groups, a developer group and an interested reader group. The former group worked 18.4 hours per week on Linux

development, 20 percent received a regular payment that was tied to Linux development and other 23 percent received unregular compensation for their contributions. Still, a majority of 57 percent reported that they never received any salary for their contributions. The results on  
580 the links between effort and compensation are comparable to those of Lakhani and Wolf (2005) and Hertel et al. (2003) report that the more developers were paid for their Linux-related work the more time they spent contributing (measured as effort in hour per week). Another similarity with the Lakhani and Wolf survey was that many of the programmers could work on open source software development during their regular working hours (38  
585 percent). The authors stress that this does not imply that such contribution was in fact part of their official job. Reviewing literature on motivation, Lattemann and Stieglitz (2005) suggested that pay may affect the role taken in the community. However, while Lakhani and Wolf (2005) distinguished between paid developers and volunteers, and Hertel et al. (2003) distinguished between an active developers group and a passive interested reader group,  
590 Lattemann and Stieglitz argued that the motivation may differ between bug fixers, programmers and managers. In particular, they suggested that managers could be more driven by monetary compensation than the other groups. Looking at one specific form of participation, Dahlander and Wallin (2006) find that paid developers contribute more to development discussions in terms of emails sent. Two-thirds of the developers which  
595 responded to the questionnaire administered by Luthiger and Jungwirth (2007) reported that they work fully or partly for free for open source projects. The average developer spent 12.56 hours on open source software development, of which 58 percent was spent on his or her spare time whereas the rest, 42 percent was spent during working hours. The authors distinguished between the work effort of professionals and hackers<sup>2</sup> and found that the former

---

2 “A professional is designated as an open source developer producing less than 10 percent of his open source engagement in his spare time. Such a programmer engages virtually exclusively during working hours. In

600 group indeed do spend more time on open source software development than do the latter  
 (14.6 hours per week compared to 9.7 hours per week, the difference is reported to be  
 significant). A more complex picture was painted by Roberts et al. (2006) when they  
 suggested and tested various interrelationships between motivating factors, as we report in  
 the next section. Although finding that paid participation indeed was associated with  
 605 increased participation, their results show that the concept of pay may have more dimensions  
 than previously investigated.

## 2.4 Summary

Table 1 summarizes the review of studies analyzed motives and their impact on the  
 contribution to open source software development. By priority we reviewed empirical papers  
 610 but also included significant theoretical contributions that proposed novel relationships. The  
 table distinguishes between empirical and theoretical contributions.

MOTIVATION	INTRINSIC				INTERNALIZED EXTRINSIC				EXTRINSIC	
	<i>Ideology</i>	<i>Altruism</i>	<i>Kinship</i>	<i>Fun</i>	<i>Reputation</i>	<i>Reciprocity</i>	<i>Learning</i>	<i>Own-use</i>	<i>Career</i>	<i>Pay</i>
Benkler 2002	—	—	—	S	—	—	—	—	—	S
Bergquist and Ljungberg 2001	—	—	—	—	—	S	—	—	—	—
Bitzer et al. 2007	—	T	—	—	—	—	—	T	—	—
David et al. 2003	T	—	—	—	—	T	T	T	—	—
Ghosh 2005	T	T	—	—	T	—	T	T	T	T
Hars and Ou 2002	—	T	T	—	T	—	T	T	T	—
Hemetsberger 2004	T	T	T	T	T	T	T	T	T	—
Hertel et al. 2003	T	—	T	T	T	—	—	T	T	T
Lakhani and von Hippel 2003	T	—	—	S	S	T	—	T	—	—
Lakhani and Wolf 2005	T	—	T	T	T	T	T	T	T	T
Lattemann and Stieglitz 2005	—	—	—	—	S	—	—	S	—	S
Lerner and Tirole 2002	—	—	—	—	S	—	—	—	S	—
Luthiger and Jungwirth 2007	—	—	—	T	—	—	—	—	—	T
Osterloh and Rota 2007	—	S	—	—	—	—	—	S	—	—
Haruvy et al. 2003	—	S	—	—	—	—	—	—	—	—
Roberts et al. 2006	—	—	—	T	T	—	T	T <sup>*-</sup>	T	T
Spaeth et al. 2008	—	—	—	—	T	—	T	—	—	—
Wu et al. 2007	—	T	—	—	—	—	T	T	T	—
Ye and Kishida 2003	—	—	—	—	—	—	S	—	—	—
Zeitlyn 2003	—	—	S	—	—	—	—	—	—	—

\*-: significant negative impact on participation

S: Suggested T: Tested

*Table 1. Work on motivation in open source software (Phase One)*

While the research in Phase One reviewed generated a clear link between the extrinsic

---

contrast, a programmer whose open source engagement takes place for more then 90 percent of spare time matches the image we have of an open source hacker.”

and intrinsic motives and contributions, it did not relate motivations to the institutional setting of open source software development. Yet, as pointed out earlier, open source software  
615 development is a form of collective action that is influenced by or influence institutions that in turn enable individuals to contribute (von Hippel and von Krogh, 2003). As shown in decades of research on other forms of collective action ranging from lobbying, via preservation of natural resources, to the money collection for a good cause, institutions and individual motivations interrelate (Morris and Mueller, 1992). In the emerging Phase Two  
620 research on the motivation to contribute to open source software seek to shed light on institutions related to open source software, and focus on this interrelationship. We turn to this next.

### **3. Phase Two: Motivation and institutions**

Institutional context characteristics such as organizational sponsorship (Shah, 2006),  
625 barriers to joining a developer community (von Krogh et al., 2003), or the software license (Stewart et al., 2006) may be outside the direct control of single individual contributors, but yet, influence their motivation to contribute. Self-determination theory (Deci and Ryan, 1985), the dominant framework in Phase One, addressed extrinsic and intrinsic motivation. Such motivation can be understood both as a predictor and outcome of institutional context  
630 depending on “the nature of the study and the way and time in which self-determined motivation is measured” (Sheldon and Krieger, 2007: 885). While authors have recognized the interrelations between the motivation of open source software developers and the institutional context factors that impact on the development (von Hippel and von Krogh, 2003; von Krogh and von Hippel, 2006; Shah, 2006), most of this research is fairly recent  
635 and difficult to review because most research efforts spawned their own framework or

dimensions across organizations and individual. We review the findings along seven institutional characteristics suggested in the literature on open source software development and, then, proceed to discuss the special role of crowding theory (Frey and Jegen, 2001 for a review) which explains mutual adjustments among the motivation factors given specific  
640 institutional contexts. For the definition of institutions we follow North (1986) and include regularities in repetitive interactions among individuals, thus, customs and rules that “provide a set of incentives and disincentives for individuals” (North, 1986: 231). Prior research on open source software has identified governance (1) and community sponsorship (2), license restrictiveness (3), the provision of rewards (4), and infrastructure such as tools (5), code  
645 architecture and design (6), and social and technical exposure to a specific community over time (7), as context characteristics related to motivation.

First, Markus (2007) defines open source software governance as “the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an open source software development project to  
650 which they jointly contribute (2007: 152)”. While governance in open source software has been described in terms of structure, practices, rules, and norms, it leaves an important question unanswered that directly relates to motivation: what is the source of direction, control, and coordination in open source software development communities (Markus, 2007: 153)? The question about the source of control points to the organizational sponsorship of  
655 open source software development and the source of authority. Shah (2006) distinguished between open and gated source communities as two modes of governance where gated referred to the limited accessibility to the development process due to the sponsorship (and control) by a firm. She found that, on the long run, developers who were mainly motivated by use value tended to contribute to gated source communities whereas developers mainly

660 motivated by enjoyment contributed to open source communities. Considerations of fairness  
and reciprocity explained this behavior. The developers were aware of the property rights  
situation in the gated source communities and suspected the firms to act “strategically,” that  
is, neglecting the needs of the community and pursuing the firm's own interests. Thus,  
developers in gated source communities contributed only if they derived direct use value  
665 (Shah, 2006: 1011).

Second, Stewart et al. (2006) additionally distinguish between market (e.g., firms) and  
non-market sponsors (e.g., universities) and conclude that developers perceive cues as to the  
project's future, and hence incentives to contribute, both from the type of sponsor and the  
restrictiveness of the license (the third point). The effects, however, are interrelated. They  
670 found that open source projects with a non-market sponsor attract greater development  
activity than projects without a sponsor. Less restrictive licenses tend to attract more  
development activity to a project (Stewart et al., 2006; Fershtman and Gandal, 2004).  
However, “the presence of a non-market sponsor may alleviate concerns as to the project's  
future in the same way as a restrictive license would, in the sense that the restrictive license is  
675 not perceived as necessary to protect the developers' interests (Stewart et al., 2006: 141)”.

O'Mahony and Ferraro (2007) explored the emergence of authority in an open source  
community and discovered a mix of bureaucratic and democratic elements that constituted a  
meritocracy involving not only merit in a technical sense (programming) but also merit in  
organization-building activities. Power in the community, hence, was gained by technical  
680 savvy, impact of the contribution in terms of use by others, and organizational achievements  
such as coordination (O'Mahony and Ferraro, 2007). The link to motivation is twofold: first,  
developers perceive control over technology, including the ability to set priorities for what  
functions of the technology to develop, as a specific incentive to contribute (Spaeth et al.,

2008). Second, incentives change over time as developers approach the core of a community  
685 and gradually assume their more established roles in the community (Rullani, 2007; Shah,  
2006; Spaeth et al., 2008).

