OPAALS PROJECT

Contract n° 034824

Workpackage 8
Open Source and Open Knowledge

Deliverable 8.1:

Papers on basic characterisation of the OSS 2.0 phenomenon

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

Contract Number: 034824
Project Acronym: OPAALS
Title: Open Philosophies for Associative Autopoietic Digital Ecosystems

Deliverable Number: Papers on the basic characterisation of the OSS 2.0 phenomenon – D8.1
Due Date: 31/11/2006
Delivery Date: 31/1/2007

Short Description:
The aim of this document is to provide papers on the basic characterisation of the OSS 2.0 phenomenon and will address how this relates to the OPAALS network of excellence. The document examines the OSS 2.0 phenomenon and the relevant surrounding areas within the scope of OPAALS.
Partners Owning: University of Limerick
Partners Contributing: University of Limerick

Versioning

Version	Date	Author, Organisation
1.0	12/9/06	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
1.1	1/11/06	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
1.2	22/2/07	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
1.3	21/3/07	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, University of Limerick
Final	30/4/07	Brian Fitzgerald, Gary Gaughan, Pär J. Ågerfalk, Maha.I.Shaikh University of Limerick

Quality check
1st Reviewer : Evangelia Berdou, LSE

Dependences:

Work Packages	This deliverable affects a number of work packages namely: WP7 Community networks and digital ecosystems WP10 Sustainable community building Tasks: T7.1- Regional development policy T7.4- Population of community network with services and users T7.5- Digital ecosystems & rural community networks Task 10.3 OPAALS Community enlargement (LSE) This work package relates to the above tasks, in that, this deliverable deals with a major change in open source software; the trend towards the commercialisation of open source software. One of the reasons behind this move by commercial companies is to attempt to leverage the potential of community which is unique in Open Source Software communities. For this reason we see the potential this deliverable could have on the above work packages and tasks.
Partners	This deliverable may be of interest to the following partners: London School of Economics (LSE) Universität Kassel (UNIK) CREATE-NET (CN) Indian Institute of Technology Kanpur (IITK) Universty of Surrey (UNIS)
Domains	The domains of, Open Source Software, Global Software Development and Open Source Service Networks are extensively discussed in this deliverable.
Targets	Targets of this deliverable include: Domain researchers OPAALS community

Table of Contents

1 Executive Summary

The aim of this document is to provide papers on a basic characterisation of the OSS2.0 phenomenon. The document examines not only the OSS2.0 phenomenon exclusively but also examines surrounding areas that provide a background to OSS2.0 The document is divided into three main areas, these areas are namely the Open Source Software 2.0 Phenomenon, Open Source and its role in Global Software Development and Open Source Service Networks (OSSN). These areas were chosen both for the background relevance to OSS2.0 and for the pertinence to the OPAALS research community. Four papers are included in the appendix section, and tackle the issues discussed in more depth.

In summary, the outcome of the study is that while many benefits and problems have been raised in relation to Open Source Software methods, not all issues and benefit are necessarily relevant to the OPAALS domain. We have however highlighted the emerging trends in the Open Source 2.0 phenomenon and surrounding areas and awareness of this may be of great benefit to the work of researchers contributing to the OPAALS community.

2 Introduction

The aim of this document is to provide papers on the basic characterisation of the OSS 2.0 phenomenon. The document examines the OSS 2.0 phenomenon and the relevant surrounding areas within the scope of OPAALS. This document will introduce the OSS 2.0 concept here in this section and examine the concept in more detail in the next section entitled "Studies on the OSS 2.0 Phenomenon". We also discuss the concept of Open Source Service Networks (OSSN) and the role these networks might play. The main papers discussed are included in the appendix section and are entitled:

- The Transformation of Open Source Software.
- Supporting Global Software development in open Source Ecosystems: A Role for Actability in the Pragmatic Web.
- Open-Sourcing in the Celtix Project: A Case of Outsourcing to an Unknown Workforce?
- Towards A Model Of Governance in Open Source Service Networks: An Exploration of social Mechanisms.

The papers above have been carefully chosen to illustrate the emerging trend of OSS moving beyond its traditional domain and into a new commercially focused age. For example, the governance structure emerging in Open Source Service Networks seems to be quite applicable to Digital Ecosystems. Also, the models of recruitment and subcontracting identified in the "Open Sourcing in the Celtix Project" are certainly applicable to Digital Ecosystems in general, and will also inform specific policies within OPAALSA in relation to software license choice and community sustainability.

The first paper *"The transformation of Open Source Software"* identifies the OSS 2.0 trend as a distinct entity from the traditional free and open source software movement. In particular the paper examines the movement of open source software practice from this traditional standing to a more commercially viable one. Traditionally, proponents of open source software suggest that it represents a paradigm shift which can solve the 'software crisis' (i.e. systems taking too long to develop, costing too much, and not working very well when eventually delivered). They point to the rapid release schedule, the fact that products are usually available without charge, and argue for the high quality and reliability of open source software. Most paradoxically, this 'silver bullet' is apparently provided for free by a global community of supremely talented volunteers. However, this characterisation is somewhat of a myth as we illustrate this first paper. We consider the transformation of open source software from its Free Software origins to a more mainstream commercially viable form—OSS 2.0, as we term it[1]. We identify the key differences between this emergent OSS 2.0 model and the initial Free and Open Source

[1] As we are interested in the evolution and transformation of the open source phenomenon, we will use the term FOSS to refer to the initial era of Free and Open Source Software Where we wish to specifically refer to the phenomenon which we see now emerging, we use the term OSS 2.0, and we use the term 'open source software' to refer to the phenomenon in general.

Software (FOSS) concept in terms of process and product characteristics. Also, we contend that the OSS 2.0 development process will become *less* bazaar-like as more rigorous project management elements will be enacted to produce a more professional product. In contrast, the OSS 2.0 product life-cycle will become *more* bazaar-like as loose federations of companies converge to create a customised and professional 'whole product' service. For OSS 2.0, 'value' will be the key term, with two of its connotations especially significant – value as in 'value for money' and value as in 'acceptable community values'. Striking an appropriate balance between the two will be key to achieving success in OSS 2.0. This paper is an excellent introduction and provides a solid foundation for the following papers.

The second paper *"Supporting Global Software development in Open Source Ecosystems: A Role for Actability in the Pragmatic Web*" examines new forms of collaboration (based on open source methodologies) between organizations. The study also examines the emerging concept of open source ecosystems from the perspective of actability and the Pragmatic Web. Actability can be defined as: 'an information system's ability to perform actions, and to permit, promote and facilitate the performance of actions by users, both through the system and based on information from the system, in some work context.' As a concept, actability thus emphasises that information technology and information systems are used by people to communicate and perform communicative action. Actability is arguable central to the idea of the pragmatic web, a concept that embraces developments in Internet technology for collaboration, including semantic technologies and "Web 2.0". In particular we identify new forms of collaboration between organizations based on open source principles that are rapidly emerging. The collaboration is typically done in a spirit of co-opetition whereby companies, often SMEs, share cost and risk by developing software jointly and openly. The paper elaborates how this emerging phenomenon of open source ecosystems can be understood from the perspective of actability and the Pragmatic Web. The concept of open source ecosystems as a form of global software development is explored and actability is presented as a useful concept for articulating design criteria for the required collaborative tools. In doing so, a possible research agenda for pragmatic web research in this domain is outlined.

The third paper *"Open-Sourcing in the Celtix Project: A Case of Outsourcing to an Unknown Workforce?"* presents the use of open source as an offshore outsourcing strategy. The paper presents a case study involving a commercial organization, the Irish-based global middleware company IONA, as the customer and the open source community and the supplier of services to IONA. The paper exposes a shift from OSS as community of individual developers to OSS as community of commercial organizations, primarily small to medium-sized enterprises. We present a psychological contract perspective on the use of open source as an offshore outsourcing strategy – open-sourcing as we term it here. Building on previous research on IS outsourcing, a theoretical framework for understanding commercial software organizations involvement in open source software (OSS) is derived. The framework is used in a qualitative case study involving IONA. The study reveals that outsourcing to the OSS community provides

ample opportunity for companies to headhunt top developers – hence moving from outsourcing to a largely unknown OSS workforce towards recruitment of talented developers from the open source community. In a similar fashion, the process allows the development community to get to know the customer organization better. Overall, the key watchwords for open-sourcing are partnership, building of mutual trust, flexibility, tact and complementariness: The customer and community need to establish a trusted partnership of shared responsibility in building an overall ecosystem to deliver the product. The customer has to be flexible in accepting consensus on the development priorities and the functionality that will be built in. The community must be flexible in affording more transparency into the development process. Also, the complementary skills offered by each stakeholder are key to successfully nurturing the ecosystem.

The fourth paper is a research-in-progress which investigates the governance of networks of Open Source Software (OSS) firms, which we are calling Open Source Service Networks (OSSN). These socio-digital business communities are an under-researched, but growing, phenomenon as small OSS firms collaborate to compete with larger firms. The paper illustrates that social mechanisms are a viable alternative to formal/legal inter-firm agreements under the exchange conditions that exist in OSSN. It develops a preliminary governance model and hypotheses from a qualitative analysis of one OSSN, and concludes with a plan for refining the model using a quantitative analysis of two additional OSSNs.

3 Studies on the Open Source Software 2.0 Phenomenon

The open source software (OSS) phenomenon has been around for quite some time now. However, the image of OSS is undergoing a transformation. What we now term OSS 2.0 has emerged as something quite different from its Free Software antecedent, not just ideologically, but also practically. Free Software was originally perceived as largely an ideological phenomenon within the domain of software engineering. However, the open source movement pragmatically shifted the centre of gravity towards a more business-friendly and hybrid concept, and OSS is now rapidly changing into a viable alternative to proprietary software also in commercial settings The OSS phenomenon is, without doubt, innovative in a number of respects. It has, for example, appeared on the software engineering scene as an apparently revolutionary paradigm shift which addresses the three central problems of software development: projects exceeding budget and schedule and software products being of poor quality. In tackling these issues, often referred to as ‘the software crisis’, the OSS development model is somewhat at odds with conventional software engineering wisdom. For example, it often does not map to current software development lifecycle models, it encourages redundant tasks and seems to turn ‘Brooks Law’ on its head.

However, since OSS products are freely available for public download and OSS processes encourage extensive collaboration, there is an obvious element of shared cost and risk. The cost issue is thus immediately addressed. From the point of view of

development speed, the collaborative, parallel efforts of globally-distributed co-developers have allowed many OSS products to be developed much quicker than conventional software. In terms of quality, many OSS products are recognised for their high standards of reliability, efficiency and robustness, and the OSS phenomenon has produced several 'category killers' (i.e. products that remove any incentive to develop any competing products) in their respective areas, such as the Apache HTTP server and the Sendmail mail routing application. The OSS model harnesses the most scarce resource of all – talented software developers. The resulting peer review model, comprising extremely talented individuals, serves to ensure the quality of the software produced. There is also growing evidence to support the claim that OSS can help to address, or even alleviate, the software crisis. For example, Forrester Research have reported that 56% of the world's 2500 largest organisations use one or more OSS products. These are organisations with considerable internal resources who choose OSS products based on technical quality and the radically different nature of software ownership created by open source terms of distribution. From a broader business perspective, several innovative business models and new business opportunities have emerged as a result of the OSS phenomenon, and many organisations have begun to capitalise on this. In terms of competitiveness, the OSS phenomenon has created a new service market for commercial enterprises to exploit and there are several examples whereby these companies have innovatively forged competitive advantage. Since purchase price and license fees are not a factor, OSS companies have to compete predominantly in terms of customer service. Since OSS counters the trend towards proprietary monopolies, the OSS model inherently promotes competitiveness and an open market. Also, by having access to source code, traditional barriers to entry which militate against new entrants are lowered. Although much of the recent OSS debate has focused primarily on desktop applications (Open Office, Mozilla Firefox, etc.), the origins and strengths of OSS have been in the platform-enabling tools and infrastructure components that underpin the Internet and Web services; software like GNU/Linux, Apache, Bind, etc. This suggests that OSS may have a particularly important role to play in the secondary software sector; i.e. in domains where software is used as a component in other products, such as embedded software in the automotive sector, consumer electronics, mobile systems, telecommunications, and utilities (electricity, gas, oil, etc.). With a focus on the secondary software sector, different vertical issues, such as embedded software and safety critical applications, are brought to the fore. The differences in how horizontal issues play out across different vertical sectors can be dramatic. For example, the nuances of the software development context in the banking sector are very different from those which apply in the consumer electronics or telecommunications sectors. A vibrant European secondary software sector provides fertile research ground for studying the potential benefits of OSS from a commercial perspective. With this backdrop we have set out to investigate the role of OSS for the European secondary software sector. We are conducting a thorough investigation of the role of OSS for European industry the output of which will be an industry-research roadmap. This will enable coordination of industry and academic research interests in this area.

The emerging commercial incarnation of OSS (i.e. OSS 2.0) – as adopted by Open Source Software, Inc. – naturally retains many of the characteristics of the 'traditional'

OSS model. However, the commercial setting introduces a number of new complications. For example, the community-based development model may not be completely compatible with a *corporate environment. Business strategy* becomes a central concern, and the very *acceptance* of OSS as a serious contester becomes a critical issue. Fig 1 shows these high-level factors that appear central to the success of Open Source Software, Inc., together with the required *social*, *technical* and *legal infrastructural foundation* on which to build it. In the remainder of this section, we will briefly discuss each of these factors and look at how the Open Source Software, Inc. stakeholders choose to characterise them during the workshop, as reflected in Section III.

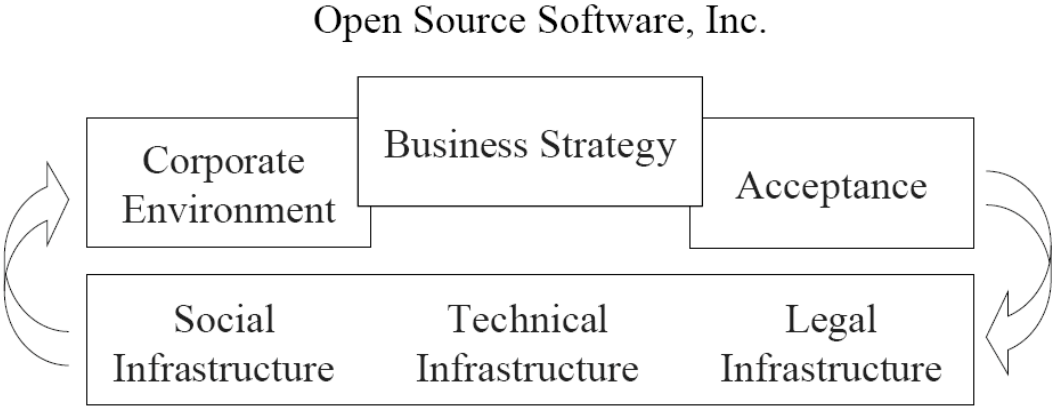

Fig. 1. Major factors in the successful construction of Open Source Software, Inc.

A. Technical Infrastructure

Since it constitutes the software technology as such, the technical infrastructure is a natural key foundation for OSS success. It is believed that the OSS model leads to the creation and adoption of open standards and increased software reuse, which are important factors in adoption of OSS. Other factors that were seen as OSS success-drivers are the perceived proven quality and increased control over the technical infrastructure. However, there are issues related mainly to reliability that needs to be addressed. Especially since the secondary software sector brings OSS to previously unproven domains, such as safety-critical systems. There were also concerns raised about a perceived lack of support and, generally, about the issue of ensuring the longevity of OSS products. Proprietary software companies have tried to undermine OSS by claiming that long-term support of OSS products is uncertain due to the voluntary and self-selected nature of the community. Obviously, Open Source Software, Inc. need to prove them wrong. Perhaps due to the general interest in and success of the OSS model, there are a multitude of projects and products to select from. This raises the question of how best to find a technical solution that is right for the current problem or project.

B. Social Infrastructure

The emphasis on social infrastructure is one of the main tenets of OSS. Leveraging the community-based model that underpins OSS development is thus key to a successful Open Source Software, Inc. Positive aspects of this is the potential of increased collaboration, and specifically joint R&D efforts. This is particularly important from an SME perspective since it may enable smaller companies, with limited resources to

participate in cutting-edge research. Increased collaboration is also likely to lead to knowledge sharing and, in the end, shared costs regarding software development. However, a basic foundation for such collaboration and sharing to happen is that companies find ways of building and sustaining OSS communities, and learn how to build trust within those communities. The relationship between commercial organisations and already established OSS communities will be a challenge for Open Source Software Inc. A model based on sharing obviously assumes that as a user of the community's results, you are also supposed to give something back. How and what to give back to 'the community' is often unclear. A further concern regarding the social infrastructure has to do with whether or not it is at all meaningful to discuss Open Source Software, Inc. as a European venture. In many ways, OSS is an international phenomenon, and it seems to be important not to overlook possible global collaboration, especially with Asia and USA.

C. Legal Infrastructure

One of the most contentious legal banana-skin for Open Source Software, Inc. relates to software patents. Patents can make a good deal of sense in theory – as initiatives which stimulate publication of non-trivial ideas. Clearly, they can play a role in improving innovation and creativity, giving small players a chance, and consequently benefiting society as a whole. However, one can certainly disagree with their implementation in practice (even if it is difficult to separate theory and practice). Interestingly, some of the largest commercial OSS players are also supporters of patents. For example, IBM, HP and Sun are high-profile examples of companies who support software patenting but who have also provided significant benefit and support to OSS. The basic argument from these companies is that patents are necessary and important, and the real problem is that too many spurious patents have been granted which have actually served to stifle innovation. The guarantee of exclusively-held monopoly rights over a long period of time is clearly not suited to the spirit of the current IT era. Interestingly, it is not too much of a stretch to define OSS as publication of non-trivial ideas, with an explicit guarantee of continued availability, which seeks to protect the interest of the small players, for the overall betterment of society, as others learn from and improve on the original ideas. This shows how close OSS and intellectual property protection are at heart. However, the stimulation of innovation and creativity, which should be the fundamental rationale behind intellectual property protection, has failed miserably in the software area [8]. Hence, a solid legal infrastructure is undoubtedly an absolute key to the success of Open Source Software, Inc. On the positive side, there seems to be a strong motivation to solve the legal, licensing, and intellectual property rights problems. Emerging open standards are important in this respect and the general feeling is that legislation is eminent. There are still uncertainties about copyright legislation and EU policy regarding software patents. Another problem is the 'ownership mindset' that prevails in many companies. People are simply used to think in terms of owning tangible assets, and there is a required conceptual move before sharing can become as strong a metaphor. Another potential problem is the lack of understanding of the OSS legal infrastructure within many organisations.

D. Corporate Environment

Leveraging the potential of OSS within a commercial setting means that it will have to adapt to and exist within a corporate environment, which could be quite different from the 'traditional' hacking culture often associated with the OSS movement. However, it is

envisaged that this cultural cross-breed will lead to increased cooperation at the personal level, increased job satisfaction and more flexible forms of employment. The latter also relates to companies believing that the OSS model will help them recruit the best developers available. Then again, there are a few corporate environment factors that seem to mitigate against the successful construction of Open Source Software, Inc. First of all, there is currently a lack of OSS expertise. Such expertise would thus be required in order to enable the recruitment of the best OSS developers. There is also a perceived resistance to change in organisations, where often a few enthusiasts advocate OSS and feel that they are not listen to by management. The OSS movement is often portrayed as an extreme collectivist approach. For example, Bob Young, the founder of Red Hat, adapted the communist manifesto to characterise it as 'from the programmers according to their skills, to the users according to their needs' [9]. Although this purported collectivism has been questioned [2, 10] it still permeates the mainstream OSS discourse and indeed complicates the integration of the OSS model and the commercial environment. This is also related to the meritocratic structure of OSS communities, which may not go well with traditional power and promotion hierarchies in commercial organisations. This makes one wonder where the next OSS leaders will come from? To date, a strong charismatic leader has often proved to be key to the success of OSS (such as Linus Torvalds in the case of Linux).

E. Business Strategy

For apparent reasons, the business strategy of Open Source Software, Inc. will be quite different from organisations built around a proprietary software model. Besides the potential benefits mentioned above, a basic incentive for adopting a OSS approach is to cut costs. The social infrastructure and the sharing mindset of OSS helps to lower barriers (not only in terms of R&D participation) and to reduce the risk involved in product development. Increased software reuse is also likely to lead to shorter time to market. However, to some, the actual value of going open source is questionable, and there is an element of risk associated with a blind faith in OSS as finally providing us with the long awaited silver bullet. There is also a perceived lack of coordination of Open Source Software, Inc. and a need for concrete and proven business models. Again, the success of Open Source Software, Inc. depends on the ability to create sustainable OSS communities, which also affects business strategy. Due to the inherent risks involved in changing the business strategy towards OSS, there may be a need for maintaining a fall-back position, and perhaps adopt OSS incrementally.

F. Acceptance

In order for Open Source Software, Inc. to become reality, it must gain acceptance. Acceptance, in this case, is twofold. First, any company going open source run the risk of being rejected by the existing OSS community. Second, Open Source Software, Inc. must be able to win customers over to the OSS camp and be able to convincingly explain the many benefits and potential shortcomings. Large commercial organisations are not always well perceived within the OSS community. However, OSS offers organisations the ability to appear progressive and 'open', thereby avoiding potential bad press and negative public opinion. However, it is increasingly important that the current enthusiasm

surrounding OSS is harnessed and promoted through well documented success stories. That is if Open Source Software, Inc. is to be widely accepted as a realistic opportunity.

From our analysis we can conclude that the European secondary software sector recognise the benefits of leveraging OSS, but are aware of key issues pertinent to its success. There are numerous pressing issues, such as intellectual property rights and patents. Other such issues include standardisation, the development of proven business models and the acquisition of skills required to successfully engage in OSS projects. These issues must be addressed before OSS can reach widespread adoption. One of the most interesting potential benefits of OSS seems to be the possibility for SMEs with limited budgets to engage seriously in research and development. There are also benefits at both individual and organisational level from a strengthened identity and autonomy. At the same time, OSS facilitates contributing to a shared pool of knowledge and expertise in increased collaboration with both customers and competitors. An integral part of this research was the development of a conceptual framework which reflects the basic factors that this research suggests will facilitate the successful emergence of OSS in a commercial setting (i.e. the success of Open Source Software, Inc.) Fundamentally, the social, technical and legal infrastructure must be in place in order to ensure the success of Open Source Software, Inc. Building on this foundation, companies will need to align their business strategies with a supportive corporate environment and ensure internal and external acceptance of the OSS model.

