Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Promise of Research on Open Source Software

Georg von Krogh, Eric von Hippel,

To cite this article:
Georg von Krogh, Eric von Hippel, (2006) The Promise of Research on Open Source Software. Management Science 52(7):975-983. https://doi.org/10.1287/mnsc.1060.0560

MANAGEMENT SCIENCE
Vol. 52, No. 7, July 2006, pp. 975–983
ISSN 0025-1909 | EISSN 1526-5501 | 06 | 5207 | 0975

informs

DOI 10.1287/mnsc.1060.0560

The Promise of Research on Open Source Software

Georg von Krogh
Department of Management, Technology, and Economics, ETH Zurich,
Kreuzplatz 5, 8000 Zürich, Switzerland, gvkrogh@ethz.ch

Eric von Hippel
Sloan School of Management, Massachusetts Institute of Technology, 50 Memorial Drive,
Cambridge, Massachusetts 02139, evhippel@mit.edu

Breaking with many established assumptions about how innovation *ought* to work, open source software projects offer eye-opening examples of novel innovation practices for students and practitioners in many fields. In this article we briefly review existing research on the open source phenomenon and discuss the utility of open source software research findings for many other fields. We categorize the research into three areas: 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. We introduce the articles in this special issue of *Management Science* on open source software, and show how each contributes insights to one or more of these areas.

Key words: innovation; user-innovation; open source software; organization; motivation; competition
History: Accepted by Wallace J. Hopp, editor-in-chief.

1. Introduction

Every once in a while, an example comes along that shows important new possibilities so clearly that the world changes for some or all of us. Consider the discovery that life flourishes near deep-sea volcanic vents: Who knew life could exist and even prosper under environmental conditions previously "known" to be so prohibitively hostile? But, once the example had been seen and understood, what a wonderful opening up of new insights and possibilities for life scientists!

Open source software projects represent the same sort of eye-opening example for students and practitioners in many fields. Who would ever have imagined that innovation could flourish under conditions like those? But once we truly see and begin to understand the phenomenon, what a tremendous source of new insights and possibilities for us all! Now many are excitedly exploring the functioning of open source software projects, while others are seeking to extend lessons learned to other areas of innovation.

Open source software is a public good: It's use is nonrival, and it involves a copyright-based license to keep private intellectual property claims out of the way of both software innovators and software adopters—while at the same time preserving a commons of software code that everyone can access (O'Mahony 2003). Open source software is typically created within open source software projects, often initiated by an individual or a group that wants to develop a software product to meet their own needs. Today, a large number of such projects exist and they produce code for many different purposes (one large host of development activity, SourceForge.net, lists in excess of 110,000 projects, in areas ranging from operating system software to software games).

Prior to the emergence and clear success of open source software projects, it was assumed that a requirement to contribute one's innovation to a commons would lead inevitably to the destruction of incentives to innovate, due to free-riding by others on the product of the innovator's labor (e.g., Dam 1995, Granstrand 1999). But once we begin to understand how the open source example functions, the entire fabric of assumptions buttressing the necessity of intellectual property protection regimes, with their well-known negative side effects on the freedom to innovate and adopt (e.g., Heller and Eisenberg 1998), can be examined with fresh eyes (e.g., von Hippel and von Krogh 2003).

Open source software development communities consist of people who contribute to the public good of open source software by writing code for the project. These people conduct their business using a few commonly shared coding languages, such as C++ and Java. Contributors also have access to computing equipment operating on a common standard and they can distribute their creations widely and essentially without cost via the Internet. The result is that anyone and everyone can immediately obtain, test, and observe the value of freely revealed new software code for themselves. These favorable factors

make it possible for us to observe the effects of really widespread, cost-free diffusion of, and peer-to-peer interaction regarding, innovation-related information. What we see is a rapid sharing and collaborative improvement of what is offered within self-forming, open access communities that share a common language.

Open source software development communities centrally involve innovating software users (von Hippel 2001, Kuan 2002, Lakhani and von Hippel 2003, Gacek and Arief 2004). User-innovators benefit from their own expected use of an innovation. In general, no one has to pay them to innovate, and users don't have to make investments that can only be recouped if others adopt what they have developed (von Hippel 1998). Who knew that users would have sufficient incentive to innovate, given their relatively limited returns from in-house use? Surely manufacturers, spurred by the prospect of profiting from an entire market, inevitably dominate the innovation scene? But in open source software projects we see that users are indeed major innovation contributors. Spurred by this example, we can rethink underlying assumptions about innovation. We then discover countervailing advantages available to user-innovators that can offset the manufacturers' advantages of potential market scale: better information on emerging needs, and a certain, even though sometimes small, internal market for what they develop.

