ELSEVIER

Long Range Planning 37 (2004) 591–610

LRP long range planning

www.lrpjournal.com

Resource Allocation Beyond Firm Boundaries:

A Multi-Level Model for Open Source Innovation

Simon Grand, Georg von Krogh, Dorothy Leonard and Walter Swap

Successful technological innovation depends upon marshalling sufficient knowledge resources to support continuous discovery, knowledge creation and technical development. Current perspectives emphasize innovation occurring either within firm boundaries or in the public arena. Open Source [OS] software development represents a third mode, where privately funded efforts contribute to the creation of a public good, which may presage future models of innovation. The authors develop a four-level management model of increasing private resource allocation based on a detailed discussion of how and why software and IT firms engage in OS development where, paradoxically, increased 'public' investment can lead to greater 'private' benefits. But each level implies greater outlay of private resources and increased dependency upon publicly available knowledge assets, so managers will need to select their firm's appropriate level of engagement carefully. Successful development of such knowledge entails understanding the nature of OS innovation and the distinction between freely available explicit knowledge and firms' privately retained tacit knowledge, participating in the dynamic cumulative process of gift exchange inherent in acceptance as a relevant player in an OS community, and optimising firm-specific engagement over the four levels of investment and involvement to establish the conditions for knowledge creation and appropriation.

0024-6301/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.lrp.2004.09.006

Introduction

Research questions

Technological innovation is a risky undertaking, requiring the allocation of financial and knowledge resources under uncertain conditions. Optimising their resource allocation for technological innovation is a constant challenge for firms in a range of technology-based industries which draw upon multiple knowledge sources including internal research and development activities, research conducted by partner organizations and scientific knowledge available through public innovation.

In our view, the Open Source [OS] software movement, which has gained increasing attention over the last few years, suggests an intriguing new model for resource allocation that combines knowledge sources inside and outside the firm in non-traditional ways. We argue that firms gain valuable knowledge-creation options by tapping into knowledge assets beyond firm boundaries, which at the same time brings particular risks and challenges. This leads to two questions:

Research Question 1: How and why do firms engage in the creation of public and private knowledge inside and outside the firm; and

Research Question 2: What factors must firms consider when making their resource allocation decisions.

At first glance such opportunities resemble those that have always been available to technology firms drawing upon publicly available scientific knowledge. However, there is a significant difference in the OS context, where individuals and firms from the private domain produce source code at their own expense, which is then made freely available to collaborators and competitors alike. And unlike most scientific knowledge, this source code can readily be used to develop technical solutions, offering both user and developer firms the chance of immediate financial gains. Some conflicts of interest would seem to obtain, especially as firms are crossing their boundaries in the significantly important area of intellectual property, leading to a game that managers and developers cannot entirely own or control. Yet it is the experience of the firms in our research that they accrue intangible knowledge assets (and thus innovative capabilities) commensurate with their contributions to the accumulation of public knowledge.

As an innovation mode, the OS software development movement exhibits some unusual characteristics in terms of enabling knowledge creation beyond firm boundaries:

- It represents a unique combination of private and collective aspects of innovation and knowledge. While the source code is freely available as public knowledge, the learning and expertise developed remains as privately held, intangible knowledge with the developers and the developing firms.
- OS software development represents an unusual collaborative effort where skill and adherence to a philosophy are the entry qualifications to a community where developers work together beyond firm, sector and national boundaries to co-create a public good
- Engagement with the OS community is a highly dynamic and cumulative process of gift exchange, both at individual and at a firm level, which goes considerably beyond simple 'gift giving' as it is normally referenced.[1]

Engagement with the OS community is a highly dynamic and cumulative process of gift exchange

These particularities are studied in the light of a series of exploratory case studies and secondary data analysis on whether, how and why software and IT firms allocate resources to OS innovation,

as well as considerations based on recent research into the creation and coordination of knowledge assets. Such empirical research and theoretical considerations leads us to identify four levels of firm response to the OS movement, which we describe as a 'ladder' of increasing resource allocation and involvement, with each higher 'rung' of involvement leading to higher levels of innovation capability, but entailing substantial resource allocation at the lower levels. While there are significant knowledge barriers to joining the community, benefits are commensurate with contributions. Acceptance as a core OS developer requires both considerable prior investments as a precondition of entry and major ongoing commitment to the evaluation of existing code and the development of valuable new code to remain involved and relevant. But, once accepted, core players benefit from substantial learning-by-doing and the subsequent creation of tacit knowledge and expertise.

We consider the four-level management model presented here as an attempt to understand firm-level engagement in Open Source (and thus fill an important gap in the existing OS literature), and a first conceptual step towards building a general model of resource allocation to technological innovation beyond firm boundaries. Although studied here in the particular OS innovation context, this cumulative hierarchy of resource allocation may also apply to other sectors sharing similar environmental conditions, and we introduce some first ideas for generalizing these insights to other industry, business and technology contexts.

The article is structured as follows. First, we discuss the context of recent research on OS software development, in particular firm-level research on OS innovation. We then introduce our four-level management model, and show how it advances our understanding of firm involvement and resource allocation in OS innovation, and challenges the ideological aura surrounding OS by showing how it can be rational for firms to innovate in this way. Thirdly, the model is illustrated by insights from case studies and secondary data from research on OS software development ventures. Finally we discuss our empirical insights as regards resource allocation and knowledge creation in the OS context, the potential generalisability of our findings, and implications for management and for further research.

Existing literature

Most research on OS software development has focussed on individual developers and particular development processes and projects.[2] There are only a few studies that deal with economic activities and incentives at the level of the firm. Two studies look at the competitive dynamics between open and closed source software. Casadesus-Masanell and Ghemawat analyse the interaction between for-profit (Microsoft) and not-for-profit (Linux) competitors, and examine the specific welfare implications of such interactions,[3] while Bonaccorsi and Rossi study the competitive rivalry between commercial and OS software, finding that the two can co-exist under many different conditions.[4] While these studies are important for understanding competition, they separate the firm from OS and do not deal with the specific engagement of firms in the OS movement itself. Dahlander draws on the resource-based view of the firm arguing that firms may experiment with various models of profit generation, often settling on a service model. In addition, he suggests that OS software allows firms to reap first-mover advantages and network externalities.[5] In a similar vein, West gives an historical account of how IBM, Apple and Sun Microsystems used OS Linux to develop their platform strategies and Garud, Jain and Kumaraswamy study how Sun Microsystems helped to create a technological standard (Java).[6]

In order to explore the firm-level implications of OS software development, it is important to put OS innovation in a broader context of models of technological innovation. Two modes of innovation are prevalent in both management practice and research today.[7]: First, *private innovation* assumes that returns to the firm result from private goods and efficient regimes of intellectual property protection taking place within firm boundaries or as part of a firm's web of partnerships.[8] Second, *collective innovation* assumes that firms collaborate with public institutions such as universities to produce a public good, where public access to the results and transparent communication of the research and development process are inherent.[9] *Open Source innovation* is