Fourth, an institutional context also provides tangible rewards that in turn influence  
motivation in open source software development. Pay is frequently used for examining  
extrinsic motivation (Roberts et al., 2006). Using a scenario experiment as part of a survey,  
690 Alexy and Leitner (2007) studied the impact of rewards and the norm for payment in a  
community on developers' intentions to contribute to a fictitious open source software  
project. They found that developers' total motivation increased when offered a monetary  
reward upon completion. In addition to the reward, a norm for no payment did not decrease  
total motivation but a norm for payment increased total motivation. Still with the reward,  
695 intrinsic motivation increased (decreased) given a norm for (no) payment (Alexy and Leitner,  
2007).

Hann et al. (2002) showed that the amount of contributions did not increase Apache  
developers' salaries. However, a higher rank within the Apache community significantly  
correlated with higher wages, suggesting that the programmers position (or role) within the  
700 community signals productivity to the employer. This finding confirms the signaling  
argument, first advanced by Lerner and Tirole (2002).

Fifth, information technology including the low-cost access to bandwidth and  
communication and coordination infrastructure enables individual contributions in two ways  
(von Hippel and von Krogh, 2003; Lee and Cole, 2003). First, the infrastructure lowers the  
705 cost of contributing by providing easy access to others' work and low costs of sharing own  
work (von Hippel and von Krogh, 2003). Second, tools such as versioning control,  
repositories, and mailing lists enable direct learning through feedback and coordination work

(Lee and Cole, 2003).

Sixth, research on contribution levels also took a new direction inspired by the work of  
710 Franke and von Hippel (2003) who demonstrated that technical design of open source  
software relates to developer motivation. This work prepared the ground for other  
researchers, such as Baldwin and Clark (2006), to investigate the relationship between design  
and motivation. These authors developed a game-theoretic model that takes into account the  
extent to which a software architecture is “modular”. Their analysis showed that developers  
715 have less incentive to “free-ride,” and thus contribute more to open source software with a  
modular architecture.

Finally, the exposure to a community over time creates opportunities in terms of  
advancement within the social structure of the community (Rullani, 2007; von Krogh et al.,  
2003) and in terms of insights that can lead to more challenging tasks (Shah, 2006).  
720 Adherence to joining scripts requires time for developers to advance to community leadership  
or other central positions (von Krogh et al., 2003; O'Mahony and Ferraro, 2007). A central  
position within the community represents an incentive for developers due to more frequent  
and direct learning opportunities as feedback arrives from many other community members  
as well as due to the power to influence the technical agenda (Spaeth et al., 2008). Rullani  
725 (2007) finds that contributions to community activities can be explained by the developers  
social exposure to the community over time. Similarly, Shah (2006) identified the  
accumulation of knowledge with long-term developers as providing opportunities to take up  
more challenging tasks or tasks that require broader knowledge of the code base, that is  
knowledge about multiple modules and components.

730 The motivational factors and institutional context characteristics in open source software  
development warrants a discussion of crowding theory which studies the mutual adjustments

among motivation factors, particularly between intrinsic and extrinsic motivation. Luthiger and Jungwirth (2007) showed that 42% of the contributions to open source software development emanated from paid working time. Several studies agree that paid contributors amount to approximately 40% of all developers (Lakhani and Wolf, 2005; Hertel et al., 2003; Hars and Ou, 2002). Counter to basic economic theory, introducing monetary incentives may not always increase supply of a public good (see Frey and Jegen, 2001 for a review) and Osterloh and Rota (2007) argue that the profit motive may crowd out sharing of software and other knowledge among developers based on reciprocity. Crowding out of intrinsic motivations to contribute to open source software development, however, has not occurred due to a balance between intrinsic and extrinsic incentives and due to the “pro-social intrinsic motivation of a sufficient number of participants to contribute to the second order public good (enforcing the rules of cooperation) (Osterloh and Rota, 2007: 169).” Indeed, Lakhani and Wolf (2005) found intrinsic and extrinsic motivations to co-exist, and Roberts et al. (2006) detected no crowding out of intrinsic motivation by extrinsic motivation. However, they found that extrinsic motivation did crowd out own-use value motivation (Roberts et al., 2006). Increased reputation and career concerns (here termed status motivation) enhanced developers' intrinsic motivations (Roberts et al., 2006).

A dynamic perspective revealed that while own-use value motivated developers to initially join an open source software development community, fun and enjoyment motives kept them contributing on the long run (Shah, 2006). Shah's findings confirmed a conjecture initially proposed by von Hippel and von Krogh (2003; see also Elster, 1986; Lattemann and Stieglitz, 2005) that the continued contribution to a open source software development requires higher and different levels of incentives, than initially motivating developers to join. Thus, as developers start to work their way into the *social practice* of open source software

development, interestingly, *their motivations change*.

To summarize, beyond distinguishing between mailing list and source code contributions, research in Phase One failed to connect motivation with the nature or quality of contributions. Yet, as scholars acknowledge, the number of messages sent or the amount of code written cannot approximate the quality of the contributions such as software (e.g., Koch and Schneider, 2002; von Krogh et al. 2003). The review of research in Phase Two show how individual motivations are constrained or enabled by institutional context characteristics that lie outside the direct influence of the individual. While the seven institutional context characteristics have been suggested by various authors, little is known about how individuals or firms should balance the effects and increase contributors' motivations. In order to advance our understanding beyond specific institutional contexts provided in Phase Two, theory is needed that explains the emergence of institutions as facilitators of open source software development and as the outcome of the very social practices of open source software. The interplay between institutions and motivation represents one aspect of the social practice that may protect the practice from the “corrupting power of institutions (MacIntyre, 1981: 181)” because the institutional context impacts the individual's motivation and, thus, the values and norms that are being applied.” The emergence of institutions through social practice is yet not well understood. In the remainder of this paper, we turn to this issue. We use Alasdair MacIntyre's theory to advance a set of research questions related to the notion of social practice, that deals with the research gap identified.

#### **4. Motivation, institutions, and social practices**

While research in Phase One and Phase Two built on theories and created and tested models that sought to explain rationale and work effort in open sources software, it neglected important aspect of social practice, ethics, and virtues that guide the work of engineers and

780 software developers. Yet, work ethics, virtues, and social practices influence technical work  
(Martin, 2002; 2000; Latour, 1999) including software engineering (Friedman and Kahn,  
1994). Hence, it is reasonable to believe that such elements also influence why, how, and  
what software developers contribute to open source software development. Alisdair  
MacIntyre's milestone work *After Virtue* (1984) presents a theory that has become  
785 increasingly important in the social sciences for understanding social practice and their  
relationship with institutions, and how they influence motivations of individuals (Knight,  
1998). The theory has also found some use in organization studies (e.g., Beadle and Moore,  
2006), but as noted by Dawson and Bartholomew (2003) applying it is not unproblematic  
since MacIntyre heavily critiques contemporary notions of management and firms as  
790 institutions. While this concern must be taken seriously, we will show that the strength of  
MacIntyre's concept of social practices is that it derives explanatory power from highlighting  
distinct differences between social practices and institutions. With some exceptions (Martin,  
2002) his work has not been widely applied to the theories of innovation and technology  
management, and we are not aware of any application to research on IS and open source  
795 software development. However, as we will argue below, the relationship between the social  
practice of open source software development and institutions such as software firms, is  
critical in understanding motives and contributions to open source software.

In this section we seek to show that individual motivation to create open source software  
is influenced by, and influences, social practice and institutions. We start by outlining  
800 individual motives related to developers technical work and professions applying a  
framework suggested by Mike W. Martin. Martin's work is based on MacIntyre's theory.  
Next, we briefly present three central elements in MacIntyre's theory; the notion of social  
practice, the nature of goods internal and external to these practices, and the relationship

between social practices and institutions. We use these elements to analyze motives that set in  
805 motion the free software foundation by Richard Stallman. Our analysis forms the basis for  
entering Phase Three of research by formulating a set of prominent research questions.

#### **4.1 Motives and the nature of work**

Recall that Deci and Ryan (2000) defined motivation as being moved to do something.  
Reviewing work in Phase One, we demonstrated the prevalent and dominant role of self-  
810 determination theory in explaining developers' contributions to open source software. To  
date, the framework of intrinsic and extrinsic motivation is detached from the nature of work  
including open source software development, and could thus be applied to any type of human  
activity. However, the nature of work and job design relates strongly to motivations to work.  
Currently, one of the most comprehensive reviews of this literature can be found in Morgeson  
815 and Humphrey (2006). Based on a survey of 540 employees in various organizations, these  
authors conclude, for example, that social context has a significant effect on the satisfaction  
of workers. In IS development, Nelson et al. (2000) found that different factors impact on the  
motivation of people contingent on whether they were working in software support or  
-development. Open source software development, like software engineering, consist of work  
820 evolving or being designed provide various levels of satisfaction for developers. For example,  
in a study of Gentoo Linux Monteiro et al. (2006) found that a technical infrastructure (IRC)  
gives rise to communication rituals that move developers to continue their contributions.  
These rituals keep developers “hooked” into development and prevent their defection, and  
hence the disintegration of whole projects based on voluntary contributions.

825 An interpreter of MacIntyre's work, Mike W. Martin (2000; 2002) analyzes motives  
related to people's performance of technical and other professional work. Martin distinguishes  
between three types of motives; craft motives, compensation motives, and motives of moral

concern. Craft motives are desires to achieve expertise and desires to manifest technical skills, theoretical understanding and creativity. When committed practitioners achieve  
830 standard of excellence, they experience satisfaction. When they do not, however, they perceive a sense of failure, shame, or regret. Standards of excellence are often set within a social practice, and we will return to this point later. Martin (2000:22) suggests that craft motives are associated with individuals embracing professional ideals, such as technical standards of excellence, that evoke their interests and talents with sustained challenge and  
835 complexity. For example, in open source software, Haefliger et al. (2008) found that given severe time and resource constraints, developers reuse existing software from other projects in order to be able to create what they consider to be more challenging and “technically sweet” software.

