OPAALS PROJECT

Contract n° IST-034824

WP12: Socio-Economic Models for Digital Ecosystems

Del12.6 – Making Sense of Open-Sourcing in Practice

Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: IST-034824

Project Acronym: OPAALS

Deliverable N°: 12.6: Making Sense of Open-Sourcing in Practice

Due date: M30

Delivery Date: M33

Short Description: This deliverable is our understanding of how large and medium sized companies are coping with open source adoption. Managerial practices have changed to adapt to the transparency required in open source processes. Our focus here is to relate and analyze the 'best practices' that companies practising open-sourcing are adopting.

Author: Maha Shaikh

Partners contributed: UL

Made available to: All OPAALS

Versioning		
Version	**Date**	**Name, organization**
Version 1	March 2009	Maha Shaikh – University of Limerick

Quality check

Internal Reviewers: Antonella Passani and Paul Krause

Dependences:

Achievements*	New literature was read. Seven–ten months of data collection in six companies with interviews. Coding of dating in a grounded theory method. Analysis and write up of the deliverable.
Work Packages	WP12 and our work on open-sourcing will help lay the groundwork for how communication, business models, and collaboration is achieved in a digital ecosystem. This work is also important to WPs 2, 3, 5, 10, 11 and other sections of 12. The lessons on community development processes are particularly relevant to ongoing work in these work packages, and to the continued sustainability of digital ecosystems.
Partners	LSE
Domains	Social science
Targets	Domain researchers, public administrations, SMEs.
Publications*	Conference papers still under review for AMCIS, and journal.
PhD Students*	NA
Outstanding features*	Scant research in the area of open-sourcing to date so the current study makes serious headway into understanding how managers can cope with the adoption of open source software. Many companies like IBM and HP have taken interest in this author's current research.
Disciplinary domains of authors*	Maha Shaikh – Social Scientist at the University of Limerick

TABLE OF CONTENTS

LIST OF TABLES

EXECUTIVE SUMMARY

In our field study we focused on six different companies, four of which were large, global companies and two SME's. This deliverable focuses on how these companies are learning to cope with open source adoption. The interesting element in such adoption is that it has had mixed success in each company, but at the same time each company has adopted open source in a different manner and with differing levels of use. Some companies use open source software for in-house use for backend functions. This usually implies that the license of the open source software will not really affect the company because they don't plan to repackage and sell/distribute the software to others. If the latter does become true then the license of the software is a bigger question which needs to be thought through properly (more on this area below in the deliverable).

Some companies choose to collaborate with open source communities external to their own company. This requires different resources, strategy, and business model. Again, how much they collaborate with the external community also varies. Some companies work with existing communities while others attempt to create a community around an in-house product it wishes to take the open source route. This approach is also explained in more detail below through an explanation of the various strategies employed by managers which we re-tell in this deliverable.

Another interesting evolution we found in companies involves them not only adopting open source software but also an open source process of development, including openness, collaboration and a change in work practices. Indeed, some companies adopted an open source process without working with open source software. Rather they chose open source process as a way to encourage greater communication within the company, and importantly, innovation.

This deliverable aims to show how, no matter what strategy or business model adopted by each company, what were the different coping and balancing tactics and long term strategies adopted by managers to work with an uncertain product which has lead to greater uncertainty about governance, rules, licenses, human resources, best practices and collaboration. This is a serious step for any company to take and our research in this area is intended to guide both practitioners and academics alike who are working in this area.

We begin the main section of this deliverable with a short introduction to the sensitizing concepts we adopted to create our initial framework of open-sourcing. This framework or characterization lays the basis for our data collection and analysis which followed. We then explain our research methodology of data collection and analysis through conceptual coding.

Our analysis is split into three main parts of product, community and process. Within each element we indicate what our data reveals to us about open-sourcing, how and why companies adopt this method of open-sourcing and then how the managers cope with the many issues that arise.

We end the deliverable with a more detailed understanding of how what we have gleaned from our data collection and analysis can guide the building of a digital ecosystem (an aim of OPAALS). We also show a forward path for our future research which we intend to cover in Phase III of OPAALS.

INTRODUCTION

Deliverable 8.3 laid the groundwork for our current study. In D8.3 we focused on the various business models we found in the field of open innovation in practice by both large and small companies. This led to a preliminary characterization of open-sourcing. In this deliverable we focus on this characterization, with the help of empirical data, to understand the situation as practised by companies. Our findings help us to strengthen our early framework and to contribute to how, why and at what level, open source adoption by companies can benefit them.

In our field study we focused on six different companies, four of which were large, global companies and two SME's. This deliverable focuses on how these companies are learning to cope with open source adoption. The interesting element in such adoption is that it has had mixed success in each company, but at the same time each company has adopted open source in a different manner and with differing levels of use. Some companies use open source software for in-house use for backend functions. This usually implies that the license of the open source software will not really affect the company because they don't plan to repackage and sell/distribute the software to others. If the latter does become true then the license of the software is a bigger question which needs to be thought through properly (more on this area below in the deliverable).

Some companies choose to collaborate with open source communities external to their own company. This requires different resources, strategy, and business model. Again, how much they collaborate with the external community also varies. Some companies work with existing communities while others attempt to create a community around an in-house product it wishes to take the open source route. This approach is also explained in more detail below through an explanation of the various strategies employed by managers which we re-tell in this deliverable.

Another interesting evolution we found in companies involves them not only adopting open source software but also an open source process of development, including openness, collaboration and a change in work practices. Indeed, some companies adopted an open source process without working with open source software. Rather they chose open source process as a way to encourage greater communication within the company, and importantly, innovation.

This deliverable aims to show how, no matter what strategy or business model adopted by each company, what were the different coping and balancing tactics and long term strategies adopted by managers to work with an uncertain product which has lead to greater uncertainty about governance, rules, licenses, human resources, best practices and collaboration. This is a serious step for any company to take and our research in this area is intended to guide both practitioners and academics alike who are working in this area.

Recap of Open-Sourcing

Drawing on open source (OS[1]) products and mass voluntary participation as a deliberate sourcing strategy for software and other IT/IS services – what we address here as open-sourcing -is a relatively unexplored concept. Until recently this term has usually been used to refer rather to cases of commercially controlled and created software switching partially or fully to open source licensing. An example of which would be Netscape's decision in 1998 to 'open' some of its browser code as Mozilla, the basis for the present day browser Firefox. Debate at that time was divided as to whether this was a capitulation to Microsoft dominance in browsers, or evidence of a radical innovation in the software model. Subsequent history does not it seems reveal a clear answer. More recently the term open-sourcing has taken on a significantly different meaning (Ågerfalk *et al.*, 2006a; Anderson, 2005), implying a deeper link with fundamental sourcing options and decisions, and in particular outsourcing strategies. Thus open-sourcing is defined by

[1] To avoid confusion between the similar terms 'open source' and 'open-sourcing' in this deliverable we refer to open source when appropriate as OS and write the full phrase 'open-sourcing' when we speak of the latter idea.

Ågerfalk *et al*. (2006b) as 'outsourcing to a global but largely unknown workforce', a definition that we take as our starting point.

A number of developments in the past decade have made such a strategy for sourcing IT services possible. For example, there is an emerging understanding of broader Web 2.0 functionality and its implications (O'Reilly, 2005). At the same time commercial organizations have changed their attitudes to OS software (Dickerson, 2004). The OS movement too has moved on from its original form (free software, counter-cultural ideology, hacker ethics) to embrace an increasingly sophisticated set of business models with strong commercial interrelationships (Deek and Mchugh, 2007). In a some cases, such as IBM's commitment to the LINUX operating system, an open-sourcing approach has been enacted on a truly strategic scale. Reflecting such shifts Fitzgerald (2006) identifies a new era of OSS 2.0, a concept intended to reflect the extent to which the OS movement has reshaped itself so as to allow major sections of the software and services industries, and their commercial clients, to accept and work with OS software, its methods, licenses, processes, and (to a degree) its ideology of open innovation (von Hippel, 2005a).

Open-sourcing should also be understood for the potential it offers in its 'global' dimension; as a means of bringing together diverse and distributed human, cultural and economic resources from across the world. It is feasible today, at least potentially, for a client organization to draw upon a very diverse and distributed collective of talent that has the potential and the tools to be able to collaborate, communicate and coordinate to achieve outstanding results. This is central to the emergence of both the OS and the outsourcing communities, for both of which the Internet has allowed new configurations of resources and expertise to be established within new and innovative on-line market places. This in turn can lead to new styles of working across boundaries, be they political, spatial, temporal, and not least cultural. Ågerfalk *et al*. (2006b) claim a distinctive global character to open-sourcing, and this is not just a repetition of a tired contemporary cliché but helps reveal some of the distinctive aspects of this scheme of production and service delivery. In this respect we argue open-sourcing is in some contrast to traditional outsourcing, which is usually accomplished within a limited number of local contexts and situations (Carmel, 1997) – as in traditional software off-shoring - and pursues a strong degree of control as well as technical and cultural homogeneity (though it has been argued that culture is always local even within this context and thus never truly homogenous (Walsham, 2002)).

Structure of this Deliverable

The sections of this deliverable are organized as follows – we begin with a short section on the sensitizing concepts we adopted to create our initial framework of open-sourcing. This framework or characterization lays the basis for our data collection and analysis which followed. We then explain our research methodology of data collection and analysis through conceptual coding.

The section to follow on from the methodology is the findings and analysis. Here we, dictated by our data analysis, split our section into three main parts of product, community and process. Within each element we indicated what our data revealed to us about open-sourcing, how and why companies adopted this method of open-sourcing and then how the managers coped with the many issues that arose.

We end the deliverable with a more detailed understanding of how what we have gleaned from our data collection and analysis can guide the building of a digital ecosystem (an aim of OPAALS). We also show a forward path for our future research which we intend to cover in Phase III of OPAALS.