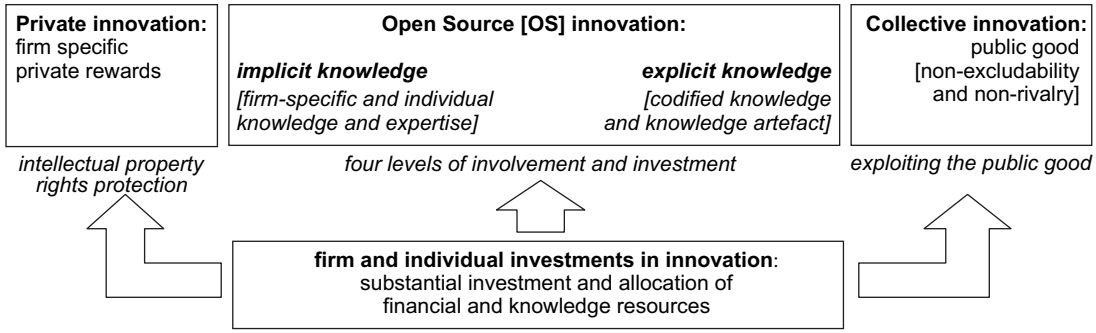

Figure 1. Resource allocation to Open Source innovation

an important new option for firms as a mode that contains elements of both private and collective innovation, (as visualised in Figure 1). In particular, it means that, although the software code developed must be publicly released, the firm-specific individual knowledge and expertise accumulated through the engagement in OS software development remains with the developing firms or individuals.

To date, there is no management framework available supporting the allocation of financial and knowledge resources to OS innovation or steering such allocations according to costs and benefits offered. Allowing software engineers to participate in OS software development projects may be an important incentive for a firm, but it is unclear how willing firm are to support the participation of (often their best and most skilful) senior people in OS projects.

while software code developed must be publicly released, the firm-specific individual knowledge and expertise remains with the developer.

Four-level model of resource allocation

The model

Given the lack of conceptual work on firm-level engagement in Open Source innovation, the article relies on an exploratory study of published information about major OS software development projects and on studies about the interplay of OS projects with the software development strategies of major IT corporations. The article also benefits from case studies of custom software ventures' successful commercialisation of scientific research and technological innovation. These studies note considerable reliance on open standards and high levels of engagement in OS projects as part of involvement in state-of-the-art software development. It is fair to conclude that a wide spread of commercial software interests see themselves as a part of the OS movement. (see Appendix for fuller details)

By definition, technological innovation involves a process of discovery, which can range from a limited search for ways to best implement a new tool, process, product or service, all the way to a business built around the continuous discovery of radically new knowledge as an internal capability. From a managerial perspective, firms must choose the extent to which they will rely upon others' innovative capabilities or will develop their own. That resource allocation decision is perhaps most stark when competitors have access to the same basic knowledge as a public good, and firms must decide how to convert such knowledge into a proprietary advantage. When the public good is generic scientific knowledge, the distance between idea and profitable product is so great that only substantial internal investments in its development will yield profits, since no firm

can use the knowledge in the form in which it exits the research facility. But OS code is much closer to a potentially profitable set of knowledge assets, and managers have a wider range of options for exploiting this form of public good.

The internal dynamic of cumulative knowledge creation increases in terms of both costs and benefits for (public) communities and firms.

In OS software development, software firms, developers and users co-exist in a symbiotic relationship, and while their motives may be different, such cooperation can benefit them all. Firms are rewarded when users innovate and share their innovations with the firm, providing important information about emerging needs and technical ideas. External users hope this cooperation will provide them with better products as well as facilitating their learning about new technologies jointly with the firm. Nevertheless, innovation based on knowledge assets beyond firm boundaries presents the challenge of mobilizing the resources needed to experiment with markets and technologies without being able to use hierarchical or incentive systems to shape the free engagement of OS developers In the OS realm, where financial incentives are not the primary driver, those resources are primarily time and people.[10] At the different levels of the model, it is therefore important to understand the logic of the mutual benefits and joint interests of all parties involved, as well as the inherent dynamic and cumulative nature of these interactions leading from one level to the next.

From a software and IT firm perspective, we suggest four levels of resource allocation for exploiting the opportunities offered by OS software (illustrated in Figure 2), ranging from a modest investment in developing knowledge about OS all the way to a 'bet-the-firm' investment. Each successively higher level encompasses and requires the kinds of knowledge developed at the lower levels, so there is a progression both in investment and in the internal capabilities developed.[11]

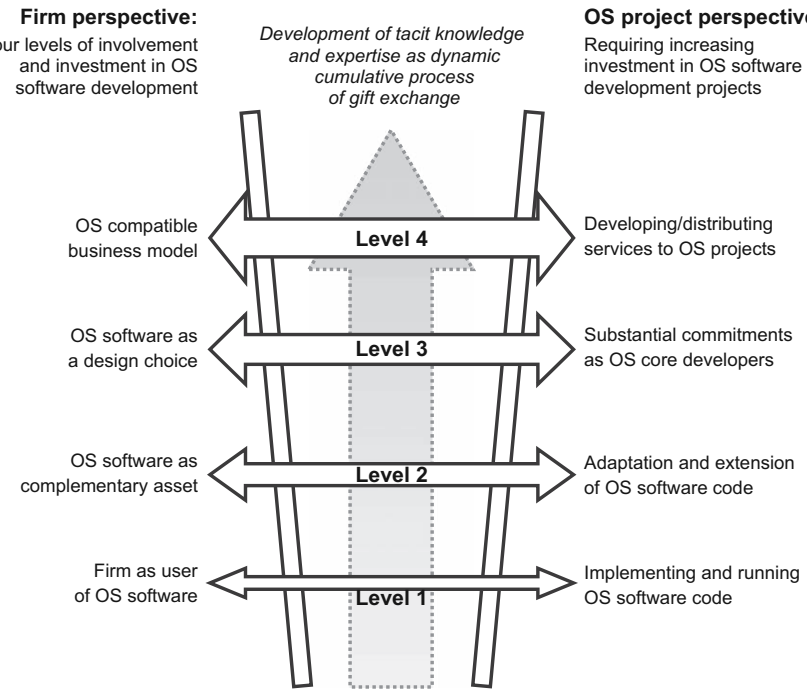

Figure 2. The Ladder : A four-level management model of resource allocation

Successful technological innovation hinges on careful balancing risks and rewards. Research in such areas must contribute to the development of a framework that aids managers in their resource allocation decisions, bringing these risks and rewards to the forefront of attention. Thus each level is associated with certain costs and benefits, as well as conditions under which it can be expected to work, given the particular mechanisms and patterns characterizing OS software development.

There is an inherent progression in levels from the exchange of minor contributions and investments at Levels 1 and 2 to the foundation of the entire company's strategy and business model on OS investments at Levels 3 and 4. Paradoxically (and providentially), such increased investment beyond a firm's boundaries can lead to greater benefits for the investing firm. Each level implies not only greater outlay of private resources and an increased dependency upon publicly available knowledge assets, but at the same time ensures greater operational benefits as well as extended firm-specific learning and knowledge creation opportunities. Substantial resource allocation to OS innovation at lower levels is thus a precondition for being able to enter higher levels of OS innovation. This internal dynamic of cumulative knowledge creation, which increases in terms of both costs and benefits for both communities and firms, is reflected in the cumulative nature of our multi-level model.

Resource allocation at different levels

This section identifies the underlying mechanisms driving successful engagement and discusses the specific costs and benefits at each level, as well as exploring the preconditions for moving from one level to the next.