4 Open Source Software and its Role in Global Software Development

There are many reasons why an organisation should consider adopting distributed development of software systems and applications, including access to a larger labour pool and a broader skills base, cost advantages, and round the clock working. However, distributed development presents many challenges stemming from the complexity of maintaining good communication, coordination and control when teams are dispersed in time (e.g. across time zones) and space, as well as socio-culturally. The open source software (OSS) development model is distributed by nature, and many OSS developments are considered success stories. The question therefore arises of whether traditional distributed development models can be improved by transfer of successful practice from OSS development models. In this paper we compare OSS with traditional distributed development models using a framework-based analysis of the extant literature. From our analysis we find that the advantages of temporal and geographical distance dominate in OSS, rather than their associated problems. Further, socio-cultural distance is lowered by active developer selection. However, there is a challenge to satisfying project goals when personal goals dominate.

Distributed development of software systems and applications (DD) is an issue of increasing significance for organisations today, all the more so given the current trend towards outsourcing and globalisation. However, as well as involvement in conventional DD many companies are getting involved in open source software (OSS) development projects, which have their own development models of DD.

The core challenges of DD seem to lie in the complexity of maintaining good communication, coordination and control when teams are dispersed in time (e.g. across time zones) and space, as well as socio-culturally. Since the OSS development model is distributed by nature, and many OSS developments are considered success stories, the question arises naturally as to whether lessons can be transferred between OSS and distributed development. However, proven methods for successful DD have not yet been formulated, and there is a need for a better understanding and transfer of lessons in relation to all distributed software development. As put by Crowston et al. (2005): "Understanding the work practices of teams of independent knowledge workers working in a distributed environment is important to improve the effectiveness of distributed teams and of the traditional and non-traditional organizations within which they exist."

In line with this sentiment, proponents of the Tigris.org development platform (www.tigris.org) claim that there is much to learn for traditional DD projects by taking a closer look at OSS projects. As stated on the Tigris.org website, the "software engineering field can learn much from the way that successful open source projects gather requirements, make design decisions, achieve quality, and support users."

To facilitate such learning, this paper compares OSS development models with traditional distributed development models by means of a framework-based analysis of the extant literature on case studies in OSS development. From our analysis we derive the underlying characterisations of OSS development models in the literature. These characterisations can help to inform anyone involved in any kind of DD. As the analysis is based on a previously established framework for DD, our analysis will allow broad comparisons to be made.

It is a widespread view that anyone who wants can contribute to an OSS project, and several well-known OSS projects have large numbers of developers. However, it has also been observed that many projects have very few contributing developers. In fact, it has even been claimed that "most OSS products are developed and maintained by a tiny number of developers" (Krishnamurthy, 2002), and that the "community model is a poor fit for the actual production of the software." (Krishnamurthy, 2002) Irrespective of the size of an OSS project, the temporal, geographical and socio-cultural separation of developers has direct consequences for communication, coordination and control in a project.

Communication in OSS projects is largely asynchronous, with developers favouring the use of email (Sarma, 2005). According to Erenkrantz and Taylor (2003), this is because it allows participants to contribute on an equal basis in discussions, and such forms of communication lend themselves well to long-term archiving. However, there have been concerns raised in, for example, the Apache project that off-list discussions take place which are not always relayed back to the larger community (Ågerfalk, 2005). Similarly, Erenkrantz and Taylor (2003) note that the use of synchronous methods amongst distributed developers implies that "some participants may not be able to contribute to a discussion." (p. 23) Although face-to-face meetings in OSS projects are rare, there are some exceptions. For example, the GNOME OSS project has a yearly conference to get developers together (German, 2003). Similarly, the Zope community (www.zope.org) and the PyPy project (pypy.org) are known for their "sprints" where developers come together to work intensively on specific problems.

In OSS development projects developers are typically also users: a good argument for benefiting from DD. For example, in a case study analysis of the FreeBSD OSS project, it was noted that the "developers of FreeBSD were clearly users" (Dinh-Trong and Bieman, 2004). Some go further, for example claiming that "all open source developers are users, but not all users are developers" (Gacek and Arief, 2004, p. 35, 36), and even that it is a "necessity that developers must be users" (Mockus and Herbsleb, 2002, p. 3).

Gurbani et al. (2005) advance a slightly different argument, stressing the benefits of having "a significant pool of users who were interested and capable developers" which seems to be "a precondition for a successful open source project." (p. 28) The reason why developers must be users relates to the role of requirements in an OSS project, and the developers' assumed domain expertise "since there is generally no requirements gathering, and the developers are assumed to be domain experts." (Mockus and Herbsleb, 2002, p. 1) In general, communications within an OSS project reflect the notion that developers are users, something which may have consequences for what can be done in this development model. As noted by Mockus and Herbsleb (2002), "only what this group of user-developers wants will actually get built." (p. 1) Of course, this is changing as paid employees play an increasingly central role in many OSS projects. It is also the case that many companies adopt OSS only as an embedded component in systems which are not used exclusively by developers (Ågerfalk et al. 2005b). In these cases, the communication processes become more complicated and more formalised.

As a final point on communication, there are many accounts in the literature of the benefits of openness, and in particular releasing code early, utilising the massively parallel peer-review process which ensues (e.g. Cubranic, 1999, p. 5). One advantage, noted by Lussier (2004), relates to the Wine project for implementing a Windows API on Unix. Through the mailing list, developers "quickly learned what their common errors were, and how to avoid them." (p. 70)

Coordination needs in OSS projects are minimised. Modularisation is generally seen as an important strategy for minimizing dependencies in software engineering, and even more so in DD. This has been strongly recognised by OSS researchers, "creating software with modules that can be worked on independently." (Crowston et al., 2005, p. 5) For example, GNOME (German, 2003) is "divided into several dozen modules, ranging from libraries (such as GUI, CORBA, XML, etc) to core applications (such as email client, graphical editor, word processor, spreadsheet, etc)" (p. 39), minimising the amount of communication between developers.

Mockus and Herbsleb (2002) raise the concern that "modularity may suffer" (p. 2) if decisions on new features in OSS projects become driven by market demands. Such concerns have been raised in the related area of traditional development tools, where Lundell and Lings (2004) have identified increased market pressures as a possible cause for vendors adopting "feature-driven" development, instead of longer-term visions for their products emphasising reliability and quality.

It has been argued that (free) tools based on standards should be adopted to support coordination in OSS development projects. However, support seems to be mainly for "low-level coordination" (Fielding and Kaiser, 1997, p. 89). Modern tools have not been widely adopted in the OSS community for supporting the software process. For example, it has been claimed that "CVS, now a de-facto standard version control system of medium-to-large open-source projects, is 10 years behind equivalent commercial

offerings" (Cubranic, 1999, p. 4) We note that in a survey of 11 successful OSS projects (Erenkrantz and Taylor, 2003), CVS is used by all OSS projects (Apache HTTP Server, GNOME, GCC, Tomcat, KDE, Linux Kernel, Mozilla, NetBeans, Perl, Python, XFree86), although it is acknowledged that since the publication of the survey, Linux "had adopted BitKeeper as its source control system" (Erenkrantz and Taylor, 2003).

Control in OSS projects might be considered anathema. In DD projects in general, and in OSS projects in particular, it is important to acknowledge the potential tension generated by the need of participants to fulfil their goals. In the context of OSS, only when "private goals are being met will participation continue" (Erenkrantz and Taylor, 2003, p. 24). Essentially, when "a difference of vision occurs between developers and organizations, projects can be forked to maintain the integrity of the private goals." (Erenkrantz and Taylor, 2003, p. 24)

Different authority models have been adopted in the OSS community, and Linux has been recognized as an example of a "central authority organizational model" (Erenkrantz and Taylor, 2003, p. 25) By way of contrast, the GNOME OSS project has a board (GNOME Foundation), which meets regularly, usually over teleconferencing. Minutes are taken from each of these meetings, which are then published in the main GNOME mailing list (German, 2003, p. 42). Gurbani et al. (2005) use the term *benevolent dictator* when describing the final arbiter on what goes into the code while "preserving the architecture" (p. 28) of an OSS product. In the case study reported, which is an OSS project within a company, the benevolent dictator needs to consider the commercial goal of the company also, and not act according to 'blind' idealism for controlling the overall vision and architecture in the OSS project. Several studies have stressed the importance of a strong leader for successful OSS projects, to maintain a unified architectural vision. Leadership in OSS projects is often based on reputation and respect. For example, Linus Thorvalds is often characterised as a strong leader of the Linux project – Moon and Sproull (2000) refer to him as a "great man", and others have commented on his devotion and strong role in spending a "considerable number of hours rewriting code submissions by others." (Krishnamurthy, 2002),

As noted above, although in most OSS projects participants are self-selected, it is becoming more common that developers are paid for their contributions. In fact, participation on a voluntary basis can be a way to obtain a paid job. For example, there are paid employees working in the GNOME project who were initially volunteers. In fact, German (2003) notes that: "Most of the paid developers in GNOME were at some point volunteers. Their commitment to the project got them a job to continue to do what they did before as a hobby." (p. 42) However, their change in status clearly has implications for control, which effectively moves into the company employing them.

In summary, when looking at communication, coordination and control issues in OSS it becomes clear that a number of characteristics emerge. Firstly, the advantages of temporal and geographical distance dominate, rather than their associated problems. Basically, routine development is asynchronous, Internet-based and assisted by having timely feedback from a large user base. There is no concept of having to bring distinct teams together for certain phases of a project, but some projects do utilise "sprint" meetings and developer conferences to increase cohesion and boost progress. Secondly, socio-cultural distance is low by virtue of active developer selection: a developer joins a project through choice, and because of a perceived alignment between a project and the

developer's own aims. Thirdly, personal rather than project goals dominate, so control in an OSS project must be through maintaining a shared vision in which contributors' goals are enhanced through participation.

5 Open Source Service Networks (OSSN)

Open source service networks (OSSN) (Feller et al, Appendix 10.4) is the label pertaining to the governance of networks illustrates how Open Source Software vendors interact with each other. These networks or socio-digital business communities are an under-researched, but growing, phenomenon as small OSS firms collaborate to compete with larger firms. The issues and area of research of OSSN are of relevance to OPAALS not only because of the open source characteristic, but because OSSN are a socio digital business community issue and as such may have a huge resonance factor throughout OPAALS.

The work of Feller et al argues that social mechanisms can, and have been observed to be an effective substitute to engaging in legal contract and formalities. This environment of OSSN much like Service Orientated Architectures (SOA), which involves firms offering a tailored and specific service offering. Just like a group of SOA offerings can be grouped together to be tailored to a customers need so too can OSSN offer this kind of dynamic offering. This leads to expertise in a specific chosen domain by the OSS speciality company and gives rise to OSSN. This environment allows the OSSN to offer a more substantial robust solution.

Open Source Software is of course at the heart of the concept of OSSN. The nature of OSS lowers the barriers to enter to the market place to multiple niche small firms. Through Open Source initiatives small firms can view code in a specific industry and spot gaps in the market place at which to enter and offer products and services. Case in point Google's Android platform providing and open source operating system for developers and companies to offer products and series on top of in order to offer a comprehensive solution where multiple companies or developers come together, each as a small piece of a larger puzzle. There are multiple other examples of this case in point, as Software as a Service (SaaS) has come of age. It is this situation where one company cannot offer a complete solution and looks to find other companies (or developers in the case of Google's Android platform) to offer the aspects of the solution that are missing or which they feel that is outside their domain of expertise or funding.
The networks that emerge in such situations as in the case of the study in Appendix10.4 Feller et al, tend to look a lot like the OSS development communities that have been studied in the past. We now examine the role of social mechanisms in the governance of networks of OSS firms.

The objective of the study in Appendix 10.4 was to explore the role of social mechanisms in the effective governance of Open Source Service Networks.
The study adopted a "soft positivist" epistemology (Kirsch, 2004) and employs both qualitative (phase 1) and quantitative (phase 2) research methods. In phase 1, the study starts with the work of Jones et al in the use of a framework for network governance. Phase 2 sees the testing of the newly developed OSSN governance model by means of a quantitative study within two other OSSN. The study consists of an analysis of 317 peer-reviewed OSS research papers (published 1998-2005). The direction of the research into

OSSN was based on the fact that of the 319 papers examined, OSS had been investigated in the studies in a multitude of theoretical point of view however, the dominant research theme was found to have been socio-cultural analysis and treats OSS communities as socio-digital networks; social networks whose interactions are predominantly mediated through ICT (e.g. Berquist and Ljungberg, 2001; Gallivan, 2004; Sagers, 2004).

The starting theoretical standing for this study is the work of Jones et al on Network Goverence and Jones et al (1997: 913) defines Network governance as:

> "interfirm coordination characterized by organic or informal social systems, in contrast to bureaucratic structures within firms and formal contractual relationships between them."

They identify four pre-conditions for network governance: demand uncertainty, customized (asset-specific) exchanges, complex tasks executed under time-pressure, frequent exchanges between partners. Jones et al states that when high levels of structural embeddedness exist in a social network, mechanisms can be employed in the event of problems occurring in the social network. These social mechanisms are:

1. Restricted Access – the "strategic reduction in the number of exchange partners in the network" (Jones et al 1997: 927)
2. Macroculture – a "system of widely shared assumptions and values … that guide actions and create typical behaviour patterns among independent entities" (*ibid*: 929)
3. Collective Sanctions – ways in which groups punish members who violate shared norms, values and goals (*ibid*: 931)
4. Reputation – "estimations of one's character, skills, reliability and other attributes important to exchanges" (*ibid*: 932)

Feller et al (Appendix 10.4) argue that the social mechanisms listed above by Jones et al are already highly prevalent in OSS literature. They suggest that while the Jones et al framework has been cited in numerous fields, the framework fails to measure network effectiveness.

6 Qualitative Study: The Development of a Conceptual Model

The qualitative study undertaken in the work of Feller et al (Appendix 10.4) was concerned with initially achieving an increased understanding of the phenomenon of OSSN. The study began with the work of Jones et al. (a framework for network governance in general) however Jones et al failed to account for network effectiveness which is an important addition to understanding the nature of the OSSN. Given this initial standing the work proceeded to an in-depth case study of one particular OSSN – the Zope Europe Association (ZEA). Zope is an application server technology and upon which ZEA (an international network of businesses), build various different applications for the offering. Zope is a particularly strong application server technology with a multitude of applications on offer and as such is a seen as a well grounded choice as an OSSN study.

In the study, it was seen to be the case that demand uncertainty was prevalent. The nature of the companies involved in ZEA was such that companies were typically small in size, which limited the contracts they could apply for at any one time. The OSSN of ZEA allows companies to pool resource and apply for bigger contracts.

In the study The ZEA Founder explained that for inter-firm coordination, social mechanisms "are the only thing." In an opt –in system where perceived reputation is a major motivating factor the use of social mechanisms works very effectively. In the analysis of the use of social mechanisms between ZEA members the study found some discrepancies between the work of Jones et al on governance and what is happening in ZEA. In particular the study revealed that the functioning of the mechanisms and how they relate to coordination/adoption has more in common with the cultural values and mechanisms in the OSS communities.

The relationship between social mechanisms and coordination/adaptation and safeguarding exchanges (Figure 2) is the same as extant research, except that the study found macroculture safeguards exchanges rather than just facilitating coordination/adaptation as predicted by Jones et al.

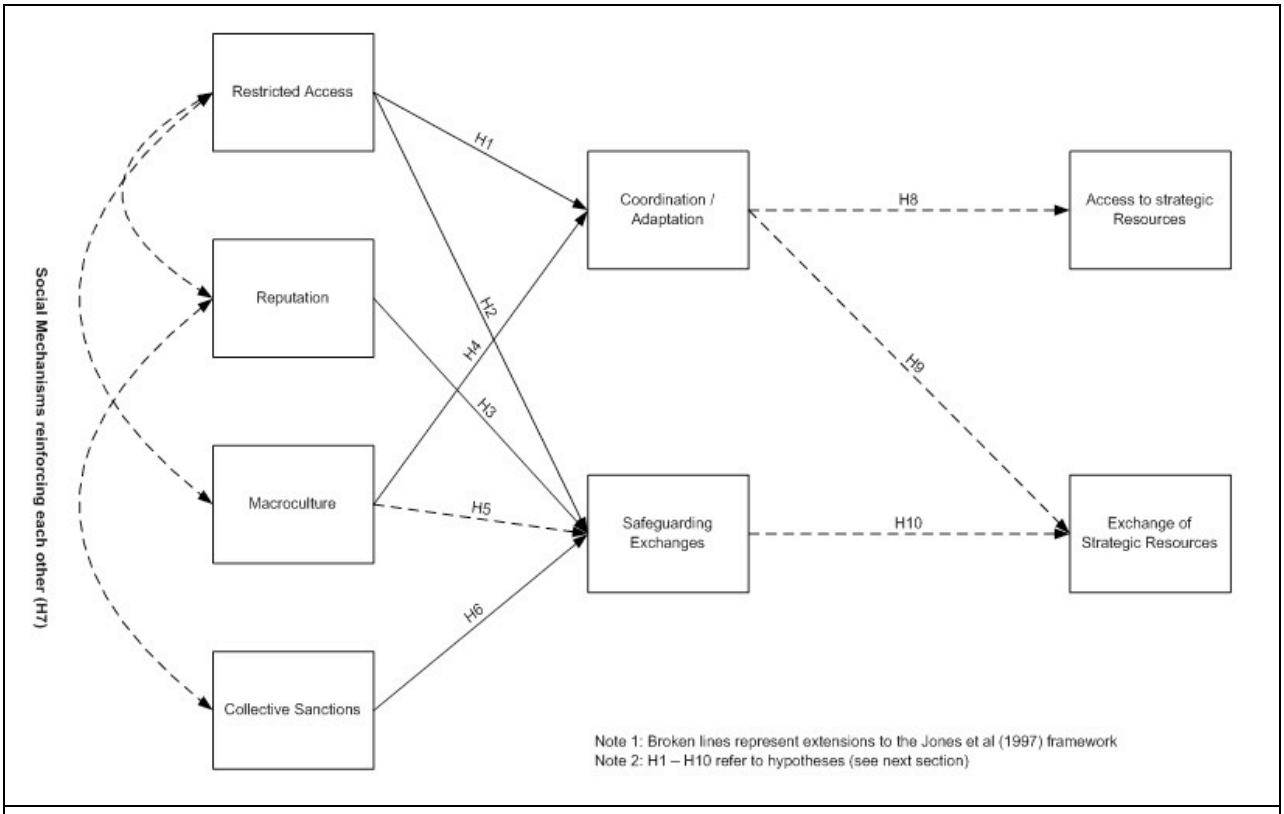

Figure 2: Model of Network Governance in OSSNs Fellar et al (Appendix 10.4)

Participants in the ZEA network consider it to be very beneficial in that it afforded the companies' access to strategic resources. Members consider the ZEA network effective as it allows them to access and share strategic resources. They cited the ability to; access and leverage the skill base of other firms; share customer contacts; compete for larger contracts; share business experiences/expertise (e.g. project management) and enjoy an enhanced reputation as a result of membership.

7 Work-in-Progress

The qualitative study developed a conceptual model of governance in OSSNs building on the Jones et al framework by (a) illustrating the interaction between social mechanisms and (b) expanding the concept of network effectiveness. The conceptual model is represented as a series of hypotheses as shown above in Figure 2. These are:

H1: Restricted access facilitates co-ordination/adaptation by facilitating the establishment of routines for working together (H1a), reducing co-ordination costs through increased visibility (H1b), adopting deliberate membership requirements facilitating a component-based approach to work allocation (H1c), giving members a voice in decision making (H1d), establishing routines for working together (H1e), and reducing variances that parties bring to the exchanges (H1f).

H2: Restricted access safeguards exchanges by reducing transaction costs (H2a), reducing the danger of opportunistic behaviour (H2b), increasing identification amongst partners (H2c), and not diluting membership benefits (H2d).

H3: Reputation within the OSSN safeguards exchanges by acting as a prerequisite for participation (H3a), rewarding good behaviour (H3b), and ensuring that there are economic consequences of a firm's reputation (H3c).

H4: OSSN Macroculture improves co-ordination/adaptation by fostering a culture of network agility (H4a), ensuring new members fit into the culture of the network (H4b), creating a sense of mutual interest (good karma) (H4c), preventing fear of lock-in (H4d), and reducing information / knowledge asymmetries (including those with customers) (H4e).

H5: OSSN Macroculture safeguards exchanges by balancing commercial interests with network objectives (H5a), taking an ethical approach to commercial issues (H5b), and creating and environment of trust (H5c).

H6: Collective sanctions safeguard exchanges because of the threat of damaged reputation within the network (H6a), and the possibility of exclusion from the network resulting in damaged reputation in the OSS community (outside the network) (H6b).

H7: Social mechanisms mutually reinforce each other as follows; restricted access and macroculture (H7a); restricted access and reputation (H7b); reputation and collective sanctions (H7c).

H8: Coordination/adaptation improves network effectiveness by providing access to strategic resources.

H9: Coordination/adaptation improves network effectiveness by facilitating the exchange of strategic resources.
H10: Safeguarding exchanges improves network effectiveness by facilitating the exchange of strategic resources.

The above hypotheses will be explored further by means of a large-scale survey of additional OSSNs, and this preliminary model will tested using regression analysis techniques.

