Available online at www.sciencedirect.com
ScienceDirect
Journal of Strategic Information Systems 16 (2007) 236–253
Strategic Information Systems
www.elsevier.com/locate/jsis

The open source software phenomenon: Characteristics that promote research ☆

Georg von Krogh *, Sebastian Spaeth

ETH Zurich, Department of Management, Technology, and Economics, Kreuzplatz 5, 8032 Zurich, Switzerland

Accepted 21 June 2007
Available online 13 August 2007

Abstract

Since the turn of the century, open source software has triggered a vast volume of research. In this essay, based on a brief review of selected work, we show that research in many different fields and disciplines of the social sciences have shed light on the phenomenon. We argue that five characteristics make the phenomenon particularly attractive to examination from various fields and disciplines using a plethora of research methods: (1) impact: open source software has an extensive impact on the economy and society; (2) theoretical tension: the phenomenon deviates sharply from the predictions and explanations of existing theory in different fields; (3) transparency: open source software has offered researchers an unprecedented access to data; (4) communal reflexivity: the community of open source software developers frequently engage in a dialog on its functioning (it also has its own research community); (5) proximity: the innovation process in open source software resembles knowledge production in science (in many instances, open source software is an output of research processes). These five characteristics also promote a transdisciplinary research dialog. Based on the experience of open source software research, we propose that phenomena-driven transdisciplinary research provides an excellent context to promote greater dialog between disciplines and fields. Moreover, we propose that the recent diffusion of the open source software model of innovation to other areas than software calls for new research and that the field of information systems has an important role to play in this future research agenda.

Keywords: Open source software; Interdisciplinary research; Innovation

☆ This research was supported by the Swiss National Science Foundation (grant # 100012-101805).
* Corresponding author. Tel.: +41 44 632 88 50.
E-mail address: gvkrogh@ethz.ch (G. von Krogh).

0963-8687/$ - see front matter
doi:10.1016/j.jsis.2007.06.001

1. Introduction

Open source software has two distinct features. First, open source software comes equipped with licenses that provide existing and future users the right to use, inspect, modify, and distribute modified and unmodified software to others (Raymond, 1999). With such open licenses, the software products aim at several market segments, covering operating systems, middleware, and end-user products, such as media players, office suites, and games. Over the last 15 years, many open source software products have made successful inroads into these segments attracting many millions of users. For example, in 2005, Apache achieved a 60% market share for web server software (Netcraft, 2007). In the same year, Firefox, the browser, achieved a 13% market share (Jano, 2006) and turned over more than 50 Million USD for the Mozilla foundation that markets and coordinates its development. Thus, business models around software development are changing and the deployment of open source solutions is a viable alternative for IT managers.

Second, while software can be classified as "open source" independently 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 innovation associated with open source software. Today, projects display a very distinct development process. Open source software projects are typically initiated by a "project leader" or "project entrepreneur." Depending on their interest in the project, volunteers (or companies) join in and contribute to designing, writing, testing, debugging, distributing, and documenting the software. Depending on their knowledge, these voluntary "project contributors" perform tasks ranging from cheering, via administration and coordination, to technical development. Popular projects such as Linux or Azureus may receive the support of several thousand contributors who emerge from a much larger group of users. These contributors, in turn, provide feedback to the open source software developers, share their ideas, report software bugs, indicate new opportunities for using the software, etc. (Raymond, 1999; Lerner and Tirole, 2002; von Krogh and von Hippel, 2003). This highlights a change in the nature of how software architecture is created in an evolutionary manner rather than in a pure top-down planned style as advocated by many (e.g., Brooks, 1995).

Over the last decade, paralleling the growth in markets for open source software, studies of the open source software phenomenon have proliferated in the social sciences. Google Scholar lists more than a thousand papers using the keywords "open source software" and "social science." In the social science citation index, out of 198 papers that have "open source software" registered as a keyword (717 when using just "open source"), roughly 75% have been published within the last 3 years. Several key journals in various fields have published special issues related to open source software, gathering focal contributions.

Research on open source software has surfaced in many disciplines and fields of the social sciences, ranging from economists over computer scientists to anthropologists, investigating various aspects of the phenomenon using a plethora of research methods. The field of information systems thrives on and contributes to "trans-disciplinary" research. According to Bob Galliers, the combination of technical, social, and human factors that impact on the performance of information systems requires an open and frequent exchange amongst researchers working in various areas and disciplines (Galliers, 2003). For the field of information systems, the open source phenomenon, thus, represents an unprecedented opportunity for such a trans-disciplinary dialog. One main question is (1) why the phenomenon has promoted this large amount of research and in the process

(2) evoked the interest of researchers working in several disciplines and fields of the social science contributing to the field of information systems? This essay attempts to address these two questions by analyzing the open source phenomenon and proposes specific characteristics that promote extensive and trans-disciplinary research. The intention is to identify some "lessons learned" about the phenomenon and highlight new research opportunities. This is particularly important since the open source model of innovation is diffusing into other fields and industries including technical design, pharmaceuticals, biotechnology, cultural goods, etc.

The paper is organized as follows. In the next section, we briefly review three research streams and show that researchers from various fields and disciplines have contributed insights to these shared research questions. In Section 3, we analyze in more detail the open source software phenomenon and propose five characteristics that have led to the proliferation of multi-disciplinary research. These are impact, theoretical tension, transparency, communal reflexivity, and proximity. In Section 4, we conclude the paper with a call for new transdisciplinary research on information systems as it applies to the diffusion of the open source model of innovation to new fields.

2. Open source software: A fertile phenomenon for research and trans-disciplinary dialog

There are a number of research areas that are emerging surrounding the phenomenon of open source software. However, for the sake of simplicity and clarity, we classify research according to three major streams (von Krogh and von Hippel, 2006): developer motivation; governance, organization, and innovation process; and competitive dynamics. Each of these streams poses its own set of research questions that are shared by researchers from all fields and disciplines. We demonstrate this trans-disciplinarity through a brief review of relevant work in each stream.