Compensation motives cover desires for social rewards such as money, power, authority,  
840 good reputation, and job stability. The studies of motivation in open source software reviewed in Phase One covers compensation motives, in particular extrinsic elements such as pay, career, reputation, and reciprocity. However, in contrast to the work reviewed, Martin concludes that compensation motives are not purely self-interested. They may relate to desires to support family and friends, philanthropic activities, and obtain resources to help  
845 others. The immediate aim of compensation motives, however, is to reap benefits for oneself. The craft motive in contrast, is aimed outwards towards other people and social practices.

In studying professions, like engineering, Martin (2000) emphasizes that practitioners provide valuable services ranging from health-care, via safe products, to education. Work attains significance by providing opportunities to make ongoing contributions to the well-  
850 being of others and often by placing special responsibilities on practitioners to act in accordance with ethical codes of work. Motives of moral concern include caring for other

people in the sense of active desires to promote their well-being for their own sake. He gives the example of medical doctors or nurses who act in accordance with ethical codes, when they care for their patients well-being. Motives of moral concern also includes a desire to  
855 maintain moral integrity and self-respect. These also include desires to meet professional responsibilities (in Martin's words (2000: 23): "Because they are ones' responsibilities") and maintain personal integrity. In addition, such motives can often be embedded in virtues like honesty, truthfulness, trustworthiness, justice, courage, loyalty, benevolence, etc. These motives cannot be viewed in isolation from a social practice.

860 In comparison with the application of self-determination theory to the problem of motivation in open source software development, these three motives are linked more directly to technical work and social practice, and thus might be an important complement to motivation studies thus far. Martin also acknowledges that his framework relates to self-determination theory, and suggests the three categories of motivation are all a "wellspring of  
865 intrinsic satisfaction" (2000: 24). However, he also suggest that craft motives, because they are directly aimed at the development of a profession, and motives of moral concern, that tend to be directed towards those who are being served by the work of practitioners, are likely to have the greatest overlap with intrinsic motivation. Compensation motives in turn would overlap more strongly with extrinsic motivation, but also here, there are social rewards that  
870 are more deeply rooted in the work itself. For example, developers may achieve reputation or status through work performance rather than pay or career. On the one hand Martin provides us with a more fine grained understanding of intrinsic motivation and an alternative take on extrinsic motivation, on the other hand, as suggested above, his motive categories can be viewed as an alternative to self-determination theory with a clearer link to social practice.  
875 Thus, his categories serve an important analytical purpose in this paper. Moreover, Martin

also points out that all these three motives tend to be tightly interwoven in the technical work of engineers and other professions. Moral concerns and compensation motives may be well aligned. Likewise, examples are given where compensation motives in fact give rise to moral concern motives. While this runs counter to previous theory that warns how extrinsic motivations could crowd out intrinsic motivations in open source software (Osterloh and Rota, 2007) it supports the result in Roberts et al. (2006) that extrinsic and intrinsic motivations can coexist and explain higher participation levels. Using the current framework, an important issue is the extent to which craft and moral concern motives are reinforced by developers being paid and achieving careers. At least, when they are paid developers do not need to worry about their ability to sustain their contribution to the open source software project.

#### **4.2. Social practice in open source software development**

Alasdair MacIntyre's defines a social practice as “any coherent and complex form of socially established cooperative human activity through which goods internal to that form of activity are realized in the course of trying to achieve those standards of excellence which are appropriate to, and partly definitive of, that form of activity, with the result that human powers to achieve excellence, and human conceptions of the ends and goods involved, are systematically extended“ (MacIntyre, 1984: 187). This definition can be applied to a range of professions and disciplines, such as architecture, medicine, journalism, science, and arts, with the precondition that a social practice should have wide and positive effects for humankind. Because of the ubiquitous presence and wide-ranging impact of information systems in all aspects of contemporary human life, we believe software development, like many other areas of engineering and technology (see Latour, 1996; van den Burg and van Borg, 2005), should be considered a social practice in MacIntyre's sense of the concept. However, Walt Scacchi

900 has written on the difference between traditional software engineering and open source software development practices. He notes that “(open source software)..enact teamwork structures and relatively flat, peer-oriented decentralized community forms that reduce/supplant functional organizational forms inherent in traditional (software engineering) techniques that increased bureaucratic tendencies. (Open source software)...avoids reliance on  
905 formal project management techniques and administrative structures that pervade industrial (software engineering) project.” (Scacchi, 2002: 3). Thus, social practice is distinctly related to the technical object. Mackenzie (2005) suggest that the open source software code itself, with modular, functional, and transparent object, gives rise to a social practice outlining rules for behavior. The technical object of the software code itself requires developers to behave in  
910 a specific way when creating and maintaining it, for example modularizing, reusing, keeping to the specification of interfaces (API), documenting, and so forth (see Baldwin and Clark, 2006; Baldwin, 2008). To “be an open source software developer”, thus, means to engage in a social practice, and adhere to its rules of software development, because those enable the creation of an “excellent or high quality product” for its users. At the same time, according to  
915 Scacchi, it also means abandoning other rules related to institutions of software engineering prevalent in industry.

### **4.3. Internal and External Goods**

Individuals might be motivated to act through the provision of incentives or goods. MacIntyre proposes that social practices creates two types of goods that motivate  
920 practitioners: “external and internal goods.” External goods include capital, status, or power, that are privately owned by individuals and organizations. “Internal goods” are partly defined by social practice. They are “public goods” and thus of benefit to the wider community. For example, knowledge is considered an internal good of science. Likewise, software publicly

available under an open source license is the internal good of open source software  
925 development.

MacIntyre's theory emanates from a criticism of the work by Aristotele on ethics and virtues in political leadership. In pursuing internal goods, MacIntyre notes that practitioners achieve excellence of character or virtue. To act in a virtuous manner is to emulate the rules of morality rather than simply abiding by them because one is commanded to do so. A social  
930 practice, therefore, is a “school of virtue,” where practitioners learn aspects of the internal good, such as ethical reasoning, argumentation, criteria for excellence and product quality, rules of communication, and so forth. Justice, courage, and truthfulness and above all love for the social practice, are cultivated through practitioners participation. Practitioners discover and commit to goals that lie beyond their own selfish, short-term, needs and desires. They  
935 also realize that they can only achieve the internal goods, of value to themselves and their social practice, as well as the wider society, when they emulate the standards of excellence already established within the practice. According to MacIntyre, to pursue internal goods (excellence) is therefore synonymous with cultivating virtues by subordinating oneself and one's relations with others to the reasoning which is internal to the social practice. Only by  
940 (being allowed to) participating in a particular practice and emulating the highest standards of excellence achieved within the social practice one can learn, as an apprentice learns, how to exercise sound judgment with regards to what is the best or most preferred example of a good yet achieved. By becoming proficient in such judgment one is able to exceed previously established standards and advance the kind of reasoning that is internal to the practice and  
945 thereby the kind of knowledge on which such reasoning is based. Knight (1998: 12) gives the following example: “To reason better about how to build a house is to advance knowledge about how houses can best be built. The rules of practices are made to be broken, but only by

those who have become so proficient in practical reasoning that they know better than those who previously framed the rules. This is how such practices as building or chess or physics progress.”

Martin (2002) extends MacIntyre's view of social practice and goods to technology. He views technological innovation as a social practice in itself, whose participants include firms, users, or other citizens. Technologies fulfilling basic human needs, such as communication, entertainment, housing, transportation and so on, result from complex cooperative social activities conducted by many individuals and groups working in a coordinated manner and according to the standards of excellence. Open source software development is a social practice that involves designers, coders, testers, but also users and those who are indirectly affected by the technology. The practice builds on and extends knowledge of software development, including developers own experience, algorithms, rules for documenting and testing, designs sheets, product roadmaps, artifacts, software modules, languages, and so forth. Following both Mackenzie and Martin, we propose that when developers create software and share this with others using an open source software repository and communication platforms, they gradually develop, as a by-product, a better understanding of the technology object simultaneously with appropriate rules for behavior. As for technology such standards could be an upper limit for the lines of code needed to solve a mathematical problem, proper ways of documenting code, or testing before releasing a work product. As for behavior, standards could include the need to learn what more experienced developers do before contributing comments or software code, the need to understand an evolving software architecture before contributing, the cycles of testing needed before an official release of the product and so forth. Several of these rules, such as observing and learning before contributing software code and technical comments, was uncovered in an important study by

Kuk (2006). Thus ,to become a “developer of open source software” means to learn the rich knowledge of the social practice and its standards of excellence with regards to technology and behavior. Martin (2002: 556) goes even further by suggesting that “These practices enter centrally into defining a way of life, with technological development entering even more centrally into the ways of life of engineers. (The technologies are parts of or aspects of ways of life..).” Thus, to become a developer means to learn standards of excellence that in many people also create a sense of moral obligation for supporting and further developing the practice.

For example, a form of moral obligations can be found in early release of software code patched, - an integral part of the social practice of open software development (Raymond, 1999). Through early release of prototypical software, dialogue, bug fixing, and technical fine tuning, developers create software that represent “mental models” or designs, unarticulated needs, and aspirations of current and future users (von Hippel, 1998; von Hippel, 2003; von Krogh et al., 2003). Motivations of moral concern lead people to empathy. Empathy is needed on the part of software developers, to achieve standards of excellence within the social practice (Klein and Herskovitz, 2007). Personal commitment goes beyond commitment to the software product per se, and to external goods such as profit, to the social practice that constitutes the activity of open source software development and its broader effects for society. Seen in this light, open source software development as social practice is meaningful work and ways of life, together with particular fascinations, pleasure, challenges, and other inherent benefits. Work offers particular goods to developers who in turn discover the particular goods of a certain kind of life, - as an open source software developer.