Level 1: the firm as user of Open Source software

At Level 1, the firm is primarily a user rather than a developer, exploiting the opportunities provided by publicly available OS software code. At this level, users need to allocate enough development resources to implement and run the software code on their IT infrastructure, as well as provide occasional feedback and code patches to the OS community. Their only formal and legal obligation is to observe the OS licenses governing software; that is, firms can function like any other OS software user, with no need to be otherwise active in the OS realm.

Such use confers a number of benefits, as OS software has proven to be technically sophisticated, scalable, reliable and cost effective even in a corporate environment. Thus, despite strong competition from commercial developers (such as Microsoft and Netscape) Apache software is still used by over 60% of the web servers which are the backbone of the World Wide Web infrastructure.

OS software modules can be incorporated with little or no change into otherwise proprietary software suites, thus reducing the costs of product development for custom software houses, whose clients increasingly request the use of OS software code to simplify future changes to applications. Such 'transparency' is a primary advantage of OS software, ensuring continued accessibility for users and installation specialists, and avoiding the potential for 'lock-in' limiting future development associated with proprietary software.

Using OS software at Level 1 is not completely costless. Technological expertise is needed for installation and integration into existing IT environments, and these competencies must be generated in-house or in collaboration with specialized software firms. Some firms remain unconvinced that re-orienting their software to align with OS is worthwhile, or of its stability and security. However, other companies (such as Unilever and IBM) are very open about the use of OS software or even proactively require it to be considered: such organisations are very important advocates for the establishment of OS software in the corporate world.

Level 2: Open Source software as a complementary asset

Some manufacturers, including IBM and Sun Microsystems, configure and sell their PCs and servers etc with OS software: in such cases OS code can be considered a complementary asset.[12] To produce and deliver new products, technological innovation often demands the use of related assets that may have alternative uses, ones that typically lie downstream in the production process.[13] Such

complementary assets are manufacturer-specific, and software developed from OS code can be applied across in different computer environments to earn downstream rents in the same way as memory chips and other hardware items.

OS software is also a complementary asset for custom software houses. Its often modular architecture allows them to 'borrow' it to avoid 'reinventing the wheel' and thus optimise both their own and their customers' allocation of resources. The context of open standards also favours the continuous integration of newly available OS software.[14] As the CEO of one of the custom software houses argues, '*… [our approach] is characterized by the use of Internet protocols, which almost automatically implies open standards, not proprietary software. This leads to a high percentage of OS components in our solutions, implying that we work on Unix, concerning platforms…Wherever OS software components do exist, be it an operating system like Linux or other stuff, we try to use them in our projects*'.

The development of OS software as a complementary asset can require major investment, as extensive development may be needed to adapt code initially developed for other purposes to new hardware. Sony recently launched a Linux kit for its Play Station 2 allowing users to transform the game console into a Linux workstation. While in principle Play Station 2 is open to porting other software, Sony chose to allocate considerable resources to the development of specific software related to Linux (instead of a proprietary operating system). Using a public good as a complementary asset requires internal investments, as the firm adapts internal practices and thinking to the external style and development philosophy of established OS processes. As a Chief Operating Officer of a custom software house argues, '*… the Internet meant for us from the beginning that [software development] is a self-organizing organism. There is no central [authority who] can say 'it must be done like this.' Rather, it is a group of reactive cells which at the same time work on the same idea and which are qualified by their openness with respect to technology standards and protocols*'. Establishing a common knowledge base is key to facilitating the combination of external and internal assets.[15]

As this quote suggests, level 2 firms investing in OS software as a complementary asset build more internal development and innovation capability than level 1 firms, in addition, of course, to using the code itself to develop a complementary asset and adapt its own assets accordingly. In sum, the increased exploitation of OS software code as complementary asset requires an active engagement in OS at Level 1, and thus increased commitment of resources to development and the concomitant increase in knowledge about OS software.

Level 3: Open Source software as a design choice

On January 23, 1998, Netscape announced that it would give away its Navigator browser and release the source code for the next generation of its communicator suite, which is closely integrated with both Java and HTML. These languages promote openness because they run across different computer platforms effectively. Netscape's intention was to mobilize individual private developers to contribute to its software development and speed up diffusion of the browser.[16] The body of source code behind Netscape Communicator is Mozilla, first released in June 2001 and developed under the OS model. Other large computer hardware and software corporations have followed suit, turning to the OS model for software development. Firms sell their computer hardware with OS software and in return contribute developer capacity, existing software, promotion and funding to OS projects.

Level 3 of our four-level management model represents a step change increase in the level of internal investment in OS software, to the point where it becomes a design choice for the way the firm develops new software. OS philosophy requires a high degree of openness to contributors and transparency about discussion of process and the resultant code as well as respect for internal and external knowledge sharing and awareness of property rights issues among both firm developers and their customers. In order to generate this awareness, one CEO points out that he proactively 'asks' his developers '*… to experiment relatively freely with [the new opportunities arising from the OS movement] and to develop new stuff together with customers.*'

(Given) network externalities or compatibility issues, products based on standard cross-platform software increase in value as more adopters choose them.

When OS is thus integrated into the firm's processes and the mindset of its developers, the technology becomes increasingly instrumental to the firm's business and interaction with customers. Firms will benefit from choosing OS software designs depending upon the extent to which they compete in an environment dominated by technological standards. A dominant piece of software (such as an operating system) decreases uncertainty associated with technological changes for both hardware and software manufacturers, as it can effectively 'lock in' both customers and suppliers.[17] When network externalities or compatibility issues are important, products based on such standard cross-platform software (such as HTML and Java) will increase in value as more adopters choose them.

Level 3 firms exploiting the potential for OS software as a development standard must make significant contributions to software development across a variety of OS projects, requiring significant investments in innovation and development, as well as in coordination and communication. As we know from in-depth studies of their internal organization by von Krogh, Spaeth and Lakhani, participants in OS projects are highly aware of differences in the quality of developers' work. The projects are governed by core groups of selected developers who have write-access to integrate software code into an official version of the software: all other developers must submit code for approval to the core group. As Kohanski reports, in order to use available OS code to solve a specific problem, software developers need considerable expertise that allows combination, adjustment and adaptation of the source code in the project.

Thus public release of the source code co-developed by many contributors coincides with the development of firm-specific knowledge and expertise that is *not* public. As one interviewee in our study of software houses commented, '*… we do not have any problems handing over our source code to the customer, since in fact he cannot do anything with the code, because he does not have the knowledge and the systems to do anything with it.*' While the source code for an OS software project (as well as the information exchange among the developers) are freely availably to both particular customers and the OS community as a whole, the specific expertise needed to create and further develop such code is mostly private to individual developers. Much of the value inherent in the collaborative development effort obtains from this tacit dimension of knowledge retained by the people directly involved, which are firm specific knowledge assets.[18] Thus, firms active at Level 3 benefit by growing an embedded capability to innovate within an emerging standard platform — and since OS software continues to evolve as a de facto standard, firms cannot fully exploit it without continuing to contribute to its progress.