First, a large amount of research has been devoted to the question: why do developers contribute to open source software? This problem is not trivial. Due to its particular license, open source software is a public good; meaning that, first, that no one can be prevented from using the software and, second, the users' utility from the software is independent. The latter is a consequence of open source software being information with negligible distribution costs. Any developer could choose to invest in (or develop) commercial software using a regime of intellectual property rights to appropriate returns from this investment, rather than contributing to open source software where the intellectual property rights guarantee free application and distribution to current and future users of the product. Due to its public good nature, any potential developer can wait to contribute until someone else has developed the software. The latter is frequently referred to as the "free-rider" problem and leads to under-supply of public goods (Olson, 1965). Yet, the enormous growth and popularity of the open source software shows that the founders of open source projects, so-called "project entrepreneurs", and other developers were perfectly capable of solving this problem.

Economists found this puzzle interesting. For example, in their pioneering work, Lerner and Tirole (2002) using economic theory suggested that developers contribute in order to increase their labor market value. By writing and sharing high-quality software in the public domain, developers could signal to current and prospective employers their level of skills and, thereby, increase their salaries and advance their careers. Other economists such as Dalle and David (2003) concurred, but these authors added that the signaling incentive

was only available to a few developers who worked on technically important and sophisticated modules and to those project entrepreneurs who initiated important and popular projects. Combining theories from economics and sociology, von Hippel and von Krogh (2003) developed a "private-collective" model of innovation incentives. They suggested that developers contribute to a public good innovation because they garner private benefits related to the innovation process. These benefits, including fun, reputation, learning, enjoyment, and peer recognition, are not supplied to the same degree to non-contributors.

Psychologists shared this research question and quickly advanced an important research agenda that covered a broad range of motives. The conjecture was that community participation, social motives, and norms relate to levels of contribution individual developers and other contributors make to open source software projects. For example, in an early survey study, Hertel et al. (2003) found that Linux developers were motivated by the need to improve the product for their own use. They also identified with the larger "community of Linux developers" and were, thus, motivated by group-related factors, such as their "perceived indispensability" for the group in which they work. Bagozzi and Dholakia (2006), using a survey design, found that participants in Linux User Groups (technical support groups) were motivated by a combination of social and sychological factors. "We-intentions" were important for predicting individual actions that contribute to group-level actions.

In the discipline of cultural anthropology, Zeitlyn (2003) pointed out that we need to better understand the culture of the open source software movement and the corresponding social norms that regulate people's behavior. Due to the social norm of reciprocity, developers are motivated to give "gifts" to the project in the form of software patches, comments, bug fixes, and so on (Raymond, 1999; Bergquist and Ljungberg, 2001). However, Zeitlyn's advice was for the cultural studies of the open source software phenomenon to focus more broadly on complementary social norms, including "familiarity" and "kinship" amongst contributors in projects, as important for motivating contributions.

Researchers in the field of management and organization studies (broadly covering technology and innovation management, management information systems, management and organization theory, organization behavior, etc.) also found the puzzle of developer- and contributor-motivation interesting. For example, gathering and analyzing data from the Apache project, Roberts et al. (2006) asked if employment in firms matters for developers' levels of contributions. These authors observed that developers attain very different status levels in the open source software projects, and that mixed motives relate to the highest levels of contribution. People are motivated both by attainment of status in the Apache project and paid participation (firm employment). Developing a new survey instrument, Franke and von Hippel (2003) studied user satisfaction in Apache security software. They found that developers who were capable of changing the software to fit their own needs were significantly more satisfied than those who could not adapt the product. This evidence indicated that the technical design of open source software relates to developer motivation and prepared the ground for other researchers such as Baldwin and Clark (2006) to investigate the relationship between design and motivation. They developed a game-theoretic model that takes into account the extent to which a software architecture is "modular." Their analysis showed that developers have less incentive to "free-ride" upon open source software with a modular architecture.

Spurred by a compelling and simple research question, a matrix of research has emerged that helps shed considerable light on the open source phenomenon. This matrix

is enabled by researchers pursuing a trans-disciplinary dialog. While they conduct their disciplinary research and make contributions within the methodological tradition of each discipline, there is also an extensive and continuous referencing of the work by researchers in other disciplines.[1] As we will argue later, this research matrix also fuels a debate within the practice of open source software development.

While the study of individual motivations clearly represents the most straightforward example of a trans-disciplinary dialog, there are also clear instances in the two remaining research streams. In governance, organization, and innovation process, research uncovered that contributors' distributed and diverse interests as well as their different capabilities make them participate with different intensity in various parts of the innovation process (e.g., Nonnecke and Preece, 2003; see also Franck and Jungwirth, 2003; Koch and Schneider, 2002; O'Mahony, 2003). In the field of legal studies, Benkler (2002) argued that open source software represents a form of "commons-based peer production." Such a form of production uses other governance mechanisms than the "market" or the "firm." In the field of management and organization studies, scholars analyzed in more detail the nature of the open source software development process and argued that open source software projects have developed a way of governing very distributed participation where people can pursue their diverse interests. This happens without the software product "forking" into new and "unofficial" versions (Kogut and Metiu, 2001; see also Demil and Lecocq, 2006). If forking were common practice, projects developing different versions of the product would compete for scarce developer resources, create confusion in the market, create incompatible versions on users' computers, etc. Forking could potentially threaten the sustainability and the quality of the software product. The open source software projects' ability to "govern" the development process relate to role differences in projects. While many people can contribute in various ways to open source software (giving advice, identifying and solving bugs, etc.), technical development is typically restricted to a much smaller group of "developers" who also decide on the design of the software product. This layered model of organization of open source software development can be thought of as an "onion" with layers of bug fixers, bug reporters, occasional participants in email subscription lists, etc. with each layer being more distant from the project's developer core (Crowston and Howison, 2006). Moving from being a "peripheral" contributor to becoming a developer may be costly for the individual (von Krogh et al., 2003). Moreover, Kuk (2006) found that this change in the status of contributors also coincides with various forms of strategic interactions amongst a few participants who are knowledgeable about the software and resourceful in other ways (Kuk, 2006).