The above discussion shows that developers' craft motives, compensation motives, and motives of moral concern are tightly interwoven with what goes on in the social practice of

open source software development. To propose that social practice gives rise to motives, assumes that practitioners can reflect on their own work in relation to the internal good of the practice. However, there is no “objective” or “universally true” point of reference from which to understand technology development. Discussing his concept of *habitus*, Bourdieu (1990: 91) underscored the difficulty of finding such a vantage point: “...there is every reason to think that as soon as he reflects on his practice, the agent loses any chance of expressing the truth of his practice...” Habermas (1991: 163) suggests practitioners develop their skills through self-reflection and thus that practice shapes their character (see also Flyvbjerg, 2001). In other words, once developers promote the social practice, are motivated by it, and reflect on these actions (Calhoun, 1988), they gain insights about their values in relation to that social practice, for example whether or not they fit with their own personal virtues. Therefore, in actions related to social practice, developers may also experience errors on a personal level. A scientist may experience errors in technical aspects of a paper as a sour comment from a reviewer on a research design. In most cases, they would not feel this as a criticism directed towards their character. However, consider motives related to moral concerns in a social practice. If a teacher fails to help a student make a moral judgment and stay away from crime, this can have grave personal consequences for the student as well as for the teacher’s self-image. Likewise, an open source software developer “accused” by members of the social practice for “stealing”, packaging, and selling software from others on a market, breaks with the rules of behavior of the social practice and may distort the self-image (see also O’Mahoney, 2003). Technology is left behind as soon as we leave the practice. For example, photocopier repairers mostly leave their toolkits behind at work and stop being “copy repair people” in their private lives. In contrast, craft motives and moral concerns has the power to determine drives and passions and it stays within a person's character *even as he or she*

1020 *leaves a practice* (see Habermas, 1991). To conclude, a social practice influences a  
motivations through the provision of internal and external goods.

#### 4.4 The (unlikely) origin of institutions

As discussed in section 3 of this paper, researchers have recently recognized that the  
1025 institutional context characteristics of open source software impact on motivations of  
developers. MacIntyre sees institutions as a prerequisite for the organization and sustenance  
of social practices. For example, the social practice of software development is  
institutionalized in a firm that sells licenses to use the software it develops. This firm applies  
behavioral rules of software engineering that schedules work, develop plans, set standards,  
1030 define roles, compartmentalize work (Scacchi, 2002; Kohanski, 2000). While indispensable  
for the social practice to evolve, MacIntyre warns institutions may also constrain or even  
corrupt the social practices and demotivate or demoralize practitioners because institutional  
goals may conflict with the internal goods of the social practice. In particular, this is the case  
when institutions have limited goal seeking behavior aimed at external goods, such as  
1035 excessive profits, at the expense of internal goods that motivate practitioners, such as  
achieving excellence in a craft.

MacIntyre suggests that to act effectively in society it is necessary to adopt society's  
ethos or presuppositions. In a controversial part of *After Virtue*, he suggests the most  
effective kinds of people are those who characteristically seek to affect the actions of others  
1040 by manipulating them. Thus, there are important, symbiotic relationships but also strong  
potential tensions between social practices and institutions. Whereas institutions provide  
incentives for practitioners that satisfy their compensation motives, they may fail to support  
or even corrupt social practices by overshadowing or conflicting with craft motives and

motives of moral concern. For example, a social practices involves taking care of other  
1045 practitioners, such as the care experts show for novices. Pekka Himanen (2001) argues that  
caring is a prevalent feature of open source software development practice. Open source  
software developers want to expand their social practice by ensuring that newcomers can  
learn the technical and behavioral rules. They need receive attention by experienced members  
who show them how to effectively contribute to the product. As found in studies on  
1050 knowledge sharing in organizations, caring behavior normally falls outside the scope of  
organizational incentive systems, such as money or careers. Yet, caring behavior can be  
instrumental for newcomers in learning the complex technical knowledge needed for  
software development from experienced practitioners (von Krogh, 1998; Zarraga and  
Bonache, 2005).

1055 Much of MacIntyre's critique of institutions aims at management (MacIntyre, 1984). In  
his view, managers justify their power over others and the monetary compensation by  
implementing techniques and systems for social change. However, because of the complexity  
of organizations techniques and systems seldom lead to predictable outcomes, and hence the  
basis for justification must be false. This critique echoes Robey and Markus's (1984) much  
1060 cited analysis on the unpredictable outcomes of the design of management information  
systems. The implementation of planned social change can do more harm than good in  
helping the social practice's capacity for producing internal goods. Moreover, managers are  
often motivated by compensation motives and they want to create and appropriate excessive  
external goods, such as profits. This in turn can undermine practitioners' craft-based and  
1065 moral concern motives, and ultimately destroy the social fabric of the practice.

MacIntyre (1984) also raises doubts that management can be considered a social practice  
because this would presume attention to the creation of internal goods and well-being of

humanity. Managers' concern are with techniques and systems transforming raw materials into products, unskilled labor into skilled labor, and investment into profit (p. 30). Managers, 1070 decide and act to achieve a desired end state through social change, but are blind to other concerns such as the wider effect of their actions on humanity. Thus, he concludes they are not able to engage in moral debates about their own actions.

MacIntyre's harsh critique neglects the empirical fact that managers often are concerned about the consequences of their actions and repeatedly engage in a wider discourse with 1075 society about the nature of their decisions and activities and the purpose of the institutions they run (see Dawson and Bartholomew, 2003). Nonetheless, his analysis fits strikingly well with early accounts of what motivated open source software movement. Consider that in the 1960's and 1970's much of software development was carried out in academic and corporate research laboratories by scientists and engineers. These individuals found it a normal part of 1080 their social practice to freely give and exchange software they had written, in order to modify and build upon each other's software both individually and collectively, and to freely give out their modifications in turn. The virtue of active and intense sharing was considered important for learning, efficiency in the development of code, better bug-free products, and overall, the development of the software engineering profession. In 1969, the U.S. Defense Advanced 1085 research Project Agency (ARPA) established ARPANET, which eventually grew to link hundreds of universities, defense contractors, and research laboratories. The network was later succeeded by the Internet, which enabled software developers to share software code, knowledge, and information cheaply and easily on a worldwide scale and thereby also quickly build and establish a new profession.

1090 The virtue of sharing work in the social practice of software development was very strong amongst a group of developers at MIT's Artificial Intelligence Laboratory in the 1960's

and 1970's (Levy, 1984). The first conflicts between the institutional goals and the virtues of social practice can be traced back to the 1980'. At this time, MIT decided to license some of the code created by this group to a commercial firm. In accordance with its commercial interests, the firm restricted the access of the source code of that software to the MIT developers who had originally participated developing it, which in turn created frustration and irritation amongst the MIT developers. This incidence illustrates well the kind of conflicts MacIntyre makes us aware. Whereas software developers at the time considered virtues such as openness, learning, and knowledge sharing a prerequisite for the social practice's creation of internal goods including product excellence, professional development, and benefits for others, they felt institutional concerns for external goods constrained and eventually would harm the practice. In the presence of such conflicts, MacIntyre's theory would predict practitioners, whose motives are influenced by their social practice, would begin to act collectively. In effect, new institutions would emerge in support of social practices that preserve virtues and internal goods. The events that transpired at MIT lends support to this theory. Richard Stallman who at the time was a programmer at MIT's Artificial Intelligence Laboratory, was distressed by the institutional pressure for restricting access to source code and selling software through licenses. He believed it would harm the software engineering profession, and in fact not aid humanity in their fast growing needs for ever better technologies. Stallman viewed these practices as "morally wrong" impingements upon the rights of software users to freely learn and create. In his own words, at the time, faced with the collapse of his community's social practice, he was faced with a moral choice (Stallman, 2002):

*"With my community gone, to continue as before was impossible. Instead I faced a stark moral choice. The easy choice was to join the proprietary software world, signing non-*

*disclosure agreements and promising not to help my fellow hackers. Most likely I would also be developing software that was released under a non-disclosure agreements, thus adding to the pressure on other people to betray their fellows too. I could have made money this way, and perhaps amused myself writing code. But I knew at the end of my career, I would look*

1120 *back on years of building walls to divide people, and I feel I had spent my life making the world a worse place...So I looked for a way a programmer could do something for the good. I asked myself, was there a program or programs I could write so as to make the community possible again?..”*

Stallman's reply to this question was to create an institutional alternative under the name

1125 of the Free Software Foundation. The purpose of this foundation was to preserve free access for all to software developed by those who shared the virtues of the practice. The legal mechanisms he developed to support this idea, was the GNU General Public License that can be affixed to a piece of software by a developer, and that guarantee a number of rights to all future developers and users. Basic rights include the right to download for free, study, and

1130 modify the source code, and the right to redistribute to others modified or unmodified versions of the software for free. Stallman firmly believed that this license and the new institution of the Free Software Foundation would support the social practice of software development and eventually help create excellent products for benefit to society. Because of his skills and the need to ensure further openness, Stallman first started work on the GNU

1135 operating system, that emulated the design of Unix. In the years to follow, different licenses and organizations emerged over such as the Open Source Software Movement, that presented varieties of this basic idea (Perens, 1999).

As this analysis shows, craft motives and moral concerns alongside compensation motives set in motion free- and open source software development. Stallman was concerned

1140 about the advancement of craft in social practice of software development, as well as the  
moral choices that new institutional goals posed on software developers. His institutional  
alternative was created to preserve the social practice's ability to create internal goods  
alongside those external goods pursued by the software industry. Later observers refer to  
Stallman's institutional alternative as a “free software” ideology that serves to motivate  
1145 developers to join and contribute to open source software (Stewart et al., 2006). However, the  
origin of collective action and subsequently, the ideology, was the social practice of software  
development that shaped craft motives and moral concerns and prompted Stallman to create  
an institutional alternative. To paraphrase an insight from MacIntyre's theory, anything that  
does not promote the common good, such as the appropriation of software code collectively  
1150 developed, may not be properly regarded as social practice, such as software development.