8 Conclusion

In this document we have provided papers that characterise the OSS 2.0 phenomenon and examine the surrounding areas that pertain to emerging trends in the movement of OSS to OSS 2.0. These areas include open source service networks and the role of open source software in global software development. In the first paper discussed we characterised the emergent OSS 2.0 model and differentiated it from its FOSS antecedent in terms of core process and product characteristics. We identify the key driving factor behind FOSS as the individual developer's achievement as part of the collective community of peers. In OSS 2.0 a corporate facet is unveiled and brings the notion of value for money to the traditional set of OSS community values.
Where we examined open source software in terms of distributed development, we have found that the advantages of temporal and geographical distance dominate in OSS, rather than their associated problems. This is achieved by making routine development asynchronous, with timely feedback from a large user base. Further, socio-cultural distance is lowered by active developer selection. Every developer has a personal stake in the success of the project. The low entry cost to projects, both in monetary terms and in terms of learning the technology used for coordination by a community, also makes it possible for contributors to participate in several projects. This could significantly increase flexibility in prioritising development effort between projects.
However, there is a challenge for OSS in relation to satisfying project goals when personal goals dominate. It is unclear to what extent goals can be unified when a project is removed from the infrastructure level, to a level where consensus is more difficult to achieve. How much this factor may dominate in OSS development has yet to be established.
The papers and issues examined and discussed here represent a broad overview of the issues that are present today in Open Source Software and provide a pointer to where we see emerging fundamental change in Open Source. In our work here we have endeavoured to provide a insight to emerging trends in Open Source Software and in doing so provide material to aid in the collective construction of the OPAALS community and in more practical term provide reference material for the decision making process required throughout the OPAALS community and OKS.

9 References

Ågerfalk, P.J. (2005). Studying Persistent Conversations in an Open Source Context: A Conceptual Framework, In Proceedings of Promote IT 2005, (Eds, Bubenko jr., J et al.) Studentlitteratur, Lund.

Bergquist, M. and Ljungberg, J. "The Power of Gifts: Organising Social Relationships in Open Source Communities" *Information Systems Journal* (11:4) December, 2004 pp.305-320.

Carmel, E. and Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. IEEE Software, 18 (2), 22-29.

Cubranic, D. (1999). Open-Source Software Development. In 2nd Workshop on Software Engineering over the Internet, 7p.

De Wever, S., Martens, R., and Vandenbempt, K. "The impact of trust on strategic resource acquisition through interorganizational networks: Towards a conceptual model," *Human Relations* (58:12) December 2005, pp. 1523-1543.

Dinh-Trong, T. and Bieman, J.M. (2004) Open Source Software Development: A Case Study of FreeBSD. In Proceedings of the 10th International Symposium on Software Metrics, IEEE Computer Society.

Eisendardt, K.M. "Building theories from Case study Research", *Academy of Management Review* (14:4) December, 1989 pp.532-550.

Erenkrantz, J.R. and Taylor, R.N. (2003). Supporting Distributed and Decentralized Projects: Drawing Lessons from the Open Source Community. In The 1st Workshop on Open Source in an Industrial Context, p. 21-30, <wwwbib.informatik.tu-muenchen.de/infberichte/2003/TUM-I0319.pdf>

Fielding, R.T and Kaiser, G. (1997). The Apache HTTP server project. IEEE Internet Computing, 1 (4), 88-90.

Gacek, C. and Arief, B. (2004). The Many Meanings of Open Source. IEEE Software, 21 (1), 34-40.

Gallivan, M.J. "Striking a Balance between Trust and Control in a Virtual Organisation: A content Analysis of Open Source Software Case Studies" *Information Systems Journal* (11:4) December, 2004 pp.277-304.

German, D.M. (2003). GNOME, a case of open source global software development. In International Workshop on Global Software Development, p. 39-43, <gsd2003.cs.uvic.ca/gsd2003proceedings.pdf>

Gurbani, V.K., Garvert, A. and Herbsleb, J.D. (2005). A Case Study of Open Source Tools and Practices in a Commercial Setting. In Proceedings of the 5th Workshop on Open Source Software Engineering, p. 24-29, ACM.

Jones, C, Hesterly, W, and Borgatti, S. "A General Theory of Network Governance: Exchange Conditions and Social Mechanisms," *Academy of Management Review* (22:4) December 1997, pp. 911-944.

Kirsch, L.J "Deploying Common Systems Globally: The Dynamics of Control" *Information Systems Research* (15:4), December 2004, pp. 375-395.

Krishnamurthy, S. (2002). Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. First Monday, 7 (6), <www.firstmonday.org/issues/issue7_6/krishnamurthy/>

Lundell, B. and Lings, B. (2004). Changing perceptions of CASE-technology. Journal of Systems and Software, 72 (2), 271-280.

Madill, A., Jordan, A. and Shirley, C. "Objectivity and Reliability in Qualitative Analysis: Realist, Contextualist and Radical Constructionist Epistemologies. British Journal of Psychology (91:1) February 2000, pp.1–20.

Mockus, A. and Herbsleb, J.D. (2002). Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?. In Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering, p. 19-25, <opensource.ucc.ie/icse2002/MockusGerbsleb.pdf>

Moore, G. *Crossing the Chasm,* Harper-Perennial, New York, NY 1999.

Moon, J.Y. and Sproull, L. (2000). Essence of Distributed Work: The Case of the Linux Kernel. First Monday, 5 (11), <www.firstmonday.org/issues/issue5_11/moon/>

Sagers, G "The Influence of Network Governance Factors on Success in Open Source Software Development Projects. *Proceedings of theTwenty-Fifth International Conference on Information Systems.* Agarwal, R and Kirsch, L. (eds.) Washington DC, December 2004, pp.427-438.

Sarma, A. (2005). A Survey of Collaborative Tools in Software Development, ISR Technical Report # UCI-ISR-05-3, Institute for Software Research, University of California, Irvine <www.isr.uci-edu/tech-report.html>

Woods, D., and Guliani, G. *Open Source for the Enterprise*, O'Reilly Media, Sebastopol, CA, 2005.

, R.K. *Case Study Research, Design and Methods*, Sage Publications, Newbury Park, CA, 1994.

10 Appendix

10.1 The Transformation of Open Source Software Brian Fitzgerald

Brian Fitzgerald, University of Limerick, Ireland

MIS Quarterly, Vol. 30, No 3, pp. 587-598.

Abstract

Proponents of open source software suggest that it represents a paradigm shift which can solve the 'software crisis' (i.e. systems taking too long to develop, costing too much, and not working very well when eventually delivered). They point to the rapid release schedule, the fact that products are usually available without charge, and argue for the high quality and reliability of open source software. Most paradoxically, this 'silver bullet' is apparently provided for free by a global community of supremely talented volunteers. However, this characterisation is somewhat of a myth as we illustrate here. We consider the transformation of open source software from its Free Software origins to a more mainstream commercially viable form—OSS 2.0, as we term it[2]. We identify the key differences between this emergent OSS 2.0 model and the initial Free and Open Source Software (FOSS) concept in terms of process and product characteristics. Also, we contend that the OSS 2.0 development process will become *less* bazaar-like as more rigorous project management elements will be enacted to produce a more professional product. In contrast, the OSS 2.0 product life-cycle will become *more* bazaar-like as loose federations of companies converge to create a customised and professional 'whole product' service. For OSS 2.0, 'value' will be the key term, with two of its connotations especially significant – value as in 'value for money' and value as in 'acceptable community values'. Striking an appropriate balance between the two will be key to achieving success in OSS 2.0.

Keywords: Open source software, free software, business models, IS development

Introduction

According to one of the more widely accepted definitions, innovation has been defined as theoretical concept plus technical invention plus commercial exploitation (Trott, 1998). This definition is actually quite useful in summarising the progression from Free Software to the Open Source Software phenomenon: the Free Software concept exhibits the characteristic of *theoretical concept* (e.g. unrestricted access to source code guaranteed through the General Public Licence) and *technical invention* (e.g. the GNU family of products, including the Linux kernel and operating system). However, the innovation was only completed with the *commercial exploitation* phase which really commenced followed the coining in 1998 of the more business-friendly term, open source software—an initiative which continues to gain momentum as an increasingly commercial phenomenon, or OSS 2.0 as we label it here. It is also the case that the real impact of an innovation is often not in the area or manner first anticipated—the phonograph, television, and so on. It is our contention that OSS 2.0 has emerged as something quite different from its Free Software antecedent, not just ideologically, but

[2] As we are interested in the evolution and transformation of the open source phenomenon, we will use the term FOSS to refer to the initial era of Free and Open Source Software Where we wish to specifically refer to the phenomenon which we see now emerging, we use the term OSS 2.0, and we use the term 'open source software' to refer to the phenomenon in general.

also practically. Free Software was originally perceived as largely an ideological phenomenon within the domain of software engineering. However, the open source phenomenon pragmatically moved the centre of gravity towards a more business-friendly and hybrid concept.

Furthermore, we contend that as a research topic, OSS 2.0 extends well beyond the disciplinary boundaries of software engineering. The observation that the average Navajo Indian family in 1950s America consisted of a father, mother, two children and three anthropologists certainly has resonances in that it would have seemed extremely unlikely just a few years ago that what would appear to be primarily a software topic could elicit so much interest from such a diverse range of research disciplines—sociology, economics, management, psychology, public policy and law, for example (Feller & Fitzgerald, 2002).

OSS 2.0 also has many facets which locate it as a central topic in the IS nomological net, to use the terminology proposed by Benbasat and Zmud (2003). There are some similarities with the DSS field – the drawing together of a wide range of researchers from across a range of disparate disciplines, for example. However, the DSS field has had a troubled fate according to one of the original researchers who helped coalesce the DSS area into a coherent research field. Keen (1991, pp.37-38) laments DSS research having been "co-opted and trivialised…by lab-experiment-academics." He concludes that "identity is easily blurred and eroded when the purposive focus of the research is lost and the topic area then dominates". We suggest that a similar situation could recur in the case of open source research as it grows in popularity and researchers take advantage of the ready availability of large online databases of data, and continue to focus their research efforts inwards on the phenomenon – repeatedly studying project characteristics and developer motivation, for example. Such research is valuable, but a more purposive research agenda which also looks outward at the open source phenomenon in general, and the emergent OSS 2.0 phenomenon in particular, is also clearly necessary.

In this piece, we begin by discussing the myth which features in some characterisations of open source software. We analyse the initial FOSS phenomenon using a framework to examine relevant process and product issues. We then analyse the emergent OSS 2.0 in terms of this framework of process and product factors. We also identify key challenges for OSS 2.0 and identify some research initiatives which may help address these challenges.

The Myth of Open Source Software

While there has been some disagreement within the free and open source software community as to the definitions of free software versus open source software, we will not dwell on this here (see www.fsf.org/philosophy/free-software-for-freedom.html). A more significant issue is the difference between free and open source software and concepts such as freeware, shareware and public domain software. Table 1 summarises the differences among these different categories of software.

A frequent characterisation of open source software is that of a collective freely volunteering their services to produce very high quality software by means of a revolutionary new software development approach. However, this is actually a myth as almost everything in this statement is open to question.

Firstly, in terms of a *collective*, the open source movement is often portrayed as a collectivist approach; Bob Young, the founder of Red Hat adapted the Marxist manifesto to characterise it as "from the programmers according to their skills, to the users according to their needs" (Young, 1999). Certainly, the massive parallel development—the Linux development community is variously estimated to exceed 40,000 (Raymond, 1999) or even 750,000 contributors (Torvalds & Diamond, 2001)—seems to support such a collectivist view. Furthermore, the suggestion that all contributions are equally valued reinforces the appearance of collectivism. Rather than just accepting code contributions, the argument is that those who cannot write code can write documentation, fulfil the role of testers, or elaborate requirements.

Table 1 A Taxonomy of Software Categories

License Feature / **Software Type**	**Zero Price**	**Redistrib-utable**	**Unlimited Users and Usage**	**Source Code Available**	**Source Code Modifiable**
Commercial Proprietary Software					
Trial Software (e.g. time-bombed evaluation products)	X	X			
Use-Restricted (e.g. Netscape Navigator)	X	X			
Shareware (e.g. WinZip)	X	X			
Freeware (e.g. Leap Frog)	X	X	X		
Royalty-free binaries (e.g. Microsoft's Internet Explorer and NetMeeting)	X	X	X		
Royalty-free libraries (e.g. class libraries)	X	X	X	X	
Open Source (e.g. Linux, Apache)	X	X	X	X	X

However, there is also very strong evidence to support the view that the open source phenomenon is at heart an individualist one. The open source culture is fundamentally a reputation-based one, and is underpinned by the economics of signalling incentives on the part of individual developers (Lerner & Tirole, 2000). The signalling incentive term is an umbrella one capturing both *ego gratification* and *career concern* incentives. Ego gratification incentives occur where developers experience a significant 'rush' from seeing their code in use more quickly in open source projects, while they also receive recognition from peers they truly respect. The *career concern* incentive relates to the fact that working on an open source project may enhance future job prospects. Linus Torvalds states that his reward for working on Linux is the fact that he will never have any difficulty in getting a job—his CV, as he puts it, contains just one word: Linux. Thus, a significant motivator of participation in open source development is the belief that it will improve a developer's own coding skills and help improve career prospects. Interestingly, reputation may scale far less than first anticipated in that only a small handful of people may actually achieve widespread name recognition. One of the findings of the FLOSS large-scale survey of open source developers worldwide (Ghosh et al., 2002) was that respondents were as likely to report knowing fictional developers (names made up for the study) as much as actual developers when it got beyond the first few well-known names.

Also, the collectivist notion that all contributions are valued, and that literally thousands of globally-located developers and users contribute unproblematically to open source projects does not bear up to examination. In the case of the BSD operating system, McKusick (1999) admits that 90 percent of contributions were thrown away. In a study of the Apache project, Mockus et al. (2000) found that almost 85 percent of modification requests from users were totally ignored. Alan Cox, a central figure in the development of Linux, has admitted that most contributions are worthless, suggesting that they actually support the argument that one should need a license to get on the Internet, and that there are a lot of "dangerously half-clued people milling around", and that those of proven ability are well known within each product development community (Cox, 1998). Such evidence is not indicative of a collectivist atmosphere.

Also, as commercial organizations have played an increasing role, the *volunteer* element more characteristic of the Free Software era is being counterbalanced by paid development – initiatives by Red Hat, IBM, SUN and MySQL being obvious examples. This is further borne out in studies of the phenomenon where estimates have been reported that more than 40 per cent of developers working on open source projects are being paid to do so (Jorgensen, 2001; Lakhani & Wolf, 2005).

Proponents of open source also argue for its *high quality*, suggesting that feedback is very prompt, the testing pool is global, peer review is truly independent, the contributors are among the most talented developers worldwide, and they are self-selected and highly motivated. However, some more rigorous analysis of actual open source code has cast considerable doubt on the universal high quality of open source software (Rusovan, Lawford & Parnas, 2005; Stamelos *et al*, 2001).

Raymond's (1999) attention-catching metaphor of the cathedral v. the bazaar was chosen to reflect the view of open source as a revolutionary new development approach which

runs counter to the fundamental tenets and conventional wisdom of software engineering. He characterised conventional software engineering as akin to a *cathedral* of highly formalized, well-defined and rigorously-followed development processes. He contrasted this with a bazaar style of open source development. The bazaar metaphor was chosen to reflect the babbling, apparent confusion of a middle-Eastern marketplace. In terms of software development, the bazaar style does not mandate any particular development approach—all are free to develop in their own way and to follow their own agenda. There is no formal design process, no risk assessment nor measurable goals, informal co-ordination and control, much duplication and parallel effort. There is no formal procedure to ensure that developers are not duplicating effort by working on the same problem unknown to each other. All of this is anathema to conventional software engineering. For example, duplication of effort would be seen as extremely inefficient and wasteful, but in the open source model, this *optimistic concurrency* leads to a greater exploration of the problem space, and is consistent with an evolutionary principle of mutation and survival of the fittest, in so far as the best solution is likely to be incorporated into the evolving software product.

However, the notion that open source represents a revolutionary software development paradigm is questionable. An examination of the details of the open source development process casts doubt on the claim that software engineering principles are actually being fundamentally overturned. Firstly, the main contributors are acknowledged to be highly-talented coders. Also, as they are self-selected, they are highly motivated to contribute. The remarkable potential of gifted individuals has long been recognised in software engineering (e.g. Brooks, 1987; Paulk et al., 1993). The Chief Programmer Team more than thirty years ago also bore witness to the potential of great programmers.

Furthermore the advancement of knowledge in software engineering has certainly been incorporated into open source projects. Linux, for example, benefited a great deal from the evolution of Unix in that defects were eliminated and requirements fleshed out a great deal over the years. Also, some of the fundamental concepts of software engineering in relation to cohesion, coupling and modularity of code are very much a feature of open source. Modularity is critical for open source development model for a number of reasons. Firstly, it allows work to be partitioned among the global pool of developers. Also, as projects progress, the learning curve to understand the rationale behind requirements and design decisions becomes extremely steep. New contributors need to be able to reduce their learning curve below the level of the overall project. Modularity helps achieve this and is a *sine qua non* for open source. Many open source projects were rewritten to be more modular before they could be successfully developed in an open source mode, including Sendmail and Samba, and even Linux itself. Configuration management, another important research area within software engineering, is a vitally important factor within open source. Also, the software engineering principles of independent peer review and testing are very highly evolved to an extremely advanced level within open source development.

In summary, then, the code in open source products is often very structured and modular in the first place, contributions are carefully vetted and incorporated in a very disciplined fashion in accordance with good configuration management, independent peer review and

testing. Thus, on closer inspection, the so-called 'bazaar' model of open source does not depart wildly from many of the sensible and proven fundamental software engineering principles. The argument that open source projects begin as a bazaar with a chaotic development process and evolve mysteriously into a co-ordinated process with an exceptionally high quality end-product is too simplistic a characterisation of what actually is taking place in practice.

Characterising FOSS

Given that the myth of open source software can be challenged, we seek to characterise the nature of open source software, particularly in relation to its transformation from the initial free/open source software (FOSS) concept to the more commercially-oriented OSS 2.0 phenomenon.

In their work on technological transformation, Tushman and Andersen (1986) propose a framework based on two sets of technological factors, namely process and product. We present a similar framework to characterise the initial FOSS phenomenon. This analysis is summarised in Table 2, and discussed more fully below. Later we extend this framework to analyse the emergent OSS 2.0 phenomenon.

Table 2 Characteristics of FOSS

Process	FOSS
Development Lifecycle	Planning – "an itch worth scratching" Analysis – part of conventional agreed-upon knowledge in software development Design – firmly based on principles of modularity to accomplish separation of concerns Implementation Code Review Pre-commit test Development release Parallel Debugging Production Release (often the Planning, Analysis & Design phases are done by one person/core group who serve as '*a tail-light to follow*' in the bazaar)
Product	
Domain	Horizontal infrastructure (operating systems, utilities, compilers, DBMS, web & print servers)
Primary Business Strategies	Value-added service-enabling Loss-leader/market-creating
Product Support	Quite haphazard – much customer reliance on email lists/bulletin boards, or on support provided by specialized software firms
Licensing	GPL, LGPL, Artistic License, BSD and emergence of commercially-oriented MPL

Key Tension	Achieving balance between collectivist v. individualist

FOSS Development Process

In conventional software development, the development lifecycle in its most generic form consists of four broad phases: planning, analysis, design and implementation. However, in FOSS development, these stages are configured differently to the conventional model. The first three stages of planning, analysis and design are concatenated and are typically performed by a single developer or small core group, while the implementation phase is elaborated considerably.

The planning phase is probably best summarised by Raymond's (1999) phrase of a single developer perceiving " an itch worth scratching". This leads to the building of the initial prototype. Given the ideal that a large number of globally-distributed developers of different levels of ability and domain expertise should be able to contribute subsequently, requirements analysis is largely superceded, as requirements are taken as generally understood and not requiring interaction among developers and end-users. In fact, FOSS developers are invariably users of the software being developed. This model is perhaps best suited to infrastructure software in horizontal domains (as will be discussed later). Design decisions are also generally made in advance before the larger pool of developers start to contribute. Systems are highly modularised to allow distribution of work, and also to reduce the learning curve for new developers to participate, as they can focus on particular sub-systems without needing to consider the system in its totality.

In the FOSS development lifecycle, the implementation phase consists of several sub-phases (Feller & Fitzgerald, 2002), very briefly summarised here:

- Code – writing code and submitting to the FOSS community for review
- Review – one of the strengths of FOSS is the truly independent and prompt peer review
- Pre-commit test – the negative implications of breaking the build ensure that contributions are carefully tested before being committed
- Development release – code contributions may be included in the development release within a short time of having been submitted – this rapid implementation being a significant motivator for developers
- Parallel debugging – the so-called Linus's Law ('given enough eyeballs every bug is shallow') comes into play as debugging can scale and the large number of potential debuggers on different platforms and system configuration ensures bugs are quickly found and fixed

- Production release – a relatively-stable debugged production version of the system is released

The management of this development process varies a great deal, as different projects have varying degrees of formalism as to how decisions are made, but the principle of "having a tail-light to follow" (Bezroukov, 1999) captures the spirit well. Often, however, the initial project founder or a small core group make the key decisions in accordance with the process outlined in the lifecycle above.

FOSS Product

Here we will discuss issues to do with the FOSS product domain, the business strategies underpinning FOSS, the nature of FOSS product support, and FOSS licensing.

FOSS Product Domains

As discussed above, due to the nature of the globally-distributed development community, most of whom never meet face-to-face, FOSS products have tended to be infrastructural systems in horizontal domains where the requirements are part of the general taken-for-granted wisdom of the software development community. Thus, the most successful FOSS products – the Linux operating system, the Apache web server, the Mozilla browser, the GNU C compiler, the Perl scripting language and MySQL database management system – are all examples of horizontal infrastructure-type software. Since these systems did not require major budgetary approval, they were typically back-office and the take-up has been extremely significant. Some statistics serve to illustrate this penetration very well: the Apache web server runs 70 percent of the world's web sites, OpenOffice has had more than 16 million web downloads (not to mention CD-ROM installations), the Mozilla Firefox web browser has had almost 11 million downloads, and MySQL has over 5 million active installations world-wide, with estimates of 35,000 downloads per day.