Sociologists shared this interest in the status of open source software contributors and developers. In particular, the open source software phenomenon provided data from a natural setting related to various circumstances that lead to the generation of status orders. In the process of status attainment, research identified that open source software developers tended to evaluate a focal actor reputation according to publicly available social references. These references emerge as part of the innovation process itself (Stewart, 2005).

Due to the public good nature of open source software, the research stream on competitive dynamics posed an important research question shared across disciplines and fields:

[1] The reference list of the contributions reviewed indicates cross-referencing.

how do firms compete (or collaborate) with "free?" In particular, as open source software gains ground in the market, what impact will it have on commercial software? In economics, early work on this topic concluded that both commercial software and open source software would continue to coexist, but that open source software would impact the commercial software vendor's strategy (Bonaccorsi and Rossi, 2003; Mustonen, 2003). More recently, authors such as Casadesus-Masanell and Ghemawat (2006) and Economides and Katsamakas (2006) have developed in more detail how open source software would change the commercial firm's pricing and the choice of a technological platform (software/hardware). The research question also spurred inquiry in the field of management and organization studies. For example, West (2003) discussed advantages and disadvantages of different strategies and technological platforms pursued by firms. Dahlander and Magnusson (2005), in turn, analyzed how software firms in Scandinavia collaborate and build relationships with communities of volunteer software developers.

To summarize, research originating in various disciplines has shed considerable light on the open source software phenomenon. This work shared some research questions that must be seen as a precondition for a fruitful trans-disciplinary dialog amongst researchers and, thereby, a proliferation of research. Without this dialog, partial views and models would still have dominated our understanding of the open source software phenomenon. For example, while early pioneering economic theory suggested some developers are motivated by career incentives, research in psychology, sociology, and management and organization studies accumulated a complementary set of models of motivation that include factors, such as the use-value of the software, the need for peer recognition, and kinship. We also believe there are particular characteristics of the open source phenomenon that made it attractive for researchers and that led to a proliferation of disciplinary research and trans-disciplinary dialog. Next, we discuss these characteristics in more detail.

3. Characteristics that promote research

Flyvbjerg (2001) calls for social sciences to have more impact on society. In particular, he notes that the problem of the social sciences is an inherent detachment from reality. Natural science, in contrast, is interwoven with reality and, thus, does not face the same problems. The criticism of social sciences typically poses an urgent question: "If you are wrong about this, who would notice?" Flyvbjerg proposes a number of remedies to the problem detachment in the social sciences, of which one is transcending the problem of relevance by grounding research in the context studied. This implies getting close to the phenomenon during data collection and remaining close during data analysis, feedback, and publication. The involvement in research of stakeholders in the field who test, evaluate, and comment upon the findings is key to making the research more impactful in the social world. Moreover, Flyvbjerg asks researchers to nurture the interaction between universal principles and the particular context in building and testing theories.

Research on the open source software phenomenon is an interesting example of research that combines scientific rigor with relevance. Open source software, perhaps because of the distributed organization of its development as well as its intellectual property regime, lends itself to close and often intense interaction with research. This has also made research relevant. On the one hand, research on the open source software phenomenon has served to uncover features of an institutional innovation and lay these open for scrutiny and debate among the wider public who have also taken an interest in the

phenomenon. Studies of developer motivations have contributed to discussions regarding software product quality and project sustainability (Lakhani and von Hippel, 2003; von Krogh, 2005). Work on open source licenses contributes to a better understanding of competitive dynamics in the software industry and possibly other industries producing digital goods (The Economist, 2005). On the other hand, the phenomenon provided sufficiently interesting research questions to trigger the curiosity of researchers and maintain their interest over several years. This is a precondition for knowledge to accumulate in the social sciences.

In the following, we speculate that certain characteristics of the open source software phenomenon make it attractive and important for researchers. These characteristics also make it a good example for the value of relevant and trans-disciplinary research. These characteristics of the phenomenon are: social and economic impact; tension with existing theory; transparency of the data; reflection of the community on its inner workings, and proximity between open source innovation processes and science. As we will show, these characteristics also lead to important questions for information systems research. As the open source software model of innovation spreads into many fields, these questions have become very important for researchers to bear in mind.

3.1. Impact

The open source software phenomenon has had a ubiquitous impact on society and the economy, as we will demonstrate in five examples. First, the phenomenon has been likened to a massive social movement in which contributors, developers, governments, and firms collaborate to create a public good that shapes society (e.g., Holtgrave and Werle, 2001). The magnitude of the phenomenon as a social movement is often emphasized by quoting data from open source software project hosts. Currently, one of these hosts, Sourceforge, lists 150,000 projects and in excess of 1.6 million contributors. In addition, open source software has millions of users all over the world. For example, the popular "OpenOffice.org" office suite lists in excess of 50 million users.

Second, as expected, open source software has altered global competition in the computer software and hardware industries where firms traditionally competed on proprietary, "closed source" software. Firms that develop and sell proprietary software products have started to adopt open source software solutions in their own product portfolio. Moreover, a new type of firm has entered the industry, such as Suse, Red Hat, Red Flag, and MySQL that package and distribute and sell open source software and auxiliary services. Research has shown that open source software may be the preferred license form of new entrants into the software industry (Bonaccorsi et al., 2006). Open source software has also changed competition in adjacent industries. Increasingly, firms ship their computer hardware equipped with open source software products. In areas such as consumer electronics or manufacturing technology, embedded software represents an increasing share of research and development costs. Here, embedded open source software is increasingly used in devices and control systems (Henkel, 2006). For example, consumer electronic firms such as Motorola, Nokia, and Palm increasingly use Linux as operating system in their products.