A social practice is the basis for making decisions regarding which virtues are called for  
in particular circumstances and the best way to enact those virtues (Noel, 1999; Fowers,  
2003; Rämö, 2004). Dunne (1993) refers to “ethical knowledge” that directs “ethical action.”  
Aristotle's discussion of the “good” refers to man's practice “of his soul's faculties in  
1155 conformity with excellence or virtue” (Aristotle, 1926: 33). The good becomes a  
metaphysical goal such as truth, justice, and beauty towards which people strive by adjusting  
their lives and actions. However, in a modern world such goods may be contested by people  
who pursue different goals. The very standards that define what is good may be subject to  
different interests and, therefore, judgment itself will be judged as more or less virtuous (be it  
1160 of good or bad, right or wrong). MacIntyre argues we should rather understand the common  
good as internal to a social practice, as a goal to be achieved by its practitioners (Knight,  
1998). For example, a common good can be a practitioner's pursuit of excellence in software  
development, and it is against this backdrop we might understand Stallman's moves. Thus, it

is important to note that the new institution of free software spring out of craft motives and  
1165 moral concerns of practitioners, that in turn are shaped by social practice of software  
development. It is the concern for product quality, work, and the wider implications of free  
and open source software that gives rise to new institutions. *The new institutional context  
does not originate in an overarching ideology that motivate people to act.* Therefore, our  
analysis is sharply different from alternative accounts of open source software as “Rebel  
1170 Code” or social movements that battle the establishment of the software industry (e.g.,  
Moody, 2001; Stewart et al. 2006; see also the discussion of ideology and collective action in  
von Hippel and von Krogh, 2003). Finally, consistent with this theory, if the emerging  
institutions of free and open source software fail to support the social practices of software  
development that create internal and external goods, and enable developers' craft motivations  
1175 and motivations of moral concern (alongside compensation motives), new institutional  
alternatives may emerge.

### **5. Entering Phase Three of research on motivation**

We are now in a position to formulate a set of questions for information systems  
research. What we term “Phase Three” of research on motivation assembles a set of research  
1180 areas with corresponding research questions. This phase is characterized by an important  
assumption, namely that contributors are primarily motivated to preserve and develop their  
social practice. In other words, what we want to do in this section is to make explicit that  
there is a link between individual motives to contribute and the social context in which  
contributions are made. From the preceding sections of the paper it has become clear that the  
1185 social aspects in framing motivations, or rather the links and tensions between the individual  
motivation and social context in open source software development is underdeveloped and  
often completely missing. This phase of research on open source software development is

characterized by a systematic categorization of individual motives and their social context.

We propose an organizing framework of motivations in open source software development

1190 that accounts for the individual motivations as presented by Martin in the preceding section  
and the social aspects as presented by MacIntyre. The resulting framework is outlined in the  
form of a matrix (labeled Motivation-Context matrix in Table 2 below). The columns  
represent the individual motivations: craft motives, compensation motives (including non-  
financial compensation), and moral concerns. These motivations thus go beyond the

1195 traditional intrinsic-extrinsic dichotomy and provide a finer grained categorization in the area  
of intrinsic motivation. The rows of the organizing framework represent the three key  
conceptual building blocks of MacIntyre's theory, that is the social practice, internal and  
external goods, and institutions. The organizing framework allows for the categorization of  
research questions of the type: “what is the relationship between craft motives and social

1200 practices”? Highlighting the social aspects of motivation has two main consequences. First,  
people's motivations are tightly interwoven with their social practice, and since social  
practices and institutions mutually support or constrain each other, motives must be  
understood by taking into account the institutional context in which social practices evolve.  
From the perspective of the individual, we claim that individual motivation is both enabled

1205 and constrained by social practices, goods produced and institutions. Second, the social  
context is not only the antecedent to individual motivation but also vice versa. In particular,  
we discuss this in the case of the emergence of institutions below. We do not claim that the  
framework has any explanatory power, rather it helps us to classify and identify new,  
challenging and hopefully fruitful avenues for future research. The section is structured along

1210 the rows of the framework, i.e. starting with the Social practice in open source software  
development, continuing with Internal and External Goods, and finishing with Institutions.

		Individual		
		Craft motives	Compensation motives	Moral concerns
Collective	Social Practice			
	Goods			
	Institutions			

Table 2. Outline of the Motivation-Context Matrix

## 5.1 Social Practice

Developers care about their social practice (OSS development) and would like to adhere to it and improve it. Over time, the social practice generates and alters motivation (Shah, 1215 2006). However, multiple motives play a role (Roberts et al., 2006). For each motive, information systems researchers should ask how the social practice generates motivation and how the motivation impacts the social practice.

### Craft motives

The desire to achieve expertise, to demonstrate technical skills, theoretical 1220 understanding and creativity are generally individual motivations to participate in the social practice. Vice versa, participation in the social practice shapes the craft motive by exposing the individual to the intricacies of the trade. A number of research questions arise, e.g.: What dimensions of virtue form the basis of the social practice of open source software development? What constitutes “beautiful code”? Why is it commonly accepted that generic 1225 functions are made modular and reusable (making things more difficult for authors in the first place)? Is producing producing excellent software code sufficient to adhere to the standards of virtue within the social practice? Anecdotal evidence suggests otherwise: In 2000, one of the hackers in the Freenet project rewrote large portions of the core code privately over a couple of weeks without community involvement. When he deemed it finished, he ripped out 1230 the existing code base and replaced it with his code. While being seen as technically high

quality source code, the community at large was not happy with the lack of transparency and community involvement. Not having being involved in the development process, they found it hard to grasp how the new code worked and criticized the development process. This developer had clearly not followed the standards of what was seen as excellent craft within  
1235 the social practice despite having single-handedly developed a technologically excellent solution.

These issues have hardly been explored so far and the concept of social practice highlights another important question: Participants strive for excellence within their social practice, however, it is not clear that this translates into technically superior output, that is the  
1240 internal goods. If, for example, highly philosophical and abstract discussions are perceived as more virtuous than sitting down and producing “beautiful” code, than craft motives may not necessarily lead to the efficient production of technically supreme software.

### ***Compensation motives***

Are compensation motives and craft values reinforcing or reducing each other? For  
1245 example, developers with a high own-use value (solving their problems) might be interested more in fixing the one specific bug or adding the one feature they crave rather than strive to produce “beautiful” code. Also, in combination with institutional factors, (financial) compensation might reduce the level of participation in the social practice. On the other hand, exposure to the social practice over time might change the perception of compensation values  
1250 for internal or external goods.

### ***Moral concerns***

Caring for others, moral integrity, self-respect, and adhering to work ethics are issues of moral concerns. In the context of open source, researchers will need to identify the

dimensions of moral concerns within the social practice. Can we, as is usually done, simply  
1255 assume that “free software” and “open source” development are equivalent and cluster them  
together in research, especially when examining motivations? While both factions might  
share standards and values on craft, their moral considerations deviate widely. As the love for  
the social practice impacts on individual motivations and vice versa, this implies that there  
might not be only one social practice (“open source”) but that “free software” might represent  
1260 a related yet separate social practice. Designing institutions that cater to these social practices  
would require different approaches then. Another potential area of research is the interaction  
between craft issues and moral motivation. Another question has not been answered yet: do  
moral obligations lead to common and accepted behavior and routines? For example, do  
moral obligations lead developers to create reusable components, even if it means an  
1265 increased burden of maintenance to them? (“This is just how you have to do it in open  
source.”, “I have the obligation to fix bugs in my code, even if it bothers only other users of it  
and not myself.”). Hitherto, morality in open source development has often been seen as an  
existing yet unimportant aspect. However, if moral considerations do shape the practices that  
help to create superior software, they are much more worthwhile to examine.

	<b>Craft motives</b>	<b>Compensation motives</b>	<b>Moral concerns</b>
<b>Social Practice</b>	<ul style="list-style-type: none"> <li>• What dimensions of virtue and standards of excellence are inherent in the social practice?</li> <li>• How do newcomers learn the standards of excellence?</li> <li>• How do the social practice's standards of excellence change over time?</li> <li>• What is the relationship between excellence in the social practice and the technical quality of the resulting internal goods?</li> <li>• Are standards of excellence global to the social practice? If not what are the boundaries (regional, programming language, training,...)?</li> <li>• How does the social practice generate and alter craft motives?</li> </ul>	<ul style="list-style-type: none"> <li>• What effect has financial compensation on the participation in the social practice?</li> <li>• Does compensation affect norms and values of the social practice?</li> <li>• Are compensation motives and craft values reinforcing or reducing each other?</li> <li>• How are internal valued in comparison to external goods within the social practice?</li> <li>• Does exposure to the social practice change the perception of compensation over time?</li> </ul>	<ul style="list-style-type: none"> <li>• What are the dimensions of moral concerns within the social practice?</li> <li>• Do moral considerations result from or precede the participation in a social practice?</li> <li>• How do moral motivations impact on other motivations?</li> <li>• What role do moral considerations play in the attracting and socialization of new practice members?</li> <li>• How are conflicts of interest between members of the social practice and other stakeholders resolved?</li> <li>• Are moral obligations a strong factor impacting on behavior within the social practice? (i.e., this is how things are done)</li> </ul>

1270

*Table 3. Motivation-Context Matrix: Social practice*

## 5.2 Internal and External Goods

In our reading of MacIntyre, the constructs of internal and external goods constitute a conduit, connecting the social practice and institutions by representing output and direct consequences of the latter constructs. The goods here are seen as collective goods in

1275

production and not necessarily in consumption (use), which means that an external good such as status is indeed a collective good in production, although it may be privately appropriated (private in consumption). At the same time, open source code can be both collective in production as well as consumption. Future research in information systems may ask how the goods relate to different motives and how the motives impact on characteristics and, in

1280

particular, the quality of the goods produced.