Primary FOSS Business Strategies

While the 'open source' term was deliberately chosen to move away from the anti-business perceptions associated with free software, initial FOSS business strategies were understandably emergent and speculative. Several FOSS business models have been proposed (Hecker, 2000; Raymond, 1999), but in effect, two categories of business strategy have been most signifcant for FOSS, namely value-added service-enabling and loss-leader/market-creating.

An early example of the value-added service-enabling model was that of Cygnus Solutions who integrated a suite of GNU tools and sold support services and other complementary software products. In FOSS terms, Red Hat are probably the most well-known proponent of this strategy, and it has also been used by Caldera. Effectively, these vendors are simplifying the task facing end-users in deploying an overall open source

solution such as Linux which requires the complex configuration of different components.

In the loss-leader/market-creating model, the open source product is distributed free, but with the end goal of enlarging the market for alternative closed source products and services. The Netscape Mozilla project is a prominent example of such a strategy. Another example is the open source Sendmail product which enlarges the subsequent market for the Sendmail Pro, a product with extra functionality which is distributed for a fee. An added bonus is the fact that voluntary contributions to the open source product can be included in the closed source commercial versions.

FOSS Product Support

The nature of product support in FOSS is quite haphazard and bazaar-like, and is very different to the proprietary model. Bulletin boards, mailing lists and the like are the forums where support requests are often routed and solutions sought. In some cases, support may be purchased from a competent third party provider. For example, Linux support is available from HP or IBM, or a specialized (often local) software firm may offer support and consultancy services. While many organisations may face internal resistance to the fact that their support essentially derives from a series of bulletin boards, they may be equally reluctant to purchase consultancy support to effectively deploy a solution. In a case study of large-scale open source implementation at Beaumont Hospital (Fitzgerald & Kenny, 2003), Beaumont's IT Manager captured the issue well:

> "If you have a product which costs €1 million – it may seem appropriate to spend €500K on support. However if the product costs nothing – then spending €500K somehow seems to be a more difficult decision to take – yet the saving is still €1 million".

FOSS Licensing

Ironically, given the collectivist anti- intellectual property perception associated with FOSS, the open source phenomenon is actually quite strong on property rights—they are vested in the author through copyright, with very liberal rights granted to others under license. In fact, the success of the open source model is largely due to the use of licensing, albeit in a form counter to the normal restrictive sense. In the FOSS era, the principal licenses have been the GNU Public License (GPL), the Lesser GPL (LGPL), the Artistic License, the Berkeley System Distribution (BSD), and this era also saw the emergence of the commercially-oriented Mozilla Public License (MPL) which has been quite influential. Each of these is discussed briefly here.

The GPL is the earliest open source license and was created in the mid 1980s to distribute the GNU project software. The majority of open source software to date has been distributed under the GPL, Linux being one very high profile example. The GPL subverts the traditional concept of restricted access through copyright by ensuring complete and unrestricted access to all open source software and any derivatives, which must also be licensed under the same terms, referred to as 'copyleft - all rights reversed'. This latter

guarantee of the same rights to subsequent users causes such licenses to be termed 'reciprocal'.

The GPL is a controversial license, requiring that all software distributed with GPL software be released under a GPL license also. This proved impractical and a modified version, the Lesser GPL (LGPL) has been created. The LGPL differs from the GPL in two main ways. Firstly, it is intended for use with software libraries (in fact it was initially known as the Library GPL), and secondly, the software may be linked with proprietary code, which is precluded by the GPL. Another early license to achieve fairly widespread use has been the BSD license (a short license containing only 225 words in comparison to the 2,968-word GPL and 4,380-word LGPL). The BSD license imposes very few restrictions, its main requirement being the retention and acknowledgement of the work of previous contributors. Another early license is the Artistic License which dates from 1987 and was chosen by the Perl scripting language community on the basis that the GPL was too restrictive.

The FOSS era also saw the emergence of the commercially-oriented Mozilla Public License (MPL) created by Netscape. The MPL was significant in that it focused on the conversion of a commercial software product to open source. This raised significant challenges in that each of the underlying software licensors incorporated into the Netscape browser would also have to use the same open source license, thus rendering the GPL problematic. Also, Netscape were concerned that an academic style license would not guarantee that developers would contribute back to the community. Thus, a new license, the MPL, was specifically created.

Key Tension for FOSS

Earlier, we discussed the issue of FOSS as a collectivist versus individualist phenomenon. Interestingly, in personal interaction with active and committed open source developers, a frequently expressed motivation for involvement in open source is that of striving for supreme individual achievement, albeit in a collectivist context. It seems that the 'engine' that drives the surprising success of the FOSS model hinges on successfully balancing this tension between collectivism and individualism, The spirit of collective network-enabled cooperation among developers within and across projects is a powerful catalyst to the individual developer heroics that arise due to the healthy competition generated among developers and projects.

Characterising OSS 2.0

The term 'open source' was coined in 1998 in order to place the phenomenon on a more-business-friendly footing than that associated with the ambiguous 'free software' as the common misperception was that you could not make money with free software. The open source initiative succeeded spectacularly well, to the extent that at this stage, a new commercial phenomenon is emerging, which we term OSS 2.0 here. This is quite different to the initial FOSS phenomenon as we illustrate in Table 3. We also identify key challenges for OSS 2.0 and some indicative references to embryonic research that appears relevant to addressing these key challenges.

Table 3 Characterising OSS 2.0

Process	FOSS	OSS 2.0	Key Challenges for OSS 2.0
Development Lifecycle	Planning – "an itch worth scratching" Analysis – part of conventional agreed-upon knowledge in software development Design – firmly based on principles of modularity to accomplish separation of concerns Implementation Code Review Pre-commit test Development release Parallel Debugging Production Release (often the Planning, Analysis & Design phases are done by one person/core group who serve as '*a tail-light to follow*' in the bazaar)	Planning – purposive strategies by major players trying to gain competitive advantage Analysis & Design – more complex in spread to vertical domains where business requirements not universally agreed upon Implementation sub-phases as FOSS but development process becoming *less* bazaar-like	Ensuring quality of open source software – e.g. that excessive modularity does not lead to maintainability problems (Rusovan Lawford & Parnas, 2005; Michlmayr et al, 2005; Schach et al, 2002, Stamelos et al, 2001) Inner source – how to transfer benefits of open source development to conventional development, especially in context of global software development (Dinkelacker & Garg, 2001;Gurbani et al, 2005)

Product	FOSS	OSS 2.0	Key Challenges for OSS 2.0
Domain	Horizontal infrastructure (operating systems, utilities, compilers, DBMS, web & print servers)	More visible IS applications in vertical domains	How to stimulate development in vertical domains not immediately attractive to global development community (Fitzgerald & Kenny, 2003)
Primary Business Strategies	Value-added service-enabling Loss-leader/market-creating	Value-added service enabling Bootstrapping Market-creating Loss-leader Dual product/licensing Cost reduction Accessorising Leveraging community development Leveraging open source brand 'Whole Product' approach	Further exploration of hybrid business models (Krishnamurthy, 2005, O'Neill, 2005, Onetti & Capobianco, 2005) Deriving appropriate total cost of ownership (TCO) measures for open source (Fichman, 2004; Russo et al, 2005; Wheeler, 2005)
Product Support	Quite haphazard – much customer reliance on email lists/bulletin boards, or on support provided by specialized software firms	Customers willing to pay for a professional 'whole product' approach	Effecting the 'whole product' approach

Licensing	GPL, LGPL, Artistic License, BSD and emergence of commercially-oriented MPL	Plethora of licenses (85 to date validated by OSI or FSF)	Safeguarding against IPR infringement
Key Tension	Achieving balance between collectivist v. individualist	Achieving balance between 'value for money' v. acceptable community values	

OSS 2.0 Development Process

As FOSS has largely been a voluntary phenomenon, there has been somewhat of a vacuum in relation to strategic planning (competing with Microsoft on the desktop being one example of a questionable strategy, for example). However, in the OSS 2.0 development lifecycle, strategic planning moves very much to the fore as the haphazard principle of individual developers perceiving 'an itch worth scratching' is superceded by corporate firms considering how best to gain competitive advantage from open source. Red Hat, for example, have published an architecture roadmap which details their plans to move open source up the software stack towards middleware and management tools. Other proprietary companies have also seen the strategic potential of open source to alter the competitive forces at play in their industry. A company may seek to grow market share or undermine competition. IBM, for example, are strong supporters of Linux due to its erosion of the profitability of the operating system market, as this adversely affects their competitors, Sun and Microsoft.

Analysis and Design

As already discussed, FOSS products were primarily targeted at horizontal infrastructure where requirements and design issues were largely part of the established wisdom, thus facilitating a global developer base. However, in vertical domains, where most business software exists, the real problems are effective requirements analysis. Students and developers without any experience in the particular domain simply do not have the necessary knowledge of the vertical application areas to derive accurate requirements which are a precursor to successful development. In OSS 2.0, the analysis and design phases will become more deliberate. In many cases, based on the earlier phase of strategic planning, paid developers will be assigned to work on open source products in vertical domains.

Given its increasingly commercial nature of OSS 2.0, more rigorous project management will be practiced to achieve a professional product. As a consequence, we will see a shift whereby the management of the development process becomes *less* bazaar-like. This is already evident in the formalised meetings for a number of popular open source products, such as the Apache conferences in the US and Europe, the regular Zope/Plone development project meetings, and the GNOME annual project conferences (German, 2003) which bring together developers to coordinate and plan further development. Also, the legal incorporation of several open source projects which ostensibly reduces the risk of litigation for individual developers (O'Mahony, 2005) also allows the project accept donations, perhaps to develop requested functionality, again a trend in keeping with the increased commercialisation and formalisation of the OSS 2.0 development process.

OSS 2.0 Product

Here we will discuss issues to do with the OSS product domain, the business strategies underpinning OSS 2.0, the nature of OSS 2.0 product support, and licensing issues in OSS 2.0.

OSS 2.0 Product Domains

At present, an exhaustive list of open source products, even if limited to those of industrial strength, would be impossible to provide, as the available portfolio of open source software products ranges right across the overall software stack. It is interesting to note that in the highly competitive software world, several open source products have nudged out proprietary alternatives to emerge as 'category killers', that is products of sufficiently high quality and popularity that they obviate the need for development of competitive products. Also, OSS 2.0 is moving from deployment as back-office invisible infrastructure to front-office highly visible deployment of IS applications in vertical domains. An example of this was evident in the Beaumont Hospital case study (Fitzgerald & Kenny, 2003) where a number of in-house developed applications are beign made available on an open source basis to other healthcare agencies. Examples of these include a staff rostering system, a tissue matching system and a casualty system. In the context of open source, this is a very significant development in that it has often been assumed that open source products will not emerge in many vertical domains as they will not be perceived as an 'itch worth scratching' by most developers. However, if organizations in these vertical application domains subscribe to the open source philosophy and contribute expertise from their domain, it will serve to grow the model in these vertical domains.

OSS 2.0 Business Strategies

A number of open source business strategies have been proposed at this stage (Hecker, 2000; Koenig, 2004; Raymond, 1999). In the FOSS era, we discussed two overarching 'families' of business scenarios – value-added service-enabling and loss-leader market-creating. While these are still applicable, they are further nuanced in OSS 2.0. Other strategies have also emerged, including leveraging community software development, leveraging the open source brand, and effecting a 'whole product' approach. Also, it is important to note that companies may not stick solely to one of these scenarios but may pragmatically employ a hybrid mix.

Value-Added Service-Enabling in OSS 2.0

The building of a lucrative service and support business on top of open source was discussed earlier. Companies like Red Hat and Novell have realised large annual revenues through annual subscriptions. However, this 'bootstrapping model' is taken to a higher level in OSS 2.0, as open source products are treated as a platform, somewhat similar the highway or telecommunications infrastructure, and a company bootstraps its own value-added speciality on top. There are a number of examples of companies operating in this manner. Amazon, Google and salesforce.com are very high profile examples of organisations who have taken advantage of the low cost and reliability of open source to create a platform on which they offer their own value-added service in their business domains. Indeed to their customers, the use of open source is pretty much invisible. Also, these companies can customise open source products to precisely suit their internal needs and since they are not redistributing the software, they are not faced with any problems of non-compliance with the GPL.

Market Creation Startegies in OSS 2.0

The other FOSS era business strategy discussed above was that of loss-leader market-creating. In OSS 2.0, the emphasis is firmly focused on market creation. This can be

through a loss-leader approach, dual product/licensing, cost reduction, and accessorising.

A loss-leader example arises in the case of Integrated Development Environments (IDEs). These have traditionally been expensive proprietary applications which have been especially lucrative due to the possibility of a huge license-paying user base. When IBM chose to open source their Eclipse IDE, the decision seemed surprising given that the source code was valued at $40 million. However, there were massive compensations for IBM. Open sourcing Eclipse dramatically increased its popularity as a development platform. Estimates currently suggest 10,000 downloads of Eclipse per day. This grows the market for other complementary products from IBM. Several other companies have now also open sourced their proprietary IDEs—SUN with Netbeans, BEA with BeeHive, for example. This snowball effect arises largely due to vendors feeling the need to try counter any potential competitive advantage that might otherwise be gleaned by competitors.

There are several examples of dual product/licensing strategies. MySQL are a high profile example of a company who have adopted such a strategy. It is estimated that 5 million free copies of MySQL have been downloaded at this stage. Of these, 5,000 customers have purchased a commercial license from MySQL. While a return of 0.1% might seem minuscule, the large numbers of downloads make this a very successful strategy—many market-leading vendors of proprietary software packages would be very pleased with a base of 5,000 customers. Other high profile examples of dual product strategies include Red Hat with Fedora and Enterprise Linux, Sun with Star Office and OpenOffice.

Companies can leverage the commodification effect that has occurred with open source. In this case, companies may take advantage of open source in terms of its low cost, reliability and portability across platforms. Oracle can reduce the overall cost of database implementation for their customers, and IBM can reduce the overall cost of servers. In the area of embedded systems, open source is fast becoming dominant. Here, companies are concerned with open standards, stability, high performance, small footprint and the ability to run on generic and minimal specification hardware, together with a vibrant and responsive development community who are willing to port to other platforms and write extra utilities. Open source products score very highly on all these issues. Philips, Nokia, BMW, and many other companies in the so-called ‘secondary’ software sector where software is becoming more pervasive and a critical part of their business, pursue such a strategy.

Several companies leverage open source as a base upon which they offer non-software products. For example, HP promote open source in areas which facilitate the deployment of HP hardware. In other cases, products such as books and other accessories can offer lucrative returns—the O’Reilly publishing house being a very successful exemplar.

Leveraging Community Software Development

In today’s marketplace, the shortage of skilled personnel is a critical problem. Therefore, leveraging the development talent of the open source community allows companies to increase development productivity, with the added benefit that much of this development work may be done for free. Thus, hundreds of Eclipse plug-ins have

been developed. Also, Apple's initiative in starting the Darwin open source project to develop part of its operating system facilitates extra development contributions which they can take advantage of as more or less free R&D, in addition to improving Apple's reputation in the open source community. This becomes a circular phenomenon, as the extra functionality increases its attractiveness to other developers who also contribute to further grow functionality, and in a rapid timescale.

Leveraging the Open Source Brand

While patents and copyrights are such key issues with respect to free software, another IP mechanism, that of trademark or brand, could become quite significant. For example, Oracle promote the "unbreakable Linux" slogan. Also, an increasing number of government agencies and public administrations (traditionally the largest consumers of software) are mandating that open source be a priority option, even to the extent that a formal case be made if open source is not chosen as a solution. This will ensure that the open source brand becomes even more important in the future.

A further variation on this strategy is to leverage the open source brand to re-energise a flagging product by re-releasing it as open source— Computer Associates' Ingres database, and SAP's SAP DB metamorphosing into MySQL's MaxDB come to mind. This strategy also seems to have elements of strategic and political opportunism, and of the value-added service-enabling strategy discussed above, as it entices some new users who are positively disposed towards open source and the company can benefit through selling support, thus revealing the hybrid nature of these business strategies.

The 'Whole Product' Approach

The particular characteristics of the OSS 2.0 position it as a good exemplar of the 'whole product' concept, proposed by Moore (1999) to reflect a market-driven business approach which seeks to deliver a complete solution to the customer in terms of products and services. The open source phenomenon is market-driven and, as discussed above, places a great deal of emphasis on services. A professional approach is taken to achieving value by establishing a profitable business venture for which customers are willing to pay the going rate. In this scenario, developers do the coding and others complete the business model by adding sales and marketing services— necessary activities but not ones that developers may be interested in. The OSS 2.0 'whole product' approach is also larger than a single company or software product or service. Indeed, the network benefits of open source arise as a result of the size of the community rather than any individual company. Thus, a network of interested parties with complementary capabilities offer a professional product and service in an agile and bazaar-friendly manner. Customer service requests can be routed to the most appropriate expert partner in the network. In this manner, the OSS 2.0 brand increases trustworthiness to achieve market leader status, and also leverages the open source brand effectively. Such a convenience network is not unknown in conventional business circles. The LVMH (Louis Vuitton Moet Hennessy) brand is an international network of almost 50 luxury brand leaders in fashion, wines & spirits, watches, jewellery and cosmetics (www.lvmh.com). From a business perspective, this network of well-known brands combine to create the ultimate luxury brand status, LVMH, but they can still pursue their own business interests independently also.

OSS 2.0 Product Support

Developers in the past have referred to the "exhilarating succession of problem-solving challenges" in installing open source products (Sanders, 1998). However, as the OSS 2.0 model moves more mainstream, it is unlikely that time-impoverished professionals will seek exhilaration in this manner. As OSS 2.0 evolves, customers will want a professional service—support, training and certification—and will be prepared to pay for it, as evidenced by the large number who choose to buy Star Office in preference to opting for the free OpenOffice. The unreasonable expectation mentioned earlier, that just because the software license cost was zero, then all other software-related costs should also be zero will be eliminated. Further, many organisations find it difficult to accept the fact that their support essentially derives from a series of bulletin boards. There is inevitably a hankering for the comfort factor of a support number, and have someone take care of the problem.

In OSS 2.0, the bazaar metaphor therefore shifts from just being associated with the development process to address the whole product, with a major emphasis on achieving value by establishing a profitable business venture for which customers are willing to pay the going rate. Many companies will find profitable opportunities supporting customers who are deploying open source products. The claim by large proprietary software companies that open source would stifle the local software industry is proving unfounded. A far more likely scenario is that small indigenous software companies will thrive in providing local training, technical support and consultancy for companies deploying open source products.

OSS 2.0 Licensing

In OSS 2.0, we see the emergence of a plethora of licenses – a total of 85 distinct licenses have been approved to date by the Open Source Initiative (OSI) and the Free Software Foundation (FSF). However, there is by no means universal acceptance – as 28 are approved by the OSI only, 26 are approved by the FSF only, and 31 licenses are approved by both (Lyddy-Collins, 2005). This plethora of licenses can be grouped into four broad categories: reciprocal licenses (as per the FOSS era), academic-style licenses, corporate licenses and non-approved (by FSF or OSI) licenses such as shared source (see Table 4).

Table 4 A Typology of OSS 2.0 Licenses

Reciprocal	GPL, LGPL, Open Source License (OSL)
Academic Style	Academic Free License, Apache License, BSD, MIT
Corporate Type	MPL, Qt Public License, Sun Public License, IBM Public License, Apple Public License, Eclipse Public License
Non-Approved (e.g Shared Source)	Microsoft Shared Source (MSS), Sun Community Source License (SCSL)

Reciprocal licenses such as the GPL and LGPL have already been covered above in our discussion of FOSS licensing. However, a recent addition here is the Open Software License (OSL) created by Rosen in 2002. The OSL was designed to be an

alternative to the GPL which would be more acceptable to corporate users and developers, thereby promoting reciprocal licenses in this constituency. Again, this emphasises the continued progression towards corporate and commercial compatibility which is at the heart of OSS 2.0. Interestingly, while the FSF have issued warnings against this license, Linus Torvalds has adopted it for open source development other than the Linux kernel.

Academic style licenses were also covered in the FOSS licensing discussion above. Again, these licenses have evolved to cover patent concerns. Also, the Apache License V2 seeks to incentivise external parties to contribute improvements back to the community, without making this compulsory as under the GPL.

Corporate-style licenses are central to OSS 2.0, and they reveal a potential friction point in OSS 2.0 since they seek to benefit corporate interests rather than the open source development community – a point we will return to later. As already mentioned, these are generally based on the Mozilla Public License (MPL). Typically, these licenses seek to allow open source code to be mixed with proprietary code, and to ensure that corporate sponsors retain control of derivative works.

Perhaps, the most interesting category of licenses for OSS 2.0 are the non-approved licenses, since these push the boundaries of the proprietary software sector as they seek to accommodate the open source model. Two significant exemplars here are the Microsoft Shared Source (MSS) License, and the Sun Community Source License (SCSL), discussed in turn.

Ironically, Microsoft will be a major player in the hybrid OSS 2.0. Microsoft has actually distributed an open source product for quite some time—Windows Services for Unix – and has publicly acknowledged that Windows 2000 and Windows XP use open source BSD code[3]. Also, it has a number of high profile open source projects on SourceForge. Microsoft has already abstracted some of the key ideas from open source. Recognizing the power of the social and community identification aspects of open source, it has introduced the Most Valued Professionals (MVP) initiative which extends source code access to this select group. The Open Value policy extends discretion to sales representatives to offer extreme discounts and zero percent financing to small businesses who may be likely to switch to zero cost open source (Roy, 2003). Also, the realisation that "transparency increases trust" (Matusow, 2005) has led to Microsoft's Shared Source License. Initially, this just allowed certain customers full access to source code, but in some cases it allows "licensees to review, modify, redistribute, and sell works with no royalties paid to Microsoft" (Matusow, 2005). However, both FSF and OSI are in agreement that this license is neither free nor open. Nevertheless, as OSS 2.0 evolves, it will be increasingly difficult to exclude such licenses which increasingly comply with the hybrid model of OSS 2.0.

The SCSL license has been chosen by Sun for its Java product. This license requires that all modifications be submitted back to Sun. Also, any modified versions of the work must be certified a compatible by Sun, and any commercial use may require payment of a royalty, a clause which is incompatible with the open source model.