SENSITIZING CONCEPTS

Our research was guided by various conceptual ideas. We briefly mention some below in this section. It needs to be made clear that these concepts guided our methodology and initial framework (see Table 1) very explicitly but our analysis is more implicit in its use of these ideas. Our aim was to have a data-driven study which did not explicitly draw on any one theoretical idea. We will approach our current data and the next

phase of collection (the community side of the phenomenon) in Phase III of OPAALS, when we have a more complete understanding of open-sourcing.

Borrowing concepts from von Hippel's (2005) user-driven innovation we analyze our data through notions of principal agent theory (Alchian and Demsetz, 1972; Jensen and Meckling, 1976). Von Hippel's main thesis of democratizing innovation is premised on the idea that collective action driven by users will lead to greater innovation and more successful adoption of it (Franke and Hippel, 2003; Franke and Von Hippel, 2003; von Hippel, 2005a; von Hippel, 2005b). Innovators include both individuals and firms, where lead users of innovations (or innovators) stand to profit the most. A lead user creates something that is useful to herself and thus perhaps to others too. If the latter is proved true then the commercialization of the idea can lead to economic gain.

In order to commercialize an idea an individual innovator needs the support of others or a firm. The basic ideas of principal agent theory are adapted by von Hippel (von Hippel, 2005a) to make sense of how and why both individuals and firms manage opportunistic behaviour in their relationship. In open-sourcing relations become a little more complicated as often there are no contracts signed between the firm and community members. A number of reasons, for example control over one's work, need or sense of urgency to solve a problem, ability to customize a product, and experience and opportunity to learn, inspire innovators to persevere in spite of potential costs. So in effect it is not just the product and what you can do with it (can you share it, distribute it, etc) but the process of innovation (von Hippel, 2005a) that leads to learning and knowledge creation.

From the other perspective, firms benefit from a larger pool of ideas and differing innovations that they can use to earn a profit. They need domain experts to innovate because knowledge is 'sticky' (von Hippel, 1994; von Hippel, 1998) and thus difficult to reproduce outside the context in which it was created. The idea of sticky knowledge can be linked to how companies that are actively encouraging and participating in open-sourcing consider resource capture (Shaikh and Cornford, 2008) to be one of the key reasons for turning to an open innovation process.

METHODOLOGY

Four large global technology companies were chosen as case studies, along with two smaller firms. Our access to the companies was agreed upon on the condition of anonymity. These companies have moved towards greater use of open source software, ideas and development methods over the last ten years or more. Also, they have a different focus on their use and adoption of open source and are thus at differing levels of adoption.

In our analysis, to maintain our promise of anonymity, we do not name any company or employee interviewed. Instead we generically call all the employees of the four large companies 'Company A', and SME employees 'Company B'.

Data Collection

Our research method included semi-structured interviews carried out in person, but mostly via telephone. Each interview lasted an hour or more. The interviewees belonged to top and middle management, and also included software developers. In two companies access was negotiated in a traditional manner of approaching a contact and then asking the contact to suggest key personnel that could prove fruitful interviewees, keeping in mind the focus of our study. However, for the other half of the companies we did not have a similar contact, and were thus forced to search the Internet for names of personnel on open source related positions. We used project websites that we knew the company was linked too and searched mailing lists for possible interviewee names. We found our first interviewees in this manner. Subsequently we successfully adopted the snowballing method whereby each interviewee was asked to offer a few more potential interviewees. Most unhesitatingly offered two or more names, people in their own company or

colleagues in similar technology based firms. We conducted thirty interviews over a course of seven months of data collection. Our focus was technology companies for a number of reasons, namely; they are leaders in the adoption of open source; have a level of maturity in using open source that allows for reflection on practices; and were easier to find, and get access too.

Data Analysis

The sensitizing concepts of innovation, which stem largely from economic theory, guided our initial framing of open-sourcing. The literature we covered in order to create the framework was part academic, and part grey literature. The latter was an important element of our work because academics are in the process of studying open-sourcing and as the concept is fairly new in both coinage and practice we were obliged to draw on current information, with less regard for its peer reviewed nature. This is not to imply that our literature was in any way questionable, indeed we were meticulous in our data collection from such material and cross checked news items against a minimum of five other cutting edge and respected sources (such as the FT, Economist etc etc). Our criteria for Table 1 were elicited through a process of coding the literature we covered.

We then focused on academic literature for outsourcing, open source and proprietary source. Due to the large number of academic papers written on all three areas, especially outsourcing, we made a judicious choice of finding 15 papers on each area which had been published most recently AND were focused on looking at the field as a research commentary or literature review. These were not necessarily literature reviews of the entire field of say outsourcing but more a perspective on concerns over-lapping with open-sourcing, such as organizational change issues (Ghemawat and Hout, 2008; Howells *et al*., 2008; Jiang *et al*., 2008; Komporozos-Athanasiou, 2008; Ling-Yee and Ogunmokun, 2008; Vivek *et al*., 2009; Warren *et al*., 2009; Winkler *et al*., 2008).

The framework (Table 1) became the basis of our interview guide for the second part of our data collection, in this case primary data. The guide consisted of close to sixty semi-structured interview questions but the guide was divided into sections. Each section was based on a higher level theme or criteria and we only asked each interviewee questions from specific sections, rather than all sixty questions (which would have been impossible considering time limitations). After basic introductions with each interviewee the interviewer made it a point to ask them about their areas of expertise. This was done in order to better filter out the sections of the guide that could not be tackled by the interviewee.

Interviews were transcribed and coded using Atlas.ti content analysis software. Using the tools of Grounded Theory (Glaser and Strauss, 1967; Strauss and Corbin, 1999), though not the full ontology, yielded a code book of forty-nine initial codes (Shaikh, 2007). Along with codes we added memos in the form of notes, concepts and broader emerging themes. These themes gravitate around the larger ideas of product, community and process. On the product side the focal ideas that we found include; license, environmental impact of the product and level of documentation and after-sales service. For community the main themes were motivation, resource capture, requirements management, match between company and community and incentive schemes. And finally for process, our coding elicited ideas such as governance model, managing innovation, organizational factors, communication and hybridization of process.

	Criteria	Proprietary Source	Outsourcing	Open Source	Open-Sourcing
Process	**Communication**	Face-to-face mostly.	Face-to-face, but also uses technology like email, telephone etc.	Mostly online, though limited face-to-face in conferences.	Combination of face-to-face and online.
	Control	Centralized, tight and rule based.	Centralized, tight and rule based – often implemented though the SLA.	Distributed, informal rules and norms.	Combination of control mechanisms used ranging from very centralized to quite distributed and

					informal.
	Infrastructure	Works on internal company infrastructure though this can be global.	Works on internal company infrastructure though this can be global.	Built on distributed OS structure and uses already in place Internet infrastructure.	Built on both global internal infrastructure and Internet.
	Governance model	Hierarchical, top down with strong management structure.	Hierarchical, top down with strong management structure, and often needs to consider governance model of client company.	Varies from OS project to project but often democratic and a combination of bottom up and top down.	Combination of hierarchical (and client based governance structure matching) and OS democratic style.
	Maintenance	Clear and distinct after sales phase with documentation.	Clear and distinct after sales phase with documentation.	Evolutionary and agile development that makes little distinction between phases of development.	Combination of clear phase and evolutionary type of development and after sales service.
	Distribution model	Software/product/service is often created for a large market so generalized product and then distributed widely through retail.	Made to measure software/product/service so distribution channel is closed and internal.	Distribution is carried out over the Internet as downloads (which are sometimes free of charge) but always with open access to product design and code. Internet provides large, cheap and effective distribution channel.	Mix of internal, closed channel with the use of the Internet (where companies often take from this latter channel but don't pour back their now copyright idea/product).
	Total cost of ownership	Clear methods to help quantify TCO in the company making decision making on this basis between products/services easier to compare.	Clear methods to help quantify TCO in the company making decision making on this basis between products/services easier to compare. Such factors have to be dealt with when creating the SLA and thus this is clear.	Too many factors that are hard to quantify thus making TCO difficult to measure in OS.	Depending on the open-sourcing route taken companies would face varying fuzziness of TCO. TCO is clearer in open-sourcing than OS as some elements must be quantified for a company wishing to sell its product/service.
	Marketing	Wide, open and global marketing strategy to pitch product/service at the largest audience.	More at the level of company ability and expertise to enrol other firms into long term contracts.	Via forums, word of mouth, use of product, and gaining critical mass of designers and developers in the community.	Global marketing strategy but also based on expertise of the company – however also rely on word-of-mouth and forums.
	Transparency of process	Limited transparency of process as proprietary license.	Limited transparency of process as proprietary license.	Greater transparency than other models and depends on the promiscuity of license.	Partly transparent – depends on license. If dual license then good level of transparency in process.
	Development model	Depends somewhat on size of project – large scale production usually entails clear phases and division of labour.	Depends somewhat on size of project – large scale production usually entails clear phases and division of labour.	Agile, evolutionary and more focused on parallel constant testing and building.	Combination of planned and clear phase production with agile methods and beta testing.
Product	**License**	Closed and proprietary	Closed and proprietary	Open source (with varying degrees of openness)	Dual licenses and some only OS or proprietary.
	Application type	Wide range of both products and services are covered.	Back-office and non-core applications/services.	Horizontal rather than vertical business applications and services. Mostly infrastructural and back-office.	Non-core applications and services but open-sourcing model used as a strategic device and propped up as strong PR for the company.
	Code/product/ idea quality	Unable to reverse-engineer product or	Quality specifications somewhat written into	Quality easier to test as large base of testers	SLA specifies quality level so quite effective