### *Craft motives*

Concerning craft motives, we suggest three future research questions. First, how and why does the type of good produced influence peer reputation ? As craft motives relate to the desire to be viewed as an expert, it ought to be important to assess how and what contributions  
1285 (internal goods) influence peer reputation, given that peer reputation is a strong motivating factor at play in open source software development (Lakhani and Wolf, 2005). It would be equally interesting to make the link to external goods, e.g. to investigate how status and career influences peer reputation. Second, as argued earlier, the individual motivations impacts on the goods produced (in essence the traditional reasoning linking motivation to  
1290 output). Thus, we suggest the question: how and why does the quest for peer reputation motivate individuals to make certain kinds of contributions? Third, we believe craft motives very much are related to the quality of goods and thus we propose: what is the relationship between technical quality and other elements of quality of internal goods?

### *Compensation motives*

1295 An important aspect of pay (the quintessential compensation motive) is how the input, process or output can be monitored. We suggest the following research question: how does the monitoring of input, process and output depend on the characteristic of the internal good produced? Second, and similarly, we can rephrase the question into exploring use-aspects. In this case, we can ask: how does the use value depend on the characteristic and quality of the  
1300 internal good produced? Third, in a very simple model, compensation can occur instantaneous or in some foreseeable future; e.g. investments in human capital and learning can lead to future compensation. If the individual is focused on his or her individual learning then there is potential room for conflict with other interests. Thus, we ask: what are the

potential conflicts between the motivation for individual learning and the quality, or standard  
1305 of quality, of the goods produced ?

### ***Moral concerns***

Moral concerns are different as they deal with the well-being of others and how these  
concerns motivate individuals to act or not to act. In relation to internal and external goods, it  
gives rise to a number of future research questions. First, the presence of moral concerns in  
1310 open source software development admits that my motivation to contribute is partly shaped  
by how my resulting actions impact the well-being of others, thus it ought to include aspects  
of helping behavior with no direct and personal payoff. This kind of helping behavior, in turn,  
can result in the production of internal and external goods. Thus, we suggest the following  
research question: How and why is the type of internal good produced through helping other  
1315 individuals different from other contributions? Second, we may explore how certain goods  
constrain or enable helping behavior. The answer may lead to insights as to why certain types  
of goods elicit or allay moral concerns. Third, moral concerns not only deal with helping  
behavior. One quite obvious avenue to walk is to explore the tensions and possible conflicts  
between these caring motives and external goods such as status, power or profit, goods which  
1320 are collective. Thus, we propose the following research question: what are the potential  
conflicts between moral concerns and the production and consumption of external goods?

	<b>Craft motives</b>	<b>Compensation motives</b>	<b>Moral concerns</b>
<b>Internal and external goods</b>	<ul style="list-style-type: none"> <li>• How and why does the type of good produced influence peer reputation?</li> <li>• How and why does the quest for peer reputation motivate individuals to make certain kinds of contributions?</li> <li>• What is the relationship between technical quality and other elements of quality of internal goods?</li> </ul>	<ul style="list-style-type: none"> <li>• How does the use value depend on the characteristic and quality of the internal good produced?</li> <li>• How does the monitoring of input, process and output depend on the characteristic of the internal good produced?</li> <li>• What are the potential conflicts between the motivation for individual learning and the quality, or standard of quality, of the goods produced?</li> </ul>	<ul style="list-style-type: none"> <li>• How and why is the type of internal goods produced through helping other individuals different from other contributions?</li> <li>• How and why do certain types of goods elicit or allay moral concerns?</li> <li>• What are the potential conflicts between moral concerns and the production and consumption of external goods?</li> </ul>

*Table 4. Motivation-Context Matrix: Internal and external goods*

### 5.3 Institutions

1325           Institutions form an important part of the context that impacts individual motivations. OSS communities represent a novel but relevant form of organization because they are not associated with one employer or workplace, because they integrate individual contributions into a common pool, and because they assume ownership of their output (O'Mahony and Ferraro, 2007). The emergence of communities as institutions give rise to a number of

1330 research questions that relate the individual motivations with institutions: on the one hand about the emergence of institutions and, on the other hand, about the relationships between institutions including communities and firms. Lastly, some open questions are tied to the interplay between institutions and moral concerns.

#### *Craft motives*

1335           The craft motivation to contribute and to share the internal goods of the social practice may lead to the foundation of institutions such as community organization in OSS. This process is not well understood yet. O'Mahony and Ferraro's (2007) work on the emergence of governance represents a step to what we consider Phase Three. Governance constitutes a

central problem of community organization and the authors show that the allocation of  
1340 authority is deeply rooted in the values of the social practice of open source software  
development: community leaders receive limited authority over technical matters and must  
defer to the majority of the community members if needed (O'Mahony and Ferraro, 2007).  
When and how do individual motives give rise to institutions? In the collective action  
literature, institutional change has been described as the result of political struggles over the  
1345 framing of issues and values (see Hargrave and Van de Ven, 2006). The perspective adopted  
here suggests future research looks into the distinction between framing as a political process  
of collective action and craft motives as a force in institutional change rooted directly in the  
social practice. The two approaches may account for different models (or forms, or stages) of  
collective action.

#### 1350 *Compensation motives*

Institutions compete for talent and, within institutions, individuals compete for resources  
that institutions provide, such as status, pay, and others. Motives are directly connected to the  
way institutions structure the work for employees, deal with ethical issues, and compensate  
individual achievements and effort (Alexy and Leitner, 2007). Communities may emerge  
1355 where firms failed at balancing individual motives, for example by providing hackers with  
the necessary freedoms associated with Free software. Vice versa, firms may prevail where  
compensation motives prove compatible with craft motives and moral concerns. Institutional  
change, we argue, closely ties in with the motivations prevalent and latent in the social  
practice: the research questions probe into a perspective on mutually dependent institutions  
1360 and motivations.

***Moral concerns***

Certain institutional characteristics may be more receptive to moral concerns: possibly consensus-oriented decision making or democratic legitimacy. But beyond compatibility, which characteristics could actively engage individuals with a strong motivations rooted in moral concerns? How are institutions established and sustained based on moral concerns? If moral concerns indeed account for significant and important individual contributions to an institution and social practice, how does competition between institutions effect the sensitivity to individuals' moral concerns?

	<b>Craft motives</b>	<b>Compensation motives</b>	<b>Moral concerns</b>
<b>Institutions</b>	<ul style="list-style-type: none"> <li>• When do craft motives warrant the foundation of new institutions?</li> <li>• How can institutions support rather than corrupt craft motives?</li> <li>• How can craft motives be distinguished from framing strategies to determine the direction of institutional change?</li> <li>• Which institutional characteristics, such as governance structures, are best compatible with craft motives?</li> </ul>	<ul style="list-style-type: none"> <li>• How can compensation structures be designed to remain compatible with other motives?</li> <li>• How can institutions balance financial and non-financial compensations?</li> <li>• How can competing institutions maintain long-term relationships with volunteers?</li> <li>• How can firms and communities interact given radically different compensation structures for members?</li> </ul>	<ul style="list-style-type: none"> <li>• Which institutional mechanisms foster the integration and respect towards the moral concerns by their members?</li> <li>• What is the role of moral concern in establishing and sustaining institutions?</li> <li>• How can conflicts between moral concerns and institutions be resolved?</li> <li>• How do competing institutions compare when catering to moral concerns?</li> </ul>

*Table 5. Motivation-Context matrix: Institutions*

Phase Three of research on motivation outlines an agenda that connects the social practice and the institutions of information systems development with the individual motivations that, on the one hand, derive from the social practice and, on the other hand, generate institutions that support and constrain the social practice. The matrix of research questions that spans between types of motivations and social practice, goods, and institutions may inspire future research to tackle fundamental problems of collective action, agency, and institutional change as functions of individual motivation embedded in the nature of work.

## 6. Conclusion

The objective of this paper was two-fold, first to provide a state of the art review of what is known about motivations to contribute to open source software development, and second to  
1380 reinvigorate the research field by providing a framework with a set of new research questions, combining motivations with a social practice perspective of open source software development. By doing so we open up for a broader array of motivations than previously laid out in studies rooted in economic theory.

Our review of motivation studies in open source software development segmented  
1385 previous work in three phases whereby Phase Three includes our suggestions for future research in information systems development. Phase One summarizes studies that shed light on the types of motivation factors (intrinsic, extrinsic, and internalized extrinsic) that incent software developers to contribute their time and effort to the development of open source software. The wealth of motivational factors identified in these studies warranted further  
1390 research into the effects connecting the motivation factors with the institutional contexts surrounding open source software development. Phase Two, as we termed this segment of research on the motivation of open source software developers, reviewed the literature that connects motivation and institutions in open source software development and included studies that uncovered the complex crowding effects between motivational factors, e.g., how  
1395 institutions such as firms impacts on motivation among open source software developers. Phase Three integrates the social practice perspective of open source software development, which relates the social and institutional settings in which open source software is embedded, to the goods and motivations connected to the social practice. We argue that by treating open source software development as a social practice we can study the consequences of the  
1400 interdependencies between individual motivations and these social and institutional contexts,

such as the nature of work.