Firms that develop such internal assets through active participation in OS projects may also have an advantage in attracting and retaining outstanding individual contributors, potentially enriching the firm's proprietary capabilities. One CEO concluded that '*[the opportunity to freely work in OS contexts] has been a main driver for a lot of people to join our company. [To work in this context is seen as] really working with state-of-the-art instruments and to develop in close cooperation with state-of-the-art people around the world. We got excellent people from established corporations, since they thought 'in this company I can talk to other people, exchanging ideas, sharing information and know-how, teaching others how to improve their work, and constantly learning something myself''*.

In sum, there are a number of benefits to a Level 3 engagement with OS software development. Such firms can take advantage of a global community of highly skilled software developers to augment their internal resources, and, by allowing some of their employees to join this global clique, they enhance internal knowledge assets - particularly intangible ones. Those rich in such assets grow richer by attracting and retaining talent.

In parallel, however, there are certain conditions necessary to operate at Level 3, including involvement at the lower levels. Because firms use OS software and developed OS capabilities as a complementary asset to bundle with some of their solutions, their customers understand the nature of property rights in the context of OS software. That understanding is a prerequisite for a substantial inclusion of OS software components at Level 3 - especially for custom software houses. As one CEO explained: *'If we work based on pre-existing components, [if we] use this or that component that we developed for another project, the customer does not exclusively own these components*'. Lower-level activities are an important precondition for generating the technological expertise, the reputation, the customer expectations and the firm-specific expertise necessary to benefit from the resource allocation to OS software development and innovation at Level 3 and 4.

At Level 4, OS software moves from being a design choice for particular projects to the design choice for a firm's overall business model.

Level 4: Open Source-compatible business model

At Level 4, OS software moves from being a design choice for particular projects to the design choice for a firm's overall business model. Firms such as Red Hat or SuSe have built businesses around product lines that augment OS software. These firms do not realize returns from selling proprietary software protected by commercial licenses but by distributing and adding services to software protected under the OS license. Such firms, with access to software code written by many developers across the Internet (using their own resources) will obviously benefit from lower software development costs than those associated with the traditional business model. By the same token, exploration and experimentation costs are normally considerably lower for level 4 firms.

The firms mentioned above provide services around the operating system GNU/Linux. Even though the Linux license allows any user to download, install and modify the software freely, it is quite cumbersome to handle without extensive skills in programming. After extensive market feedback, Red Hat realized that this difficulty represented an opportunity to meet customer needs, and Red Hat's revenues are now partly generated from selling CD ROMs to assist with Linux installation.[19] The firm also provides services such as management of Linux-based computer networks and installation and training services. Whereas Linux software in general is in constant drift, Red Hat's Linux software follows a versioning system that allows its customers to plan the integration of software and hardware. Over the last five years, several firms have been founded on similar ideas. However firms that have attempted to combine non-OS products with Linux have met with considerable resistance from the OS community.[20] This opposition underscores the necessity for firms to be sensitive to the social norms of the independent community when developing a business based on OS technologies.[21]

At this top level, firms must contribute significantly to the OS software. For instance, Red Hat is a major contributor to Linux software development by having developers release software files under the OS license. Bob Young (company co-founder), calls this Red Hat's *'commitment to OS software',* And the company's advanced development laboratory focuses on developing a desktop environment for Linux. One of the complaints often heard about OS software is that, while it is a great model for building operating systems and other challenging software, it does not as readily support development of middleware and application software, perhaps because high-power developers tend to be less interested in less challenging or prestigious developments. Yet, for the diffusion of the Linux operating system, application software is key, and such development is likely to bring companies like Red Hat closer to those customers for whom it provides services, allowing it to better explore users' needs and technologies. Young further comments that the firms' developers

(like other Linux developers), are themselves customers of the infrastructure that make Linux development possible over the Internet, as well as being regular users of the software product, and so look for pragmatic solutions to their problems.[22]

Working on an innovation in concert with its users has the benefit of automatically bringing the firm closer to that user community. Constant interaction with the community can help a firm avoid drifting away from user needs and devoting internal resources to the 'pet projects' of developers. Thus, since the OS community directs the evolution of the base product (the software code), a firm operating at Level 4 is more market-driven.

As suggested above, in order to benefit from OS software at Level 4, firms must give special attention to intellectual property. Many Free and OS licenses guarantee that the software is freely available to all and that one user's utility of the software does not diminish that of another user. In effect, because users agree that contributing to a public good cannot be withheld from any user, developer or firm, a return appropriation from investment in OS software code through the traditional mechanism of intellectual property rights cannot be secured. As OS licenses guarantee the rights of future users, the OS community faces the particular challenge of preventing the appropriation of work.[23]

Businesses operating at Level 4, identifying revenue streams and building businesses based on a 'public good' technology must break with the conventional 'intellectual property rights' business models. However, our analysis demonstrates exploitation at this level is possible as many potential users will need expert knowledge to benefit directly from OS software in firm-specific environments. Moreover, software firms adopting the level 4 business model who actually contribute to the OS software development process (and thus abide by their OS license agreement), will be developing the internal knowledge and competence base to then exploit their input to its full potential.

As mentioned earlier, the internal dynamic of knowledge creation explains the cumulative pattern of costs and benefits for firms' involvement in OS Innovation, as illustrated in Table 1.

Discussion of empirical insights and management model

Resource allocation patterns and business model implications

To best of the authors' knowledge, this article is first to offer a managerial framework for resource allocation decisions for OS innovation. While the existing literature on OS innovation follows the ideological premise of a clear-cut dichotomy between OS projects on the one hand and software and IT firms on the other hand, we raise two questions which go beyond this dichotomy and study the actual fruitful interplay between the two:

(1) How and why do firms engage in the creation of public and private knowledge inside and outside the firm?
(2) What factors must for firms to consider when making their resource allocation decisions?

Each level is inherently different as far as the resource allocation for the business approach, internal development processes and OS license commitments

The four-level management model proposed is essentially confirmed by the exploratory empirical insights we discuss. Each level is inherently different as far as the resource allocation required for the

Table 1. Costs and benefits of firm involvement in OS innovation on 4 levels

	Costs and conditions of involvement:	Benefits of involvement and investment:
Level 1: The firm as user of OS software	• Technological expertise to install the OS software • Resources to implement and run the OS software • Legitimation of using OS software • No property rights protection	• Access to increasingly sophisticated, scalable, reliable, cost effective software • Cost reduction for new software development • Increasing transparency and openness of software • No lock-in into particular commercial software
Level 2: OS software as complementary asset	• Investment into adaptation and asset-specific software development • Adaptation of internal processes and practices • Establishment of organizational knowledge base • No property rights protection	• Software development in cooperation with other companies (inc. competitors) and the OS community (including users) • Access to substantial external development resources (and related knowledge and expertise) • Efficient and affective leverage of internal resources and the customers' resources
Level 3: OS software as a design choice	• Significant contributions to software development across a variety of OS projects • Substantial investment of high-level development time • Investment of considerable knowledge resources to allow combination, adjustment and adaptation of software code • Adaptation of the internal software development processes [OS software development as a template] • High degree of openness and transparency as well as respect for internal and external knowledge sharing • Dependence on external technology becoming instrumental to the firm's business and interaction with customers • Creation of in-depth understanding of the property rights implications of being involved in OS software development	• Development of firm-specific, highly implicit and dispersed knowledge and expertise (that is not public) • Establishment of an embedded capability to innovate within an emerging standard platform • Extension of development resources by leveraging a global community of highly skilled software developers to augment internal resources • Increased robustness and reliability of new software due to the extensive involvement of software users into the software development process • Decrease in uncertainty associated with technological changes
Level 4: OS compatible business model	• Significant contributions to software development across a variety of OS project • Necessity for firms to be sensitive to the social norms of the hacker culture • Break with the conventional wisdom of business models based on intellectual property rights	• Distribution of services to software protected under the OS license • Substantial reduction of development costs for new software • Substantial reduction of exploration costs for new technologies and customer needs • Close continuous operational interaction with user community

distinct business approaches to the integration of OS code, and the resulting internal software development processes and license-based commitments to the OS community:

- On level 1, OS software is used in order to replace or supplement existing software within the firm, without implications for the user-firm's business model. Any firm in any technological or industrial context can become involved in OS innovation as a user of the software - however, the fact that substantial expertise may be needed to run OS code within existing IT infrastructures is partly responsible for the emergence of business approaches for specialized software/IT firms at the other three levels.
- On Level 2, OS software is used as a complementary asset, which is combined with other software or hardware to deliver integrated solutions for level 1 OS users. The firm integrating OS software as complementary asset must invest substantial development resources to adapt the code to its particular services and turning it into a stable technological solution. In addition, the software thus developed must be publicly released, according to the licensing agreement of the OS software used.
- On Level 3, OS software is not only seen as a complementary asset, but as core for the entire software developed by the specialized firm. At this level, companies both complement their own software with OS 'building blocks', but essentially build their own software based on the freely available code. As, in either case, the entire developed code must be freely released back to the OS community, this clearly implies a substantial commitment a firms' own business model to OS.
- At Level 4 finally, the packaging, bundling and distribution of stable services and solutions for OS software becomes the business model itself, complementing the interests of the many level 1 firms. The emergence of Level 4 firms can be seen as the natural outcome of the success of the OS software community.

In sum, these insights allow us to further specify the inherently cumulative nature of our four-level model in terms of committing the firm and its business model to OS. The increasing interest of firms to become users of OS software at Level 1 leads directly to the emergence of software firms specializing on integrating OS software in their development process, be it as complementary assets at Level 2, as the basic design choice at Level 3, or as the business model itself at Level 4.

Resource commitment and knowledge creation

The cumulative nature of the four-level model emerges not only from the increasing link of the firm's business model to OS innovation, but also from the knowledge and expertise gained from activity at each level. Firms participating in OS innovation need to understand the software development process: while OS source code is freely available, firms need particular degrees of understanding and expertise to be active at each level. Thus at Level 1, firms need to understand the software sufficiently well to adapt it to their particular technology infrastructure or to involve a specialized software firm doing it for them; at Level 2, firms need to understand the detailed functioning and development of the OS software, in order to relate it to their own technology development; engagement at Levels 3 and 4 depends on active involvement among the core developers of the respective OS project, defining the architecture and the quality standards of the software developed within the project. Full engagement at respective lower levels is a precondition for involvement in higher levels, for a number of reasons:

- First, the four-level management model and its empirical illustration confirm that the simple usage of OS software At Level 1 essentially requires the factual, codified and tangible *knowledge-that* of the respective software code, in particular the specifications and the functioning of the software, as well as a basic understanding of the *knowledge-how* to use and adapt the software to the specific software/IT environment of the firm. At Level 2, an in-depth understanding of the more process-based intangible and dispersed *knowledge-how* becomes more important, including knowledge of how to use and adapt the software, as well as how to further develop and advance

the software, in order to align it with other technology as a complementary asset. Firms active at Levels 3 and 4 confirm that beyond an understanding of knowledge-that and basic knowledge-how, it is important to gain access to the *knowledge-why* of software development, including insights into the rationales behind knowledge development, in this case the underlying architecture and the development style of the software.[24]

Entering core developer groups, necessary for gaining access to and creating knowledge, requires a substantial resource commitment at lower levels

- Second, the firms studied explain the step-wise advancement of knowledge and expertise over different levels by the dispersed and inherently collective nature of knowledge in general, and in OS contexts in particular. A proactive involvement in the OS community at levels 3 and 4 is essential to benefit from its full knowledge creation and learning potential, and a precondition to understand how the software is actually developed in terms of knowledge-why, beyond a discussion of software features in terms of knowledge-that and knowledge-how. The proactive engagement into knowledge creation, which implies the development of valuable source code and of related knowledge and expertise, according to the community's own valuation criteria, allows firms to benefit from the full knowledge creation and learning potential of the OS community, residing in its core developer groups. Entering such core developer groups, necessary for both gaining access to and creating knowledge-how and knowledge-why, requires a substantial resource commitment at lower levels.[25]
- Third, these observations confirm Wenger's insights from management research on knowledge creation, arguing that situated experiential learning and proactive knowledge creation are a necessary precondition for entirely exploring and exploiting knowledge and expertise created by others.[26] One's own level of knowledge creation determines the level at which one can benefit from external knowledge creation. The study of this interplay is particularly interesting in our context, since the knowledge developed is to some extent publicly available; but this source code alone is not sufficient to understand entirely the software development process and the expertise residing in the development of the software code. Furthermore, joint knowledge creation is necessary to exchange and transfer certain expertise and knowledge, which requires the establishment of a common knowledge creation community. Thus, innovation beyond firm boundaries implies investment in knowledge-creation communities beyond firm boundaries.
- Fourth, we learn from our exploratory empirical case studies that being involved in OS core groups implies the creation and development not only of specific, application-oriented knowledge assets, necessary at Level 2 (and to a limited amount at Level 1), which are normally close to the resolution of practical tasks based on knowledge-how. In addition, we echo Spender's judgement that the co-development of shared knowledge 'infrastructures', including software frameworks, technology architectures and dominant worldviews are of basic importance. This investment into the knowledge-how and knowledge-why of the community itself implies major resource allocation, as well as major learning opportunities, due to the reasons discussed above. And, besides implying increasing involvement with particular OS projects, it betokens explicit interest and engagement with OS innovation and the philosophy in general, with an impact on the firm's overall approach and business model.

In sum, contrary to the basic ideological premise of most OS community initiatives (as well as of existing OS innovation literature) we find that it is both reasonable and beneficial for firms to allocate substantial resources to OS innovation — and, in parallel, for OS projects to gain firm

resources for development activities. Firms and OS communities not only share codified knowledge developed in their collaboration, but also substantially invest in shared knowledge 'infrastructures' — and all of this is created and established beyond firm boundaries. Our four-level management model suggests that it is the cumulative nature of these community-based knowledge-creation processes that determines the particular options for firms to both invest in and benefit from OS innovation.

knowledge creating expertise is inherently cumulative (leading) from knowledge-that to knowledge-how and on towards knowledge-why

Implications, limitations and future research

Implications for management and research

We have examined OS innovation and the particular approaches that managers have available to exploit the knowledge created by OS community developers as a new model of innovation that allows individuals, organizations and for-profit firms to collaborate, but for a public good they have created and hold in common to nevertheless still yield proprietary benefits. Managers may exploit that common pool of knowledge at four distinct levels of involvement, but each higher level entails involvement and investment at the lower ones, with increasing resource commitments a precondition for entering the project core of OS software development, which yields the greatest returns of developing particular expertise and knowledge. The four levels are characterized by a dynamic, cumulative logic of gift exchange: the greater the knowledge creation benefits to be derived, the greater the resource investment managers must make and the more interdependence between their firm's business model and a freely available public good they must accept.