Open source software projects have created an institutional alternative to firm-based innovation (Tuomi 2003, Ulhoi 2004).[1] Open source software projects are based on voluntary contributions and involve only very light coordination activities by a central project team (Asklund and Bendix 2002, Kogut and Metiu 2001). Generally, beyond mailing lists and versioning software (e.g., concurrent versions system (CVS)), no roadmaps or milestones or other coordination devices, customarily regarded as mandatory to prevent product development projects from falling into chaos, may be sought or offered. But, as Galileo is said to have murmured after officially recanting his statement that the earth moves around the sun: "And yet it moves!" What is going on here? Once the successful counterexample to conventional wisdom regarding the organization of innovation is seen in open source software projects, our minds are opened to ask exciting new questions, and we will then find exciting new answers.

Research on open source software is helping to define an entirely new model of innovation of relevance to many fields beyond open source software. This model helps us understand how private needs and wants can create public good innovations of major importance to the society and the economy. The necessary conditions for this form of innovation include the proper mix of user-oriented incentives, a community that creates and keeps innovations in the commons, low-cost project coordination mechanisms, and negligible out-of-pocket expenses in diffusing the innovation. These conditions could apply beyond software development, and we are currently witnessing other fields, ranging from cultural products (e.g., Jeppesen and Frediksen 2006) to the biomedical and life sciences (e.g., Duyk 2003), exploring the impact of an open source model. Indeed, open source software research promises to offer insights of great relevance to these efforts.

In the remainder of this introductory article we briefly outline a framework for organizing the research on open source software, and introduce the articles in the special issue along three topical areas: motivation of contributors to open source software; the governance, organization, and process of innovation in open source software projects; as well as the influence of open source software on competitive dynamics. We conclude the article and express our gratitude to all who contributed to this special issue. Next, we turn to a framework for organizing open source software research.

2. Research on Open Source Software

Since the turn of the century, a very impressive body of research on open source software has emerged based in different academic disciplines and drawing on a variety of methodological approaches. In earlier work, we proposed a framework for organizing this research in three areas: motivations of contributors, innovation process, and competitive dynamics. In each area, the open source software phenomenon posed fundamental puzzles that called for *both* extended or entirely new theory *and* novel empirical research. In the call for papers for this special issue, we encouraged very broad contributions from scholars working in diverse fields, and we placed few restrictions on the theoretical perspectives and methods used. The primary interest was to shed extensive light on the phenomenon and thereby help take theory and research one step further. The submissions advanced our understanding of open source software contributors' motives and competitive dynamics, and interestingly, expanded the work on innovation processes to include governance and organizational issues. In what follows, we briefly review existing research in the three areas of this research framework, and show how the articles in the special issue contribute to the current state of knowledge about the open source software phenomenon. Table 1 summarizes the contents of this section.

[1] See also Dalle and Julien (2003) for some of the challenges the open source software movement faces creating institutional alternatives.