Third, in many countries the government has adopted explicit policies towards open source software (Cook and Horobin, 2006; see also Comino and Manenti, 2005 for an analysis). The reasons given include reduction of procurement cost, better bargaining

positions, the need to support local software and service firms, the adaptability of the software to the government's needs, transparency of the software, and security issues. While many governments have enthusiastically embraced open source software policies, there are also reports of significant organizational changes needed to install and run the software, unanticipated costs of training people, and slow migration from previous systems (e.g., Waring and Maddocks, 2005).

Fourth, open source software has been advocated by many as a solution for closing the "digital divide" by assisting developing countries in their efforts to apply information technology (Bokhari and Rehman, 1999; May, 2006). Because open source software is free and easily accessible online, it is attractive for many users including government, home users, schools, or business accounts. Moreover, the access to the source code as well as the transparency of the development process also enables the local education and training of information technology professionals (see Kogut and Metiu, 2001; Weber, 2004). According to James (2002) who analyzed open source software in developing economies, one of the challenges in such economies is to prolong the "life" of computers. Often actors in these economies cannot afford to upgrade their systems at the same pace as customers in developed countries. James quotes an example from the Philippines where schools had to raise their tuition fee every time they needed to upgrade their information systems due to a new version of a software. Linux comes with free upgrades and ameliorates this problem for local educational institutions. In addition, based on previous experiences from buying commercial software licenses, many local users in developing nations are wary of locking their institutions into software that in the future may force them to pay license fees.

Due to the rapidly growing impact of open source software on society and the economy, it became attractive for researchers from many disciplines to investigate different aspects of the phenomenon. Due to increasing investments in information systems, including open source and commercial software around the world, society and the economy also had a direct interest in the results of this research. For example, governments were interested in the effect of their policies towards promoting open source software versus promoting commercial software. This raised questions to economists whether open source software was a response to market failure. Moreover, governments and firms too were interested in understanding issues such as the quality of open source software in comparison with commercial software, the cost of migrating to open source solutions, and to what extent the phenomenon would be sustainable in the long run (see Evans, 2002).

Research that uncovered the inner workings of the open source projects also achieved importance in the public debate and assisted in bringing the field of information systems to the attention of the general public. It raised debates and issues about user involvement in systems development, the advantages and disadvantages of intellectual property rights, the cost of information systems to the public, and so forth (The Economist, 2006). With the impact of open source software and the attractive research opportunities that open source software affords, the interaction between the phenomenon and researchers and the larger landscape of the economy and society in itself emerges as an important area of inquiry. For example, Longino (1990) proposes that different fields and disciplines should investigate how social values influence scientific research. Social values such as the ability to close the digital divide, empowerment of consumers ("stuff by us"), or less costly access to intellectual property may have played a role in the drafting of research agendas. This topic remains very important as open source software continues to gain importance and as its model of innovation is diffused to other fields. Information systems research will also

benefit from an examination of the interaction between social values and information systems in society. Open source software may be just the "thing" to do this.

3.2. Theoretical tension

As researchers started to investigate the phenomenon in more depth, many came to realize a growing tension between the data they were retrieving and established assumptions and mainstream theories. Consider two examples. First, the process of open source software development strongly deviated from proposed models, frameworks, and approaches to software engineering (Feller and Fitzgerald, 2002; Scacchi, 2002). Software development typically requires a dedicated team of software engineers and other specialists who assume different roles in the process, including specifying requirements for the products, creating a high-level roadmap of development, writing and documenting the code, and assessing and testing the product. Open source software development, on the other hand, consists of hundreds or perhaps thousands of volunteers who assume different roles too, including founding the project and formulating its goals, writing, reporting and fixing bugs, etc. However, in many projects roles are assumed through "self-allocation" based on people's knowledge and interests. According to Madey et al. (2002) this way of developing software is often "counterintuitive" and they called for more research in order to understand the inner workings of the process. In particular, they underscored that open source software development emerged as a new way of organizing a very large number of volunteer contributors and that this approach had no precedent and no satisfactory explanation from theories of software engineering. Second, open source software runs counter to many established theories of innovation (Schoonhoven, 2003). von Krogh and von Hippel (2003) noted that conventional theory incentives to innovate relate to the regime of intellectual property rights that offer innovators the possibility of appropriating returns from innovation-related investments. In open source software, licenses ensure that the software product remains a public good and companies sometimes even support projects associated with direct competitors (Mustonen, 2005). Therefore, von Hippel and von Krogh called for more research to investigate the incentives to create public good innovations.

These are just two brief examples among many that show the theoretical tensions researchers were confronted with as they intensified their interaction with the phenomenon. It required researchers to rethink many established theories and assumptions (e.g., Bessen, 2002). On the one hand, this enhanced the attractiveness of the research on the phenomenon. Open source software represents a "critical case" or the possibility to "falsify" existing theory. It also offers a possibility to combine theories, for example from economics and sociology, in order to explain hitherto distinct and isolated phenomena (e.g., Bonaccorsi and Rossi, 2003). Because of new research questions (Section 2), there are also growing possibilities for grounded theorizing along the lines called for by Flyvbjerg (e.g., O'Mahony, 2003; Shah, 2006).

On the other hand, open source software also represents an additional investment for researchers who might be familiar with and sometimes have contributed to established theories and research. In many cases, an additional investment is also needed to develop an understanding of software development, in general, and the context of open source, in particular. However, judging by the growth of research over the last years, the research opportunities seem to outweigh the cost.

In our view, the popularity of open source software research was enhanced because early research contributions pointed to theoretical tensions, rather than "confirming" existing models and frameworks. This also raises an important question for information systems research in general. What are the current phenomena that deviate from existing and accepted theory and assumptions? It seems that often the cost of searching for such opportunities outweighs the potential benefits (which in the case of open source software is only available when a certain amount of research has produced new insights about the phenomenon). However, the field of information system research will benefit strongly from encouraging early-stage, high-risk research into areas with theoretical tension.