This leads to various *implications for managers* (illustrated as Figure 3):

- First, in a knowledge-intensive environment, the knowledge creation and learning processes are directly dependent upon the financial and knowledge resources allocated to these processes: business models, resource allocation and knowledge creation are interdependent;
- Second, given the dispersed and collective nature of knowledge, resource allocation goes into the development of specific knowledge assets and the common knowledge 'infrastructures' at the

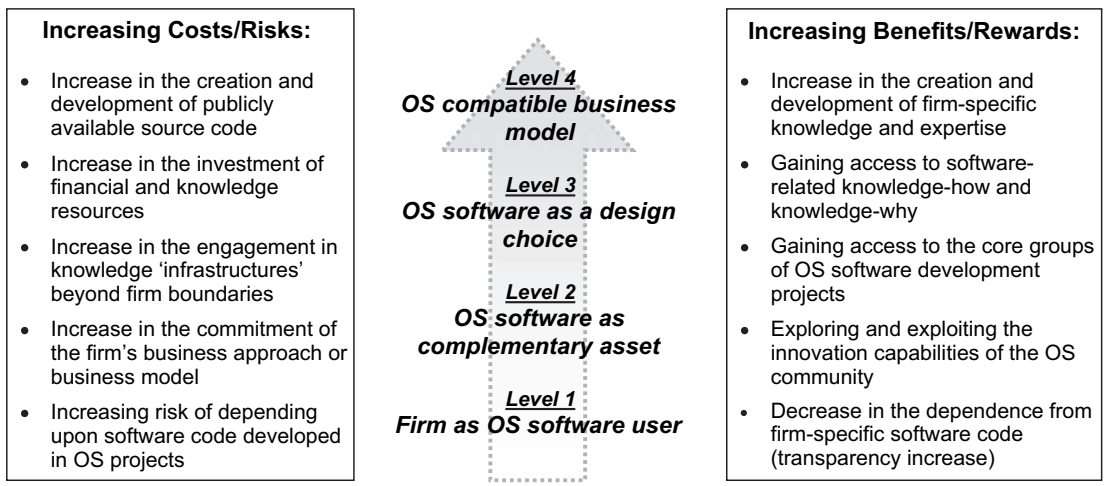

Figure 3. Resource allocation to OS innovation: managerial implications

same time: understanding the nature of knowledge-creating communities and developing firm-specific expertise in investing and participating in such communities is key;
- Third, the establishment of such knowledge creating expertise is inherently cumulative and leads from the development of *knowledge-that* to *knowledge-how* and on towards *knowledge-why*. This move marks the fundamental shift from simple gift giving as it is often discussed in the existing literature to cumulative gift exchange as the dynamic of relations between the different involved parties;
- Fourth, at the higher levels of the model, a substantial investment into the development of public knowledge is a precondition for gaining true access to and effectively learning from such public knowledge.

In parallel, this leads to *insights for management research* on OS innovation in particular (illustrated as Table 2), as well as into technological innovation and knowledge creation in general:

- First, this article is among the first to explicitly study the beneficial interplay between OS communities and firm level resource allocation to OS innovation; while the existing literature is dominated by an explicit or implicit confrontation of OS software development and firm-specific software development, we show the importance of an integrative framework which focuses on the interplay between the two;
- Second, while most research on resource allocation processes and business model identification focuses on financial resources and tangible assets, understanding the allocation of knowledge resources to knowledge creation is equally important and influences the appropriate firm-specific resource allocation and business model; thereby, the knowledge creation, within and beyond firm boundaries, is inherently cumulative and collective in nature, leading to an important interdependence between resource investment and learning opportunities, confirming Dyer and Singh's recent insights from the relational view of strategic partnering;
- Third, the collective nature of knowledge implies, as Spender notes, that contributions to and benefits from knowledge creation require a substantial investment into the establishment and maintenance of knowledge 'infrastructures', which goes far beyond the development of knowledge assets and the resolutions of specific tasks;
- Finally, OS innovation is an interesting context to study the differences and the interactions between encoded explicit knowledge, here in the form of source code, and implicit knowledge, which can be highly individual and firm specific.

OS software may have unique origins, but the strategies that have grown up to exploit this valuable knowledge resource could apply any time that a similar public good became readily available for exploitation.[27] Several *environmental conditions* make this more likely today than ever before.

- First, the very success of OS software itself has contradicted traditional economic expectations. It has proved possible to use OS resources outside firm boundaries to augment firms' internal innovation capability, and for for-profit firms to derive sustained financial benefit from drawing upon and contributing to a commonly held good;
- Second, the modularity of OS architecture has enabled contributions to the innovation to be made from widely dispersed sources. (The benefits of such modular designs have been highlighted in the literature on innovation by Garud (among others));
- Third, the design of standard physical objects is increasingly represented in bits and bytes that can transverse the globe in seconds. Thus it is increasingly easy to create an ever-growing collective of knowledge about an innovation, and that collective can include brainpower anywhere in the world.

In short, we suggest that the OS model may be a herald of innovation modes that involve a worldwide collective of creative contributors, suggesting that it might be interesting to rethink

major knowledge-related *developments in other industries* in the light of the experience of the software industry with OS innovation:

- First, the emergence and successful establishment of the *consulting industry* over the last few decades, benefiting from public insights from management research as well as from private knowledge from their client firms, while itself also investing into the creation of (partly) public knowledge;
- second, the extensive engagement of large corporations and new ventures in the *pharmaceutical and biotechnology industry* (as well as in other technology driven industries) in the creation of scientific knowledge, which is (to some extent) publicly presented and published in academic contexts, but at the same time protected;
- third, the open discussion of technology standards and common product features among all major players in the *semiconductor industry*, due to the enormous investments necessary to develop a new technology generation, complemented by highly secret firm-specific knowledge creation and technological innovation activities.

In all these cases, the optimal resource allocation between firm-specific knowledge creation and knowledge creation beyond firm boundaries, and the implications for the particular business model and the specific interactions and partnerships with the scientific community, industry partners and other institutions represent major challenges for strategic management.

the optimal resource allocation and implications for business models and partnerships with the scientific community and industry are major challenges for strategic management.

Limitations and future research

Several limitations in our analysis should be acknowledged. Our multi-level model emerges from a combination of conceptual and theoretical reflections on the innovation in the context of OS — as compared to private or collective innovation — and from a series of exploratory empirical insights illustrating the characteristics and aspects of each level. A true empirical evaluation would require

Table 2. Resource allocation to OS innovation: research implications

Key Insights for Research on OS Innovation:

- There is a mutually beneficial interplay between community-based OS innovation and firm-level resource allocation to OS innovation, on four different levels of engagement, which contradicts the ideological antagonism between the two often emphasized in the literature.
- The distinction between encoded knowledge in the form of software source code, which is freely available, and implicit knowledge in the form of experience and expertise, which is highly individual and firm specific, is essential to understand firm-level engagement in OS innovation.