		service so need to accept quality as provided by vendor.	SLA which ensures certain level of quality. Unable to reverse-engineer product or service.	and bug fixers. Source and process are fairly transparent so if needed the product/service can be analyzed.	yet made even better when both product and process are transparent and open to scrutiny.
	Ownership	Owned solely by company or license holder.	Owned solely by company, license holder, or client company that pays for the product or service.	Owned by the collective and not by any one individual or company solely.	Partly owned by the collective and some strands of the product/service owned by a company.
	Architecture	Closed architecture.	Closed architecture.	Open architecture.	Early part of the product/service is open but the final product is often closed architecturally.
	Level of documentation	Detailed documentation is a requirement.	Detailed documentation is a requirement and specified in the SLA.	Often patchy or non-existent but sometimes this is mitigated by speedy help provided through discussion forums by experts.	Documentation of good quality and detail is a must and this counters a serious problem companies have with OS products/services.
	After sales service	Either part of the contract or non-existent.	Either part of the contract or non-existent – this can lead to issues of vendor lock-in just as the maintenance issue.	No real after sales service at all but discussion forums are often used to tap into community expertise and help.	Companies often bridge between the community and client and ensure that after sales services are provided through the company but tapping into community expertise.
	Reusability of code/idea/ product	Reusability limited to development/designer of idea but often the company dictates what can be reused.	Reusability limited to client company as they pay for the product or service and thus own it. However they often don't have the expertise to reuse it so are locked into vendor company.	Great reusability of idea, code, product as the process and product is open and transparent – and available for anyone to reuse.	Good reusability level as company usually releases much of the design or product back into the community. Indeed this is encouraged as a way to build trust between company and community.
Community/Organization	**Motivation**	Mostly financial, promotion and job-related.	Mostly financial, promotion and job-related.	Learning from each other, reputation, ego, potential job-seeking and creativity.	Community and company link building to retain expertise over time, promotion of product and company, and sustaining the community and its members.
	Contributor profile	Company based and can be a global company. Paid employees.	Company based and can be a global company where outsourcing and offshoring implies mixed cultural background of developers and designers. Paid employees.	More truly global than any other model, mixed culturally and mostly male. Mostly working in their free time/while on another job and not usually paid.	Global as the OS community it links too is very global but the company members are often mostly from one country, paid employees.
	Level of interest and contribution	Paid to work on project so interest varies and contribution is expected and dictated by senior management.	Paid to work on project so interest varies and contribution is expected and dictated by senior management and client company.	High level of interest but most contributions are small and can take the form of using the application/service and providing feedback/bug reports.	Good level of interest and contribution may be small but is consistent.
	Mobility of developers/ creators	Mobility limited to company but company maybe global.	Mobility limited to company but also to client sites and often experts of the vendor are hired by the client.	Great mobility between projects and often we see the same person is a member of more than one community.	Very good mobility between company sites, client sites and OS community.
	Access to	Companies' offer training	Training is offered by	Training (traditional	Employees/members

	learning/ training	options but learning from other colleagues is limited by the number of people you have communication with and access too.	companies. Learning is enhanced through access to a larger population of people from working on client sites.	form of it) is not usually available but learning from colleagues is a key reason why people contribute and belong to a community.	have both training and collaborative learning resources. Training is provided by the company and all members have access to a global community of OS experts outside the company walls.
	Size of community	As large as the number of employees of the company.	As large as the number of company employees and often the client site too.	Varies from project to project but potentially this is a global and very large workforce.	A mix of both company employees and access to the global workforce accessible through the OS community.
	Sanctions on rule breaking	Companies have clear rules and enforcement strategies.	There are clear rules and enforcement techniques which stem from both the company of origin and client/vendor companies.	Informal but very effective enforcement techniques to control the community.	Mix of informal and legally enforceable sanctions.
	Status of core developer/ creator group	Position of authority but still answerable to manager.	Position of authority but still answerable to manager and client/vendor management.	Position of control and authority which is based on recognition of expertise and thus very influential.	It is a position of control and authority yet there is some measure of answerability to the company.
	Global distribution	Large companies have global offices which create a global presence and workforce.	Large companies have global offices which create a global presence and workforce, and this is enhanced when outsourcing is done in the form of offshoring.	Internet as infrastructure provides few temporal or spatial boundaries to such communities thus creating a more globally distributed community/ organization than possible in any one company.	Beneficial combination of resources which gives rise to a considerably globally distributed presence.

Table 1: Comparative Characterizing of Open-Sourcing within Global Sourcing Types

FINDINGS AND ANALYSIS

The business models that have emerged in response to increased organizational acceptance of OS ideas as part of a sourcing strategy need to be considered with regard to their sustainability and comparative effectiveness. Not all innovative business models work – indeed most fail. In this section we attempt to consolidate and assess these ideas in terms of three key aspects: the products and services offered through open sourcing, the work processes they imply, and the potential for a relevant global collective (community) that can support such activity (Fitzgerald, 2006; Shaikh, 2007). Using this framework we explore the advantages and problems that organizations may face as they come to assess and participate in open-sourcing activities.

Product and Service

Open-sourcing is about some degree of trust or belief in the ability of a 'largely unknown workforce' to deliver what a client wants. In the established area of open source this is usually understood as software, and most often software that has generic capabilities as part of an IS infrastructure and thus has obvious value to multiple users. In contrast, in open-sourcing the expectation must be that the product or service sought is in some degree unique or specific to the client's needs. Of course this too is part of the open source model, open code allowing customization, but research seems to show that relatively little such customization actually takes place. But if specific needs are to be served through an open-sourcing approach, then we must

expect that product (as in code) will become more and more bound into a service, an observation that is born out by the examples cited in the section above.

Consideration of product or service also leads us to consider the implications of the distinctive licensing of intellectual property that is central to OS. To be convincing for clients open-sourcing will need to carefully address the issue of license (Clarke, 2007).The transparency that stems from adherence to such licenses is fundamental to allowing mass participation and peer review processes. However, current experience with open source code suggests that corporate lawyers are not always well-versed in all the different OS licenses (the Open Source Initiative (OSI) has approved 58 distinct licenses2), or able to identify the implications of their use. The movement to less strong licenses such as the LGPL (Lesser GPL) and the BSD license, and the practice of dual licensing, may help build this understanding.

In some areas OS software is recognised as of the highest quality. As a web server Apache is a match for it proprietary competitors, as arguably is LINUX in large segments of the operating systems market. However, beyond the top 20 OS products quality may be at least less easy to judge, notwithstanding the availability of code. The same may be true, by extension, of other open-sourced services and ideas, particularly if the peer review aspects are less than rigorous as may well be the case in global sourcing situations exploiting weaker licenses. Companies may also fear losing intellectual property rights to the software or service they source in this way, and a switch to a different sourcing model may not be accepted without effort or some desperate circumstance. There are also relevant concerns over security implications of wider access to code (Hoepman and Jacobs, 2007; Neumann, 2000). It is often proposed that OS software, because the code is visible, should be more secure through the scrutiny it has by a large body of testers and reviewers (Schneider, 2000; Schneier, 1999, September 15th). But there is the counter argument that when code is visible it can be manipulated in more subtle ways (Wheeler, 2003; Witten et al., 2001).

An Open Process

The OS process, in which code is reviewed and commented on by a larger community, where competing ideas are encouraged and which parallels development with debate, has been shown to be able to crack open some enduring software woes. More generally it is part of a broader movement towards open innovation that adopts a philosophy of mass participation, innovation through iteration, with a fast and fluid cycles from idea to critique to testing to use. This is supported by the ability to freely borrow and rework what already exists, based on a strong faith in peer review and adherence to (and creation of) standards. For many corporate IT/IS tasks these characteristics seem to offer an attractive possibility of achieving new levels of quality, responsiveness and timeliness.

But not surprisingly it is not so easy to pick up and exploit such ideas. Companies have endured many problems when adopting open-sourcing products, let alone processes, not least in the managerial and organizational change issues faced (Dinkelacker and Garg, 2001). These include basic lack of understanding, continuity concerns for relevant experts and module designers (West and O'Mahony, 2005), the variable code quality, lack of full documentation and lack of understanding of the range of support services (Melian, 2007). And even if higher management are comfortable with taking such a route to source major elements of their IT/IS, it may not be acceptable to front line staff who have a large investment in a different type of proprietary skill for which there is a keen market (West and Gallagher, 2006).

Collectives and Community

Finally, we must understand that fundamental to the potential of the open-sourcing idea is the existence of that 'global but largely unknown workforce', willing and able to undertake intensive knowledge work. Indeed many hands may make light work, or as the open source movement has it, 'Given enough eyeballs all bugs are shallow', but is the positive emphasis that OS scholars place on motivation - both community and individual - and the value it creates through knowledge creation and constant learning, as relevant in an open-sourcing model? Sustained access to a collective or to organization-wide expertise is a necessary

[2] http://www.opensource.org/licenses/category

condition for open-sourcing, but is almost impossible to engineer and once achieved poses many problems to maintain long-term. Within the wider world there is increasing competition for the attention of the talented, and more and more opportunities for them to achieve realization of their goals through conventional work, while inner source strategies rely on a pool of experts that is not large.

Product

Product concepts relate closely with the license of the product. Proprietary software has a very distinct license from open source products and services but open–sourcing is able to capitalize on a dual licensing scheme. Some open source software is also released under a dual license but that is because the company or community that has made such a decision is working on an open-sourcing model. Ideas of ownership are laid down by and resolved by the license too. The General Public License (GPL) for instance, ensures that no one person or company owns the product. The GPL gives ownership of only his/her contribution to each individual, so each person owns what they create but no more. Individual owners then when they adopt the GPL give rights to others to use and amend their work creating the impression of collective ownership. Open-sourcing is able, unlike other approaches to take advantage of both proprietary licenses and rights of ownership and those of open source.

The reason why many consider open source products to be of better quality is because the source is open and visible to all. This draws back into our earlier point about how transparency of product is designed through the license adopted. Companies are interested in open-sourcing because the source is visible which reduces vendor lock-in thus allowing companies to take their product/service to an alternative vendor if the original one begins to charge too much.