3.3. Transparency

Often research on commercial product development is hampered by restricted access to the development process and convoluted, or selectively, released data. Firms that commercialize their software products are in most cases not interested in sharing the product's source code due to the risk of code spilling over to competitors or "software pirates." In contrast, due to their development practices, open source software projects provide a very high transparency of data for research. The software's source code is generally available from repositories that host the projects, such as Sourceforge, Freshmeat or Savanna. This enables researchers to investigate the inner, technical workings of the software and product development process. This transparency of technical data afforded an unprecedented opportunity to study such issues as functionality, software architecture, file size, language, software component reuse, application protocol interfaces, bug identification and fixing, and individual contribution levels (e.g., MacCormack et al., 2006). Depending on the projects' hosts, in many cases data on software downloads would also be available. This was used by some researchers to gauge the popularity or "market share" of the project in conjunction with the number of contributors to the project (e.g., Crowston et al., 2003; Bagozzi and Dholakia, 2006).

Moreover, most projects host mailing lists dedicated to various aspects of the software product and the project. Some lists may focus on technical development issues, while others may deal with user assistance, general user feedback, or discussions regarding the "philosophy" of the project. These lists represent very valuable data for researchers because they make discussions available that can be used to examine a host of organizational and behavioral issues. Some important issues that can be investigated using such data include the number of participants in the discussions, intensity of the mail exchanges, topics of discussion, the number of messages related to particular topics, people's influence in the development process, coordination of work, decision making, conversational protocol and etiquette, and so forth. The retrieval of both technical data and mailing lists data is in many cases possible for all researchers, and not only restricted to those who have exclusive relationships with developers or project entrepreneurs. For many open source software projects, such data is extensive, covering millions of lines of code and thousands of messages and, thus, offers itself to various forms of quantitative analysis.

As research progresses on the open source software phenomenon, it becomes increasingly clear that the mailing lists in most cases represents the primary means of communication between developers who work in different locations. Yet, researchers who did extensive field work and interviewed open source software developers also found that developers used online chats to solve issues. These chats are not stored and often not

available for real-time inspection by researchers. In other cases, developers would organize meetings or attend conferences in order to discuss development issues (O'Mahony, 2003). It has also become increasingly clear that many developers in large projects, such as Apache or Linux, work for the same firm (see Roberts et al., 2006; West and O'Mahony, 2005) and, thus, may meet offline. With time, research combined different types of data to produce new insights, ranging from source code and message threads to participant observations during off-line meetings and interviews with developers and other contributors. While many are attracted to the open source software phenomenon because of the immediate transparency of the phenomenon and the access to and the ease of data retrieval, in many projects researchers soon discover that the pursuit of rich data is important in order to provide "thick descriptions" of the phenomenon (Geertz, 1973). The search for better explanations of open source software development posed some critical questions on the data used by researchers (e.g., von Hippel and von Krogh, 2003). This, in turn, initiated "learning and improvement" in the research process that triggered even more new research.

For information systems research an important lesson from open source software research regards the nature of data gathering. First of all, due to the social, human, and technological dimensions of information systems research (as is the case with open source software), high-quality empirical research is characterized by the retrieval and analysis of data from multiple sources. Moreover, the extent and availability of one type of data (in particular, quantitative) can lure researchers into neglecting other types of data (in particular, qualitative) that may be more costly to retrieve. Therefore, what characterizes the quality of data in information systems research that explores a new phenomenon? Likewise, considering the learning process of empirical research on open source software, an important question is what are the conventions for gathering and analyzing data in information systems research and how do these need to be adjusted to the phenomenon under investigation.

3.4. Communal reflexivity

Delanty (2001) suggested that university-based research undergoes a transformation in the process of knowledge production in society. Historically, the university was the primary user of the research-based knowledge it produced, predominately when educating people. However, increasingly research-based knowledge is also produced as well as used by many other actors in society. Interestingly, the interaction between research and the open source software phenomenon epitomizes this shift and may help to set a precedent for future research on information systems. The community of open source software developers, contributors, and users share a strong interest in why, what, and how the community operates. We term the ongoing engagement in a dialog about the functioning of the community "communal reflexivity." Communal reflexivity is evident in the many online discussions about the roles and functions of open source software and its impact on the economy and society. In addition, a search on 'Google Blogs' reveals more than one million entries using the term: "open source software" of which several thousand are devoted to the development process and the functioning of the community. Google also archives nearly 4000 Usenet discussion groups dedicated to the Linux open source operating system. Many books about the open source software phenomenon have been written by developers and very active participants in the open source software community. For exam-

ple, the ground-breaking and famous book by Eric S. Raymond (1999) can be seen as initiating much of the research and dialog on the functioning of the community. Raymond, himself an open source project entrepreneur, developer, and trained anthropologist, embodies the dialog between research and open source software practice. Furthermore, a number of conferences about the open source software development model include practitioners in the field and, increasingly, developers and participants are also involved in tracks of mainstream academic conferences.

We would like to offer a reason for the evolution of communal reflexivity. Open source software development can be seen as an "institutional innovation" (Hargrave and Van de Ven, 2006; O'Mahony, 2002). As discussed in Section 2, open source software involves a very large number of contributors, it uses simple technologies for coordinating work, it draws upon direct feedback and improvement by users, it produces a public good with considerable market and economic impact, and so forth. Observers have noted that open source products, such as Linux, emulate already existing software products in their design, such as commercial Unix variants. The same holds for OpenOffice.org, an office suite that emulates Microsoft Office. These previous designs provided guidelines for software developers and allowed them to observe and design important features and benchmark their design for performance.