Key Insights for Research on Knowledge Creation in general:

- Resource allocation processes and business model identification must be extended beyond the allocation of tangible assets; however, not without explicitly considering the inherently dispersed and collective nature of knowledge.
- Firm-level knowledge creation beyond firm boundaries requires the substantial and cumulative investment in publicly available knowledge 'infrastructures', in terms of collective knowledge-how and knowledge-why.

a more detailed and systematic empirical study. The work also draws from a particular sample of firms that all see OS software development as a positive and promising opportunity for their existing business model: it would be interesting to understand more precisely why some firms decide not to participate in OS software development. And, while we suggest the multi-level model to be of general relevance beyond the particular case of OS software, our discussion only explores OS innovation in this context: further work is needed to explore the potential relevance or actual existence of this innovation mode in other technological areas.

The arguments presented here have implications for future research.[28] First, the extent to which our four-level model might be valid in technological fields other than software remains an unanswered question beyond our development of a series of environmental conditions, which suggest potential application contexts. A promising area of future empirical studies for research would be science-based, knowledge-intensive innovations, particularly since science itself depends to a large extent on collective innovation in which the concerted efforts of scientists around the world create 'public good'-type knowledge.

Second, while we explore the rationale of firms for allocating resources to OS innovation, the performance implications of the model remain under-explored. Although we discuss the efficiency, quality and effectiveness of involvement and investment in knowledge creation in the OS context, the implications for overall firm performance are not analysed. Studies of innovation investments in OS could compare firms that exploit OS software at the various levels to identify benefits accruing from a certain level of shift in a particular industry context. The second research strategy would be to compare firms using the Level 4 business model with those using traditional business models.

Third, software innovation is subject to network externalities, with beneficial emerging standards lowering communication and software use costs. OS offers opportunities for firms to use their own resources more efficiently by employing resources from outside firm boundaries. But, in the presence of network externality effects, will the highest quality be achieved under an OS or a commercial software regime?

Conclusion

Internal software development, firm-specific competence-building and the use of external software resources and source code are highly interdependent and mutually beneficial, requiring an in-depth understanding of the costs and benefits of proactively contributing to OS software development, as well as detailed, explicit and careful management and development of the preconditions and processes appropriate for each cumulative level of engagement.

But, while the decisions about at what level to 'join the party' may not be straightforward, for the 'insiders', those individuals and firms who understand the inherently logic and dynamism of cumulative knowledge creation implicit in OS, it is the only way it can be. For them, the process of cumulative dynamic gift exchange has led to new understandings of the potential for knowledge creation across the public/private boundary, and new strategic puzzles to solve about managing the interplay between business model, resource allocation and knowledge creation.

For those not yet involved, the OS 'story' can yield lessons for both strategic management in the IT context and for strategy in other industries. The pattern of cumulative gift-exchanges between a loosely-linked but growing global community of knowledge creators (individuals and firms) willing to contribute increasing resources to a public good - but one from which private value can nevertheless be gleaned - will gain increasing importance for the more traditional business and corporate world.

Old certainties give way to new possibilities - and OS software innovation offers a fascinating model for future technological innovation and knowledge creation, where the old 'fenced-field' certainties of knowledge ownership give way to more fluid and creative blueprints for innovation in the world to come.

OS offers a fascinating model for future innovation and knowledge creation, where 'fenced-field' certainties of knowledge ownership give way to more fluid and creative blueprints for innovation in the world to come.

Acknowledgements

We would like to thank the guest editors of this special issue, Michael Gibbert and Liisa Valikangas, LRP editor-in-chief Charles Baden Fuller, copy editor Jonathon Morgan, two anonymous reviewers and an anonymous industry expert for their detailed feedback, dedicated support and excellent suggestions. The first author gratefully acknowledges the financial support from the company partners of the RISE Research Center; the second author gratefully acknowledges funding support from the SNF grant #101412.

Appendix

The article relies on a series of secondary case studies, taken from published information about a series of major OS software development projects such as Linux, Mozilla and Apache. In addition, we have drawn upon studies about the interplay of such projects with the software development strategies of major IT corporations, including Hewlett Packard, IBM, Sun Microsystems and Sony.

The article also benefits from five in-depth case studies of successful European custom software ventures. These case studies were conducted in the context of a larger research project about how and why high-tech ventures in different technological and industrial contexts successfully commercialise scientific research and technological innovation. The studies rely on:

a) a series of in-depth interviews (from 1½ to 7 hours) with company founders and other key personnel (5 to 20 interviews with individuals from each company, including up to 4 interviews with the same person in some cases);
b) a detailed analysis of additional material (including business plans, business presentations, internal documents); and
c) participatory research (one team member following particular software development projects within the firm, by participating in formal meetings and informal discussions of the projects).

It should be noted that the case studies of the sampled software companies did not primarily focus on the involvement of these companies in OS innovation. However, the cases illustrate and complement the insights gained from the secondary data on Apache or Mozilla. To a surprising degree, given the discussion about the ideological distance between OS software development and software companies in the literature on OS, these companies were found to be involved in OS projects, using OS software internally, installing OS software for their customers, and proactively investing in OS software development. All the companies seemed to regard this engagement in OS projects as a natural part of their involvement in state-of-the-art software engineering and development.

The software ventures studied focus mainly on distributed software systems in the Internet and Intranet environment of large corporations, especially in financial services, transportation and telecommunications. Companies have been in existence for between 5 and 15 years; have between 60 to 100 employees (mainly senior software engineers with an academic background) and each has a turnover of several million Euros. The companies rely heavily on open standards, believing reliance on proprietary software for the development of customer specific solutions to be highly

problematic both for themselves and their customers. It is fair to conclude that these software ventures see themselves as a part of the OS movement.