Company Motivation	***Key Areas***	**'Product' License and IPR**	**'Community' Requirements**	**'Process' Managing with Open-Sourcing**
	'Product' Innovative Products/Ideas Focus	• Balance viral versus promiscuous licenses with dual licensing or non-distributive strategies for software • Release open source version of product to harness ideas and usage	• Create in-house team of developers that liaises with community for two-way exchange • Attract community and companies with challenging technical issue or prototype release	• Innovation through use of product so allow community to 'guide' its direction • Offer gratis technology for experimentation to community enabling a support infrastructure and funding avenue
	'Community' Resource Capture	• Create industry standard • Building in-house legal expertise • Creation of Open Source Board in the company	• Agree to support activities eg documentation • Actively approach OS community for ideas and innovation	• Create a strong Steering Committee for greater control • Offer job (both short and long term contracts)
	'Process' Steering the Environment	• Create new OSI approved licenses • Contribute to creation of 'best practices' for the industry to follow	• 'Follow success' – rely on positive experience of other companies • Employ people that become part of the OS community and can 'talk shop'	• Creating test cases: testing and debugging • Create industry standard by leading the way and openness of process (for trust)

Table 2: Open-Sourcing Motivation and Mechanisms

License and Ownership

In our fieldwork we found that managers indicated the main issue that they needed to consider when adopting open source software concerned the license of the software. Licence becomes an issue when the software they adopt might be re-distributed to their clients. In such a case, should some open source software be used in combination with proprietary software then that could become a problem.

To work with open source is to accept the fundamental implications of an open license. But such licenses are a serious concern to commercial corporations whose natural instinct is to retain knowledge resources, or at least to retain the ability to mix code, proprietary and open, as needed. Not all open source licenses are as reciprocal as the GPL, and this allows companies to at times choose an appropriate one when initiating their own open source project. However, there is little or no choice for a company when they decide, for strategic reasons, to nurture or back an open source product which already has a community working on development and where the license has already been chosen. In practice being prepared to work with strong licenses is necessary.

A consequence of this is that, if companies want to seriously work with open source software, especially its distribution, there is a need for them to build in-house expertise in the various open source licenses and how they can be used. Interviewees from both companies indicated their legal department had become more skilled in open source licenses, and established some form of open source review board. The role of the latter group being to scrutinize code that is created for clients or is distributed in any way to check for various license implications. This move has been accelerated by the increasing demand for open source products and software by customers encouraged by the prospect of reduced vendor lock-in.

Dual Licensing

There are possibilities of dual licensing (Valimaki, 2003; Olson, 2005; Comino and Manenti, 2008) and other complex ‘work arounds’. Dual licensing made to release software under two different licenses, one which is usually open source and the other proprietary (Valimaki, 2003). However, according to our respondents, dual licensing and other work-arounds should usually be avoided because they cause complexity and critically, uncertainty as to the validity of potential IPR claims. If dual licenses are not the ‘magic fix’ envisaged then managers need to consider other issues when using, adopting or mixing company code with open source software.

Most Oft Adopted Licenses

Within the broad open source movement there are a large variety of different licenses in use ranging from the weak to the strong. Unsurprisingly, large companies appreciate and (when able to make a choice) implement weaker licenses such as Apache or BSD, rather than the viral/reciprocal GPL. However, often there is no choice and the license that will be used is dictated to by the existing licensing regime of the product and community that they choose to work with. More significantly, companies will try to affiliate with a product that is successful in terms of sustaining a strong community of developers around it, what ever the license they use. Thus IBM, SUN and many other large companies contribute to and participate in LINUX development. The technical credibility and developer base is ultimately the more relevant issue since resource capture – of experience and knowledge held by the community – is often the primary goal.

Environmental Impact on the Product

The question here is exactly who is the customer? If a company has assumed a more inner source form of approach then unless the product or service produced is open source, and importantly, distributed as well then open source develpment processes will affect the customers of this company. If there is no re-distribution this is not something that customers need to worry about. Inner source implies strongly that the product created is often NOT open source which again brings us back to the same conclusion as above. However, in cases when some form of sourcing is adopted where an open source license is used AND the product or service is then distributed to customers then the latter will be affected in a few ways, some useful and some less so ways (Shaikh and Cornford, 2008).

Thus, under conditions when the product/service is open source and customers appropriate it then they need to evaluate some issues such as; license; viability of community; strategic opportunities of product/service;

after-sales service offered; reduced vendor lock-in; ability to innovate in-house; lack of documentation; and lack of clear contractual obligations from community in relation to support (Shaikh and Cornford, 2008; Shaikh and Cornford, 2009).

License is and always will be a slightly contentious issue depending on the degree of reciprocity required (Fitzgerald, 2006). Some companies have chosen to use a dual form of licensing to side-step concerns with GPL use (Välimäki, 2003; Ven and Mannaert, 2008) but in general companies favour a less reciprocal license such as the BSD or the LGPL. This ensures that they can take the open source software and mix the code with proprietary software without making the latter open source, unless they so wish (Fogel, 2005).

Customers also have the possibility to use open source software as a way to increase in-house development, expertise, and innovation (of software and process) (Shaikh and Cornford, 2009). Open source development processes if appropriated in-house, usually require a degree of organizational change (Lindman *et al.*, 2008). Such a shift can lead to greater distribution of control (Pulkkinen *et al.*, 2007; Shaikh and Cornford, 2009), and thus a change in governance and reporting structure, a more emergent and bottom-up approach to development and innovation, greater spread of domain knowledge and expertise and more communication and collaboration within the company (Shaikh and Cornford, 2009).

Level of Documentation and After-Sales Service

Finally, a lack of or limited documentation can become an issue for customers but this is where a company offering open source products or services will ensure that this concern is addressed (Koponen *et al.*, 2006; Melian and Mahring, 2008) by filling the gap left by an open source community (which is often reluctant to create copious amount of documentation). Customers are naturally concerned if they are required to rely on community support alone because there is no likelihood to have a contractual relationship with a community. Contracts can, and are signed with individual members of the community but as a whole a community cannot be 'controlled' by a company (Shaikh and Cornford, 2009).

Viability of community refers to how sustainable is the community that is busy concentrating on the product of company interest. This is only relevant if the customer is keen to develop the product in different and innovative ways (Hyatt and Mickos, 2008) otherwise the openness of the source implies that the customer is free to take the code and find any vendor that offers the most financially viable support. There is less vendor lock-in with open source software so customers have more options to reduce their costs without being tied into one company. After sales services are, in principle at least more flexibly sourced.

Theme/Memo/sub-theme	Quotations	Respondent
License and IPR Regime	*"We help our customers make better use of open source… we can negotiate any support agreement they want. In some cases we provide consulting and outsourcing services… customers are more comfortable with open source because they know that at the end of the day they know that nobody can take it away from them. It is a matter of getting (our company)to support them – they will trust that from an open source standpoint because in the worst case if one vendor doesn't meet their support needs they can get another vendor to support them – they are not locked into any one vendor for support".*	Open source Consultant in Company B
Dual licensing	*"We don't do dual licensing, because we release some stuff for free, for free download and then you can buy a support contract. But our main way is by including components within bigger software product… It's down to, there is a different sort of business model. The business model of Linux is essentially around the ecosystem".*	Open Source Manager in Company A
Most oft adopted licenses	*"Where the project has been set up externally by another group then, we work obviously with that. So we contribute code to Linux, which is under the GPL. We contribute code to Apache, which is under the Apache license. Where we actually open source a project and create a new project … we tend to do it around similar to our*	Open source Consultant in Company A

	Apache license, but one we've defined. We prefer the Apache type license. Because it enables open source code to be mixed with closed source code and to create combined products, which is what we see as a lot of the way the industry is going and the usage is going and also, it enables a good sort of business model around. It works well from our point of view".	

Table 3: Glimpse into Product Related theme data

Community/Organization

Companies wishing to take the open-sourcing route may need to pay attention to the various tactics used by open source governance structures to intrigue, enrol and sustain the community over time. Open source development projects cannot expect contributions but can only encourage them by creating a product that will be useful to a large population of people. The more domain centric products have a much smaller community of people contributing to it. All open source projects have some form of governing body which is often comprised of the core development team. The core group tends to work more full-time on the project but other members of the community usually make very tiny contributions (Benkler, 2002) however, this is an effective way of not burdening the community and scaring off contributions.

It is attractive for developers if they can contribute when they want and to whatever part of the product/service that interests them, however another reason why they contribute and play any part in such development is because of the access it gives them to cutting edge work, ideas and experts. A key motivation of OS developers to contribute to projects is what they can learn from each other as a community (Lakhani *et al.*, 2003; Lakhani and Wolf, 2005). Open-sourcing provides not only the community the same free access to good ideas and work but gives the same right of entry to its company employees as well.

The community and the company enjoy a great level of freedom in open-sourcing yet this is not to imply that there is no control. Any behaviour that is counter to the open source way can and is punished heavily. Control mechanisms in open source projects may be informal but they are just as severe as those implemented by companies. Companies that practice open-sourcing are able to exercise a mix of enforcements which are more effective when used together.

Theme/Memo/sub-theme	Quotations	Respondent
Community Approach	*"The goal in developing a community, you know, beyond the desire to have that community to support the open source project, what we wanted is a self sustaining venue in which people could get their questions answered .Find information. And in which we had Evangelists that were not from (our company). If you look at a traditional software company like, they may have a news group or a forum that people can ask questions and sometimes can answer … you go in there and you interact with a support team. That tends to be a fairly heavyweight process. It takes time. What we wanted was this… we wanted a self sustaining community where you know, seven, twenty four, sixty five, you know. There is just people discussing, you know, chatting about one of their experiences and their problems, possible solutions. Once you get enough people together, they begin to answer one another's questions. It's not … clearly, from a development standpoint, we still stay engaged and monitor … we keep our finger on the pulse. It's self sustaining. It's out there and it's a value".*	Consultant of Company B
Match between company and community	*"To share the costs of polishing that piece of software, if you are the only company involved then you should ask yourself the question why is this. Are we so innovative that we were the first ones to think of this? It might happen that there is something about the software that makes it tricky to manage… so being able to share the development and maintenance costs is important".*	Open source Manager at Company B
The issue of	*"There is no formal requirement to process, right, in typical open*	Open source

requirements	*source. Part of our job is, when we are getting ready to think about an extra release, we'll ask the (Company A) community… What are your requirements? What do you guys need? It's our job to kind of on the (Company A) side to collect all that and to sift through it and represent what we believe are the best set of priority requirements. At the end of the day, (our company) is the one paying the salary for the open source programme. It's a set of requirements and because these are our people who are paying for it, gives us the right to say, we want to work on this stuff. It had to happen within a set of constraints. Our play here is the collaborative one with other community members including business partners. So, to be honest, it's a bit of a balancing act. We've got a set of (company) priorities and there is a set of things that community is interested in and we are interested in how the community is interested in our stuff, because it pays off. It's got to be something here that is also interesting to the community. I don't know that there is any magic there, other than it's just an experience that we've gathered over the years of having done this".*	Manager at Company A
Incentive schemes	*"A viral plank. We wanted something that would grow from the bottom up, you know. Not only would the chief technology, chief probation officer say, here's our strategy for solving in this part of our IT strategy…So, in other words, capturing the hearts and the minds of the developers out there".*	Company B employee

Table 4: Glimpse into Community Related theme data

Motivation

In our interviews with managers and technical developers employed by these two large technology companies we have heard many distinctive accounts of their open source involvement. It is clear from our data that these companies have developed extensive involvement in open source activity over almost a decade and indeed rely on it as a non-trivial and integral part of their technology strategy. Integral, but not in general to the point of total dependence – in most of the detailed cases we have been told about open sourcing has been used to build up expertise and activity in certain areas, but is often complemented by other, proprietary technologies and products. Here we explore three of the major perspectives that we have seen in our data. These are Innovative Products Focus, Resource Capture, and Steering the Environment.