[3] http://support.microsoft.com/default.aspx?scid=KB;en-us;q306819

Key issues for OSS 2.0 licensing are the issue of warranties and indemnification against IP infringements, an issue which came very much to the fore in the wake of the SCO Group's lawsuit for patent infringement in Linux against IBM. Currently, Red Hat offers a warranty against any infringement in their Red Hat Enterprise Linux distribution (although this just promises that Red Hat will replace any infringing code). Similarly, Novell has offered customers of its SuSE Linux an indemnification against copyright (but not patent) infringements. HP also offers its customers an indemnification, but only for claims made by SCO. While these initiatives may offer only very limited protection, nevertheless, in OSS 2.0, companies will seek such protections.

Key Tension Driving OSS 2.0: Value for Money v. Adhering to Acceptable Community Values

OSS 2.0 serves to further blur the distinction between open source and proprietary software as key open source players such as the Red Hat and SuSE position their Linux distributions to be more similar to the proprietary model, whereas traditionally proprietary companies, such as HP, move more towards open source. However, in the OSS 2.0 model, they must still satisfy certain criteria in relation to acceptable community values.

The ambiguous term 'free' may have been the key word for FOSS initially, with both its meanings (i.e. free as in zero cost and free as in unrestricted access) significant. For OSS 2.0, an even more ambiguous term, value, will be the key term, with two of its connotations especially significant – 'value for money' and 'acceptable community values'. The integration of open source into the commercial arena and the associated desire to create profit represents a critical source of tension, given the concomitant need to achieve a balance with the collectivist, public-good community-values, which are an inevitable legacy given its origins in the more ideologically-driven Free Software community, and the continued development contribution by that community. Both connotations of value are discussed here.

Value for Money:

OSS 2.0 can dramatically alter the economic dynamics of the marketplace. Despite the vast sums of money involved and the enormous economic potential of OSS 2.0, it is a fact that it erodes certain hitherto profitable markets, the multi-billion dollar operating system market being the most obvious example. Red Hat who have been by far the most financially successful company distributing Linux have annual revenues of about $90 million. However, this pales into insignificance beside Microsoft's $32 billion dollar annual revenue. A similar scenario is likely to occur in the erosion of the MS Office market-share by OpenOffice. Thus, OSS 2.0 decimates the traditional routes to profitability of many sectors.

As OSS 2.0 emerges, those involved are pragmatically seeking to make money from their endeavour. While OSS 2.0 organisations may not aspire to Microsoft-like annual revenues, or even to match Red Hat annual revenues, they are seeking to make payroll and earn a reasonable livelihood. Both the customer and developers need to perceive value for money in OSS 2.0. Free as in zero cost is replaced by a value for money concern, and OSS 2.0 customers are prepared to pay for a professional service. This is

very evident in the fact that many companies are prepared to pay a fee for Star Office with associated support and warranty, rather than the zero-cost OpenOffice alternative.

Acceptable Community Values

Maintaining open source-compatible community values will represent a very significant balancing act for OSS 2.0. Large commercial organisations are not always well perceived within the open source community. This issue will become even more important as organisations seek to profit financially with OSS 2.0. They will have to adhere to a set of acceptable community values. This represents a very interesting dilemma for traditional proprietary software companies. Companies such as IBM, Sun and HP, for example, while clearly supporting open source initiatives, are also at odds with the open source community in their support for patents. Also, the quintessential patron of open source, Red Hat, could struggle in future as its policies increasingly conflict with community spirit and values. The use of subscription agreements and effectively locking customers in with confidential service bulletins are probably moving close to the boundary of acceptable community values. The power of community should not be underestimated. A telling example of this was the attempt by Caldera to sell its Linux distribution which failed due to the extremely negative reaction of the open source community.

In the Beaumont Hospital case (Fitzgerald & Kenny, 2003), some examples of behaviour reinforcing positive community values were evident. Firstly, as already mentioned, Beaumont is making a suite of in-house developed applications available in an open source mode. What has also been particularly striking has been the sense of ownership that the end-user nursing staff have developed for the nurse rostering system which is being made available as open source. These end-users have been very active in demonstrating the system to other hospitals. Interestingly, their nursing counterparts in other hospitals appear to be swayed very much by the positive attitudes of their professional colleagues. Ironically, female participation in open source development has been minuscule as reports of almost 99% dominance by males are reported (Lakhani & Wolf, 2005; Ghosh et al, 2002)—the linuxchix initiative notwithstanding (www.linuxchix.org). Given the predominance of females among the nursing staff and the enthusiasm with which they have embraced the ideology of open source, this suggests that the movement overall could leverage this sector of the population to a greater extent.

In a similar fashion, the spirit of OSS 2.0 can lead to some very positive network externality effects. In the Beaumont Hospital case, users of the same products from other countries travelled to Beaumont to volunteer support and offer extra functionality that they had developed. The expectation was that Beaumont would reciprocate by making available any extra functionality they developed. Cooperation of this nature is pretty much unheard of in the proprietary marketplace, but again is symptomatic of the very strong community value orientation of open source.

Again, just as maintaining a healthy balance between collective cooperation and individualist competition was the primary ‘engine’ driving FOSS, a further impetus that will drive OSS 2.0 will be achieving an appropriate balance between the value for money proposition while still satisfying the scrupulous community values required by

the development community. This is a very complex issue as the following two vignettes illustrate:

In the Beaumont Hospital case, the hospital managed to achieve considerable savings by deploying an X-ray system using open source components. However, there was some resentment to the new system in the radiology department. Some staff felt somewhat 'short-changed' in that they expected to spend about €4m on an X-ray system, just like their counterparts elsewhere using proprietary systems, and the deployment of a much cheaper open source alternative was perceived as 'undervaluing' the radiology department to some extent.

A similar issue arises in the relation to open source in developing countries. The seemingly obvious attraction to poorer institutions there is especially interesting as the issue is actually not as simple as it is often portrayed. An excellent description of failed attempts to initiate open source projects in Ghana by Zachary (2003) identifies fundamental problems due to the widespread belief among Ghanaian programmers and users that nothing of value could be done for free, and he concludes that open source concepts would need to be considerably 'Africanized' in order to have a chance of success. Indigenous software developers there could not accept that an initiative which was based on free software could have any value. They aspired to a software industry based on the same profit and market values as in developed countries. Perception of value depends on values, and the above examples illustrate that this varies considerably across cultural and organisational contexts.

Key Challenges for OSS 2.0

Table 2 identifies a number of key challenges for the emergent OSS 2.0 phenomenon. Here we discuss these and identify some indicative research which is relevant to addressing these key challenges.

Ensuring quality of open source software

We have already discussed the myth that open source products are of universally high quality. While some open source projects do exhibit very high quality, probably higher than commercial equivalents (Michlmayr et al, 2005), this is by no means the case universally (Stamelos et al, 2001). However, ensuring the quality of the product is critical for OSS 2.0. Software quality is a complex and multifaceted area, but to focus on just one issue relevant to open source – the implications of modularity for maintainability. We have already discussed the importance of modularity in open source projects, as it provides both a way of partitioning work among a global development community, and also reduces the learning curve for new developers who not have to become familiar with the overall system. This is a classic case of information hiding (Parnas, 1972). However, the principles governing how modules should be defined are quite complex, and if not properly applied, coupling problems between modules can lead to severe software maintainability problems, a feature already identified in open source projects (Schach et al, 2002; Rusovan, Lawford & Parnas, 2005)

Transferring benefits of open source to conventional development - Inner Source

While open source may not represent a real paradigm shift in software development, it is an extremely successful exemplar of globally distributed development. In the current climate of outsourcing and off-shoring, the open source development model is attracting considerable attention as organisations seek to emulate open source success on traditional development projects, through initiatives variously labelled as 'inner source', 'corporate source' or 'community source', in an effort (Dinkelacker & Garg, 2001; Gurbani et al, 2005).

Other open source principles such as open sharing of source code, large-scale independent peer review, the community development model, and the expanded role of users are being experimented with. In traditional software development, users and developers are typically located in separate departments, all too frequently without a lot of mutual respect or voluntary interaction. The user-developer relationship has typically been quite different in open source. Early open source developers were users of the actual products. However, as OSS 2.0 has emerged, the situation has changed somewhat. Inevitably, developers are not always the users of the software being developed. In the absence of a traditional vendor, users need to become much more intimately involved in the development process, as technical staff cannot simply send a checklist of requirements to the vendor to ascertain if needs will be met. Deploying open source can lead to a sense of shared adventure which is not a common scenario in the proprietary software arena. Similarly, Norris (2004) reports that users were willing to sacrifice certain desired functionality if it could not be easily provided by open source products. Also, the open source development model offers benefits in terms of network externality effects due to increased and proactive cooperation among the users of open source software who seek to establish reciprocal arrangements to their mutual benefit (Fitzgerald & Kenny, 2003).

Stimulating open source in vertical domains

Open source products initially have tended to be in horizontal infrastructure where requirements are part of the conventional wisdom, thus allowing developers with different backgrounds or even students to contribute. However, in vertical domains, the business requirements are more complex, requiring extensive domain knowledge, and thus a significant challenge is to stimulate open source development in these domains. An example of this occurring in the healthcare sector in the Beaumont Hospital case was mentioned earlier (Fitzgerald & Kenny, 2003). As more purposeful strategic planning takes place in OSS 2.0, complementary development strategies will be enacted to provide a complete portfolio of OSS products.

Exploration of hybrid business models

Quite a lot of research is already being undertaken to refine and elaborate the business strategies discussed earlier (e.g. Krishnamurthy, 2005, O'Neill, 2005, Onetti & Capobianco, 2005). However, there is a need for some careful definition of concepts in this area. For example, the term 'business model' is frequently used very loosely in the context of open source. A useful definition of the business model concept is provided by Mahadevan (2000) who suggests it comprises three components: value, revenue and logistics. The value component represents the value proposition for customers and vendors, the revenue component focuses on how organisations can earn revenue, and the logistics component focuses on supply chain issues. To date, most research on open source business models has focused primarily on revenue

generation issues. However, as the earlier discussion of OSS 2.0 business strategies illustrates, the value proposition and the logistics of the whole product across the overall supply chain are paramount in OSS 2.0, and more analysis is needed in these areas.

Deriving appropriate Total Cost of Ownership measure for OSS 2.0

Calculating the total cost of ownership (TCO) of software is a very complex multi-faceted issue. It requires consideration of a vast number of areas, including software purchase, maintenance and upgrade costs, hardware purchase and maintenance costs, personnel training, legal and administrative costs (Russo et al, 2005). Given this complexity, it is perhaps unsurprising that leading vendors, such as Microsoft and Sun, for example, have each been able to claim a lower TCO (Wheeler, 2005). However, conventional TCO measures may not be suited to the open source phenomenon in which less obvious benefits accrue due to network externality effects and the more co-operative developer-user relationship, for example.

A promising strand of research which could suit the dynamics of open source is based on the theory of real options investment analysis (Fichman, 2004). Real options thinking is appropriate in conditions of high degrees of flexibility and uncertainty. The characteristics of open source fulfil these criteria. In terms of flexibility, there is considerable scope in which products or functions may be implemented, and in what unique customisation might be built in. Also, the zero cost aspect offers a great deal of flexibility in terms of choosing when to implement. However, there is also much uncertainty in that there is no 'royal road' to fail-safe open source implementation.

Effecting the 'whole product 'approach

Given the plethora of open source products currently available and the lack of vendors to proactively supply information, there is a need for an up-to-date catalog of open source products to counter the knowledge gap and reduce the search cost for suitable solutions from the myriad of products available. Such a catalog could provide details as to the functionality offered by various products, the types of support available, training needs, reference sites of deployment, companies offering support etc. This kind of brokerage service is one which intermediaries can use to build a lasting and profitable business relationship with customers. Conecta in Italy and OpenApp in Ireland are examples of two small software companies following such a model. These companies are able to win high margin consulting contracts supporting open source deployment. Interestingly, these small companies are aware of the need for other competitors to enter the space to raise general awareness of open source, but also realise the need to limit this to ensure that the profit margin is not eroded by too much competition. Thus co-opetition becomes an important watchword in OSS 2.0.

Safeguarding against IPR Infringement

While open source was a fringe phenomenon, there was a certain safety in its relative obscurity. Once it entered mainstream and threatened the livelihood of the established players, the stakes shifted. One effect of this is the increased fear of litigation due to fear of IPR infringements, particularly in the area of software patents. Patents can make a good deal of sense in *theory*—as initiatives which stimulate publication of non-trivial ideas. Clearly, they can play a role in improving innovation and creativity,

giving small players a chance, and consequently benefiting society as a whole. However, one can certainly disagree with their implementation in *practice*. The guarantee of exclusively-held monopoly rights over a long period of time is clearly not suited to the current IT *zeitgeist*. Interestingly, it is not too much of a stretch to define the open source phenomenon as publication of non-trivial ideas, with an explicit guarantee of continued availability, which seeks to protect the interest of the small players, for the overall betterment of society, as others learn from and improve on the original ideas. This shows how close the goals of open source and IP protection are at heart, However, the stimulation of innovation and creativity, which should be the fundamental rationale behind IP protection, has failed abjectly in the software area (Bessen & Hunt, 2003). Ironically, even though open source has often been about replicating proprietary products, the ingenuity of the global development community has allowed for extremely innovative new functionality to emerge also—the OpenOffice suite and the Mozilla Firefox browser, being two recent examples of this.

Given this context, the idea of a central software conservancy to hold intellectual property for a range of open source products is an attractive one. This would benefit developers as it would remove them from the risk of litigation, a phenomenon already identified by O'Mahony (2005) in the move towards legal incorporation for several open source projects. Such a conservancy would also reduce the risk of IPR infringement, much desired by companies, who would prefer to deal with a single entity rather than a multitude of developers (Everitt, 2004). It also allows an open source project to accept donations, again a trend in keeping with the increased commercialisation and formalisation of the OSS 2.0 development process.

Conclusion

Here we have sought to deconstruct the myth of open source software as a 'silver-bullet' panacea capable of addressing all problems in software development. We have also characterised the emergent OSS 2.0 model and differentiated it from its FOSS antecedent in terms of core process and product characteristics. We identify the key driver behind the success of FOSS as that of supreme individual achievement in a collectivist collaborative context which is primarily a developer-oriented issue. For OSS 2.0, a more corporate-oriented tension is added, that of achieving a balance between a 'value for money' proposition while at the same time satisfying acceptable community values. We also identify a number of key research challenges for the emergent OSS 2.0 phenomenon, and some indicative research which appears relevant to investigating these areas.

At the outset, we mentioned the similarities between the OSS and DSS phenomena (beyond the fact that they share two sibilant consonants) and identified some of the problems which surfaced in DSS research. There is a danger that open source research could suffer the same problems as DSS – indeed, the problems could be even more accentuated. One of the apparent benefits of the open source phenomenon from a research point of view is the ready availability of a vast amount of raw data on portals such as Sourceforge and Freshmeat. Much research has focused inward on the open source phenomenon itself, studying the characteristics of projects, and the motivations of developers to contribute, for example. While these studies are clearly useful, not everything that can be measured, should be. A purposive research agenda which looks outward at the open source phenomenon and its implications is also necessary. One of

the goals of this piece is to identify such a broader set of concerns that could be the subject of further research.

References

Benbasat, I and Zmud, R (2003), The identity crisis within the IS discipline: defining and communicating the discipline's core properties, MIS Quarterly, Vol. 27, No. 2, pp. 183-194.

Bessen, J and Hunt, R (2003) An empirical look at software patents, Working paper 03-17/R, http://www.researchoninnovation.org/swpat.pdf

Bezroukov, N (1999) OSS research as a special type of research, Vol. 4, No 10, http://www.firstmonday.org/issues/issue4_10/bezroukov/

Brooks, F. (1987) No silver bullet: essence and accidents of software engineering. *IEEE Computer*, **April**, 10-19

Cox, A. (1998) Cathedrals, bazaars and the town council, http://slashdot.org/features/98/10/13/1423253.shtml

Dinkelacker, J and Garg, P (2001) Applying Open Source Concepts to a Corporate Environment, in *Proceedings of 1st Workshop on Open Source Software Engineering*, ICSE2001, Toronto, available at http://opensource.ucc.ie/icse2001.

Everitt, P (2004) Zope: open source, revisited, First CALIBRE International Conference, Hague, Nov 2004 (available at www.calibre.ie).

Feller, J. and Fitzgerald, B. (2002) *Understanding Open Source Software Development*, Addison-Wesley; UK

Fichman, R (2004) Real Options and IT Platform Adoption: Implications for Theory and Practice, *Information Systems Research*, Vol 15, No 3

Fitzgerald, B. and Kenny, T. (2003) Open Source Software in the Trenches: Lessons from a Large Scale Implementation, *Proceedings of 24th International Conference on Information Systems*, Seattle, December 2003, pp. 316-326.

German, D. M. (2003) GNOME, a case of open source global software development, In International Workshop on Global Software Development, (co-located with ICSE 2003), pp. 39-43, <gsd2003.cs.uvic.ca/gsd2003proceedings.pdf>

Ghosh, R, Glott, R, Krieger, B and Robles G (2002) FLOSS: Free/Libre/Open Source Software Study, International Institute of Infonomics/MERIT, http://floss.infonomics.nl/report/

Gurbani, V.K., Garvert, A. and Herbsleb, J.D. (2005) A Case Study of Open Source Tools and Practices in a Commercial Setting, In *Proceedings of the 5th Workshop on Open Source Software Engineering*, 27th International Conference on Software Engineering: ICSE 2005, St. Louis, 17 May 2005, pp. 24-29.

Hecker, F (2000) Setting up shop: the business of open-source software, http://www.hecker.org/writings/setting-up-shop

Jorgensen, N (2001) Putting it All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project, *Information Systems Journal*, Vol 11, No 4, pp 142-157.

Keen, P. (1991) Keynote address: relevance and rigor in information systems research, in Nissen, H., Klein, H. and Hirschheim, R. (eds) (1991) *Information Systems Research: Contemporary Approaches and Emergent Traditions*, Elsevier Publishers, North Holland, 27-49.

Koenig, J. (2004) Seven open source business strategies for competitive advantage, *IT Manager's Journal*, May 15 2004

Krishnamurthy, S. (2005) "An Analysis of Open Source Business Models", in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge

Lakhani, K. and Wolf, B. (2005) Motivation and Effort in Free/Open Source Software Projects: The Interplay of Intrinsic and Extrinsic Motivations, in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge.

Lerner, J. and Tirole, J. (2000) The simple economics of open source, Harvard Business School Working Paper #00-059, www.opensource.mit.edu

Lyddy-Collins, N (2005) Perspectives on open-source software licensing policy in a commercial software development environment, Unpublished Masters Thesis, University of Limerick.

Mahadevan, B (2000) Business models for internet-based ecommerce: an anatomy, *California Management Review*, Vol. 42, No 4, pp. 55-69.

Matusow, J (2005) Shared source: the Microsoft perspective, forthcoming in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge.

McKusick, M. (1999) "Twenty Years of Berkeley UNIX: From AT&T-Owned to Freely Redistributable," in *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates, USA.

Michlmayr, M, Hunt, F & Probert, D (2005) Quality practices and problems in free software, in Scotto, M and Succi, G (Eds) *Proceedings of First International Conference on Open Source (OSS2005)*, Genoa, 11-15 July 205, pp. 24-28.

Mockus, A., Fielding, R. and Herbsleb, J. (2000) A case study of open source software development: the Apache server, in *Proceedings of 22nd International Conference on Software Engineering*, pp. 263-272.

Moore, G (1999) *Crossing the Chasm,* Harper, NY.

Norris, J (2004) Mission-critical development with open source software: lessons learned, *IEEE Software,* Vol. 21, No. 1, pp. 42-49.

O'Mahony, S (2005) Non-Profit Foundations and their Role in Community-Firm Software Collaboration, in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge, pp. 393-414.

O'Neill, Eoghan (2005) *An analysis of open source business models and partner networks*, Unpublished MSc Thesis, University College Cork.

Onetti, A & Capobianco, F (2005) Open source and business model innovation. the Funambol case, in Scotto, M and Succi, G (Eds) *Proceedings of First*

International Conference on Open Source (OSS2005), Genoa, 11-15 July 205, pp. 224-227.

Parnas, D. (1972) On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15, 12, pp.1053-1058.

Paulk, M., Curtis, B., Chrissis, M. and Weber C. (1993) Capability Maturity Model for Software, version 1.1, *IEEE Software*, Vol. 10, No. 4, pp.18-27

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

Roy, A. (2003) Microsoft vs. Linux: Gaining Traction, *Chartered Financial Analyst*, Vol. 9, Issue 5, pp. 36-39.

Rusovan, S, Lawford, M and Parnas, D. (2005) Open Source Software Development: Future or Fad? in Feller, J, Fitzgerald, B, Hissam, S, and Lakhani, K. (2005) (Eds) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge

Russo, B, Braghin, B, Gasperi, P, Sillitti, A & Succi, G (2005) Defining TCO for the transition to open source systems, in Scotto, M and Succi, G (Eds) *Proceedings of First International Conference on Open Source (OSS2005)*, Genoa, 11-15 July 205, pp. 108-112.

Sanders J. (1998) Linux, Open Source, and Software's Future, *IEEE Software* September/October 1998 pp 88–91

Schach, S., Jin, B., and Wright, D. (2002) Maintainability of the Linux kernel, in in J Feller, B Fitzgerald, S Hissam, and K Lakhani (Eds.) *Proceedings of 2nd Workshop on Open Source Software Engineering*, ICSE2002, Orlando, Florida, available at http://opensource.ucc.ie/icse2002.

Stamelos, I., Angelis, L. and Oykonomou, A. (2001) Code Quality Analysis in Open-Source Software Development, *Information Systems Journal*, Vol 11, No. 4, pp. 261-274.

Torvalds, L. and Diamond, D. (2001) *Just for Fun: The Story of an Accidental Revolutionary*, Harper Collins, New York.