Regardless of the software product design, however, the "institution" of open source software development evolved without templates or guidelines. In order to resolve issues that could potentially threaten the survival and advancement of the institution, intense self-observation and dialog is called for by the community. For example, professional and corporate users of open source software frequently question both the sustainability of project organization and the quality of the software. Migrating systems from commercial to open source software entails significant costs for many professional and corporate users. This prompted a discussion on the risk of developers losing interest in the software and so discontinuing a project. Because open source software does not necessarily guarantee upgrades and updates, but rather relies on the initiative of voluntary developers, many professional and corporate users felt uneasy about implementing systems migration. However, the Open Source Initiative, the Apache foundation, the Free Software Foundation, and other entities attempted to analyze if and why open source software would rival commercial software in terms of sustainability and, thereby, counter these arguments. One of their claims, for example, was that software firms that went bankrupt would represent an equal level of risk to professional and corporate users. Without a firm to back it up, commercial software would not be developed further. Discussions, therefore, began to center around the motives and incentives for developers to contribute to open source software projects. Observers of the phenomenon offered their own reflections on incentives and an important dialog evolved between academic research on innovation incentives and the community of open source developers and contributors. Examples of studies that fueled this dialog include Lakhani and Wolf (2005), Ghosh et al. (2002), and Hertel et al. (2003).

Critics of open source software and corporate users also questioned the quality of open source software. Research using bug report data (e.g., Kuan, 2001) showed that flaws in open source software did not exceed those of commercial software that performed the same functions. Research by Franke and von Hippel (2003) also showed that users in general were more satisfied with the software product which they could change to fit their specific needs. Moreover, studies revealed that software product "quality" was secured by

allowing only a select group of developers the possibility to implement changes to the software code, although the ideas, bug fixes, or software patches might have come from a large number of contributors (von Krogh et al., 2003). These and similar studies were noticed and commented upon by the open source community in conjunction with discussions about product quality.

A very vocal community of contributors and developers, resident contributors who also researches the phenomenon, and an emerging stream of scientific research evolves a communal reflexivity. This communal reflexivity, we believe, is critical to the institutional innovation represented by open source software development. As this institution gradually forms and enhances its economic and social impact, it also draws more attention amongst researchers who engage in trans-disciplinary research. Thus, open source software represents an interesting case of institutional innovation where a condition is communal reflexivity interacting with academic research. When Bent Flyvbjerg asks for social science to matter, we believe that the impact that social science can have is very much demonstrated in the interaction between the stakeholders surrounding the open source software phenomenon and academic researchers. Based on this observation, an important question is how information systems research can increase its impact on the economy and society in other areas too by building similar interactions with a broad range of phenomena.

3.5. *Proximity*

Finally, commentators have noted that open source software development resembles the process of doing science. Science has the objective of creating a public good (David, 2005) and its results are mostly possible to digitize and distribute over the Internet. Bezroukov (1999) classifies programming as a special kind of scientific activity in which an idea or a proposition is encoded. In open source software, modifications are based on the input from many peers around the world. Likewise in science, capturing research results in papers, presentation of papers at academic conferences, revising papers for journal publications, and so forth requires good feedback from peers. Both open source software development and increasingly science build on the work of "virtual teams" where collaborators are located all over the world. Bezroukov (1999) also suggests that contributors to open source software and science are motivated by factors such as learning, peer recognition, and fun rather than monetary incentives. Yet, an important difference in motivation also marks an important difference between science and open source software. While open source software developers are often motivated by the use value of their software products (von Hippel, 2002), most scientists cannot directly apply what they create in their laboratories to their lives.

Moreover, Bezroukov (1999) also notes that financing of open source projects may be similar to the financing of science. In many cases, scientific research is funded indirectly. In this scheme, individuals employed in an institution become interested in a particular phenomenon and step forward to conduct research with existing slack funds. Many open source software developers also work either in academic institutions or large corporations. For example, 50% of all developers in Linux work for firms (LWN, 2007). Large corporations may provide the necessary slack funds for open source software development just as they do for research projects.

An increasing number of large corporations, including IBM, Hewlett–Packard, Novell, and SAP, sell packaged open source software with their hardware or commercial software

products and, thus, view open source software as a strategic asset to their business. Many of these firms choose to dedicate development resources to open source software projects, either internally or through collaborative development projects such as the Open Source Software Development Lab. This approach resembles dedicated funding of research projects in universities and research labs.

In many cases, the outcome of applied science is itself open source software code. For example, Mascialino et al. (2006) report on a project for developing an open source software statistical toolkit for data analysis in experimental particle and nuclear physics. The example is interesting because the authors report on how improvements have been made to facilitate better statistical analysis in a scientific field. The paper resembles a "message" in a developer mailing list, introducing and reporting on the performance of new features such goodness-of-fit tests, new implementations of existing tests, a new component to extend the usability of the toolkit with other data analysis systems, and new tools for software configuration, etc.

We believe the strong proximity between open source software and science contributes to a faster and easier understanding of the phenomenon amongst researchers. We also believe this makes the phenomenon very attractive for researchers coming from many fields and disciplines. In many disciplines, the open source software development process resembles the creation of science and, hence, the phenomenon opens up an interesting dialog between researchers from these various disciplines. Whereas researchers may pursue very different research questions and gather and analyze data differently, we all find ourselves embedded in similar processes of creating science by many similar rules. Observing the mirror image of open source software development makes this apparent and it may also bring to the forefront the strengths and weaknesses of science and indicate a way to improve the scientific enterprise.

Information systems are embedded in academic life which allows a strong proximity between the actual use of systems in a university and research setting and their use in a corporate setting. For example, the development, adaptation, and implementation of an enterprise resource planning software may encounter many of the same opportunities and challenges in universities and corporations. This proximity is an opportunity for information systems research that can be formulated as a question: in which areas of inquiry is "proximity" instrumental in enabling researchers to pose interesting and relevant research questions?

4. Conclusion

In this essay, we briefly reviewed select research contributions that shed light on the open source software phenomenon. These contributions come from different fields and we argued that the phenomenon brought forth an important trans-disciplinary dialog. Subsequently, we attempted to define the characteristics of the open source software phenomenon that triggered a massive amount of research from different fields and disciplines. These were impact of the phenomenon, transparency of the phenomenon, theoretical tension arising from the phenomenon, communal reflexivity, and proximity between science and the phenomenon. We also showed that these characteristics of the open source software phenomenon and the lessons learned from research on this phenomenon lead to important questions for information systems research. These questions are:

1. What is the relationship between social values and the evolution of information systems in society?
2. What are the current and emerging social and economic phenomena that deviate from existing and accepted theory and assumptions in information systems research?
3. What characterizes the quality of data in information systems research that explores a new phenomenon?
4. How can information systems research strengthen its interaction with a broad set of phenomena?
5. In which areas of inquiry is proximity between science and the phenomenon instrumental in enabling information systems researchers to pose interesting and relevant research questions?