Table 1 Research on Open Source Software and the Special Issue Contributions

Some prior contributions	Research focus (examples)	Special issue article and its contributions
Motivations for contributions		
Bergquist and Ljungberg (2001) Dalle and David (2003) Franke and von Hippel (2003) Ghosh et al. (2002) Hann et al. (2006) Hars and Ou (2002) Hertel et al. (2003) Lakhani and von Hippel (2003) Lakhani et al. (2002) Lerner and Tirole (2002) Osterloh et al. (2004) Stenborg (2004) von Hippel and von Krogh (2003) Zeitlyn (2003)	• Individual incentives • Impact of firms' participation on individual motives • Impact of community participation on individual motives • Relationship between incentives and technical design	Roberts et al. • Characteristics of individual motives • The motives of firm's employees engaged in open source software development • Relationship between intrinsic and extrinsic motivation in producing a contribution to an open source software project Bagozzi and Dholakia • Psychological and social factors explaining engagement in open source software user groups (Linux user groups) • Motivation to conduct mundane work in an open source software project Baldwin and Clark • Incentives for developers to join and contribute to a modular open source software architecture • Relationship between an open source software architecture and free riding
Governance, organization, and innovation process		
Franck and Jungwirth (2002) Kogut and Metiu (2001) O'Mahony (2003) Raymond (1999) Dempsey et al. (2002) Gallivan (2001) Koch and Schneider (2002) Nonneke and Preece (2000) Scacchi (2002) West and O'Mahony (2005) Feller and Fitzgerald (2000) Lanzara and Morner (2003) Lee and Cole (2003) Lin (2003) Mockus et al. (2002) Monteiro et al. (2004) von Hippel (2005) von Krogh et al. (2003) von Krogh et al. (2005) Yamauchi et al. (2000)	• Reconciliation of diverse and distributed contributor interests • Governance of project architecture to prevent "forking" • Governance of the public good • Functioning and types of organizations in open source software projects • Roles taken by contributors to open source software projects • Coordination of innovation • Processes of open source software maintenance and development • Factors explaining the evolution of the open source software architecture	Shah • Relationship between governance and innovation in open source software projects • Characteristics of "gated" versus "nongated" communities • Types of tasks attended to by developers in the two communities Grewal et al. • Characteristics of networks between project leaders and projects • Impact of "network embeddedness" of projects and project leaders, on technical and commercial success of open source software projects Kuk • The characteristics of knowledge sharing in open source software development • Adverse impact of extreme concentration of development on knowledge sharing MacCormack et al. • Relationship between project organization and product design in open source software development • Application of Design Structure Matrices
Competitive dynamics		
Bonaccorsi and Rossi (2003) Cusumano and Gawer (2002) Dahlander and Magnusson (2005) Garud et al. (2002) Grand et al. (2004) Henkel (2003) Mustonen (2003) Mustonen (2005) West (2003)	• Impact of open source software on competition in the software industry • Hybrid strategies for melding commercial and open source platforms • Firms' resource allocation to open source software projects • Relationship between firms and open source software projects • Free revealing amongst competitors of improvements to common software platforms	Casadesus-Masanell and Ghemawat • Mixed duopolies in which competing firms and others have heterogeneous objective functions • Interaction between a profit-maximizing firm and a competitor that prizes at zero • The profit-maximizing firm's ability to persist in face of competition from an open source alternative Economides and Katsamakas • The impact of commercial and open source platforms on the evolution of the software industry • Firms' optimal two-sided pricing strategies Bonaccorsi et al. • Empirical data on firms' entry strategies based on hybrid offerings of proprietary and open source software • Factors influencing the degree of firms' openness towards open source software

2.1. Research on Motivations of Project Contributors

Motivations for contributions to open source software projects constitute a core issue in research on the open source software. Early empirical work on this topic documented a range of motives for participation among project contributors, such as fun, enjoyment, reputation building, learning, and the private use value of the software being developed (Lakhani et al. 2002, Hars and Ou 2002, Ghosh et al. 2002). Theory followed, with Lerner and Tirole (2002) proposing that those who contributed code gained private benefit by signaling to prospective employers about their programming skills and thereby getting better jobs or salaries. von Hippel and von Krogh (2003) then made a more general argument about private benefits that could accrue from engaging in the creation of public goods in the form of a "private-collective" model of innovation incentives. Three articles in this special issue move this line of research forward by exploring how motives produce a mix of outcomes, and how various motives interact to produce contributions to open source software projects. In particular, the articles provide new insights on how firm participation, community participation, and technical design relate to the contributors' motivations.

First, as firms have become increasingly important participants in open source projects, a new mix of incentives includes private returns obtained from participation in the open source project *and* related impacts on employees' firm-based careers paths (e.g., Hann et al. 2006; see also Stenborg 2004). The article by Roberts, Hann, and Slaughter (this issue) titled "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: Longitudinal Study of the Apache Projects," explores this matter. Developers' motivations can be intrinsic, satisfying basic human needs for competence, control, and autonomy, or they can be extrinsic, mostly applied by a third party. The authors develop and test a model that relate motivations, participation, and performance of open source software developers. Their data is collected from longitudinal research on software developers in the Apache projects. They find that developers' motives are related in complex ways. For example, many developers are paid to contribute to the Apache projects. These developers are motivated more strongly by status in the project than by the pure use value of the software. In contrast to predictions (e.g., Deci et al. 1999), extrinsic motivations did not displace intrinsic motivations of developers, which was further enhanced by status motivation. In the Apache project, paid participation and status motivation predict above-average developer participation, and interestingly, the use-value motivations predict below-average contribution levels. Finally, the developers' contribution levels positively impact on their performance rankings in the projects to which they contribute.

Second, a conjecture in the literature is that community participation, social motives, and norms relate to individual levels of contribution to open source software projects (e.g., Bergquist and Ljungberg 2001, Zeitlyn 2003, Osterloh et al. 2004). In short, an individual's rewards are in part tied to her participation in the community surrounding the project. Rewards from this participation complement the rewards from the public good innovation, and jointly they outweigh the potential rewards incurred from straightforward free-riding on the public good others produce (von Hippel and von Krogh 2003). Following up this conjecture with empirical research, Hertel et al. (2003) found that the nature of participation matters for individuals' motivation. The authors tested two extant models in the social psychology and sociology literatures. The first model explained the incentives for people to participate in social movements. The second model dealt with motivational processes in small work teams where members work in different places and coordinate their work mainly via electronic media. There was a good fit between both models and the data derived from a survey of 141 contributors to the Linux kernel. In this project, contributors identify with the Linux developer community and are motivated by "pragmatic" wishes to improve their own software but also by group-related factors such as their perceived indispensability for the team with which they are working.