Trott, P (1998) Innovation Management and New Product Development, Trans-Atlantic Publications.

Tushman, M and Anderson, P (1986) Technological discontinuities and organizational environments, *Administrative Science Quarterly,* Vol, 31, pp.439-465.

Wheeler, D. (2005) Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers! http://www.dwheeler.com/oss_fs_why.html

Young, R. (1999) "Giving it Away: How Red Hat Stumbled Across a New Economic Model and Helped Improve an Industry," in *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates, USA.

Zachary, G (2003) Barriers to the formation of open-source software communities in an African city: the case of Accra, Ghana, http://archives.linux-aktivaattori.org/discussion/att-0408/ICSI.Paper_Ghana

10.2 Supporting Global Software development in open Source Ecosystems: A Role for Actability in the Pragmatic Web

Pär J. Ågerfalk, University of Limerick, Ireland

Proceedings of the First International Pragmatic Web Conference (PRAGWEB 2006),
Stuttgart, Germany, 21-23 September 2006.

Abstract

New forms of collaboration between organizations based on open source principles are rapidly emerging. The collaboration is typically done in a spirit of co-opetition whereby companies, often SMEs, share cost and risk by developing software jointly and openly. The paper elaborates how this emerging phenomenon of open source ecosystems can be understood from the perspective of actability and the Pragmatic Web. The concept of open source ecosystems as a form of global software development is explored and actability is presented as a useful concept for articulating design criteria for the required collaborative tools. In doing so, a possible research agenda for pragmatic web research in this domain is outlined.

1 Introduction

The open source software (OSS) landscape is a rapidly changing one. While OSS and its Free Software antecedent were primarily driven by ideology and individual commitment, the main driving force of OSS today is commercialization and opportunities for inter-organizational collaboration [Fi06]. OSS is no longer mainly about enthusiasts contributing to SourceForge projects, but increasingly about commercial organizations developing software in 'co-opetitive ecosystems' [ÅFH06]. Many small and medium-sized enterprises (SMEs) see this mode of working as a way of sharing risks and costs and an opportunity to participate in cutting edge research that does not require a significant R&D budget [ÅDF05]. This trend is in line with recent studies which show that a significant proportion of contributors to OSS projects are paid employees [FL02, DWA03] and supported by many recent commercial OSS events[4]. Understanding the underlying communicative mechanisms of such ecosystems and designing information technology (IT) to support the collaboration is a challenging task for information systems research.

From a software/information systems development perspective, this collaboration (or rather co-opetition since the collaborators are often also competitors in the marketplace) can often be seen as a particular form of globally distributed software development [LLÅ06]. This mode of software development, often referred to as GSD (global software development), emphasizes the problems associated with increased geographical, temporal and socio-cultural distance between people [ÅFH05]. The implications of increasing distance are also central in recent work on information systems actability [Åg04] which emphasizes that the introduction of IT for communication in business processes also increases distance between people. Particularly, communication through and by means of IT moves the business interaction away from an "ideal" face-to-face setting [Cl96] towards an IT mediated one [Åg03, Åg04]. Actability is founded on a pragmatic view of IT use and thus

[4] See, for example, www.calibre.ie and www.osbc.com.

appears as a fruitful concept in a discussion about the Pragmatic Web. The notion of the Pragmatic Web, as it has been proposed by Schoop and colleagues [SMD06] captures many of the problems faced in open source ecosystems generally and in supporting their GSD efforts particularly: "to augment human collaboration effectively by appropriate technologies … in [distributed] communities of practice."

This essay introduces the concept of open source ecosystems as a form of GSD and explores how actability principles can be useful in articulating design criteria for the required collaborative tools. In doing so, a research agenda for Pragmatic Web research in this domain is outlined. Given that OSS communities have traditionally been viewed as collections of loosely coupled individuals with a common 'itch worth scratching' [Ra99], it is important to emphasize that the kind of OSS development we are concerned with here is open collaboration between commercial organizations. Such collaboration is more structured and organized than the traditional 'bazaar style OSS development model' [Fi06] and thus more in line with the sort of institutionalized setting that the actability concept has been explored in previously (i.e. a work context).

2 GSD in Open Source Ecosystems

Recent research on the open source software (OSS) phenomenon suggests that we are currently witnessing an ongoing shift from OSS as community of individual developers to OSS as community of commercial organizations, often SMEs, operating as symbiotic ecosystems in a spirit of co-opetition [ÅDF05, ÅFH06, Fi06]. These ecosystems are not restricted to software development collaboration, but rather draw on open philosophies in a wide range of domains (open content, open knowledge, open standards, etc). However, the impact on software development organizations is of particular interest since OSS can be seen both as an approach to inter-organizational collaborative software development and as the result of such collaboration to be employed by other, non software developing organizations and by the software developing organizations themselves. Interestingly, many commercial organizations involved in this emerging second wave of OSS can be characterized as 'secondary software companies' [ÅFD05]. These are companies whose main product is not software, but of which software plays an important part; telecom, automotive and medical devices are good examples.

The OSS development model is by its very nature globally distributed [LLÅ06, ÅFH06]. Globally distributed software development (GSD) has been characterized as a setting where increased geographical, temporal and socio-cultural distance affects the processes of communication, coordination and control in software projects [ÅFH05]. Temporal distance is here thought of as 'a directional measure of the dislocation in time experienced by two actors wishing to interact', caused by, for example, time zone difference; geographical distance is 'a directional measure of the effort required for one actor to visit another at the latter's home site'; and socio-cultural distance is 'a directional measure of an actor's understanding of another actor's values and normative practices'. In the context of GSD, the communication process concerns transfer of information and creation of common understanding, and the tools used to facilitate such interaction. Coordination has been defined as 'the act of integrating each task with each organisational unit, so the unit contributes to the overall objective.' [CA01] The coordination process thus concerns interdependencies between actors: 'Two people have a coordination problem whenever they have

common interests, or goals, and each person's actions depend on the actions of the other.' [Cl96] Finally, control is 'the process of adhering to goals, policies, standards, or quality levels.' [CA01] The control process thus concerns the management and reporting mechanisms needed to make sure a development activity is progressing. Table 1 summarizes these concepts and points at some of the characteristics of GSD in an OSS context.

Process	Dimension		
	Temporal Distance	*Geographical Distance*	*Socio-Cultural Distance*
Communication	Practically all routine communication asynchronous, through the Internet.	Typically, developers acting as the market. Internet used creatively for communication channels.	Responsive communities of motivated, self-selected contributors.
Coordination	Preponderance of modular, plug-in style architectures, reducing the need for coordination.	Dynamic and flexible labour pools. High critical task awareness throughout the community.	Common environments based on free, lightweight tools and lightweight process infrastructures.
Control	Typically 24x7 working. Control primarily through the commit process.	Mechanisms in place for identifying and addressing the issue of non-active key members.	Shared project goals and no project forking. Protection of OSS values. High level of activity on mundane tasks.

Table 1: Characteristics of GSD in an OSS context [LLÅ06].

As can be seen from Table 1, communication, and, as a consequence, mechanisms for coordination and control, are primarily implemented through IT-based solutions and predominantly over the Internet. Typically, the technologies used provide fairly low-level support and include simple configuration management tools such as CVS, mailing lists, and IRC.

3 Actability Principles

To fully support workpractice action and communication, it is important to see IT as a tool that mediates social action. To understand the particular features of IT in such settings, we may draw on the casual face-to-face conversation made by Clark [Cl96]. Clark suggested this model as a benchmark for understanding other communication situations, and by using this setting as an ideal type we can see how the introduction of IT changes the situation. To understand the face-to-face situation, Clark suggests ten typical features: *co-presence* (participants share the same physical environment), *visibility* and *audibility* (they see and hear each other), *instantaneity* (they recognize each other's action at no perceivable delay), *evanescence* (the medium fades immediately), *recordlessness* (actions do not leave any record or artefact), *simultaneity* (participants may receive and produce at once and simultaneously), *extemporaneity* (actions are formulated and executed in real time), *self-determination* (participants determine for themselves what actions to take when), *self-expression* (participants take actions as themselves) [Cl96]. Table 2 summarizes an interpretation of these features from the perspective of actability, as used in the remainder of this paper and defined below. The comparison and interpretation is based on the notion that workpractice communication is a special type of norm-based context (a work context) and that IT is a special type of medium for conversation (speech act exchanges).

The concept of actability was introduced as a way of conceptualizing the use of IT in organizations from a pragmatic point of view. Within actability theory [GÅ02, Åg03, Åg04] IT is viewed primarily as a tool for social (inter-personal) action and communication. This is in line with the language/action perspective which suggests that the real power of computers is to support communication, not computation *per se* [Fl98, Sc01, Di03, Di04]. Actability can be defined as: 'an information system's ability to perform actions, and to permit, promote and facilitate the performance of actions by users, both through the system and based on information from the system, in some work context.'

In essence, actability promotes systems that are 'easy to use' and also explicit about what actions are possible to perform. They should furthermore encourage users to benefit from acting through them so that an organizational action memory can be maintained by the IS. That is, information systems should *permit, promote and facilitate the performance of actions by users*.

Feature	*Face-to-face conversation*	*Workpractice action through IT system*
Co-presence	Participants share the same physical environment	Participants may not share the same physical environment
Visibility	Participants can see each other	Participants may not see each other
Audibility	Participants can hear each other	Participants may not hear each other
Instantaneity	Participants perceive each other's actions at no perceptible delay	Participants may perceive each other's actions with considerable delay
Evanescence	Medium is evanescent – it fades quickly	Medium is persistent – it may stay until the system is shut down
Recordlessness	Participants' actions leave no record or artefact	Participants' actions may leave a record in an "action memory" (e.g. a database)
Simultaneity	Participants can produce and receive at once and simultaneously	Participants either produce or receive as separate acts
Extemporaneity	Participants formulate and execute their actions extemporaneously, in real time	Participants may formulate and execute their actions reflectively during extended amounts of time
Self-determination	Participants determine for themselves what actions to take when	Workpractice norms and system design determine (to a large extent) what actions to take when
Self-expression	Participants take actions as themselves	Participants may take action on behalf of other people and their organization

Table 2: Features of face-to-face conversations [Cl96] as compared to workpractice communication through and by means of IT. (After [Åg04])

Actions are here thought of as 'social actions', i.e. intentional actions that takes into account the behaviour of others [We78], or, more specifically, as speech acts [Au62, Se69] or communicative actions [Ha84], following the language/action perspective. Within actability theory, such actions are referred to as elementary communicative actions (or e-actions), which generates action-elementary messages (or ae-messages) communicated through and by means of the system. Ae-messages are elementary information units carrying a propositional content (what is talked about) and an associated action mode (representing the speaker's intention, or what Searle termed 'illocutionary point' [Se69]). The concept of the ae-message is thus based on the fundamental language/action thesis that language use is not restricted to making descriptions of reality, i.e. to refer and to predicate. Rather, language is often used to perform actions. People do things when speaking, such as promising, ordering and declaring [Au62, Se69]. Someone performs an e-action to effect a social change

(which may or may not have material consequences). This creates an action relationship between the speaker and one or more listeners [Ha84, GÅ02, Åg02].

Saying that information systems have an ability to perform actions implies that they can be seen as agents performing actions on behalf of some human actor. There is always a human actor ultimately responsible for such 'automatic actions', which are always derived from predefined rules. Actions can also be performed *through the system*, such as when a user performs actions with an information system as a tool for communication. Conversely, *based on information from the system* implies that an information system can also be used to create possibilities for action. The actability achieved in a certain situation is always related to a particular *work context* in which the information system is used. The work context includes actors' pre-knowledge and skills regarding both the information system and the work tasks performed. [GÅ02]

To facilitate actability design and evaluation, nine actability dimensions have been suggested along which the pragmatic usefulness of IT can be analysed [Åg04]. These dimensions should be understood in the light of many years of research into the success of information systems from a use perspective [Da89, DM92, GT95, Be99, DM03]. Several lists of criteria focusing on usability and user interface design and its relation to user and task characteristics have been proposed, including Nielsen's ten usability heuristics [Ni94] and Shneiderman's eight golden rules [Sh98]. Such criteria are often grounded in cognitive psychology and tend to employ an overly instrumental view on IT use [ÅE06]. The actability dimensions, on the other hand, concern institutionalized settings in which IT is used as a tool to perform communicative business action. As shown below, each dimension highlights a number of criteria for design and evaluation by means of questions to ask. The actability dimensions and associated criteria/questions are derived from the concept of actability and the set of ten significant features of casual face-to-face conversations introduced above (Table 2). Note that the following description of the nine dimensions is only meant as a brief summary and the reader is referred to [Åg04] for more in-depth elaboration.

The *action elementariness dimension* reflects the notion that information systems are systems for handling messages as (semiotic) results of communicative actions. This dimension addresses questions such as:

- Is it clear who says what to whom with what intentions, and, that this is done on behalf of someone else, if that should be the case?
- Are separate messages kept separate? That is, are users forced to do (or to make sense of) more than one thing at a time?

The *recorded action dimension* reflects the notion that users' actions may leave a record in the action memory of the information system. This dimension addresses questions such as:

- Does the system store and provide access to what has previously been said and done using the system?
- Does the system keep track of who said what to whom?

The *action potentiality dimension* reflects the notion that an information system can be understood as the set of communicative actions it affords and supports. This dimension addresses questions such as:

- Are required actions afforded?

- Are known and understandable effects of possible actions communicated?
- Are expressive interactive user interface components used (icons, labels, *et cetera*)?
- Is information that the system requires from users meaningful and easily provided to the system?
- Is information shown adequate (necessary and sufficient) so that it can be readily used as a basis for action?
- Does the language used correspond to the users' professional language?
- Does the system support justification by explanations, and possibly negotiation, of the action potential and its communicative validity?

The *structured action dimension* reflects the notion that business rules to a large extent determine what actions to take, and when to take them. This dimension addresses questions such as:

- Does the system admit focus and work task changes?
- Is the navigation style made explicit?
- Are sequence restrictions enforced when necessary and desirable, and only then?
- Does the system assist performers in knowing what they are doing, and what they are supposed to be doing?
- Is choice of course of action to take legibly informed by the system?
- Does the system support the following-up of previous commitments made?

The *irrevocable action dimension* reflects the notion that business messages may be formulated and executed reflectively during extended amounts of time. This dimension addresses questions such as:

- Is the system explicit about when a social action is actually performed?
- Is rollback (undo) provided as far as socially acceptable?

The *remote activity dimension* reflects the notion that participants may not share the same physical environment. This dimension addresses questions such as:

- Is the receipt and interpretation of messages possible at desired places?
- Is the receipt and interpretation of messages possible in desired ways?
- Is action potential provided where and when it is needed?

The *delayed interpretation dimension* reflects the notion that participants may perceive each other's actions with considerable delay. This dimension addresses questions such as:

- Can we (always) tell when an (important) action was performed?
- Do messages reach intended interpreters in due time?

The *delayed feedback dimension* reflects the notion that communicating users either produce or receive, but not simultaneously. This dimension addresses questions such as:

- Do users understand that no feedback on communication effects is given until a message has been delivered, interpreted and acted upon?
- Is delayed feedback on communication effects minimized and, if known or anticipated, communicated to users?

The *delegated action dimension* reflects the notion that users and systems may take action on behalf of other people and of their organization. This dimension addresses questions such as:

- Is performance of action allocated to human actors and IT systems so that users gain maximum support?
- Are descriptions and explanations of the system's performed and scheduled future action(s) readily available?
- Are users aware of their action relationships?

4 A Case for Pragmatic Web Research

The actability dimensions were developed and validated primarily in a traditional business information system context. Such a setting means that organizational norms and business rules are, if not well-known at least possible to elicit and document; as is actors and actor roles with associated action responsibilities. When moving into the realm of open source ecosystems, many of these properties do not hold. For example, as in any 'virtual community', the actual people involved and their norms, values, needs, etc are not always possible to identify and explicate beforehand. Similarly, the communication structures and interaction patterns evolve during the course of evolution of the ecosystem. To support GSD efforts in these emerging ecosystems of people and organizations there is a great need for collaborative environments that can adapt to local needs, yet provide facilities for implementing the required control structures. We believe that the concept of actability and its associated dimensions are crucial in such an endeavour. This is because, as described above, actability immediately addresses the challenges of increased distance and provides detailed design guidelines for how to make sure that communication through and by means of IT is supported at both the semantic and the pragmatic level [Åg04, ÅE04, ÅE06]. Also, the pragmatic orientation of actability goes well with a value-based phenomenon such as OSS. Understanding the underlying values (for example, the conflict between business/economic value and 'open' community values) is arguably key for co-opetition to be successful [Fi06]. In a Pragmatic Web context this would leverage the current use of Internet-based technologies in these ecosystems while making sure that new tools support emerging social action contexts properly.

A possible mapping of the actability dimensions to the open source GSD ecosystem context and a number of areas where the different dimensions could be particularly useful are outlined in the remainder of this section (and summarized in Table 3). Indeed, this is an initial tentative mapping and further research is required to fully realize the potential of actability in the Pragmatic Web in this context. Also, as indicated above, the actability dimensions and associated design guidelines need to be tailored to cater for the emergent and dynamic nature of open source ecosystems as compared to traditional in-company information systems.

Process	Dimension		
	Temporal Distance	*Geographical Distance*	*Socio-Cultural Distance*
Communication	*Recorded Action* to make sure that the asynchronous communication through the Internet is captured and maintained at both semantic and pragmatic level.	*Remote Activity* to explore what type of communication solutions best support different ecosystems.	*Action Elementariness* to facilitate bridging of possible socio-cultural mismatch and creation of appropriate pragmatic ontologies to support communication across organizations.
Coordination	Although typical OSS architectures reduce coordination needs, understanding properties of *Structured Action* and *Irrevocable Action* becomes critical in more formalized projects, which appears to becoming the norm.	Given high critical task awareness throughout the community, understanding effects of *Delayed Interpretation* and *Delayed Feedback* is crucial to maintain proper coordination.	*Action Potentiality* and *Delegated Action* to support the proper design of the required common environments and to understand the delegation structures implemented and supported by those.
Control	Although control in OSS is primarily through the commit process, understanding the *Structured Action* and *Irrevocable Action* is crucial in more formalized projects.	*Delegated Action* as a tool to understand how to implement mechanisms for identifying and addressing the issue of non-active key members and other emerging issues related to the agency of the ecosystems.	*Delegated Action* as a way of understanding the agency properties of the ecosystems and to make sure that all agents work towards common goals.

Table 3: GSD in open source ecosystems from an actability perspective.

4.1 The Temporal Distance Dimension

Since communication in open source ecosystems is generally asynchronous over the Internet, the *recorded action* dimension could be useful in designing a communication infrastructure such that relevant aspects of this communication is captured and maintained at both semantic and pragmatic level. Today's communication tools, mailing lists and IRC, for example, are typically not utilizing the potential in a pragmatic approach to structuring information so that previous conversations and commitments are easily traceable and retrievable.

As OSS projects are becoming more formalized, understanding properties of *structured action* and *irrevocable action* becomes critical. Although typical OSS architectures reduce coordination needs, communication support should be aligned with the software development process used. Since OSS projects are typically using flexible and lightweight process infrastructures, there is a need to understand these dimensions in light of fluid and agile action structures. Still, making sure that everyone knows when an action has been performed and how that action relates to other actions (previous and future) is essential. This is probably especially important if coordination of the project relies on the commit process, which is often the case in OSS development. A relevant question to ask is, for example, what does it mean, socially, to commit a changed software artefact to a repository.

4.2 The Geographical Distance Dimension

Each open source ecosystem will have their own particular needs when it comes to supporting action and communication in the software development process. It is becoming increasingly common in OSS development to organize events where developers get together and work collocated for short periods. This should of course be considered in the development of open source ecosystems, but may not be suitable

in every case. The *remote activity* dimension allows for exploration of what type of communication solutions best support different ecosystems.

Understanding effects of *delayed interpretation* and *delayed feedback* is likely to be crucial in maintaining proper coordination in open source ecosystems. Especially since coordination in OSS development often relies on high critical task awareness throughout the community.

The *delegated action* dimension should be explored in order to understand how to implement mechanisms for identifying and addressing the issue of non-active key members and other emerging issues related to the agency of the ecosystems. In general, since agent technology is likely to be of increasing importance, understanding what action responsibilities that are created and maintained throughout the ecosystem is essential.

4.3 The Socio-Cultural Distance Dimension

A key problem in GSD is the possible socio-cultural mismatch between the people involved. Although this is less of a problem in OSS, perhaps because OSS developers are often self-selected and share the same 'itch', the notion of *action elementariness* could help facilitate bridging socio-cultural gaps. One way to do this could be to create appropriate pragmatic ontologies to support communication across organizations and cultures. This way, intentions, understandings and ways of working have to be explicitly agreed upon and codified.

Investigating the *action potentiality* and *delegated action* dimension could support the design of collaborative work environments and help ensuring that these are aligned with the delegation structures implemented and supported in these environments.

Finally, and returning to *delegated action*, understanding the agency properties of the ecosystems and making sure that all agents work towards common goals is probably significant in order to bridge possible socio-cultural differences.

5 Conclusion

This essay has introduced the emerging phenomenon of open source ecosystems and explored how actability and ideas underlying the Pragmatic Web can be useful for supporting global software development in such a context. We have identified a number of areas where actability principles may be useful in this endeavour and also pointed at some issues that may require tailoring of these principles for this new and emerging context. The suggested mapping between actability dimensions and GSD in open source ecosystems (Table 3) can thus be seen as a proposed agenda for research on actability in the Pragmatic Web to support this emerging mode of collaboration (or co-opetition) between competitors.

References

[Åg02] P. J. Ågerfalk. Messages Are Signs of Action: From Langefors to Speech Acts and Beyond, In Barjis J, et al. (eds.), *Proc. 7th Int. Workshop on the Language-Action Perspective on Communication Modelling* (Lap 2002), pp. 81–100, Delft, The Netherlands, 2002. Delft University of Technology.

[Åg03] P. J. Ågerfalk. *Information Systems Actability: Understanding Information Technology as a Tool for Business Action and Communication*, Doctoral dissertation, Department of Computer and Information Science, Linköping University, Linköping, Sweden, 2003.