As open source software has shown, an important challenge of social science is to identify new theories and research designs to understand existing phenomena more efficiently and/or effectively. In many ways, open source software emerged and needed an explanation. However, another, sometimes forgotten, challenge is to actively search for phenomena that call for new theories and research designs. Because social science like open source software is a collective effort that hinges on the mobilization of people and other resources, the distinctiveness of the phenomenon is important. Over time, when more people join in, researchers are likely to develop a deeper and more collective knowledge of the phenomenon. Given the resource and time constraints facing all researchers, the wish to make a contribution where science matters is often substantive. Hence, in the early stages of research, interested researchers need to "envision" such new and distinct phenomena and reflect upon the contributions that science can make. Based on the analysis of the open source software phenomenon, perhaps the five characteristics listed will be helpful in this process.

It is uncertain to what extent the five characteristics of the open source software phenomenon can "replicate" themselves to other industries, sectors, and systems. However, open source principles are currently "experimented with" in other areas. For example, in cultural goods, a new type of news reporting involves thousands of citizens who write stories about what they observe and experience. These stories are then submitted for editing to a newspaper desk. In biotechnology, GNU-like licenses are applied to tools for genetic research. Currently, medicines for malaria and drugs for other neglected deceases are being developed using open source models where universities, firms, individual researchers, and NGOs cooperate in drug development. In technical design, new open source market places are emerging where "customers" can pose problems and interested engineers and industrial designers worldwide can design and deliver possible solutions. However, it is still uncertain if the "open source principles," in particular regarding public good innovations and intellectual property rights, will lead to the same level of social and economic impact in these other fields. Yet, we believe the gradual adoption of open source principles in other fields represents an opportunity for researchers from various fields and disciplines to explore the principles' general applicability. With increasing public attention and allocation of resources to the diffusion of open source principles in other fields, it will be social science that matters (research with a very high value and impact on society).

The field of information systems is one of the core pillars in open source software. Information systems products emanate in great numbers from open source software projects. Equally interesting is that the mailing lists and concurrent versions systems discussed

above are lightweight information systems that enable effective software development. Given how open source software is transforming industries, the field of information systems research is at the heart of a new model of innovation that allows thousands of collaborators globally to create high-quality products. Since the design of most products (even physical products) can be digitized and, therefore, shared at low cost, the diffusion of this innovation model to other fields is likely to be rapid. In order to secure the best possible understanding of the open source phenomenon as it continues to spread, information systems research should remain open to dialog with other areas and disciplines. It is then that creativity can flourish, new unprecedented insights can be gained, and new phenomena discovered (Galliers, 2003). These are wonderful times for information systems scholars.

References

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., 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.

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. Open source software: Free provision of complex public goods. Technical report, Research on Innovation.

Bezroukov, N., 1999. Open source software as a special type of academic research. First Monday 4 (10).

Bokhari, S.H., Rehman, R., 1999. Linux and the developing world. IEEE Software 16 (1), 58–64.

Bonaccorsi, A., Giannangeli, S., Rossi, C., 2006. Entry strategies under competing standards: Hybrid business models in the open source software industry. Management Science 52 (7), 1085–1098.

Bonaccorsi, A., Rossi, C., 2003. Why open source software can succeed. Research Policy 32 (7), 1243–1258.

Brooks Jr., F.P., 1995. The Mythical Man-Month: Essays on Software Engineering. twentieth Anniversary ed., Addison-Wesley.

Casadesus-Masanell, R., Ghemawat, P., 2006. Dynamic mixed duopoly: A model motivated by Linux vs. Windows. Management Science 52 (7), 1072–1084.

Comino, S., Manenti, F.M., 2005. Government policies supporting open source software for the mass market. Review of Industrial Organization 26 (2), 217–240.

Cook, I., Horobin, G., 2006. Implementing eGovernment without promoting dependence: Open source software in developing countries in Southeast Asia. Public Administration and Development 26 (4), 279–289.

Crowston, K., Annabi, H., Howison, J., 2003. Defining open source software project success. In: Proceedings of the 24th International Conference on Information Systems (ICIS 2003).

Crowston, K., Howison, J., 2006. Hierarchy and centralization in free and open source software team communications. Knowledge, Technology, and Policy 18 (4), 65–85, Special Issue on Open Source.

Dahlander, L., Magnusson, M.G., 2005. Relationships between open source companies and communities: Observations from Nordic firms. Research Policy 34, 481–493.

Dalle, J.-M., David, P.A., 2003. The allocation of software development resources in 'open source' production. Discussion Paper of The Stanford Institute For Economic Policy Research.

David, P.A., 2005. From keeping 'nature's secrets' to the institutionalization of 'open science'. In: Ghosh, R.A. (Ed.), Code: Collaborative Ownership and the Digital Economy. MIT Press, Cambridge, MA, pp. 85–108.

Delanty, G., 2001. Challenging Knowledge: The University in the Knowledge Society. Open University Press, Buckingham.

Demil, B., Lecocq, X., 2006. Neither market nor hierarchy nor network: The emergence of bazaar governance. Organization Studies 27 (10), 1447–1466.

Economides, N., Katsamakas, E., 2006. Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry. Management Science 52 (7), 1057–1071.