We also argue that open source software does not exist in competition to proprietary commercial software development, but rather, as a complement to secure the social practices upon which standards of excellence in software development can be nurtured and developed.

1405 Open source software is, thus, an inevitable and additional practice where professional craft and moral concern motives guide practitioners in learning and innovating beyond what is possible within other institutions. The internet enabled an inexpensive evolution of such practices where thousands of software developers can interact, learn and formulate ever higher standards of excellence.

1410 The history of Free and Open Source Software development demonstrates how the birth of the social practice of open source software development responded to institutional pressure in commercial software production by creating its own institutions to protect the virtues associated with the social practice. The perspective advocated by MacIntyre elucidates the role of individual motivations in carrying forward a social practice that in turn complements  
1415 and gives rise to new institutions. We argue that an understanding of motivation in conjunction with the context of the social practice could lead to fruitful research on institutional change. Other perspectives on the emergence of institutions, such as collective action (see Hargrave and Van de Ven, 2006, for a review), argue convincingly that the struggle between fractions and social movements brings about change and institutional  
1420 innovation. However, framing and political behavior hardly grasp the diversity of motivations and the notions of quality that drive one social practice to generate and maintain its standards against odds from within and from without. The views are compatible in that a social practice can become a social movement, as can be argued for open source software development (von Hippel and von Krogh, 2003; Hertel et al., 2003), and may create institutions with the

1425 primary goal of protecting work against appropriation by others (O'Mahony, 2003). The  
views are complementary in that collective action emphasizes the “struggle over meanings of  
new issues and technologies (Hargrave and Van de Ven, 2006: 884)” across social  
movements, whereas social practice focuses on the quest for better products, deeper  
knowledge, and improved collaboration within and across social movements, that is, on the  
1430 internal goods that characterize social practice. The research questions that can best be  
tackled with this perspective are subsumed under Phase Three. To conclude, we suggest  
research into the development of information systems along a matrix of three theoretical  
building blocks and three types of individual motivation: The social practice, internal and  
external goods, and institutions in combination with craft motives, compensation motives,  
1435 and moral concerns. Our review proposes to understand motivations in open source software  
development as an evolution of open source software development caused by the intricate  
interplay between individuals, social practices and institutions.

## References

- 1440 Alexy, O. & Leitner, M. (2007), 'Norms, Rewards, and their Effect on the Motivation of Open Source Software Developers', available online from <http://opensource.mit.edu>, Working Paper.
- 1440 Bagozzi, R. P. & Dholakia, U. M. (2006), 'Open Source Software User Communities: A Study of Participation in Linux User Groups', *Management Science* **52**(7), 1099-1115.
- Baldwin, C. Y. (2008), 'Where do transactions come from? Modularity, transactions, and the boundaries of firms', *Industrial and Corporate Change* **17**(1), 155-195.
- Baldwin, C. Y. & Clark, K. B. (2006), 'The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?', *Management Science* **52**(7), 1116-1127.
- Beadle, R. & Moore, G. (2006), 'MacIntyre on virtue and organization', *Organization Studies* **27**(3), 323-340.
- 1445 Becker, G. S. (1974), 'A Theory of Social Interaction', *Journal of Political Economy* **82**(6), 489-520.
- 1445 Benabou, R. & Tirole, J. (2003), 'Intrinsic and Extrinsic Motivation', *Review of Economic Studies* **70**(3), 489-520.
- Benkler, Y. (2002), 'Coase's Penguin, or Linux and the Nature of the Firm', *Yale Law Journal* **112**(3), 369-447.
- Bergquist, M. & Ljungberg, J. (2001), 'The Power of Gifts: Organising Social Relationships in Open Source Communities', *Information Systems Journal* **11**(4), 305-320.
- Bessen, J. (2002), 'What Good is Free Software', in Robert W. Hahn, ed., 'Government Policy Toward Open Source Software', Brookings Institution Press, Washington, DC.
- 1450 Bitzer, J.; Schrettl, W. & Schröder, P. J. (2007), 'Intrinsic Motivation in Open Source Software Development', *Journal of Comparative Economics* **35**(1), 160-169.
- 1450 Bonaccorsi, A. & Rossi, C. (2006), '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), 40-64.
- Bourdieu, P. (1977), *Outline of a theory of practice*, Cambridge University Press.
- Calhoun, C. (1988), 'The radicalism of tradition: Community strength or venerable disguise and borrowed language?', *American Journal of Sociology* **88**(5), 886-924.
- Csikszentmihályi, M. (1975), *Beyond Boredom and Anxiety*, Jossey-Bass, San Francisco, CA.
- 1455 Csikszentmihályi, M. (1990), *Flow: The Psychology of Optimal Experience*, Harper and Row, New York.
- 1455 Dahlander, L. & Wallin, M. W. (2006), 'A man on the inside: Unlocking communities as complementary assets', *Research Policy* **35**(8), 1243-1259.
- David, P. A.; Waterman, A. & Arora, S. (2003), 'FLOSS-US: The Free/Libre/Open Source Software Survey for 2003', Stanford University.
- Dawson, D. & Bartholomew, C. (2003), 'Virtues, managers and business people: Finding a place for MacIntyre in a business context', *Journal of Business Ethics* **48**(2), 127-138.
- Deci, E. L. & Ryan, R. M. (1985), *Intrinsic Motivation and Self-Determination in Human*

- Behavior*, Plenum, New York.
- 1460 Deci, E. L. & Ryan, R. M. (1987), 'The Support of Autonomy and the Control of Behavior',  
*Journal of Personality and Social Psychology* **53**(6), 1024-1037.
- Elster, J. (1986), *An Introduction to Karl Marx*, Cambridge University Press.
- Feller, J. & Fitzgerald, B. (2002), *Understanding Open Source Software Development*,  
Addison-Wesley, London, UK.
- Fershtman, C. & Gandal, N. (2004), 'The Determinants of Output Per Contributor in Open  
Source Projects: An Empirical Examination', Available at SSRN:  
1465 <http://ssrn.com/abstract=515282>, CEPR Discussion Paper No. 4329.
- Flyvbjerg, B. (2001), *Making social science matter. Why social inquiry fails and how it can  
1465 succeed again*, Cambridge University Press.
- 1465 Fortes, M. (1969), *Kinship and the social order: the legacy of Lewis Henry Morgan*, Aldine,  
Chicago.
- Fowers, B. J. (2003), 'Reason and Human Finitude', *American Behavioral Scientist* **47**(4),  
415-426.
- Franke, N. & von Hippel, E. (2003), 'Satisfying Heterogeneous User Needs via Innovation  
Toolkits: The Case of Apache Security Software', *Research Policy* **32**(7), 1199-1215.
- Frey, B. S. (1997), *Not Just for the Money: An Economic Theory of Personal Motivation*,  
Edward Elgar.
- Frey, B. & Jegen, R. (2001), 'Motivation crowding theory', *Journal of Economic Surveys* **15**,  
1470 589-610.
- 1470 Frey, B. S. & Meier, S. (2004), 'Pro-social behavior in a natural setting', *Journal of Economic  
Behavior and Organization* **54**, 65-88.
- Friedman, B. & Kahn, P. (2004), 'Educating computer scientists: Linking the social and the  
technical', *Communications of the ACM* **37**(1), 64-70.
- Ghosh, R. A. (2005), *Understanding Free Software Developers: Findings from the FLOSS  
Study in 'Perspectives on Free and Open Source Software'*, MIT Press.
- Ghosh, R. A.; Glott, R.; Krieger, B. & Robles, G. (2002), 'Free/Libre and Open Source  
Software: Survey and Study (FLOSS)', <http://www.infonomics.nl/FLOSS/report/>.
- Haefliger, S.; von Krogh, G. & Spaeth, S. (2008), 'Code Reuse in Open Source Software  
1475 Development', *Management Science* **54**(1), 180-193.
- 1475 Hann, I.; Roberts, J.; Slaughter, S. A. & Fielding, R. (2002), 'Economic Incentives for  
Participating in Open Source Software Project', ICIS Conference Proceedings.
- Harsanyi, J. C. (1978), 'Bayesian Decision Theory and Utilitarian Ethics', *American  
Economic Review* **68**, 223-228.
- Hars, A. & Ou, S. (2002), 'Working for free? Motivations for participating in open-source  
projects', *International Journal of Electronic Commerce* **6**, 25-39.
- Haruvy, E.; Prasad, A. & Sethi, S. (2003), 'Harvesting Altruism in Open-Source Software  
Development', *Journal of Optimization Theory and Applications* **118**(2), 381-416.
- Heider, F. (1958), *The Psychology of Interpersonal Relations*, Wiley, New York.
- 1480 Hemetsberger, A. (2004), 'When Consumers Produce on the Internet: The Relationship  
between Cognitive-affective, Socially-based, and Behavioral Involvement of Prosumers',  
available from: <http://opensource.mit.edu/papers/hemetsberger1.pdf>, Working paper.