[Åg04] P. J. Ågerfalk. Investigating Actability Dimensions: A Language/Action Perspective on Criteria for Information Systems Evaluation, *Interacting with Computers*, 16(5): 957–988, 2004.

[ÅDF05] P. J. Ågerfalk, A. Deverell, B. Fitzgerald, L. Morgan. Assessing the Role of Open Source Software in the European Secondary Software Sector: A Voice from Industry, In M. Scotto, G. Succi (eds.), *Proc. 1st Int. Conf. on Open Source Systems*, pp. 82–87, Genoa, Italy, 2005.

[ÅE04] P. J. Ågerfalk, O. Eriksson. Action-Oriented Conceptual Modelling, *European Journal of Information Systems*, 13(1): 80–92, 2004.

[ÅE06] P. J. Ågerfalk, O. Eriksson. Socio-Instrumental Usability: IT Is All About Social Action, *Journal of Information Technology*, 21(1): 24–39, 2006.

[ÅFH05] P. J. Ågerfalk, B. Fitzgerald, H. Holmström, B. Lings, B. Lundell, E. Ó Conchúir. A Framework for Considering Opportunities and Threats in Distributed Software Development, In *Proc. Int. Workshop on Distributed Software Development (DiSD 2005)*, Paris, France, 2005.

[ÅFH06] P. J. Ågerfalk, B. Fitzgerald, H. Holmström, E. Ó Conchúir. Open-Sourcing as Offshore Outsourcing Strategy, In *Proc. 29th Scandinavian Information Systems Research Seminar (IRIS'2006)*, Helsingør, Denmark, 2006.

[Au62] J. L. Austin. *How to Do Things with Words*. Oxford University Press, Cambridge, 1962.

[Be99] N. Bevan. Quality in Use: Meeting User Needs for Quality, *Journal of Systems and Software*, 49(1): 89–96, 1999.

[CA01] E. Carmel, R. Agarwal. Tactical Approaches for Alleviating Distance in Global Software Development, *IEEE Software*, 18(2): 22–29.

[Cl96] H. H. Clark. *Using Language*. Cambridge University Press, Cambridge, 1996.

[Da89] F. D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology, *MIS Quarterly*, 13(3): 319–340, 1989.

[DM92] W. H. DeLone, E. R. McLean. Information Systems Success: The Quest for the Dependent Variable, *Information Systems Research*, 3(1): 60–95, 1992.

[DM03] W. H. DeLone, E. R. McLean. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update, *Journal of Management Information Systems*, 19(4): 9–30, 2003.

[Di03] J. L. G. Dietz. The Atoms, Molecules and Fibers of Organizations, *Data & Knowledge Engineering*, 47(3): 301–325, 2003.

[Di04] J. L. G. Dietz. Towards a LAP-Based Information Paradigm, In M. Aakhus, M. Lind (eds.), *Proc. 9th Int. Working Conf. on the Language-Action Perspective on Communication Modelling (Lap 2004)*, pp. 59–76, Rutgers University, The State University of New Jersey, New Brunswick, NJ, USA.

[DWA03] P. A David, A. Waterman, S. Arora. *FLOSS-US: The Free/Libre/Open Source Software Survey for 2003*. Stanford University, California, USA, Available online at http://www.stanford.edu/group/floss-us/report/, Accessed 2006-08-03.

[Fi06] B. Fitzgerald. The Transformation of Open Source Software, *MIS Quarterly*, 30(3): 587–598, 2006.

[Fl98] F. Flores. Information Technology and the Institution of Identity: Reflections since Understanding Computers and Cognition, *Information Technology & People*, 11(4): 352–372, 1998.

[FL02] *Free/Libre and Open Source Software: Survey and Study*, FLOSS Final Report, International Institute of Infonomics, University of Maastricht, The Netherlands and Berlecon Research GmbH, Berlin, Germany, Available online at http://www.infonomics.nl/FLOSS/report/index.htm, Accessed 2006-08-03.

[GÅ02] G. Goldkuhl, P. J. Ågerfalk. Actability: A Way to Understand Information Systems Pragmatics, In K. Liu et al. (eds.), *Coordination and Communication Using Signs: Studies in Organisational Semiotics 2*, pp. 85–113, Kluwer Academic Publishers, Boston, 2002.

[GT95] D. L. Goodhue, R. L. Thompson. Task-Technology Fit and Individual Performance, *MIS Quarterly*, 19(2), pp. 213–236, 1995.

[Ha84] J. Habermas. *The Theory of Communicative Action*. Polity Press, Cambridge, 1984.

[LLÅ06] B. Lundell, B. Lings, P. J. Ågerfalk, B. Fitzgerald. The Distributed Open Source Software Development Model: Observations on Communication, Coordination and Control. In *Proceedings of ECIS'2006*, Gothenburg, Sweden, 2006

[Ni94] J. Nielsen. Heuristic Evaluation, In J. Nielsen, R. L. Mack (eds.), *Usability Inspection Methods*, pp. 25–64, John Wiley & Sons, New York, NY.

[Ra99] E. S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Sebastopol, CA, 1999

[Sc01] M. Schoop. An Introduction to the Language-Action Perspective, *ACM SIGGROUP Bulletin*, 22(3): 3–8, 2001.

[SMD06] M. Schoop, A. de Moor, J. L. G. Dietz. The Pragmatic Web: A Manifesto, *Communications of the ACM*, 49(5): 75–76, 2006.

[Se69] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.

[Sh98] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd ed. Addison Wesley Longman, Reading, MA, 1998.

[We78] M. Weber. *Economy and Society*. University of California Press, Berkeley, CA, 1978.

10.3 Open-Sourcing in the Celtix Project: A Case of Outsourcing to an Unknown Workforce?

Pär J Ågerfalk, Brian Fitzgerald, Helena Holmström, Eoin Ó Conchúir

Proceedings of 27th International Conference on Information Systems (ICIS2006),
Milwaukee, Wisconsin, USA, December 10-13 2006.

Abstract

This paper presents a psychological contract perspective on the use of open source as an offshore outsourcing strategy – open-sourcing as we term it here. Building on previous research on IS outsourcing, a theoretical framework for understanding commercial software organizations involvement in open source software (OSS) is derived. The framework is used in a qualitative case study involving a commercial organization, the Irish-based global middleware company IONA, as the customer, and representatives from the open source community, as suppliers of services to IONA. The study reveals an ongoing shift from OSS as community of individual developers to OSS as community of commercial organizations, primarily small to medium-sized enterprises. It also reveals that outsourcing to the OSS community provides ample opportunity for companies to headhunt top developers – hence moving from outsourcing to a largely unknown OSS workforce towards recruitment of talented developers from the open source community. In a similar fashion, the process allows the development community to get to know the customer organization better. Overall, the key watchwords for open-sourcing are partnership, building of mutual trust, flexibility, tact and complementariness: The customer and community need to establish a trusted partnership of shared responsibility in building an overall ecosystem to deliver the product. The customer has to be flexible in accepting consensus on the development priorities and the functionality that will be built in. The community must be flexible in affording more transparency into the development process. Also, the complementary skills offered by each stakeholder are key to successfully nurturing the ecosystem.

Keywords: Open source, open-sourcing, offshoring, outsourcing, global software development

Introduction

Companies have been offshoring manufacturing processes to lower-cost destinations for at least thirty years. However, it is really only since the mid 1990's that a significant portion of software development work is being performed offshore (Carmel, 1999). There are many potential advantages to be gained in offshoring software development, including reduced development costs; reduced cycle time arising from 'follow-the-sun' software development; cross-site modularization of development work; access to a larger skilled developer pool; innovation and shared best practice; and closer proximity to customers (Ågerfalk et al., 2005). Of these, most emphasis has been placed on potential cost savings, which accrue largely due to the availability of a trained pool of development staff at a lower salary base. To take an example, the average annual base salary of a software development engineer in India in 2004 was less than one-seventh of the median annual salary of a US software engineer. Such potential savings have rapidly increased the amount of work being offshored from high-cost countries such as the US and UK to lower cost economies such as India, China and Malaysia. The U.N. World Investment Report 2004 predicted that offshore outsourcing of IT-enabled business processes will increase 18-fold to almost $25 billion by 2007 (United Nations, 2004).

Given that the primary force driving offshore outsourcing appears to be cost savings, it is perhaps natural that companies might eventually focus on the open source software (OSS) development model as a potentially even cheaper alternative, as there are significant cost savings associated with the acquisition of OSS (Fitzgerald and Kenny, 2003; Wheeler, 2004). Carmel and Tjia (2005) have characterized offshore outsourcing as 'outsourcing to a global workforce', in this study we investigate the emerging trend towards open-sourcing; that is, outsourcing to a global but largely *unknown* workforce. To do this we adopt the perspective suggested by Koh et al. (2004), using psychological contract theory (PCT) as a basis for understanding the mutual relationships between managers of outsourcing organizations and members of the open source community. Building on Koh et al.'s (2004) results, our research question is thus: *What are the critical customer-community obligations in an open-sourcing relationship?*

Koh et al. (2004) note that most research on offshore outsourcing tends to focus on the customer perspective, and argue for the importance of studying both the customer and supplier side of the relationship. Interestingly, research to date on open source has focused inwards on investigating the characteristics of the development process and projects, that is on the supplier side of the relationship, and far less has been conducted on the customer side, in the sense of investigating the consequences of the OSS phenomenon for organizations, for example.

This study focuses on open-sourcing, as we term it – that is, adopting open source as an offshore outsourcing strategy for the software development process in organizations. One of the novel features of the study is that it considers both sides of the process – that is, the organization which is 'commissioning' the open-sourcing, and, in turn, the OSS community who are doing the development work. The work presented here also builds on and extends the work of Koh et al. (2004) in two significant ways. First, it considers the unexplored concept of open-sourcing, as opposed to 'traditional' outsourcing. Second, Koh et al. (2004) focused on onshore outsourcing while this work considers offshore outsourcing, as implicated by the open-sourcing strategy.

The structure of the paper is as follows: In the next section, we define basic terminology to distinguish the related concepts of outsourcing, offshoring and open-sourcing. We then focus on PCT, explaining why we have selected it for this study, and how it has been used. Following this, our research approach is discussed, and the case study findings presented and analysed. Finally, the overall conclusions are presented in terms of a set of obligations that the customer must bear responsibility for and a set of obligations that the OSS community must bear responsibility for in order for the open-sourcing arrangement to be successful. In the final section we also point at some possible future directions for this research particularly and for OSS research in general.

Basic Terminology

While the terms offshoring and outsourcing are often used almost as synonyms, here we distinguish between the two: Offshoring is about location – when an activity is offshored it is performed in a different location to the main operation (which is then the onshore location). Outsourcing, on the other hand, is about governance – when an activity is outsourced it is performed by another organization, as opposed to 'in-house' by the organization itself. Consequently, the two concepts are orthogonal and any particular activity can thus be performed either offshore or onshore and can be

performed in-house or be outsourced. Figure 1 shows the distinction and relationship between the two concepts.
OSS – open source software – may be defined as software released under the terms of a license which basically allows the licensee to use, modify and redistribute, either gratis or for a fee. Of particular interest in this research, however, is the recent phenomenon of open-sourcing as a software development model. Similar to outsourcing, the OSS development model allows companies to 'subcontract' development activities to another party. Since anyone (in principle) can join any open source project, the development community can be assumed to be global (Millar et al., 2005). Hence, open-sourcing, by definition, falls in the offshore outsourcing category of Figure 1.

	In-house	**Outsourced**
Onshore	In-house (traditional model)	Subcontractor in the same locale
Offshore	Foreign branch of the same company	Subcontractor in a foreign locale

Figure 1: Offshoring versus Outsourcing

A PCT Perspective on Open-Sourcing

An early and influential contribution to psychological contract theory (PCT) was that of Argyris (1960). PCT has since been widely used in studies on employment relationships (Anderson and Chalk, 1998) and has featured in several IS research studies (e.g. Ang and Slaughter, 2001; Piccoli and Ives, 2003; Koh et al., 2004; Raghu et al., 2004; Miranda and Kavan, 2005; Pavlou and Gefen, 2005). A psychological contract represents 'the contractual parties' mental beliefs and expectations about their mutual obligations in a contractual relationship, based on perceived promises of a reciprocal exchange.' (Koh et al., 2004:357) Three aspects of the psychological contract are particularly important in an outsourcing context: First, the importance of mutuality and reciprocity of obligations in a social context. This is distinct from the usual one-sided perspective (customer's *or* supplier's) commonly adopted in outsourcing studies (Koh et al., 2004). Second, psychological contracts are distinct from legal contracts. Specifically they encompass people's beliefs about both written terms and unwritten implicit terms. Third, it promotes an individual level of analysis, focusing on the individuals' beliefs and expectations in a social relationship. This is distinct from the more common inter-organizational level of analysis.
These three issues are also central to the OSS concept. Firstly, mutuality and reciprocity are critical to the success of the OSS development model. These values are effectively enshrined in OSS development through the 'copyleft' terms found in OSS licenses, which decree that software can be used, modified and redistributed provided subsequent modifications are made freely available to others. Also, development is

accomplished through the fulfilment of mutual obligations with respect to the activities of coding, debugging, testing and documentation. One of the most significant threats for the OSS movement has been suggested to be the 'free rider' phenomenon whereby someone profits from OSS without reciprocating, which thus contravenes these values of reciprocity and fulfilment of mutual obligations (Von Hippel and Von Krogh, 2003).

Secondly, in relation to the PCT focus on psychological contracts (which differ from legal ones as they encompass both written and unwritten terms), much OSS development is done in the absence of any legal employment contract for developers. Also, the norms of how development is conducted are both written and unwritten. Developers are expected to be familiar with written rules and coding standards, for example, before they attempt to contribute (Feller and Fitzgerald, 2002). However, the unwritten rules must also be learned by developers over time. New recruits to development ranks serve their apprenticeship in learning these unwritten rules and norms of expected behaviour (Gorman, 2003; Raymond, 1999).

Finally, in relation to the focus on the individual level of analysis, research suggests that OSS developers display more loyalty to the OSS phenomenon than to the organizations where they may be employed (Feller and Fitzgerald, 2002). Also, the signalling incentives identified by Lerner and Tirole (2002) as the basic motivation for developers to contribute to OSS projects (i.e. *career concerns* and *ego gratification*) applies primarily at the level of the individual. Furthermore, since many OSS developers are not affiliated with any formal organization, an individual level of analysis is to some extent required in any study of OSS developers.

In their study, Koh et al. (2004) used an initial qualitative case study approach to derive a set of obligations that customers and suppliers need to fulfil in order to achieve a successful outsourcing relationship. They followed this with a quantitative survey analysis to validate and refine these factors. They recommend that their framework of obligations be applied in a context in which clear and traditional authority structures do not exist. In this study, we drew on their finalized set of validated obligations and sought to apply it in an open source context. In this case, we use the 'customer' entity in the same sense as the Koh et al. study – i.e. as the organization commissioning the open-sourcing, seeking to grow a community around a product. However, the 'supplier' entity becomes the open source community in our study. Below, we consider the obligations and discuss their specific relevance to an OSS context.

Customer Obligations

Koh et al. conclude that there are four specific obligations for which the customer must bear responsibility and which are associated with outsourcing success. These are:

- Explicit and comprehensive requirements specifications for the services covered by the outsourcing project
- Prompt payment to suppliers and no unreasonable withholding of payments
- Close project monitoring with active overseeing of project progress, attending project meetings and regular discussions
- Project ownership to ensure that senior management provides strong leadership, support, and commitment toward the project

Explicit and Comprehensive Requirements Specifications

At first glance, explicit and comprehensive requirements specifications might seem to be at odds with the OSS development model which is predicated on the principle of a developer perceiving 'an itch worth scratching,' to use Raymond's (1999) memorable phrase, and thus not normally associated with comprehensive requirements specifications. Also, OSS developers have typically been users of the software being developed (Dinh-Trong and Bieman, 2004; Gacek and Arief, 2004; Mockus and Herbsleb, 2002), and the software was often targeted to a horizontal domain (such as office desktop applications or software development tools). In such situations, clear requirements specifications are not necessary as these are widely understood and internalized by the individual developers. However, these aspects of the OSS development context are changing. Increasingly OSS development is being purposively 'steered' as customers seek to stimulate OSS development in specific vertical domains (such as automotive or telecoms) where a developer may not perceive an itch worth scratching (Fitzgerald, 2006). In these development situations, the specifications are not part of conventional software development knowledge and thus clear specifications are becoming more important. This increasingly explicit formalization of specifications is already evident in the commercially sponsored projects that are increasingly becoming a feature of the OSS landscape.

Prompt Payment to Suppliers and No Unreasonable Withholding of Payments

Again, prompt payment to suppliers and no unreasonable withholding of payments might seem at odds with the OSS model where actual monetary payment is often not a factor. However, recent studies show that a significant part of OSS developers are indeed employed within professional organizations (Lakhani and Wolf, 2005). Also, the Lerner and Tirole (2002) study illustrates that 'payment' can come in forms other than mere monetary compensation. Many OSS developers report the primary motivation as the rush they get from seeing their code in use and getting prompt feedback from peers they really respect (Feller and Fitzgerald, 2002). This is in marked contrast to the proprietary software development model, where developers may wait months or even years to see their code in use. Thus, there is a sense in which prompt 'payment' can arise, albeit in the form of surrogates such as peer feedback. When payments in the normal sense are not part of the equation, then the issue of unreasonable withholding of (monetary) payments is not likely to arise in the case of OSS.

Close Project Monitoring with Active Overseeing of Project Progress

Raymond's (1999) characterization of the cathedral v. the bazaar to differentiate OSS development from traditional development caused the perception that OSS development was merely about developers following their own agenda developing in parallel in a spirit of optimistic concurrency. However, Raymond's characterization was based on a limited sample of OSS projects which didn't reflect the heterogeneity of the OSS development landscape even at the time. In recent times, OSS development has become more formalized. This is evident in the regular project meetings which are now a feature of a number of popular open source products, such as the Apache conferences in the US and Europe, the Zope/Plone development project

meetings, and the GNOME annual project conferences (German, 2004) which bring together developers to coordinate and plan development.

Project Ownership and Senior Management Leadership and Support

The importance of strong management support has been verified in several studies of ICT adoption (e.g. Agarwal, 2000; Chatterjee et al. 2002; Fichman, 2004; Gallivan, 2001). Project ownership and senior management championship is undoubtedly critical for radical, high-risk initiatives such as OSS deployment since it contravenes the traditional model where ongoing support is legally guaranteed by a vendor. Indeed, top management championship is likely to become even more important in the future as OSS adoption moves out of the domain of invisible infrastructure systems to more visible, high-profile desktop systems and IS applications.

Supplier Obligations

Koh et al. (2004) identify five specific obligations for which the supplier must bear responsibility and which are associated with outsourcing success. These are:

- Clear authority structures which delineate the decision-making rights and reporting structures in the project, in terms of the roles and responsibilities of all parties involved
- Taking charge in terms of completing the job and solving problems independently, with minimal customer involvement
- Effective human capital management in assigning high-quality staff to work on the project, and seeking to minimize staff turnover during the project
- Building effective inter-organizational teams – investing time and effort to foster a good working relationship among the team of customer and supplier staff working on the project
- Effective knowledge transfer in educating the customer in terms of the necessary skills, knowledge, and expertise associated with using the outsourced system or service

Clear Authority Structures

In the absence of traditional organizational sanctions, some form of structure is necessary to coordinate development. In many OSS projects, this takes the form of ‘benevolent dictatorship’ as initially suggested by Raymond (1999). Several studies of OSS development have detailed the complex authority structure that evolves over time to ensure that all code contributions are vetted and incorporated in a disciplined fashion (Mockus et al, 2002).

Taking Charge and Solving Problems Independently

OSS development has typically been characterized by the developers proactively taking charge, solving problems independently with minimal customer involvement. Initially, OSS developers did not engage in formal requirements analysis with customers (Scacchi, 2002), but took direct responsibility for development decisions. Even though the OSS development process is becoming more formalized (Fitzgerald, 2006), OSS developers are still more likely to retain a strong sense of independence.

Effective Human Capital Management

Research has also focused on the human capital management aspects of OSS (e.g. Hann et al, 2002). This suggests that participation in OSS projects allows developers to gain highly marketable technical skills which in turn can lead to higher earnings in

the future, which is also facilitated by the fact that the opportunity cost of participating in OSS development can be quite low as developers can choose the amount of work they do and organize it to fit their own personal timescale and agenda. Also, OSS developers have long been acknowledged as of high quality (Raymond, 1999), and the loyalty of developers in the longevity of their commitment to their development projects has been remarkable (Feller and Fitzgerald, 2002). Indeed, the cardinal sin of OSS, that of project forking (whereby a project is divided in two or more streams, each evolving the product in a different direction), is a strong community norm which acts against developer turnover on projects.

Building Effective Inter-Organizational Teams

The particular characteristics of OSS position it as a good exemplar of the 'whole-product' concept of a market-driven business approach that seeks to deliver a complete solution to the customer in terms of products and services (Moore, 1999). In this scenario, developers do the coding while others complete the business model by adding sales and marketing services – necessary activities but ones in which developers may not be interested. The OSS 'whole-product' approach is also larger than a single company or software product or service. Indeed, the network benefits of open source arise as a result of the size of the overall community and ecosystem. Thus, an inter-organizational network of interested parties with complementary capabilities can form an ecosystem to offer a professional product and service in an agile, bazaar-friendly manner. Customer service requests can be routed to the most appropriate expert partner in the network, perhaps even to the developer who wrote the actual code. Also, the problem of questionable long-term product support often associated with OSS (Fitzgerald and Kenny, 2003) can be minimized if a sustainable ecosystem is created and properly maintained.

Effective Knowledge Transfer in Educating the Customer

Again, this is a topic that resonates well in the case of OSS on a number of aspects. In the cases where the developer is also the user/customer, the issue of knowledge transfer does not arise. However, the role of the user/customer is significantly elaborated in OSS as they can contribute to debugging, testing, documentation, etc (Feller and Fitzgerald, 2002). Thus, a close working relationship can emerge between developer and user.