Contributing to this emerging body of work, the article by Bagozzi and Dholakia in this special issue, titled "Open Source Software User Communities: A Study of Participation in Linux User Groups," looks closer at the influence of community on why open source software users participate and remain engaged in user groups. This matter is seen in especially sharp relief when the tasks being performed do not offer great private rewards at face value. For example, one might expect the private reward of learning to be higher for group participants writing code for an open source project relative to the rewards for those contributing field support (Lakhani and von Hippel 2003). Bagozzi and Dholakia frame participation in Linux User Groups (LUG) in terms of group-referent intentional actions, and examine cognitive, affective, and social determinants of participation. Surveying 402 active LUG members representing several user groups in several countries, they find that a combination of social and psychological variables explain LUG member participation. "We-intentions" predict the users' joint LUG participation where each person performs individual actions that contribute to group actions. LUGs are influential and

cohesive communities of Linux users, and LUG members become more engaged in these with increasing experience.

Third, there is a very important and interesting notion emerging in the literature, that the technical design of the open source software relate to the incentives to contribute to the project creating the software. Franke and von Hippel (2003), in their study of the Apache security software project, found that developers who were capable of changing the technical characteristics of the software were significantly more satisfied than noninnovating software users. Dalle and David (2003) theorize that working on visible and technically advanced modules in a software architecture will be more rewarding for contributors than working on obscure and technically trivial modules. The article in this special issue that contributes to this body of work is titled "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?" The authors, Baldwin and Clark, use game theory to draw a link between the design of the software and incentives. They define open source software architectures as modular when the software is organized into distinct components. These components have "option value" when developers have the right but not the obligation to choose among modules that are offered. Baldwin and Clark explore developers' incentives to join and contribute to (or free-ride upon) an open source software project having a modular architecture, and find there is an incentive to contribute.

2.2. Research on Governance, Organization, and the Process of Innovation

In the following we review some existing research on governance, organization, and innovation processes in open source software projects, and then introduce the articles in the special issue that offer new results in this area of research. First, researchers have demonstrated a growing interest in the challenge of governing open source software projects. Open source software projects are based on contributions by many and sometimes thousands of volunteer developers, and "project owners" or "maintainers" (Raymond 1999) must resort to other governance mechanisms than those available to firms that pay developers to work. Franck and Jungwirth (2002) proposed to distinguish these volunteers in two groups, people who expect to receive rewards from contributing to open source software (investors), and people who do not (donators). They argued that open source software projects have succeeded at creating a new governance structure that reconciles the interests of people in both groups. Moreover, project governance is also challenged with self-interested contributors sometimes "forking" the project by developing their own versions of the software (e.g., Raymond 1999). In contrast, firms can closely monitor what their developers do and use career and monetary incentives to keep their work on track (e.g., Sawyer and Guinan 1998). Kogut and Metiu (2001) suggested there is a distinct governance structure in open source software projects that organizes work around an emergent product design and, at the same time, prevents the software project from forking into many fragmented versions of the code base. A related challenge of governance is to keep the software a public good. In her empirical study of six open source software projects including the Apache Web server and the Linux kernel, O'Mahony (2003) identified how communities governed their work product and used social norms, foundations, brands, and other mechanisms to ensure the open source software remained in the commons.

Second, beyond examining forms of governance, an emerging body of research have provided detailed accounts of open source software project organizations (e.g., Gallivan 2001, Dempsey et al. 2002, Scacchi 2002, West and O'Mahony 2005). A striking feature in the organization of open source software projects is the pronounced difference in roles taken by contributors. Koch and Schneider (2002), in their pioneering study of the GNOME project, showed highly concentrated developer activity, with a mean value of 1.8 contributors per file. In an early study, Nonneke and Preece (2000) showed that open source software organizations do not only consist of a few developers listed on a project's website and a larger group of participants in discussion forums, but also of "lurkers" who are passive and peripheral listeners and observers of project activity. Understanding these roles is critical in order to grasp the complexities and intricacies of innovation in open source software.