Innovative Product/Idea Focus

A primary reason for participation in open-sourcing for companies is to lead the market with innovative ideas and products. Innovation, for companies, implies success, because as one open source manager reported to us "*open source invention becomes innovation when it has business value. It's that connection. It's not just a great invention. It's that it's helped the business*" (OS Manager at Company A). The product or idea needs to generate profit or business through indirect means for it to qualify as an innovation. A product is thus not important in itself but for the value it can produce and if it is vital then this makes resource capture all the more relevant for long term sustainability.

Companies motivated to approach open-sourcing for reason of innovative products employ strategies that span across issues that affect the product being created, community values and sustainability, and an evolution in process. At the product level companies focus primarily, but not exclusively, on the license. Non-viral licenses are preferred but as managers pointed out, the more important issue is how healthy the community is that is working on the product. This again links the product back to community issues of resource capture (see below).

It is also not uncommon for companies to protect against community breakdown or disinterest. Long-term resource building around the product usually entails creating a team of in-house developers that work closely with the open source community. Learning and expertise building over time motivates the company to contribute financial resources to an in-house team. At the same time, companies manage their interaction with the community through strategies that maintain community interest without explicitly controlling the latter. Our interviewees mentioned different approaches taken by companies such as offering community

members free hardware, sponsorship, short and long term contracts etc, but also some version of the product they are working on. The latter is an approach to grab the attention of the community and to encourage new idea infusion into the product and the process of development.

Resource Capture

At the heart of the idea of open sourcing is the notion that there is something to be gained from a 'global but largely unknown workforce' (Agerfalk et al 2006). But notions of what exactly is gained are, not surprisingly, quite varied. At least four general ideas emerge from our data - and they are not quite the same as the wider open source literature seems to propose (West and Gallagher, 2006; Huston and Sakkab, 2007).

Perhaps the least important aspect of open sourcing from the perspective of our respondents is programmer activity – good people who will work on a given problem. Of course this does occur, and some of this activity is identified by our respondents as being of a very high quality. The relationship is, however not usually seen as so linear – that is the people in the 'unknown workforce' don't in general just do what they are asked, nor do they just do what *they* want, but what is appreciated is that they comment, critique and improve ideas they are presented with. So the resource that is captured is more a thinking resource than doing resource. In our data probably the most common resource that these companies seek to capture is not, as such, code or people to write it, rather the resource that these companies seek is a validation of their requirements or design.

We also have seen classic open source ideas of using the community as a source of talent to recruit – though from our respondents this is not likely to be the hiring of a star hacker (though it does occur), but more often the identification of a likely graduate trainee. For example, active participation in an open source project, even if modest, is seen as a very positive endorsement of the real programming ability and potential for computer science graduates, "*If I find somebody with open source involvement on a CV and demonstrated track record in open source, which of course is something, if they put it on a CV I can immediately go and check it by looking at that project. Then, they will go right to the top of my priority list for hiring*" (OS Senior Developer at Company A).

Another ambition that our interviewees identify in their participation in open source projects is to feed into their own organization new ideas about software process. Here the resource captured represents new, perhaps lighter weight and more flexible approaches to software. For example, most open source activity is very light on documentation, and very minimal in design materials. This quality is seen by some as a useful antidote to more bureaucratic tendencies within their own organizations. We also have heard programmers and systems development staff speak about how pleased they are to deal with real 'users'. This is in a way a puzzling statement, since most open source programmers that they will interact with are far from users in the usual sense of the word, but for our respondents just talking to other people beyond their own organizations feels like a liberation.

Steering the Environment

A third approach to participation in open source projects we have seen is the view that participation allows a degree of influence on the direction that a project takes. It should be clear, given the relatively high degree of open source code in network infrastructures, programming environments and some database products, that many large systems companies will be interested in influencing how these products evolve. This may be expressed in terms of steering towards a chosen technology, provision of certain interfaces, or support for certain data standards. The means of steering are various and can include the direct proposal of these as requirements, but will also often be matched by the provision of people to undertake some of the desired work and perhaps the release of technical information to allow the work to be done. Other strategies can include taking on the less desirable elements of a project, for example documentation tasks, or testing. Setting up and maintaining test suites may be a very powerful way to steer a project, or to benchmark it against some other software.

Participation in open source projects is not the only way that large technology companies find to collaborate with other people or other companies. As described to us by one senior manager, an open source project is just one option among a number that can be considered and used. For example, given a need to collaborate, the company might choose between different routes – a formal company to company alliance, a pooling or

sharing of IPR, a standards body (existing or perhaps newly created), or a contractual relationship. Of course, all of these are more focused on relationships between corporate bodies – but so too on occasions are open source projects. It might go against the grain of traditional notions of open source, but as described to us, an open project might be seen as the appropriate choice when two large companies choose to work together. It has certain advantage as a way to achieve collaboration; given the choice of license the lawyers are kept largely away, openness can help build trust, software processes are generally light weight, and it can be set up very rapidly, "*based on the companies, apart from you that they also involved there. Then, at least, you know that they are going to share the costs of policing that piece of software. ….there is many reasons to use open source as one of them is to share the development and the managing and costs*" (Development Platform Product Manager at Company B).

Requirements Management

An important question for companies supporting open-sourcing is the direction a software product will take in the future. More specifically, how, and by what means they can influence this direction by feeding in new requirements for development effort, or encourage appropriate interfaces for their own in-house efforts. This poses the question of how companies can 'dictate' requirements for a product that is part produced by a community and part by company developers? The community may be able to offer ideas, code contributions and support, however there are many cases where the open-sourcing company wants to focus on one particular aspect of the software, but the wider community disagrees. In such cases one coping strategy described by our respondents is to keep an in-house team of developers who are also familiar with the community and the code and who do the tasks and changes that the community is reluctant to do. Such a group may develop the required functionality either for proprietary use, or to feed back to the open source project. Though it must be recognised that this is an added expense for the company it may have value in fostering wider in-house knowledge and work as an insurance against over-dependence on the open source community. Indeed, it is understood that the community will always be a little more risky.

Managing a Project

Our respondents report strategies such as creating a Steering Committee comprised of employees but also developers from the community. Usually companies begin such a committee with a situation where more members belong to the company. However, the case we studied showed how this needed to be addressed if the company wanted any feedback from the community or to sustain its interest and commitment. Very quickly community interest in the project began to flag and the company was forced to reassess its attempt at exercising control over the wider project. As we were told, it learnt that in order to keep any control it had to '*let go of control*'. More community members were then asked to join the steering committee, voted on board depending on their experience and contribution to the project. But still, high profile developers are often voted onto such a committee as a strategic move by the company to appease the community and to sustain an informal route for control of what happens to the code. Our study also presents data on crisis mitigation strategies, for example to prevent a fork. While forking is often seen as a safety valve for open source projects, in these contexts if it occurs and splits the community a project may lose its critical mass of eyeballs and be far less useful to the open-sourcing company.

Managerial Strategies in Product, Community and Process

We see companies attempt to gain greater control over open-sourcing. It begins with micro activities at the product level (which also feed into the process of development) and go on to encompass the macro process ideas of the entire supply chain.

Some strategies employed by companies are well thought out and planned but we found through our interviews that often the uncertainty of dealing with the unknown workforce and license forced managers to step back and let the process of coping emerge. Large companies, like the ones we studied, have the resources (human and otherwise) to allow such unfolding to emerge but it would be interesting to study open source use, appropriation and distribution practices of SME's, where financial and human resources are more limited.

Innovation in companies is an aspect that has been highlighted by exciting firms like Google that focus on hiring people *to innovate*. The companies we studied clarified that their desire to branch into open-sourcing

was not principally to innovate, but rather to manage the environment and set industry standards for adoption. The strategies they employ in order to achieve their aims, however, are very innovative. It is with such practices that they manage to push changes into the product, direct the community and 'control' the environment.

Our research in this area has lead to a focus on how technology and artefacts that are created in an open source process engage with the community to change practices, product and the process of development itself. This is part of our current, and future agenda of research in the domain of open-sourcing.

All our interviewees, when asked why they feel that companies like theirs need to liaise with open source communities, cited their main reasons to be as access to a support network for expertise, testing, and ideas. These reasons form a key driver for pursuing an open source strategy. It is the community, perhaps more than the code that it provides, that interests large companies. The continued support, help, and importantly ideas (collective innovation) that attracts companies to open source collectives. Companies also understand that they need the goodwill of open source developers and must be seen to contribute back to the community. But like all other decisions concerning open source, this brings with it a number of concerns for managers. Before aligning itself with any community a company must take into consideration how well the community matches with the company needs, governance structure, and the complementarity of goals.

Match between Company and Community

While conducting our interviews the respondents identified a number of aspects that they consider when making such decisions. They include (in order of relevance, again indicated by the interviewees); a healthy community; viable and needed technology; compatible license (if possible); and whether a community has members that are employees of other companies.