Evans, D.S., 2002. Politics and programming: Government preferences for promoting open source software. In: Hahn, R.W. (Ed.), Government Policy Toward Open Source Software. Brookings Institution Press, pp. 34–49, chapter 3.
Feller, J., Fitzgerald, B., 2002. Understanding Open Source Software Development. Addison-Wesley, London, UK. Foreword by Eric S. Raymond. Companion Website.
Flyvbjerg, B., 2001. Making Social Science Matter: Why Social Inquiry Fails and How it Can Succeed Again. Cambridge University Press.
Franck, E., Jungwirth, C., 2003. Reconciling investors and donators: The governance structure of open source. Journal of Management and Governance 7, 401–421.
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.
Galliers, R.D., 2003. Change as crisis or growth? toward a trans-disciplinary treatment of information systems as a field of study: A response to Benbasat and Zmud's call for returning to the IT artifact. Journal of the Association for Information Systems, 4(6), 337–351. Available at http://jais.isworld.org.
Geertz, C., 1973. The Interpretation of Cultures. Basic Books, New York.
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/.
Hargrave, T., Van de Ven, A.H., 2006. A collective action model of institutional innovation. Academy of Management Review 31 (4).
Henkel, J., 2006. Selective revealing in open innovation processes: The case of embedded Linux. Research Policy 37 (7), 953–969.
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.
von Hippel, E., 2002. Horizontal innovation networks – by and for users. Working Paper.
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.
Holtgrave, U., Werle, R., 2001. De-commodifying software? open source software between business strategy and social movement. Science Studies 14 (2), 43–65.
James, J., 2002. Low-cost information technology in developing countries: Current opportunities and emerging possibilities. Habitat International 26, 21–31.
Janco Associates, Inc., 2006. Browser market share study. http://www.e-janco.com.
Koch, S., Schneider, G., 2002. Effort, cooperation and coordination in an open source software project: Gnome. Information Systems Journal 12 (1), 27–42.
Kogut, B.M., Metiu, A., 2001. Open-source software development and distributed innovation. Oxford Review of Economic Policy 17 (2), 248–264, Special Issue on The Internet.
von Krogh, G., 2005. Open for business. In Global Agenda: The Magazine of the World Economic Forum, 3, 186–189. World Economic Forum.
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.
von Krogh, G., von Hippel, E., 2003. Special issue on open source software development. Research Policy 32 (7), 1149–1157, Editorial.
von Krogh, G., von Hippel, E., 2006. The promise of research on open source software. Management Science 52 (7), 975–983.
Kuan, J., 2001. Open source software as consumer integration into production. Working Paper.
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: Feller, J., Fitzgerald, B., Hissam, S., Lakhani, K.R. (Eds.), Perspectives on Free and Open Source Software. MIT Press.
Lerner, J., Tirole, J., 2002. Some simple economics of open source. Journal of Industrial Economics, 52.
Longino, H.E., 1990. Science as Social Knowledge. Princeton University Press, Princeton.
LWN.net, 2007. Who wrote 2.6.20? published on February 20, 2007 by Jonathan Corbet., Available at http://lwn.net/Articles/222773/.

MacCormack, A., Rusnak, J., Baldwin, C.Y., 2006. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. Management Science 52, 1015–1030.

Madey, G., Freeh, V., Tynan, R., 2002. The open source software development phenomenon: An analysis based on social network theory. In Americas Conference on Information Systems (AMCIS2002), Dallas, TX, pp. 1806–1813.

Mascialino, B., Pfeiffer, A., Pia, M.G., Ribon, A., Viarengo, P., 2006. New developments of the goodness-of-fit statistical toolkit. IEEE Transactions on Nuclear Science 53 (6), 3834–3841.

May, C., 2006. The FLOSS alternative: TRIPs, non-proprietary software and development. Knowledge, Technology, and Policy 18 (4), 142–163.

Mustonen, M., 2003. Copyleft – the economics of Linux and other open source software. Information Economics and Policy 15 (1), 99–121.

Mustonen, M., 2005. When does a firm support substitute open source programming? Journal of Economics and Management Strategy 14 (1), 121–139.

netcraft.com, 2007. March 2007 web server survey. Accessed March 16, 2007.

Nonnecke, B., Preece, J., 2003. Silent participants: Getting to know lurkers better. In: Leug, C., Fisher, D. (Eds.), From Usenet to CoWebs: Interacting with Social Information Spaces. Springer, Amsterdam, Holland.

Olson, M., 1965. The Logic of Collective Action: Public Goods and the Theory of Groups. Harvard University Press.

O'Mahony, S.C., 2002. The Emergence of a New Commercial Actor: Community Managed Software Projects. Ph.D. thesis, Stanford University. Doctoral Dissertation.

O'Mahony, S.C., 2003. Guarding the commons: How community managed software projects protect their work. Research Policy 32 (7), 1179–1198.

Raymond, E.S., 1999. The Cathedral & The Bazaar, 1 Ed. O'Reilly, Sebastopol, CA.

Roberts, J.A., Hann, I.-H., 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.

Scacchi, W., 2002. Understanding requirements for developing open source software systems. IEE Proceedings – Software 149 (1), 24–39.

Schoonhoven, C.B., 2003. Open source software and the "private-collective' innovation model: Issues for organization science. Organization Science 14 (2), 208.

Shah, S.K., 2006. Motivation, governance, and the viability of hybrid forms in open source software development. Management Science 52 (7), 1000–1014.

Stewart, D., 2005. Social status in an open-source community. American Sociological Review 70, 823–842.

The Economist, 2005. The triumph of the commons. The Economist, pp. 55–56, February 12th.

The Economist, 2006. Open-source business. The Economist. March 16th.

Waring, T., Maddocks, P., 2005. Open source software implementation in the UK public sector: Evidence from the field and implications for the future. International Journal of Information Management 25 (5), 411–428.

Weber, S., 2004. The Success of Open Source. Harvard University Press.

West, J., 2003. How open is open enough? Melding proprietary and open source platform strategies. Research Policy 32 (7), 1259–1285.

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).

Zeitlyn, D., 2003. Gift economies in the development of open source software: Anthropological reflections. Research Policy 32 (7), 1287–1291.