References

1. Interestingly, the classical references on gift exchange describe this dynamic and cumulative process, see in particular M. Mauss, *The gift: The form and reason for exchange in archaic societies*, W.W. Norton & Company, New York (2000); G. Bataille, *Accursed share*, Zone Books, Cambridge, MA (1991).
2. Major insights into the mechanisms underlying OS software development can be found in: M. Bergquist and J. Ljungberg, The power of gifts: Organizing social relationships in open source commutes, *Information Systems Journal* **11**(4) (2001); J.M. Dalle and P.A. David, The allocation of software development resources in "open source" production mode, Stanford Institute of Economic Policy Research, Discussion Paper No. 02-27 (2003); G. Hertel, S. Niedner and S. Herrmann, Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel, *Research Policy* **32**(7), 1159–1177 (2003); P. Himanen, L. Torvalds and M. Castells, *The Hacker Ethic*, Random House, New York (2001); D. Kohanski, *Moths in the machine: The power and perils of programming*, St.Martin's Griffin, New York (2000); S. Mahony, Guarding the commons: How community managed software projects protect their work, *Research Policy* **32**(7), 1179–1199 (2003); R. C. Pavlicek, *Embracing insanity: Open source software development*, Sams, Indianapolis IN (2000); G. von Krogh, S. Spaeth and K. Lakhani, Community, joining script and specialization in open source software innovation: A case study, *Research Policy* **32**(7), 1217–1242 (2003).
3. R. Casadeus-Masanell and P. Ghemawat, Dynamic mixed duopoly: A model motivated by Linux vs. Windows, *Working Paper* (2003) retrieved from opensource.mit.edu.
4. A. Bonaccorsi and C. Rossi, Why open source software can succeed, *Research Policy* **32**(7), 1243–1258 (2003).
5. L. Dahlander, Appropriating returns from open innovation processes: A multiple case study of small firms in open source software, *Working Paper*, Chalmers University of Technology (2004).
6. J. West, How open is open enough? Melding proprietary and open source platform strategies, *Research Policy*, 32(7), 1259–1285 (2003); R. Garud, S. Jain and A. Umarasawamy, Institutional entrepreneurship in the sponsorship of common technological standards: The case of Sun Microsystems and Java, *Academy of Management Journal* **45**(1), 196–214 (2002).
7. E. von Hippel and G. von Krogh, Open source software and the private-collective innovation model: Issues for organization science, *Organization Science* **14**(2), 209–223 (2003).
8. J. P. Liebeskind, Knowledge, strategy, and the theory of the firm, *Strategic Management Journal* **17**, 93–107 (1996).
9. K. Knorr Cetina, *Epistemic cultures: How the sciences make knowledge*, Harvard University Press, Cambridge, MA (1999).
10. D. Leonard, W. Swap and G. von Krogh, Exploratory Capacity and the Adaptive Organization, *Working Paper*, Harvard Business School (2002).
11. C. Baden-Fuller and J. Stopford, *Rejuvenating the mature business*, Harvard Business School Press, Cambridge, MA (1994).
12. However, it is important to mention that OS software code does not exactly fit the usual description of a complementary asset. In an interesting dynamic, OS software does not result from a manufacturer's previous commercialisation processes, does not constitute intellectual property for the manufacturer, and is nearly costless to obtain. In fact, manufacturers often join with competitors to build OS software; both Hewlett-Packard and IBM contribute to the development of Linux. Thus, a software manufacturer may find it worthwhile to contribute to such complementary assets by discovering indirect ways to profit. For instance, IBM may profit from developing improvements to the OS program GNU/Linux if these improvements enhance Linux functioning with the proprietary computer software or hardware that IBM sells – see A. Gawer and M. A. Cusumano, *Platform Leadership*, Harvard Business School Press, Cambridge MA (2002).
13. D. Teece and G. Pisano, The dynamic capabilities of firms: An introduction, in G. Dosi, D. Teece and J. Chytry (eds.), *Technology, organization, and competitiveness: Perspectives on industrial and corporate change*, Oxford University Press, Oxford (1998).
14. It is interesting to see that while the public and some people in the OS movement heavily emphasize the ideological component of OS software and the OS movement, these software companies have a very pragmatic understanding of OS software as an interesting and attractive opportunity to explore and

exploit software beyond firm boundaries, somehow very natural to software engineering in general, rather than a unique and particular approach to software development.

15. J. H. Dyer and H. Singh, The relational view: Cooperative strategy and sources of interorganizational competitive advantage, *Academy of Management Review* **23**(4), 660–679 (1998).
16. J. Hamerly, T. Paquin and S. Walton, Freeing the source: The story of Mozilla, in C. DiBona, S. Ockman and M. Stone (eds.), *Open Sources: Voices from the open source revolution*, O'Reilly, Cambridge, 197–206 (1999).
17. J. Wade, Dynamics of organizational communities and technological bandwagons: An empirical investigation of community evolution in the microprocessor market, *Strategic Management Journal* **16**, 111–133 (1995).
18. Such skills are not readily transformed into text directives or rules; however, they may be transferred to others within the firm boundaries through apprenticeships and joint problem solving, see D. Leonard and W. Swap, *Deep smarts: How to cultivate and transfer enduring business wisdom*, Harvard Business School Press, Boston, MA (2005).
19. R. Young, *Under the radar*, Coriolis, Scottsdale (1999).
20. In parallel, it seems not to be a problem for project-based software ventures to almost naturally combine their participation in OS projects and developing specialized customer-specific software.
21. B. Perens, *The Open Source Definition*http://perens.com/Articles/OSD.htm (1998).
22. E. Raymond, *The Cathedral and the bazaar: Musings on Linux and Open Source by an accidental revolutionary*, O'Reilly, Sebastopol, CA (1999).
23. For example, a firm may attempt to weave OS code and commercial code into one single product violating a license such as GPL. As of this writing, no data exist on the costs involved in enforcing the rights granted under an OS license, and more research is needed on the disincentives preventing such malicious behaviour.
24. R. Garud, On the Distinction between Know-how, Know-what and Know-why, in A. Huff and J. Walsh (eds.), *Advances in Strategic Management*, JAI Press, 81–101 (1997); G. von Krogh and S. Grand, From Economic Theory towards a Knowledge Based Theory of the Firm: Conceptual Building Blocks, in N. Bontis and C. W. Choo (eds.), *The Strategic Management of Intellectual Capital and Organizational Knowledge*, Oxford University Press, New York, 163–184 (2002); B. J. Loasby, The Organization of capabilities, *Journal of Economic Behavior and Organization* **35**, 139–160 (1998).
25. R. M. Grant, Toward a knowledge-based theory of the firm, *Strategic Management Journal* **17**(2), 109–123 (1996); J.-C. Spender, Making knowledge the basis of a dynamic theory of the firm, *Strategic Management Journal* **17**, 45–62 (1996); E. Wenger, *Communities of practice*, Cambridge University Press, Cambridge (1998).
26. E. Wenger (op cit at Ref 25); R. Garud and M. A. Rappa, A socio-cognitive model of technology evolution: The case of cochlear implants, *Organization Science* **5**(3), 344–362 (1994).
27. Communities of highly specialized professionals who cooperate to produce bodies of collective knowledge have existed for decades, of course. For example, CERN, the community of high-energy physicists who share their research and publish papers with literally hundreds of names as authors, has *some* qualities similar to the OS community. But the differences are profound: for one, no for-profit organization could take research findings of CERN and rapidly apply them.
28. A summary of other ways of thinking about boundaries and innovation can be found in the following papers: M. Boisot and I. C. MacMillan, Crossing Epistemological Boundaries: Managerial and Entrepreneurial Approaches to Knowledge Management, *Long Range Planning* Special Issue on Boundaries and Innovation, **37**(6), 505–524 (2004); S. Karim and W. Mitchell, Innovating through Acquisition and Internal Development: A Quarter-Century of Boundary Evolution at Johnson & Johnson, *Long Range Planning* Special Issue on Boundaries and Innovation, **37**(6), 525–547 (2004); J. Roos, B. Victor and M. Statler, Playing Seriously with Strategy, *Long Range Planning* Special Issue on Boundaries and Innovation, **37**(6), 549–568 (2004); D. Dougherty and C. H. Takacs, Team Play: Heedful Interrelating as the Boundary for Innovation, *Long Range Planning* Special Issue on Boundaries and Innovation, **37**(6), 569–590 (2004).

Biographies

Dr. Simon Grand is Founder and Academic Director, RISE Research Center for Innovation, Strategy and Entrepreneurship; Institute of Management, University of St. Gallen HSG.

Prof. Georg von Krogh is Professor of Management, Director of the Institute of Management, University of St. Gallen HSG.

Prof. Dorothy Leonard is William J. Abernathy Professor of Business Administration, Harvard Business School.

Prof. Walter Swap is Professor of Psychology, former Chairman of the Psychology Department, Tufts University.