Third, as mentioned in the introduction to this article, innovation processes in open source software significantly deviate from predictions made by existing innovation theory (e.g., Lee and Cole 2003, Lin 2003, Lanzara and Morner 2003, von Hippel 2005, von Krogh et al. 2005). This fact spurred a great interest among scholars in the detailed, inner-workings of innovation processes. Authors including Feller and Fitzgerald (2000) and Scacchi (2002) provided conceptual frameworks for understanding the unique character of, and the requirements for, innovation processes in open source software. Mockus et al. (2002) and Monteiro et al. (2004) provided rich case studies of the development and maintenance process in Mozilla and Apache, and the open source distribution Gentoo Linux, respectively. In their study of the FreeBSD and GNU GCC projects, Yamauchi et al. (2000) showed how very lean coordination tools, such as mailing lists and CVS, enabled highly effective

development and maintenance of open source software. von Krogh et al. (2003), in their research on the Freenet project, found that the evolutionary design of the open source software hinged on the process by which developers joined the open source software project. Someone who wanted to become a developer in the project had to follow a social "joining script" by making considerable contributions in terms of technical advice and high-quality software files.

In this special issue, four articles advance the research on governance, organization, and innovation process in open source software projects. In her article titled "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development," Shah compares the patterns and reasons for initial and continued developer participation in "nongated" versus "gated" open source software projects. (A gated project is one in which a sponsoring firm retains some control over, and obtains profits from, the commercialization of code developed by project contributors.) Shah finds that users' urgent needs for software are the typical cause for code contributors' initial participation in both types of projects. Most users, once their needs are satisfied, tend to leave the communities. However, in the case of nongated projects, a small subset of participants remain engaged with the project and undertake various necessary administrative tasks within it. In contrast, in the case of gated projects, code contributors tend to *not* take on project support roles, instead leaving those tasks—and costs—to the firm sponsoring and profiting from the project. Shah's work makes a significant contribution to this research area by showing how project governance relate to the process of innovation in open source software projects.

Grewal, Lilien, and Mallapragada authored the article, "Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems." They investigate how the "network embeddedness" of projects and project leaders influences a project's success, where the project success measures are both technical (CVS commits) and commercial (in terms of number of downloads) (see Crowston et al. 2004). Grewal et al. collected data from SourceForge.net, where they start with 10 Perl foundry projects to develop a data set of 108 projects and 490 developers working on these projects. For these projects, the impact of network embeddedness is stronger for technical success than commercial success. The authors suggest that technically successful projects do better at attracting talented developers. However, the attraction of technical talent might be less apparent to users who in turn drive commercial success. Another important finding is that projects with more developers are more technically successful in the later stages of the project's development. The work by Grewal et al. demonstrates that the networks between projects and project leaders are key in the organization of open source software development.

Kuk authored the article, "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List." He explores the extent to which open source organizations are fully participative "bazaars" or rather small, selective groups of people that carry out most of the work. Using an innovative research design, the author gathers data from 128 discussions threads on the KDE mailing list (KDE is a desktop environment included in most Linux distributions). Echoing an idea in the previous work on the governance and organization of open source software projects (e.g., Koch and Schneider 2002), Kuk demonstrates that participants interact strategically in order to expand knowledge sharing, but that extreme concentration of development could reduce knowledge sharing.

MacCormack, Rusnak, and Baldwin improve our understanding of open source software project organization and innovation processes significantly in their article titled "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." An important conclusion from their work is that different modes of organizing development relate to different product designs. The authors use Design Structure Matrices to map dependencies between the elements of a design, and to compare the structure of different designs. They compare the design structure of Linux with the first version of Mozilla (the product of proprietary development at Netscape). They find that Linux had greater modularity and that, later, when Mozilla was released as an open source project it was made more modular by reducing the number of dependencies in the design. The article teaches that analyses of code structures via Design Structure Matrices enable targeted action to reduce unwanted dependencies and increase modularity.

2.3. Research on Competitive Dynamics: The Impact of Free

The public good nature of open source software raises a number of issues for competitive strategy. Among these is the critical question: How do firms seeking to sell products compete with *free*? A number of past contributions have investigated this issue. Bonaccorsi and Rossi (2003) studied how new technologies such as open source software can compete in an environment dominated by commercial technological standards. Based on a simulation, they concluded that under many plausible conditions, open source software and commercial software are likely to coexist, even in the limit. Mustonen (2003) simultaneously modelled the impact of open source (copyleft)

licensing on both the developer environment and the market for software. He showed that although both commercial and open source software can coexist, the (monopolist) commercial software firm must take into account both the scarcity in labour due to open source alternatives and, when pricing her products, the cost for the consumer of implementing the "free" open source alternative.