The most important factor suggested by all the respondents to the question of how they choose to work with a community was whether the community is healthy or not. If the community isn't healthy then it is unlikely that a company will want to get involved. The idea of a healthy community is interesting because, when probed, the interviewees explained that they meant a community that has sustained itself for at least some period of time, and that other companies should also be interested and working with the community. This latter point was well argued by an interviewee who explained that other companies' involvement is a good indicator that the product is of good quality. More companies taking an interest will cut the effort of each one and costs can be shared, "*to share the costs of polishing that piece of software, if you are the only company involved then you should ask yourself the question why is this. Are we so innovative that we were the first ones to think of this? It might happen that there is something about the software that makes it tricky to manage… so being able to share the development and maintenance costs is important*".

The second most important factor was stated to be the viability of the technology and the business value (direct or indirect) that the technology could bring to the company. This issue was closely linked to the question of license. However, the related issue to be considered is whether the application the open source software is being combined with will eventually be distributed to customers. If the answer to this question is no, then there is less reason to be concerned about the license. But if the answer is yes, then the potential for leaking their own IPR through viral licenses is a serious concern.

Incentive Schemes

Companies use communities to get access to ideas, and innovation through collaboration. Various incentives are utilized to harness the sustained interest of a mass of developers around a product of interest. Managers from both companies studied provided a similar list of strategies used including short-term contracts, bounties, even full-time employment, but the most relevant form of enrolment was the technology itself, "*It was the technology. They were really excited about the technology*" (Company B employee). The interviewees were clear that the best way to enroll open source community members was to offer them an interesting technological challenge and access to leading edge ideas. Of course the other strategies mentioned are employed, and usefully, but technology and what it offers is the most reciprocally valuable approach for community-company collaboration.

Process

There are ten criteria that we found to be indicative of a process (see Table 1). The main aspects that require focus in process revolve around questions of communication, control and governance. The interesting elements arise when these three criteria are seen together. There is a clear link between more online based communication, distributed control and a democratic governance structure to match, and this is possible because of the transparency afforded by the license of the product/service. Open source offers the most democratic governance structure but open-sourcing is able, through its combination of offshoring and open source techniques and ideas to enjoy a variety of benefits. Often the problems that companies face with open source ideas/process or software adoption like lack of documentation, license problems etc can be countered by some level of company adoption thus making open-sourcing a very attractive model financially and process-wise.

Theme/Memo/sub-theme	Quotations	Respondent
Modified Development Process	*"An internal open source bazaar, and it is run rather like a bazaar. Anyone can start a project. There are minimum controls around it to ensure that people don't put code in it that is inappropriate. But because it draws on people from different parts of the corporation with different levels of knowledge you can have some interesting mixtures of people who understand how to connect the internal systems together with some behaviour from external systems".*	Open source developer at Company A
Shift in governance and control	*"Letting go of control…* (which is only possible) *if you've got a shared vision. Obviously, there needs to be something that's sort of binds people together. If you've got the shared vision then you can sort of let go of control".*	Company A Manager
Managing innovation	*"(Our company) is a big company with a lot of access. We are really concerned about the pedigree, the providence of source code of a commercial product. That was easy when we all wrote it and was all behind (our company) firewall and we knew exactly where it came from and we knew who wrote it and we know when they did. But now, when you throw open source into that… number one is people are not (our company) employees. Sometimes we don't even know… the people who have written it have long since gone... The last thing we want is to see a law suit. There is a big part of what (our company) does to continue to assure the integrity both of what we contribute to open source and then we bring that back into (Company A) that everything is clean and it can go out in the commercial product. It's just making sure that what we've got is clean and it can go out…. The amount of risk that we are willing to take there is going to be very different from the amount of risks some start ups company will get".*	Open source Director at Company A

Table 5: Glimpse into Process Related theme data

Governance Model

Our respondents report a number of strategies such as creating a Steering Committee comprised of employees and community developers. This may have started with a committee where more members belong to the company. However, the cases we studied showed how this needed to be re-thought if the company wanted positive feedback from the community or to sustain their interest and commitment. Very quickly community interest in a project began to flag and the company was forced to reassess its attempt at exercising any direct control over the wider project. As we were told, it learnt that in order to keep any control it had to 'let go of control'.

More community members were then asked to join the steering committee, voted on board depending on their experience and contribution to the project. Our study also reveals data on crisis mitigation strategies, for

example to prevent a fork. While forking is often seen as a safety valve for open source projects, in these contexts if it occurs and splits the community, a project may lose its critical mass of eyeballs and hence become far less useful to the open-sourcing company.

Managing Innovation

Part of the problem of attempting to encourage innovation is how to manage 'what happens next' - and it can suggest other open innovation opportunities that may be less easy to control. This can for example lead to the possibility of security leaks, divergent technology strategies or creation of competing products, not to mention contamination of work by viral licenses. Code itself can become vulnerable, especially if not all the employees are well-versed in the particularities of open source licenses and their implications for code hygiene. A positive aspect of easing control and distributing it more widely in and beyond the organization are changing work practices of employees, their motivation and their sense of collectivity through the need to share ideas and work.

The companies studied informed us about various initiatives that had emerged from the adoption of not only open source software but the open source way of working and belief system (i.e. sharing, collaboration, and better coordination). One of the companies has "*an internal open source bazaar, and it is run rather like a bazaar"* (Company A developer). Innovation encouraged internally in what used to be a traditional organization. Interviewees claimed that such changes were triggered by the adoption and awareness of open source.

Organizational Factors

Skill development – Working with an open source community gives quick and sometimes very aggressive feedback so if employees are able to cope in such a clearly meritocratic form of contribution they show skill, patience and endurance. Due to quick feedback developers do not have to wait for long before they know if they are working in a productive manner. Skill development, through greater transparency of your work, reputation based style of moving higher in the open source echelons, more peer-review and aggressive feedback leads to good training and skills development (Shaikh and Cornford, 2009).

Career possibilities – The large companies that have turned to adoption of open source develpment processes we see a trend that clearly shows that in-house employees that are creative with open source, or are able to work well with open source communities follow a similar path of career possibilities as other in-house developers as far as promotions are concerned (Lindman *et al.*, 2008; Pulkkinen *et al.*, 2007; Shaikh and Cornford, 2009). With an increased interest by companies in open source, if any developer is able to show that they have contributed to open source projects and had their ideas and code accepted, this is seen as an asset and sign of strong peer review. Such people are being hired more easily than college graduates that are not able to show a similar manner of peer reviewed work (Shaikh and Cornford, 2009).

Communication

Large and small companies use various means of communication, not all of which are based on face-to-face communication. Open source development processes make it possible for globally distributed employees in very large companies to find it more acceptable to imitate open source ways of working, coordinating and communicating. However, this is not to imply that open source has lead the way in this respect because video conferencing, email, forum chat and other forms of real time chat are quite the norm for most companies. The Internet (Castells, 1996; Castells, 2001) made this possible rather than open source practices.

Fluctuation – open source development processes are able, through their innate flexibility to manage fluctuation in a manner that hierarchical companies cannot. Open source development processes are open, transparent to quite a degree, loosely coupled and based on collaboration in an agile manner that helps deal with a fluctuating external market (Butler *et al.*, 2008). A change in the direction of code is a norm in OSSD. In fact open source thrives on different and often conflicting requirements and the process has a built in safety valve – to fork the code. Should the need arise, code can be forked and the community break apart and follow the fork or the original code. This practice is not encouraged as it can destroy critical mass of eyeballs around code but if problems become too severe then this is a healthier possibility for the code and the community (Fogel, 2005).

Hybridization of Process

While traditional software has a clear separation between developer and user it is not true that all OSS projects have this high degree of overlap of the two groups. Indeed studies indicate that the question of requirements gathering is more fluid and emergent in open source projects partly because there is a heavy overlap between users and developers – in effect they scratch a personal itch (Raymond, 1999). Thus if we speak of 'pure' open source communities then it is not easy to differentiate between users and developers as they so often create software for personal use. The software grows and evolves quickly when more than one user/developer feels that this code can help solve their own problem (Ljungberg, 2000).

With an increased adoption of open source development processes, software and ideas by companies (Agerfalk and Fitzgerald, 2008) we have noted a slight hybridization of each (Shaikh and Cornford, 2008). Criteria that are key for an open source development process like collaboration, sharing, openness of source, testing, debugging, community effort, and 'loosely' structured governance model (Capra and Wasserman, 2008; Demil and Lecocq, 2006; O'Mahony and Ferraro, 2007; O'Reilly, 1999) are still operational when users and developers are different yet there are subtle changes (Shah, 2006).

Even when companies adopt open source development processes they usually do liaise with an open source community so the idea of a clear separation between both is not a reality (Shaikh and Cornford, 2009). In order to foster open source development processes in-house it requires a degree of inspiration and help from an open source community. The company needs to nurture the community to keep it interested in helping the company and to not view the latter as a parasite (Shaikh and Cornford, 2008). Thus if a company is serious about adopting open source development processes then invariably most examples reflect a need for community involvement. The community, in such cases is used as a testing ground for the software that the company is interested in developing for commercial purposes.

Most companies need to create a middle structure where their internal employees are able to communicate with developers in a manner that they are accepted by the community and not seen as outsiders (Shaikh and Cornford, 2009). This usually requires a degree of contribution by the employees towards code, advice and testing the community product (Agerfalk and Fitzgerald, 2008). Such employees have to fit the 'profile' that makes them suitable people for such a job. Often companies choose such people from an open source community, hire them in-house, and then allow them to devote their time to creating strong links between the company and community (Shaikh and Cornford, 2009). Such developers are usually well-respected members of the community so have little trouble gaining access to the community, in spite their new role within a company.

CONCLUSION

The basic premise of company involvement stems from a worldview that sees collaborative and collective work as leading to the ability to harness more ideas, more eyeballs, which will in time lead to cutting edge innovation. Sometimes the goal is to foster competition in the market to disturb possible monopolistic practices, but on other occasions open sourcing collaboration can be a means to reduce competition and to better secure a place within a stable market. Open source, and all that it implies, - open process, greater flexibility, better scalability and faster time to market – suggest that it is no surprise that companies have attempted to find a way to work with this movement. It has nevertheless been a difficult process for companies who have different goals, different governance structures and software practices.