- Hertel, G. (2002), Management virtueller Teams auf der Basis sozialpsychologischer Modelle [management of virtual teams based on social psychology models], in E.H. Witte, ed., 'Sozialpsychologie Wirtschaftlicher Prozesse', Pabst Publishers, Lengerich, Germany, pp. 172-202.
- Hertel, G.; Niedner, S. & Herrmann, S. (2003), 'Motivation of software developers in open source projects: An internet-based survey of contributors to the Linux Kernel', *Research Policy* **32**(7), 1159-1177.
- Himanen, P. (2001), *The Hacker Ethic and the Spirit of the Information Age*, Secker & Warburg, London, UK.
- 1485 von Hippel, E. (1998), 'Economics of product development by users: The impact of "sticky" local information', *Management Science* **44**(5), 629-644.
- 1485 Klandermans, B. (1997), *The Social Psychology of Protest*, Basil Blackwell, Oxford.
- Klein, E. & Herskovitz, P. (2007), 'Philosophy of science underpinnings of prototype validation', *Information Systems Journal* **17**(1), 111-132.
- Knight, K. Knight, K., ed. (1998), *The Macintyre Reader*, University of Notre Dame Press.
- Koch, S. & Schneider, G. (2002), 'Effort, Cooperation and Coordination in an Open Source Software Project: GNOME', *Information Systems Journal* **12**(1), 27-42.
- 1490 Kohanski, D. (2000), *Moths in the Machine: The Power and Perils of Programming*, St. Martin's Press, New York.
- 1490 Krebs, D. L. (1970), 'Altruism - Examination of Concept and a Review of Literature', *Psychological Bulletin* **73**(4), 258-302.
- Kuk, G. (2006), 'Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List', *Management Science* **52**(7), 1031-1042.
- Lakhani, K. R. & von Hippel, E. (2003), 'How Open Source Software Works: "Free" User-to-User Assistance', *Research Policy* **32**(6), 923-943.
- Lakhani, K. R. & Wolf, R. G. (2005), Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects, in J. Feller; B. Fitzgerald; S. Hissam & K. R. Lakhani, ed., 'Perspectives on Free and Open Source Software', MIT Press, pp. 3-22.
- 1495 Latour, B. (1996), *Aramis, or the love of technology*, Harvard University Press.
- 1495 Latour, B. (1999), *Pandora's hope: essays on the reality of science studies*, Harvard University Press.
- Lattemann, C. & Stieglitz, S. (2005), Framework for Governance in Open Source Communities, in 'Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)', IEEE Computer Society, pp. 192b.
- Lave, J. & Wenger, E. (1991), *Situated Learning: Legitimate Peripheral Participation*, Cambridge University Press.
- Lee, G. K. & Cole, R. E. (2003), 'From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development', *Organization Science* **14**(6), 633-649.
- 1500 Lerner, J. & Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), 197-234.
- 1500 Levy, S. (1984), *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday, New York.

- Lindenberg, S. M. (2001), 'Intrinsic Motivation in a New Light', *Kyklos* **54**(2/3), 317-342.
- Luthiger, B. & Jungwirth, C. (2007), 'Pervasive fun', *First Monday* **12**(1).
- LWN.net (2007), 'Who wrote 2.6.23?', published on September 11, 2007 by Jonathan Corbet, available at <http://lwn.net/Articles/247582/>.
- MacIntyre, A. (1981), *After Virtue*, University of Notre Dame Press, Notre Dame.
- 1505 MacIntyre, A. (1984), *After Virtue: A Study in Moral Theory*, University of Notre Dame Press.
- MacKenzie, A. (2005), 'The performativity of code', *Theory, Culture, and Society* **22**(1), 71-92.
- Markus, M. L. (2007), 'The governance of free/open source software projects: monolithic, multidimensional, or configurational?', *Journal of Management and Governance* **11**(2), 151-163.
- Martin, M. W. (2000), *Meaningful Work: Rethinking Professional Ethics*, Oxford University Press.
- 1510 Martin, M. W. (2002), 'Personal meaning and ethics in engineering', *Science and Engineering Ethics* **8**(4), 545-560.
- 1510 Mauss, M. (1959), *The gift. The form and the reason for exchange in archaic societies*, Routledge, London, UK.
- Monteiro, E.; Østerlie, T.; Rolland, K. & Røyrvik, E. (2006), 'Keeping it going: The Everyday Practices of Open Source Software', *Working paper Norwegian University of Science and Technology*.
- Moody, G. (2001), *Rebel Code: Linux and the Open Source Revolution*, Penguin, London.
- Morgeson, F. P. & Humphrey, S. E. (2006), 'The Work Design Questionnaire (WDQ): developing and validating a comprehensive measure for assessing job design and the nature of work.', *Journal of Applied Psychology* **91**(6), 1321-1339.
- 1515 Morris, A. & Mueller, C. M., ed. (1992), *Frontiers in social movement theory*, Yale University Press, New Haven.
- 1515 Nelson, K.; Nadkarni, S.; Narayanan, V. & Ghods, M. (2000), 'Understanding software operations support expertise: A revealed causal mapping approach', *MIS Quarterly* **24**(3), 475-507.
- netcraft.com (2007), 'March 2007 Web Server Survey', Accessed March 16, 2007.
- North, D. C. (1986), 'The New Institutional Economics', *Journal of Institutional and Theoretical Economics* **142**, 230-237.
- O'Mahony, S. C. (2003), 'Guarding the Commons: How Community Managed Software Projects Protect Their Work', *Research Policy* **32**(7), 1179-1198.
- O'Mahony, S. & Ferraro, F. (2007), 'The Emergence of Governance in an Open Source Community', *Academy of Management Journal* **50**(5), 1079-1106.
- 1520 Osterloh, M. & Rota, S. G. (2007), 'Open Source Software Development - Just Another Case of Collective Invention?', *Research Policy* **36**(2), 157-171.
- Perens, B. (1999), The Open Source Definition, in Mark Stone Chris DiBona, Sam Ockman, ed., 'Open Sources: Voices from the Open Source Revolution', O'Reilly, pp. 171-188.
- Raymond, E. S. (1998), 'Homesteading the Noosphere', *FirstMonday* **3**(10).
- 1525 Raymond, E. S. (1999), *The Cathedral & the Bazaar*, O'Reilly, Sebastopol, CA.

- Roberts, J. A.; Hann, I. & Slaughter, S. A. (2006), 'Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects', *Management Science* **52**(7), 984-999.
- 1530 Robey, D. & Markus, M. L. (1984), 'Rituals in Information System Design', *MIS Quarterly* **8**(1), 5-15.
- Rullani, F. (2007), 'Dragging developers towards the core. How the Free/Libre/Open Source Software community enhances developers' contribution', EURAM annual meeting, available from <http://opensource.mit.edu>.
- 1535 Scacchi, W. (2002), 'Understanding Requirements for Developing Open Source Software Systems', *IEE Proceedings - Software* **149**(1), 24-39.
- Shah, S. K. (2006), 'Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development', *Management Science* **52**(7), 1000-1014.
- Sheldon, K. M. & Krieger, L. S. (2007), 'Understanding the Negative Effects of Legal Education on Law Students: A Longitudinal Test of Self-Determination Theory', *Personality and Social Psychology Bulletin* **33**(6), 883-897.
- 1540 Spaeth, S.; Haeffliger, S.; von Krogh, G. & Birgit, R. (2008), 'Communal Resources in Open Source Software Development', *Information Research* **13**(1).
- Stallman, R. (1999), The GNU Operating System and the Free Software Movement, in Chris DiBona; Sam Ockman & Mark Stone, ed., 'Open Sources: Voices from the Open Source Revolution', O'Reilly & Associates, Inc., pp. 53-70.
- 1545 Stallman, R. (2002), 'The GNU Project', <http://www.gnu.org/gnu/thegnuproject.html>.
- Stewart, K. J. & Gosain, S. (2006), 'The Impact of Ideology on Effectiveness in Open Source Software Development Teams', *MIS Quarterly* **30**(2), 291-314.
- Stewart, D. (2005), 'Social Status in an Open-Source Community', *American Sociological Review* **70**(5), 823-842.
- 1550 Stewart, K. J.; Ammeter, A. P. & Maruping, L. M. (2006), 'Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects', *Information Systems Research* **17**(2), 126-144.
- Torvalds, L. & Diamond, D. (2001), *Just for Fun*, Texere, London, UK.
- Ulhoi, J. P. (2004), 'Open Source Development: a Hybrid in Innovation and Management Theory', *Management Decision* **42**, 1095-1114.
- 1555 Van den Burg, S. & van Borg, A. (2005), 'Understanding moral responsibility in the design of trailers', *Science and Engineering Ethics* **11**(2), 235-256.
- von Hippel, E. & von Krogh, G. (2003), 'Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science', *Organization Science* **14**(2), 209-223.
- 1560 von Krogh, G. (1998), 'Care in knowledge creation', *California Management Review* **49**(3), 133-153.
- von Krogh, G.; Spaeth, S. & Lakhani, K. (2003), 'Community, Joining, and Specialization in Open Source Software Innovation: A Case Study', *Research Policy* **32**(7), 1217-1241.
- 1565 von Krogh, G. & von Hippel, E. (2006), 'The Promise of Research on Open Source Software', *Management Science* **52**(7), 975-983.
- West, J. & O'Mahony, S. C. (2005), Contrasting Community Building in Sponsored and Community Founded Open Source Projects, in 'Proceedings of the 38th Annual Hawaii

International conference on System Sciences (Jan 2005)'.  
1570

Wu, C.; Gerlach, J. H. & Young, C. E. (2007), 'An Empirical Analysis of Open Source Software Developers' Motivations and Continuance Intentions', *Information & Management* **44**(3), 253-262.

XiTi Monitor (2007), 'Firefox frôle les 28% d'utilisation en Europe', <http://www.xitimonitor.com/fr-fr/barometre-des-navigateurs/firefox-juillet-2007/index-1-1-3-102.html>, accessed and archived 17 March 2008.

Ye, Y. & Kishida, K. (2003), 'Toward an Understanding of the Motivation of Open Source Software Developers' Proceedings of 2003 International Conference on Software Engineering (ICSE2003)', Portland, Oregon, 419-429.  
1575

Zárraga, C. & Bonache, J. (2005), 'The Impact of Team Atmosphere on Knowledge Outcomes in Self-managed Teams', *Organization Studies* **26**, 661-681.  
1575

Zeitlyn, D. (2003), 'Gift economies in the development of open source software: anthropological reflections', *Research Policy* **32**(7), 1287-1291.  
1575