While direct competition might be widespread in the software industry, some manufacturers also choose to circumvent rivalry with the open source software movement, and cooperate with it (e.g., Mustonen 2005).[2] West (2003) showed significant differences among manufacturer's resource commitments to open source software and their hybrid strategies of melding proprietary and open source platforms. He also proposed a set of conditions where manufacturers may prefer a mix of strategies to the pure-open or pure-closed alternatives. Garud et al. (2002) studied the strategic implications resulting from firms' sponsorship of "open" and "closed" technological platforms, in particular the case of Sun Microsystem's Java software. Following on this work, Grand et al. (2004) studied software firms that used open source software in delivering customized products. They developed a framework of increasing resource allocation to open source software development and showed that, paradoxically, increased "public" investment can lead to great "private" benefits for the open source–oriented firms. Dahlander and Magnusson (2005) worked out a description of the relationships such firms may develop with the open source software communities. Finally, Henkel (2003) studied the role of open source software in the interaction between competitors that had built improvements and extensions to a type of software known as embedded Linux. (Such software is "embedded in" and used to operate equipment ranging from cameras to chemical plants.) These firms freely revealed improvements to the common software platform that they shared and, after a delay, also revealed much of the equipment-specific code they had written. Even under adverse competitive conditions, there may be practical reasons for innovators to freely reveal information.

Three articles in this special issue contribute further to our understanding of competitive dynamics. The article by Casadesus-Masanell and Ghemawat titled "Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows" considers mixed duopolies in which competing firms and others have heterogeneous objective functions (e.g., Merrill and Schneider 1966). The authors analyze the interaction between two competitors; one that prices at zero (or marginal cost) and the other attempting to maximize profits, taking into consideration the dynamics of demand-side learning that enhances the benefits to users, and the proprietary competitor's (Windows) initial market power. The authors demonstrate that due to its capacity to make strategic choices, the profit-maximizing firm will likely persist in the face of competition from an open source alternative. The authors also provide a summary of new data on both governments' promotion and adoption of open source software.

Technology platforms mediate competition in many industries, including computer software (Cusumano and Gawer 2002, West 2003). Economides and Katsamakas's article titled "Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry" investigates how commercial platforms (built by firms) and open source platforms impact the evolution of industries. A firm's optimal two-sided pricing strategy is considered; prices offered to direct users of the platform and to firms developing application technologies based on the platform. When the platform is proprietary, equilibrium prices for the platform, the applications, and the fee for the applications' access to the platform may be below marginal cost. In contrast, when proprietary applications are based on an open source platform, the applications sector of the industry may be more profitable than the total profits of a proprietary platform industry. Another finding is that when a system based on the open source platform with an independent proprietary application competes with a proprietary vertically integrated or disintegrated system, both these proprietary systems may dominate the open source platform in terms of market share and profitability, with the vertically integrated system being dominant.

Bonaccorsi, Giannangeli, and Rossi's empirical research article, "Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry," is a complement to the two articles just discussed and contributes to the work on collaboration between firms and the open source software movement. The authors collect and analyze survey data on 146 Italian software firms. They find that the majority of firms entering the open source software industry adapted to that environment, dominated by incumbent standards, via hybrid offerings of proprietary and open source software. The authors examine the nature of these hybrid business models, and the factors that influence the degree of openness of firms toward open source software.

3. Conclusion and Acknowledgments

In conclusion, we return to the theme of the general utility of the research into open source software and

[2] At a general level, there are many reasons why firms might choose to freely reveal their innovation, such as more rapid and cost-effective diffusion of innovation. For more on this, see Harhoff et al. (2003).

open source software projects. Open source software contributors have pioneered new ideas and practices with respect to licensing of intellectual property and the organization of innovative effort. There is no reason to believe that these practices cannot spread to other areas of economic and social activity. The open source software example has opened the eyes of many researchers and practitioners to new possibilities in the fields of their interest. In this article, we reviewed some existing contributions to open source software research and introduced the articles of this special issue along three areas: motivations of contributors, governance, organization, and innovation process, as well as competitive dynamics. We as editors are proud to have participated in publishing some of the wonderful research studies related to open source software completed to date, and we look forward to the many more to come!

Acknowledgments

With respect to specific acknowledgments, we would first and foremost like to thank the many authors who sent in their manuscript for review. Many were young academics who contributed excellent work. We are extremely grateful to Wallace Hopp for giving us the opportunity to edit this special issue of *Management Science* on open source software. This provided great opportunities to an active community of researchers working on open source and user-innovation. Many thanks also to Betty Martin who helped manage all the administrative processes from the side of the journal. We are also very grateful to Stefan Haefliger for his invaluable assistance in managing a very high number of submissions and the complex editorial process. We are very grateful to the 121 reviewers who assisted in evaluating the papers for the special issue. Finally, we thank our colleagues at the Sloan School of Management, MIT, Department of Management, Technology, and Economics, ETH Zurich, and the University of St. Gallen for their encouragement, and for the financial support from the Swiss National Science Foundation (Grant 100012-101805) to this project and related research.

References

Asklund, U., L. Bendix. 2002. A study of configuration management in open source software projects. *IEE Proc.* **149**(1) 40–46.