Our data reveals an evolution in practices as the companies studied have worked through their understanding of what it means to work with open source communities. Such change occurs through use and engagement with the software or product itself, as much as with the community. Each company that has taken an open source route, be it in software or only process, has had a different business model structured around open innovation ideas. It is the open innovation business model based on collaboration and a slightly divergent manner of creating a profit, be it through the sale of complementary services or a strategic move of limiting competition that has been sought after by companies. Open source, according to our data, has in most companies lead to a 'culture of innovation' where innovative outcomes (products) have become the catalyst

for a structure of innovation (process change). There is a degree of sharing and process change that indicates an evolution in practices in all companies which our respondents believe to be stimulated by a greater adoption of open source software and ideas in-house.

Our company focused study has facilitated our understanding of when, how, and why companies turn to open source adoption, and the various strategies they employ to ameliorate their concerns about such a strategy. The next stage of the study intends to provide a more balanced understanding of this relationship through an analysis of the implications of such collaboration for open source communities themselves. Some initial findings suggest that these communities do increasingly encourage company involvement, seen as providing sponsorship, and supporting the longevity of both the product and community, job possibilities and a chance to prove developer ability.

Best Practices of Open-Sourcing for OPAALS	
Process	Open-sourcing encourages consortium forming with other large and SME companies that can create a new set of standards with which open source software/service/idea development is compatible. The open source element engenders collaboration, transparency and sharing – we found that companies aim to collaborate on various open source projects but also on the creation of open standards. This is a useful way to build infrastructure, but it also ensures that a large number of companies begin in the same place and then build mutually agreed vertical applications on top of the infrastructure.
	Adoption of open-sourcing ideas can speed up time to market products and services thus benefiting any company that forms part of the ecosystem. SME's are better able to compete with large monopolistic firms by innovating with their business models to incorporate open source ideas. This is conditional on a number of factors such as; how closely connected is the company to a community; how alive and healthy is the community in terms of number of developers contributing or eyeballing the code; the type of open source product (is it a cutting edge product, strong competitor, saves cost etc).
	Open-sourcing ideas can help cultivate both top-down and bottom-up innovations of products and services thus making adopting SME's more competitive. – SME's can encourage greater innovation by adopting an open source process which makes for a more transparent, collaborative and sharing environment. Tangible techniques are pair working (taken from agile practices of development); maintaining a database of ideas where all the employees of a company are encouraged to add ideas, products etc thus allowing for greater dispersion and mixing of ideas; and fostering collaboration with external communities.
	Open-sourcing can help establish and innovate to create a standard platform for collaboration between companies and between companies and open source collectives thus enlarging the ecosystem. – Creating a standard locks other companies into your standard thus ensuring the continued existence and relevance of your standard (and making you an important player).
	Open-sourcing can help reduce exploration and R&D costs for new idea development and a more collaborative form of competition is encouraged which can help reduce wasteful duplication of work. – the openness created by shared databases of ideas, transparency of process, shared standard etc implies that companies only need compete on value added services and products rather than the basic infrastructural elements.
Product	Avoid dual licensing schemes as they lead to complexity and at times, ambiguity about ownership. Usually best for a company to adopt a less reciprocal license like LGPL or BSD when starting a new project. However, if the company wishes to collaborate with a community and/or company then if the product is core to their success, the company is obliged to accept the license of the product chosen by the community. Again, a choice need to be made on the basis of what the company intends to do with the open source product – ie use it in-house only, amend it, distribute it, provide services to facilitate its use, or adopt it to encourage a change in the organizational practices and work practices of sharing and collaborating.
	Better quality product/service through greater scrutiny but the author is now also aware that he is building a reputation so will avoid releasing bad code/or ideas. Perhaps a slow and steady adoption of open source ideas, methods etc would be appropriate to disprove the myth of insecurity, questionable quality and reliability surrounding open source software/ideas. – Companies can encourage best practices amongst the team of employees across companies. Openness of process will and has lead to better quality of products and software through greater scrutiny – reputation based work.
	With open-sourcing there is increased transparency and openness of product/idea and process. This fact allows trust building in an ecosystem making an ecosystem possible and sustainable over time. – Examples such as FOSSOLOGY where best practices from various companies are pooled to create trust and show good faith towards each other. Building shared open standards, sharing best practices, ideas and even employees across companies

	and communities helps build a trusting ecosystem.
Community/ Organization	Open-sourcing is useful for leveraging company relations with open source communities thus potentially giving the company access to a large pool of developers and testers. Possible close and continuous operational interaction with the user community thus providing a constant pool of expertise to SME's who can often ill-afford expensive consultants on their payroll. – SME's are able to deliver maintenance at a lower cost to its clients, around the clock support because of the global nature of open source communities, and hire developers from open source communities. Indeed, in our research we found that many small companies were start-ups by a handful of open source developers who continued to work closely with the community they belonged to prior to creating their own company.
	Open-sourcing fosters greater focus on individual contributions thus allowing every contributor 'face' value through the electronic trail s/he leaves. – Companies can focus promotions and appraisals based on the reputation that employees garner through their work with communities. But this is also true of internal work in a company. The transparency almost forced by open-sourcing makes employees work diligently and to offer their best work due to high visibility. This can have a slightly reverse effect on some employees who are not able to cope with the level of monitoring possible with such an approach.
	Open-sourcing promotes innovation in not only open business models but also in methods that companies can use to reach out to open source communities which stress 'trust building bridges' between various communities – a more cohesive digital business ecosystem. Companies that we studied had shown real innovation in their methods of reaching out to communities which they honed over time. Their techniques included championing schemes, bonus systems, bounties etc for short-term collaboration. However for long term collaboration the techniques involved building trust and so implied relationship building methods such as exchanging staff/ developers between community and company, sharing software and ideas with the community as a way to give back, hiring developers from the community, allowing their own employees time in their contract to work for the community to build a strong relationship and gain expertise.
	Open-sourcing indicates that companies would probably benefit from reaching out to an open source collective (if possible) whose governance structure is similar to the organization. Governance is a key issue in OPAALS and open-sourcing provides strong indication of how to achieve some balance between distributed and centralized control and hierarchy. – Larger companies in our study faced this issue more strongly than SME's, and they called it letting go of control rather than losing control. This is easier for them to do at one level because they have more flexibility with their greater resources. They can hire employees to become insiders in the community to sway decisions, create steering committees, create an in-house team of development etc. SME's on the other hand may not have such resources but have another benefit of being more tightly linked to communities through their employees. Such SME's often, according to our study, have a less hierarchical structure. What also needs to be noted is that many open source communities are also quite hierarchical in governance structure.
	Open-sourcing provides some relief to SME's (who are particularly vulnerable as they do not have the necessary influence) from lock-in into particular commercial software/product/service. It also promotes rapid redeployment of skilled developers across various projects and companies. – Shared resources, best practices, open standards means that developers and their skills become transportable. This creates a more flexible and mobile community of open source developers and employees.
	Open-sourcing fosters more inclusive forms of interaction and idea sharing not only encouraged but almost a must for OSS 2.1 and for cultivating a digital business ecosystem in OPAALS.

Table 6: Relevance of Open-Sourcing for OPAALS

Best Practices of Open-Sourcing

In this deliverable we explained our initial framing of open-sourcing. It was this detailed preliminary work that guided our data collection in six different companies which have adopted open source at various levels and manners. There is much from this research that can guide our understanding of managing, creating, governing and sustaining a digital ecosystem. In Table 6 we outline some of the clear best practices that we found adopted by employees and companies in the six firms that we researched. For clarity and the sake of continuation, we have adopted the same breakdown of product, community and process. This makes understanding the various best practices easier and more practical to follow.

The next steps in our research involve studying the communities that are working closely (or perhaps obliquely too) with companies. This is our attempt to understand the 'other side of the picture' because we have a much better idea of why, when and how firms reach out to communities but in Phase III we want to interview and study open source developers that belong to communities busy liaising with companies. We feel this will give us a more complete understanding of the open-source phenomena and lead us to make a

deeper contribution to how open-sourcing ideas can be adopted by SME's attempting to create a digital ecosystem.

We find that our current research places us in an even stronger position to collaborate both within and outside OPAALS. Within OPAALS there are a number of researchers working in related areas such as IPR (WP11) but the themes of this work have the potential to translate to WPs 2, 3, 5, and 10. External to OPAALS we are able to communicate, and indeed have much overlap with other researchers working on open innovation, business models, and open source.

REFERENCES

Agerfalk, P. and B. Fitzgerald (2008) "Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy", *MIS Quarterly,* **32 (2),** pp. 385-400.

Ågerfalk, P. J., B. Fitzgerald, H. Holmström and E. Ó. Conchúir (2006a) "Open-Sourcing as Offshore Outsourcing Strategy". in *29th Information Systems Research Seminar in Scandinavia (IRIS 29) - Paradigms Politics Paradoxes,* LO-skolen, Denmark,

Ågerfalk, P. J., B. Fitzgerald, H. Holmström and E. Ó. Conchúir (2006b) "Open-Sourcing in the Celtix Project: A Case of Outsourcing to an Unknown Workforce?" in *27th International Conference on Information Systems (ICIS2006),* Milwaukee, Wisconsin, USA, December 10-13 2006,

Alchian, A. and H. Demsetz (1972) "Production, Information Costs, and Economic Organization," *American Economic Review,* **62 (5),** pp. 777-795.

Anderson, K. (2005) *Open Outsourcing* Last accessed: Last updated: Address: http://kelly.anderson.name/openoutsourcing/.

Benkler, Y. (2002) "Coase's Penguin, or, Linux and the Nature of the Firm", *Yale Law Journal,* **112 (3),** pp. 369-446.