Bergquist, M., J. Ljungberg. 2001. The power of gifts: Organizing social relationships in open source communities. *Inform. Systems J.* **11**(4) 305–320.

Bonaccorsi, A., C. Rossi. 2003. Why open source software can succeed? *Res. Policy* **32** 1243–1258.

Crowston, K., H. Annabi, J. Howison, C. Masango. 2004. Towards a portfolio of FLOSS project success measures. Working paper, MIT, Cambridge, MA. Retrieved April 4, 2006, http://opensource.mit.edu/papers/crowston04towards.pdf.

Cusumano, M., A. Gawer. 2002. The elements of platform leadership. *MIT Sloan Management Rev.* **43**(3) 51–58.

Dahlander, L., M. G. Magnusson. 2005. Relationships between open source software companies and communities: Observations from Nordic firms. *Res. Policy* **34**(4) 481–493.

Dalle, J.-M., P. A. David. 2003. The allocation of software development resources in "open source" production mode. Discussion Paper 02-27, Stanford Institute for Economic Policy Research, Stanford, CA.

Dalle, J.-M., N. Julien. 2003. "Libre" software: Turning fads into institutions? *Res. Policy* **32**(1) 1–11.

Dam, K. W. 1995. Some economic considerations in the intellectual property protection of software. *J. Legal Stud.* **24**(2) 321–377.

Deci, E. L., R. Koestner, R. M. Ryan. 1999. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psych. Bull.* **125**(6) 627–668.

Dempsey, B. J., D. Weiss, P. Jones, J. Greenberg. 2002. Who is an open source developer? *Comm. ACM* **45**(2) 67–72.

Duyk, G. 2003. Attrition and translation. *Science* **302**(5645) 603–605.

Feller, J., B. Fitzgerald. 2000. A framework for understanding the open source software development paradigm. *ICIS Conf. Proc., Atlanta*. Association for Information Systems, Atlanta, GA, 58–69.

Franck, E., C. Jungwirth. 2002. Reconciling investors and donators: The governance structure of open source. Working Paper 8, University of Zurich, Zurich, Switzerland.

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

Gacek, C., B. Arief. 2004. The many meanings of open source software. *IEEE Software* **21**(1) 34–40.

Gallivan, M. J. 2001. Striking a balance between trust and control in virtual organizations: A content analysis of open source software case studies. *Inform. Systems J.* **11**(4) 277–304.

Garud, R., S. Jain, A. Kumaraswamy. 2002. Institutional entrepreneurship in the sponsorship of common technological standards: The case of SUN Microsystems and Java. *Acad. Management J.* **45**(1) 196–214.

Ghosh, R. A., R. Glott, B. Krieger, G. Robles. 2002. Free/Libre and open source software: Survey and study. Working paper, International Institute of Infonomics, University of Maastricht, Maastricht, The Netherlands.

Grand, S., G. von Krogh, D. Leonard, W. Swap. 2004. Resource allocation beyond firm boundaries: A multi-level model for open source innovation. *Long Range Planning* **37**(6) 591–610.

Granstrand, O. 1999. *The Economics and Management of Intellectural Property.* Edward Elgar, Cheltenham, UK.

Hann, I., J. Roberts, S. Slaughter, R. Fielding. 2006. An empirical analysis of economic returns to open source participation. Working Paper 2006-E5, Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA.

Harhoff, D., J. Henkel, E. von Hippel. 2003. Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations. *Res. Policy* **32**(10) 1753–1769.

Hars, A., S. Ou. 2002. Working for free? Motivations for participating in open-source projects. *Internat. J. Electronic Commerce* **6**(3) 25–39.

Heller, M., R. Eisenberg. 1998. Can patents deter innovation? The anticommons in biomedical research. *Science* **280** 698–701.

Henkel, J. 2003. Software development in embedded Linux: Informal collaboration of competing firms. W. Uhr, W. Esswein, W. Schoop, eds. *Proc. 6. Internat. Tagung Wirtschaftsinformatik Physica*. Verlag, Heidelberg, Germany.

Hertel, G., S. Niedner, S. Herrmann. 2003. Motivation of software developers in open source projects: An internet-based survey of contributions to the Linux kernel. *Res. Policy* **32**(7) 1159–1177.

Jeppesen, L. B., L. Fredriksen. 2006. Why firm-established user communities work for innovation: The personal attributes of innovation users in the case of computer-controlled music. *Organ. Sci.* Forthcoming.

Koch, S., G. Schneider. 2002. Effort, cooperation, and coordination in an open source software project: GNOME. *Inform. Systems J.* **12**(1) 27–42.

Kogut, B., A. Metiu. 2001. Open source software development and distributed innovation. *Oxford Rev. Econom. Policy* **17**(2) 248–264.

Kuan, J. 2002. Open source software as lead-user's make or buy decision: A study of open and closed source quality. Working paper, Stanford Institute of Economic Policy Research, Stanford University, Stanford, CA.

Lakhani, K., E. von Hippel. 2003. How open source software works: "Free" user-to-user assistance. *Res. Policy* **32**(6) 923–943.

Lakhani, K., B. Wolf, J. Bates, C. DiBona. 2002. The Boston Consulting Group hacker survey. Boston Consulting Group Report. Retrieved April 4th, 2006, http://www.osdn.com/bcg.

Lanzara, G. V., M. Morner. 2003. The knowledge ecology of open source software projects. *19th EGOS Colloquium, Copenhagen, Denmark*.

Lee, G., R. Cole. 2003. From a firm-based to a community-based model of knowledge creation. *Organ. Sci.* **14**(6) 633–649.

Lerner, J., J. Tirole. 2002. The scope of open source licensing. Working paper, Harvard Business School, Boston, MA.

Lin, Y. 2003. The institutionalization of hacking practices. *Ubiquity* **4**(4) 18–24.

Merrill, W., N. Schneider. 1966. Government firms in oligopoly industries. A short-run analysis. *Quart. J. Econom.* **33** 400–412.

Mockus, A., R. T. Fielding, J. Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Trans. Software Engrg. Methodology* **11**(3) 309–346.

Monteiro, E., T. Oesterlie, K. R. Rolland, E. Royrvik. 2004. Keeping it going: The everyday practices of open source software. Working paper, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway.

Mustonen, M. 2003. Copyleft: The economics of Linux and other open source software. Discussion Paper 493, Department of Economics, University of Helsinki, Helsinki, Finland.

Mustonen, M. 2005. When does a firm support substitute open source programming? *J. Econom. Management Strategy* **14**(1) 121–139.

Nonneke, B., J. Preece. 2000. Lurker demographics: Counting the silent. *Proc. SIGCHI Conf. Human Factors Comput. Systems*. Association for Computing Machinery, ACM Press, New York, 73–80.

O'Mahony, S. 2003. Guarding the commons: How community managed software projects protect their work. *Res. Policy* **32** 1179–1198.

Osterloh, M., S. Rota, I. von Wartburg. 2004. Open source: New rules in software development. Working paper, University of Zurich, Zurich, Switzerland.

Raymond, E. 1999. *The Cathedral and the Bazaar*. O'Reilly, Sebastopol, CA.

Sawyer, S., P. J. Guinan. 1998. Software development: Process and performance. *IBM Systems J.* **37**(4) 552–569.

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

Stenborg, M. 2004. Explaining open source. Working Paper 947, The Research Institute of the Finnish Economy, Helsinki, Finland.

Tuomi, I. 2003. *Networks of Innovation: Change and Meaning in the Age of the Internet*. Oxford University Press, Oxford, UK.

Ulhoi, J. P. 2004. Open source development: A hybrid in innovation and management theory. *Management Decision* **42**(9) 1095–1114.

von Hippel, E. 1998. Economics of product development by users: The impact of sticky local information. *Management Sci.* **44**(5) 629–644.

von Hippel, E. 2001. Innovation by user communities: Learning from open source software. *MIT Sloan Management Rev.* **42**(4) 82–86.

von Hippel, E. 2005. Democratizing innovation: The evolving phenomenon of user innovation. *J. Betriebswirtschaft* **55**(1) 63–78.

von Hippel, E., G. von Krogh. 2003. Open source software and the "private-collective" innovation model: Issues for organization science. *Organ. Sci.* **14** 209–225.

von Krogh, G., S. Spaeth, S. Haefliger. 2005. Knowledge reuse in open source software: An exploratory study of 15 open source projects. *Proc. 38th Annual Hawaii Internat. Conf. System Sci.*, IEEE, Los Alamitos, CA.

von Krogh, G., S. Spaeth, K. Lakhani. 2003. Community, joining, and specialization in open source software innovation: A case study. *Res. Policy* **32** 1217–1241.

West, J. 2003. How open is open enough? Melding proprietary and open source platform strategies. *Res. Policy* **32** 1258–1286.

West, J., S. O'Mahony. 2005. Contrasting community building in sponsored and community founded open source projects. *Proc. 38th Annual Hawaii Internat. Conf. System Sci.*, IEEE, Los Alamitos, CA.

Yamauchi, Y., M. Yokozawa, T. Shinohara, T. Ishida. 2000. Collaboration with lean media: How open source software succeeds. *ACM 2000 Conf. Comput. Supported Cooperative Work, Philadelphia*. ACM Press, New York, 329–338.

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