Butler, S., D. Adebanjo and H. Ismail (2008) "Open Source Software and Leveraging of Business Effectiveness in Smes - a Case Study", *Ice-B 2008: Proceedings of the International Conference on E-Business,* pp. 93-100.

Capra, E. and A. I. Wasserman (2008) "A Framework for Evaluating Managerial Styles in Open Source Projects", *Open Source Development, Communities and Quality,* **275** pp. 1-14.

Carmel, E. (1997) "American Hegemony in Packaged Software Trade and the "Culture of Software" ", *The Information Society,* **13 (1),** pp. 125-142.

Castells, M. (1996) *The Rise of the Network Society, the Information Age: Economy, Society and Culture,* Blackwell, Oxford.

Castells, M. (2001) *The Internet Galaxy: Reflections on the Internet, Business and Society,* Oxford University Press, Oxford.

Clarke, G. (2007) *Management 'Scared' by Open Source* Channel Register Last accessed: Last updated: Address: http://www.channelregister.co.uk/2007/03/09/open_source_licensing/.

Deek, F. P. and J. A. M. Mchugh (2007) *Open Source: Technology and Policy,* Cambridge University Press, Cambridge.

Demil, B. and X. Lecocq (2006) "Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance", *Organization Studies,* **27 (10),** pp. 1447-1466.

Dickerson, C. (2004) "Open Source or Outsource?" in *InfoWorld,* IDG Network (October 22, 2004.

Dinkelacker, J. and P. Garg (2001) "Corporate Source: Applying Open Source Concepts to a Corporate Environment (Position Paper)". in *International Conference for Software Engineering (ICSE) - 1st Workshop on Open Source Software Engineering,* Toronto, Canada, May 15th, 2001, ICSE and Hewlett-Packard.

Fitzgerald, B. (2006) "The Transformation of Open Source Software", *MIS Quarterly,* **30 (3),** pp. 587-598.

Fogel, K. (2005) *Producing Open Source Software: How to Run a Successful Free Software Project,* O'Reilly, Sebastopol, CA.

Franke, N. and E. v. Hippel (2003) *Finding Commercially Attractive User Innovations* MIT Sloan School of Management Last accessed: Last updated: Address:

Franke, N. and E. Von Hippel (2003) "Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software", *Research Policy,* **32 (7),** pp. 1199-1215.

Ghemawat, P. and T. Hout (2008) "Tomorrow's Global Giants? Not the Usual Suspects", *Harvard Business Review,* **86 (11),** pp. 80-+.

Glaser, B. G. and A. Strauss (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research,* Aldine, Chicago.

Hoepman, J.-H. and B. Jacobs (2007) "Increased Security through Open Source", *Communications of the ACM,* **50 (1),** pp. 79-83.

Howells, J., D. Gagliardi and K. Malik (2008) "The Growth and Management of R&D Outsourcing: Evidence from Uk Pharmaceuticals", *R & D Management,* **38 (2),** pp. 205-219.

Hyatt, J. and M. Mickos (2008) "The Oh-So-Practical Magic of Open-Source Innovation", *Mit Sloan Management Review,* **50 (1),** pp. 15-19.

Jensen, M. C. and W. H. Meckling (1976) "Theory of the Firm: Managerial Behavior, Agency Costs, and Ownership Structure", *Journal of Financial Economics,* **3** pp. 305-360.

Jiang, B., S. Talluri and R. Calantone (2008) "Determinants of Interoutsourcing: An Analytical Approach", *Decision Sciences,* **39 (1),** pp. 65-84.

Komporozos-Athanasiou, A. (2008) "Information Technology Outsourcing in the Service Economy: Client Maturity and Knowledge/Power Asymmetries". in *International Working Conference on Information Technology in the Service Economy - Challenges and Possibilities for the 21st Century,* Toronto, CANADA, Aug 10-13, pp. 301-310,

Koponen, T., H. Lintula and V. Hotti (2006) "Defects Reports in Open Source Software Maintenance Process - Openoffice.Org Case Study", *Proceedings of the 10th IASTED International Conference on Software Engineering and Applications,* pp. 434-439.

Lakhani, K., B. Wolf, J. Bates and C. DiBona (2003) "The Boston Consulting Group Hacker Survey". in *Linux Conference Australia 2003,* Perth, Australia, January 22-25, 2003,

Lakhani, K. R. and R. G. Wolf (2005) "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects" in *Perspectives on Free and Open Source Software*, (Feller, J., B. Fitzgerald, S. Hissam and K. R. Lakhani eds) MIT Press.

Lindman, J., M. Rossi and P. Marttiin (2008) "Applying Open Source Development Practices inside a Company", *Open Source Development, Communities and Quality,* **275** pp. 381-387.

Ling-Yee, L. and G. O. Ogunmokun (2008) "An Empirical Study of Manufacturing Flexibility of Exporting Firms in China: How Do Strategic and Organizational Contexts Matter?" *Industrial Marketing Management,* **37 (6),** pp. 738-751.

Ljungberg, J. (2000) "Open Source Movements as a Model for Organizing", *European Journal of Information Systems,* **9 (4),** pp. 208-216.

Melian, C. (2007) Progressive Open Source, PhD Thesis, Stockholm School of Economics.

Melian, C. and M. Mahring (2008) "Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard", *Open Source Development, Communities and Quality,* **275** pp. 93-104.

Neumann, P. G. (2000) "Robust Nonproprietary Software". in *IEEE Symposium on Security and Privacy,* May 2000,

O'Mahony, S. and F. Ferraro (2007) "The Emergence of Governance in an Open Source Community", *Academy of Management Journal,* **50** pp. 1079-1106.

O'Reilly, T. (1999) "Lessons from Open Source Software Development", *Communications of the ACM,* **42 (4),** pp. 33-37.

O'Reilly, T. (2005) *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software* O'Reilly Media Last accessed: Last updated: Address: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

Pulkkinen, M., O. Mazhelis, P. Marttiin and J. Meriluoto (2007) "Support for Knowledge and Innovations in Software Development - Community within Company: Inner Source Environment", *WEBIST 2007: Proceedings of the Third International Conference on Web Information Systems and Technologies, Vol SeBeG/eL,* pp. 141-150.

Raymond, E. (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary,* O'Reilly & Associates, Sebastopol, California.

Schneider, F. B. (2000) "Open Source in Security: Visiting the Bizarre". in *Proceedings of the 2000 IEEE Symposium on Security and Privacy (the ``Oakland Conference''),* Berkeley, CA. Los Alamitos, May 14-17, pp. 126-127, IEEE Computer Society.

Schneier, B. (1999, September 15th) *Open Source and Security* Crypto-Gram: Counterpane Internet Security, Inc Last accessed: Last updated: Address: http://www.schneier.com/crypto-gram-9909.html.

Shah, S. K. (2006) "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development", *Management Science,* **52 (7),** pp. 1000-1014.

Shaikh, M. (2007) Version Control Software in the Open Source Process: A Performative View of Learning and Organizing in the Linux Collectif, Doctoral Thesis in Information Systems, London School of Economics and Political Science.

Shaikh, M. and T. Cornford (2008) "Open-Sourcing: On the Road to the Ultimate Global Source?" in *2nd Global Sourcing, Services, Knowledge and Innovation Workshop* Val d'Isere, France, March 2008,

Shaikh, M. and T. Cornford (2009) "Managerial Strategies for Open-Sourcing Innovation". in *Third Global Sourcing Workshop: The Impacts of Global IS Sourcing on Engineering, Technology and Innovation Management,* Keystone, CO, USA, 22-25 March 2009,

Strauss, A. and J. Corbin (1999) *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory,* Sage Publications,

Välimäki, M. (2003) "Dual Licensing in Open Source Software Industry", *Systemes d´Information et Management,* **8 (1),** p. 63+.

Ven, K. and H. Mannaert (2008) "Challenges and Strategies in the Use of Open Source Software by Independent Software Vendors", *Information and Software Technology,* **50 (9-10),** pp. 991-1002.

Vivek, S. D., R. G. Richey and V. Dalela (2009) "A Longitudinal Examination of Partnership Governance in Offshoring: A Moving Target", *Journal of World Business,* **44 (1),** pp. 16-30.

von Hippel, E. (1994) "Sticky Information' and the Locus of Problem Solving: Implications for Innovation", *Management Science,* **40 (4),** pp. 429–439.

von Hippel, E. (1998) "Economics of Product Development by Users: The Impact of Sticky Local Information", *Management Science,* **44 (5),** pp. 629-644.

von Hippel, E. (2005a) *Democratizing Innovation,* MIT Press, Cambridge, MA.

von Hippel, E. (2005b) "Open Source Software Projects As "User Innovation Networks" in *Perspectives on Free and Open Source Software*, (Feller, J., B. Fitzgerald, S. Hissam and K. R. Lakhani eds) MIT Press, Cambridge, MA, pp. 267-278.

Walsham, G. (2002) "Cross-Cultural Software Production and Use: A Structurational Analysis", *MIS Quarterly,* **26 (4),** pp. 359-380.

Warren, E., A. Tyagi, *et al*. (2009) "Breakthrough Ideas for 2009", *Harvard Business Review,* **87 (2),** pp. 19-+.

West, J. and S. Gallagher (2006) "Challenges of Open Innovation: The Paradox of Firm Investment in Open Source Software", *R&D Management,* **36 (3),** pp. 315-328.

West, J. and C. S. O'Mahony (2005) "Contrasting Community Building in Sponsored and Community Founded Open Source Projects". in *Proceedings of the 38th Annual Hawai'i International Conference on System Sciences, Waikoloa, Hawaii,* Hawaii, January 3-6, 2005, IEEE.

Wheeler, D. A. (2003) *Secure Programming for Linux and Unix Howto,* (v3.010, 3 March 2003)

Winkler, J. K., J. Dibbern and A. Heinzl (2008) "The Impact of Cultural Differences in Offshore Outsourcing - Case Study Results from German-Indian Application Development Projects", *Information Systems Frontiers,* **10 (2),** pp. 243-258.

Witten, B., C. Landwehr and M. Caloyannides (2001) "Does Open Source Improve System Security?" *IEEE Software,* **September/October** pp. 57-61.