Summary of Obligations

As illustrated above, the obligations identified and verified by Koh et al. (2004) map well to both the customer and the community in an OSS context. The manner in which they have been refined initially to suit an OSS context is summarized in Table 1.

Table 1. Initial Proposed Customer and Community Obligations in an OSS Context
Customer obligations (i.e. obligations for which the customer must bear responsibility): (1) Explicit and comprehensive requirements specifications for the services covered by the outsourcing project. *Although initially, requirements specifications were not part of the OSS landscape, this appears increasingly to be the case.* (2) Prompt feedback to supplier community. *Although payment in the monetary sense is not always a factor in OSS development, prompt feedback and recognition by peer developers and users is important, and promote further development.* (3) Close project monitoring with active overseeing of project progress, attending project meetings and regular discussions. *Again, project monitoring is increasingly a part of the OSS development process as it becomes more commercially focused.* (4) Project ownership to ensure that senior management provides strong leadership, support, and commitment toward the project. *Given the high risk, radical initiative that OSS deployment represents, strong project ownership and management championship is critical.*
OSS Community obligations (i.e. obligations for which the community must bear responsibility): (1) Clear authority structures which delineate the decision-making rights and reporting structures in the project. *Given the absence of normal organizational authority, a form of 'benevolent dictatorship' and meritocracy in OSS projects seem to be necessary.* (2) Taking charge in terms of completing the job and solving problems independently, with minimal customer involvement. *OSS development has traditionally been characterized by developer independence and prompt problem solving, although customer involvement in terms of user feedback has been a marked feature.* (3) Effective human capital management in assigning high-quality staff to work on the project, and seeking to minimize staff turnover during the project. *OSS developers are acknowledged to be high quality, and exhibit strong loyalty to projects due in part to avoidance of project forking and the freedom to choose what development tasks to work on.* (4) Building effective inter-organizational teams – investing time and effort to foster a good working relationship in the customer and community project teams. *Community networks of OSS companies are becoming a common mode of delivering 'whole product' OSS offerings to customers.* (5) Effective knowledge transfer in educating the customer in the skills, knowledge, and expertise associated with using the outsourced system or service. *The user/developer relationship is very close in OSS thus facilitating knowledge transfer.*

Research Approach

Much of the research on OSS to date has focused inward on the phenomenon itself, studying the motivations of individual developers to contribute to OSS projects, or investigating the characteristics of specific OSS products and projects, for example. Far less has been done in looking outward at the organizational use and leverage of OSS in practice. Given this emphasis and the relative newness of the open-sourcing concept, it is unsurprising that there is not a solid research base to date on this phenomenon. Bearing this in mind, this study was concerned with initially achieving an increased understanding of this phenomenon. Thus, an interpretivist approach which sought to inductively develop a richer understanding based on analysis of a single case was deemed appropriate, as such 'revelatory cases' (Yin, 1994) may provide the required rich insight. The case selected for this study was the Celtix project, an open source Java Enterprise Service Bus (ESB) sponsored by IONA Technologies.

IONA Technologies – The Celtix Project

IONA Technologies, a NASDAQ-quoted company, is headquartered in Dublin, Ireland, with U.S. headquarters in Waltham, Massachusetts and offices worldwide. IONA was founded as a campus company at Trinity College Dublin in 1991, and provides products and services to help organizations build B2B enterprise portals. IONA is currently rated as the leading provider of standards-based platform middleware technology, with more than 4,500 blue-chip enterprise customers worldwide, who use IONA products to address large, complex application integration and achieve interoperability by means of a standards-based, service-oriented architecture. In June 2005, Iona extended its business model to incorporate open source by leading a community project to develop Celtix, an open source Java ESB that will co-exist with Artix, the company's flagship integration product. The Celtix project is hosted by an established open source community, ObjectWeb, who specialize in developing open source middleware products. Most of ObjectWeb's members are small to medium-sized enterprises based in continental Europe. The Celtix project has achieved an impressive development productivity schedule, proceeding through four significant development milestones, a beta release to a fully stable 1.0 release in just over 10 months.

Data Collection and Analysis

Data was gathered over an 11-month period from July 2005 to May 2006, and a number of sources were drawn on (see Table 2). These ranged from workshops, to a series of interviews, both formal face-to-face and informal telephone interviews. An interview protocol guide was developed based on the obligations identified above. Marshall and Rossman (1989) identify the importance of being able to gain entry to a company and maintain continuity of presence for as long as necessary. We sought to achieve this by conducting initial interviews with the Chief Scientist at IONA (the 'customer' in our study) and the Chairman of ObjectWeb (the supplier 'community'). These interviews served to give a good strategic overview of the project and the high level obligations that were in place. Both these individuals initially identified key figures in the project and facilitated access to these interviewees. Following this, as other key informants emerged during the interview process, support from leadership in both the customer and community entities greatly facilitated achieving access. Most studies of open source developers up to now have relied on anonymous surveys (e.g. Ghosh, 2005; Lakhani and Wolf, 2005), the studies by Hann et al. (2002) and Mockus et al. (2002) being notable exceptions. This is caused in part by the difficulty in getting personal access to key developers, but this study was notable in achieving such access.

The interview guide approach, as proposed by Patton (1990), was used to conduct the interviews. This approach was chosen as it is more comprehensive and systematic for data collection than purely the conversational interview, and more flexible than the standardized open-ended interview or the closed, fixed response interview. The duration of interviews ranged between 30 minutes and 90 minutes. Interviews were recorded so as to minimize data loss due to note-taking, and these recordings were subsequently coded. An interview protocol guide was prepared, both to act as an *aide memoire* during interviews, and also to act as a backup if interviewees were unwilling to be recorded. This was emailed to interviewees in advance to allow them an insight into the overall issues we wished to focus on. Informal follow-up interviews took place to clarify and refine issues that emerged following the interview transcription process. These interviews were complemented by comprehensive reviews of documents and communications on the mailing lists, project wiki and web sites. Also,

the project findings were presented to the participants and other researchers over a series of workshops with in-depth discussions feeding back into the analysis process.

Table 2. Data Sources

Workshops	Interviews	Supplementary Sources
Sep 2005: Presentation and discussion of Celtix business model and strategy. Apr 2006: Workshop presentation on open-sourcing strategy. July 2006: Debriefing presentation of findings.	July 05 – May 2006: Interviews with o Chief Scientist, IONA o Chairman, ObjectWeb o Admin, IONA o Open Source Program Director, IONA o Two Project Managers, IONA o Two Developers, ObjectWeb	IONA and ObjectWeb maintain detailed and comprehensive web portals for the Celtix project. We also had access to mailing lists and project development wiki pages.

Qualitative analysis was undertaken using coding techniques proposed by Strauss and Corbin (1998), and exemplified by the research of Baskerville and Pries-Heje (1999). This approach recognizes that social phenomena are complex, and seeks to develop theory systematically in an intimate relationship with the data. This form of analysis facilitates the development of substantive theory without prior hypotheses (Baskerville and Pries-Heje, 1999) and can be utilized in the absence of, or in conjunction with, existing theory (Strauss and Corbin, 1998). In line with Patton's (1990) recommendations, interviews were transcribed and then coded according to the constructs represented by the customer and community obligations derived earlier, and analytical memos were written as patterns and themes emerged from these field notes. This was combined with an open-ended analysis in which we sough to break free from the initial obligations to allow for new categories to emerge. As a result, our revised set of obligations, presented below, deviate but yet incorporate constructs from the initially proposed set of obligations. The analytical memos provide a comprehensive grounding of these revised obligations in both existing theory (OSS and outsourcing) and in our gathered empirical data.

A problem that has been identified in relation to qualitative research is what is termed multiple realities. This refers to the unavoidable fact that the understanding of reality is based on an individual interpretation of the data, and that different individuals may interpret the same data in different ways (Kaplan and Duchon, 1988). This problem was addressed in a number of ways. Firstly, our approach to data analysis explicitly recognizes this problem of subjective data interpretation, and to address it, uses rigorous coding and memoing processes which provide a traceable, documented justification of the process by which research conclusions were reached, thereby providing an audit trail of the process (Guba, 1981). Secondly, the method of venting was used. This is a process whereby results and interpretations are discussed with professional colleagues (Goetz and LeCompte, 1984). The findings were formally presented and discussed with colleagues in detail on several occasions at practitioner/researcher workshops and conferences. Also, in this study, IONA and ObjectWeb representatives were active participants in an EU-funded research project led by the authors. Thus, as findings were presented and discussed at the project

workshops, quite detailed member-checking of our interpretation of the findings was possible.

Research Findings and Discussion

Here we discuss the obligations raised by the interviewees. In our approach, we asked both customer and community interviewees to discuss their perceptions of their own obligations, and also the obligations they would expect from each other. Although the interviews were based on the distinct obligations identified above, it quickly emerged that many of these obligations were symmetrical. It was difficult to allocate companies or individuals to a purely customer role, or purely development community role; rather it emerged that both were cooperating on ecosystem development. Here we present the refined set of obligations based on the interview evidence.

Revised Customer Obligations

Achieving Consensus on Development Roadmap

Initially we expected that the customer would provide an explicit requirements specification of required functionality, which would be in keeping with the evolution of open source towards more formalized commercial phenomenon (Fitzgerald, 2006). However, this was not in fact the case. Interviewees stressed the manner in which requirements specification here differed from traditional outsourcing. Companies may have a clear idea of what functionality they would like to see the community adding to the product. However, there has to be consensus as to what functionality will be added. If a company pushes its own agenda too much in driving the development agenda, there can be problems. The Celtix project manager within IONA expressed it well:

> "A company cannot just go onto the mailing list or the community, and say 'Can you guys build this.' When kicking off the project in the open source community, it's about stating the overall goal and the top-level requirements you are trying to achieve. Then it's driven by consensus. If people perceive you as driving your own agenda, then you will get pushback on having things accepted."

This again emphasizes the delicate equilibrium that must be maintained between acceptable community values and customer desire for value creation (Fitzgerald, 2006). Interestingly, it was stressed that within the ecosystem formed around this mode of development, it was quite permissible for customers to engage in more traditional outsourcing relationship directly with some developers in the community, outside the strict remit of the open-sourcing project.

Project Ownership

The customer interviewees identified with the project ownership obligation. One customer interviewee stressed the radical change in mindset represented by open-sourcing, suggesting that it represented a strategic initiative which differed from the normal business model where developers could see that their salary was pretty much directly derived from the sales of the commercial product that they developed. In the open-sourcing model, their work could appear to be benefiting the open source

community, and not leading to an obvious direct remuneration. Thus, as it was more a strategic initiative with a different business model, top management championship was necessary. However, the Celtix project also grows the market for IONA, enabling additional support contract revenue.
In keeping with demonstrating strong ownership and commitment to the project, IONA established a full time position – Open Source Program Director. This person is responsible for engaging with the community, and helps ensure that issues relevant to the open source community receive prompt attention. Interestingly, however, there is a delicate equilibrium to be maintained here also. The Celtix project manager at IONA suggested that if the project is seen as too much an IONA project, the developer community may have less interest in getting involved.

Marketing Project to Increase Visibility

Several interviewees emphasized the need for the company to market the attractiveness of the project and improve its visibility. This has a two-fold purpose. Firstly, the development community will perceive their reputation has been improved through involvement in a high-profile project with a high profile company. As one community interviewee expressed it: "working with them [the company] is almost a sort of professional honour, as it were".
This helps ensure the vibrancy of the project, and can attract further developers to the community, which cannot be taken for granted in an open source project. After all, OSS is an emergent phenomenon and very few projects to date have been deliberately started and nurtured to be successful; rather some extremely successful ones have emerged over time, whereas others have died off. Again, the Celtix project manager offered an interesting insight:

> "There are a lot of open source projects which don't go anywhere, even though they have built good code. It also needs to be pushed so that it gets noticed and used by other projects, documented and marketed. This is a big overhead, and vendors have structures in place to help achieve that."

Interviewees also suggested that the company could dare on their expertise in relation to R&D, software commercialization and productization in creating a professional OSS product and subsequently marketing that product. This would involve a holistic approach and proactive marketing to ensure that all who could usefully consume the product were made aware of it, and could contribute. It was felt that a commercial vendor could usefully complement the OSS community by providing this expertise. These efforts grow public awareness of the product which can then create business consulting opportunities for members of the development community.

Transparency and Close Project Monitoring

Given that there is a strong external element in open-sourcing, customer interviewees stressed the necessity for clear project milestones and more visibility about product releases. This was contrasted with traditional proprietary development where internal milestones and actual release times are perhaps deliberately kept vague (think of the Microsoft Vista and Longhorn projects!). Also, the frequency of product releases was identified as a by-product of the open source approach.
Also, it was suggested that the customer could not insist on a particular project monitoring regime. Rather, different open source communities may have different

norms and approaches in this area, and the customer has to be flexible and prepared to adapt to the particular regime in vogue in the particular open source community.
The OSS community interviewees also stressed the importance of clarifying the governance of the project. Open source development is usually characterized by a clear project authority structure based on a meritocracy. Thus, it is hardly surprising that the community would expect this of the customer in leading the project, in that it does not depart from the usual norms for OSS development.
Related to transparency there was also a community expectation that the company have an open process for accepting community contributions. Interestingly, the company can have unrealistic expectations as to the level and significance of contributions.

Creating a Sustainable Ecosystem – Customer Responsibilities

It is important that both company and the community members strive towards the creation of a sustainable ecosystem around the product. A vital factor here is to create an atmosphere of trust. One important aspect of this has to do with licensing and Intellectual Property (IP). A senior community interviewee with a background in commercial development suggested that it was sensible and pragmatic to have a clear IP policy. They had requested that IONA release the Celtix project under the Lesser General Public License (LGPL) and IONA agreed. The Celtix project manager suggested that IONA were keen to embrace open source and build trust within the community, and the choice of license is a key determinant for developers in deciding whether to participate in a particular OSS project, and also for companies to adopt. Thus, the development community tend to have a preference for a restrictive (so-called viral or reciprocal) GPL-style license which safeguards their contributions, adhering to the 'copyleft' principle discussed above. However, companies may tend towards more permissive licenses which afford greater control over the future of the software. IONA had perceived the need to be even more open to other companies and communities, and hence dual licensed Celtix under the less restrictive Eclipse Public License also.
Creating an ecosystem and engendering trust is also facilitated by the fact that the project interaction tended to be "very much techie to techie" as one interviewee put it. The project management committee is chaired by a Distinguished Engineer at IONA who would garner respect from the technical OSS development community.
There was broad agreement from the community interviewees on this issue also. However, one community interviewee stressed the importance of meaningful content in the feedback. While promptness was appreciated, this evaporated if the content of the feedback was "empty". However, the "techie to techie" nature of the relationship helped ensure that feedback was meaningful.
Another issue with significant implications for project management practices within the customer company emerged in relation to development staff rotation. Normally, within proprietary software development companies, development staff are rotated after a few months onto other projects. However, because of the strong techie-to-techie interaction between developers in the company and community, there was a strong pressure from the community that developers would remain on the project for a lengthy period. This is matched by the company expectation that community developers show strong loyalty and commitment to the project. However, in the open source community, long-term commitment and loyalty to open source projects is an established feature (Feller and Fitzgerald, 2002).

Community Obligations

We now turn towards the open source community obligations. Again, the five that were initially derived were used to drive the interviews and below is the refined set of community obligations that transpired from the interviews.

Clear and Democratic Authority Structure and Process Transparency

Community representatives identified clear and transparent authority structures as important – indeed, one stressed that these are "not only important, but mandatory". It was argued that since more and more professional people are involved in OSS, the community is expected to show the same level of quality and transparency as could be expected from any professional organization – people need to understand how decisions are made.

The customer interviewees also agreed that clear authority structures are important. However, in open-sourcing, authority structures are framed by a strong belief in democratic principles:

> "It would be good to ensure that that the [democratic] process is working, but I'm not sure that it is possible to see any authority structures other than that. It will always be shared responsibility." – Distinguished Engineer, IONA.

It was furthermore pointed out that such structures are important in two different respects. Firstly, they provide consistency between projects, which means that developers can easily contribute to more than one project. Contributing to several projects is not uncommon in OSS (Feller and Fitzgerald, 2002), and with the increasing interest in the so-called "whole product approach" (Fitzgerald 2006), this is expected to be increasingly important, as pointed out by the Open Source Program Director at IONA. Secondly, they provide for consistent terminology within and across projects which makes sure people are "on the same page, and really focus on innovation".

As a complement to clear authority structures, the lack of a traditional written outsourcing contract means that an open source community must be clear also about their actual work processes. Interestingly, this mirrors the "transparency – close project monitoring" obligation that is expected on the part of the customer.

Responsible and Innovative Attitude

Although community and customer interviewees alike believed it to be essential that the community takes responsibility and delivers on what has been committed to, this outsourcing obligation becomes somewhat blurred in the open-sourcing context. Since part of the development community in our study are paid IONA employees, the customer does have the power to manage part of the development effort more directly than would be possible in a traditional offshore outsourcing context. As explained by an IONA project manager, there currently seems to be a feeling that "there will always be customer involvement". As the open-sourcing phenomenon matures, however, "there will be a lot more of developer independence."

Interestingly, creativity and innovation is stimulated by multi-disciplinary teams operating outside conventional organization structures. Previous studies of OSS have shown that about 40% of OSS developers are employed within professional organizations, suggesting an 'open' community of about 60% (Lakhani and Wolf, 2005). This 60% may provide the creative and innovative spark as OSS loses its

image of being merely about imitation of proprietary products and innovation becomes the defining feature. This obligation also supports the customer obligation of "achievement consensus on development roadmap", which assumes that the OSS community is innovative and contributes constructively to the development roadmap. As this will have a positive impact on the project, the community is expected to help achieving positive impact among (the customer's) customers.

Creating a Sustainable Ecosystem – Community Responsibilities

As mentioned above, both company and community members are expected to aim for a sustainable ecosystem. Consequently, community members are expected to be loyal and committed to continued involvement in the project, which, as mentioned above, is a marked feature of OSS projects. From a community perspective, it was suggested that the high-quality software associated with the OSS model is an indication that the 'human capital management' is working. It was also perceived that the quality of the code is a way to attract more business, which is essential for the ecosystem to develop. As OSS is moving away from networks of individuals to networks of companies, if a contributor earns a reputation for producing high-quality code, customers will keep coming back for more. It is also the case that customers sometimes use the OSS model to identify the best suppliers, who are then approached directly and contracted in a 'traditional' outsourcing model.

IONA, as the open-sourcing customer, expects the OSS model to attract "high calibre people" who understand the project domain very well without requiring additional training. The Open Source Program Director even argued that it attracts a certain personality, with traits not necessarily those traditionally associated with a "top-notch programmer". In her view, people are attracted by the OSS model because they want to "build something better", they want to "get involved", and they want to "be part of a community" – in summary, "these are the kind of people that I would want on my team, whether I was doing open source or not."

Interestingly, OSS community members do not necessarily see themselves as suppliers commissioned by a customer in the traditional sense. Rather, customer and community are seen as "part of the same ecosystem". In fact, Object Web sees its members not as OSS developers but as "ecosystem developers". There seems to be a definite trend towards more organized open source communities, such as that of Object Web. Hence, building effective inter-organizational teams is to a large extent what open-sourcing is all about. As indicated above, part of this trend is the merging of customer and community into one ecosystem: "I don't consider IONA as a customer. Iona is a member" was how the situation was described by the Chairman Object Web. Open-sourcing is thus not just about building good working relationships between customer and supplier. It is about "ecosystem development". Hence, although "everybody knows there are business reasons why people are there", there is a lot more collaboration than in traditional outsourcing:

> "In a traditional market you don't call up your competitor and be like, oh, well tell me what your stuff does. But in open source you do." – Open Source Program Director, IONA.

In open-sourcing, the software developed is typically not aimed for end-users but is more likely to be tools and infrastructure components. Consequently, the customer and community participants typically share the same level of technical expertise – "it is mostly developer-to-developer communication." Therefore, there is no need for

formal training. Instead, knowledge transfer is happening continuously "from one research lab to another" within the ecosystem. This was emphasized by the Object Web Chairman who asserted that "I don't speak about education or anything like that, I speak about exchange between researchers". This view was acknowledged by the Open Source Program Director at IONA who referred to it as "cross pollination". According to a Project Manager at IONA knowledge transfer was also facilitated by an early and proactive focus on documentation as part of the work towards achieving consensus on a development roadmap discussed above (a customer obligation).

Conclusion

Table 3 summarizes the refined list of obligations in the context of open-sourcing. Overall the key watchwords for open-sourcing customers are flexibility, tact, partnership, building of mutual trust, and complementariness. The customer must be prepared to compromise at all stages: For example, rather than just providing a requirements specification for desired functionality, the process requires that development priorities be consensually agreed upon by the customer and community. Also, the open-sourcing customer has to provide complementary expertise in relation to product commercialization and marketing. Hence we have modified the obligations to reflect these issues. Furthermore, the customer must reach consensus on the project monitoring system that will be instituted, on trying to show leadership and ownership of the project but not so strongly as to deter the development community. Overall, the customer must achieve that delicate equilibrium between value creation in creating a successful business model for itself while not transgressing the community values which seek to benefit the overall community. Also, the standard practices that tend to apply in the customer company may need to change. For example, more clarity is required in relation to release milestones, also more frequent product releases are likely rather than artificially separating functionality on the basis of the price that can be charged. Furthermore, the policy in companies of rotating developers onto different projects after a period of several months may not be sustainable as developers become associated with the project in the OSS community.

Table 3. Summary of Refined Customer and OSS Community Obligations in an Open-Sourcing Context

Customer obligations (i.e. obligations for which the customer must bear responsibility):

(1) Achieving consensus on development roadmap. *Holistic approach to requirements specification whereby high-level requirements are identified and consensus achieved with the community on implementation of these requirements.*

(2) Project ownership. *Given the high risk, radical initiative that OSS deployment represents, strong project ownership and management championship is necessary. However, a delicate equilibrium must be maintained as the development community may be reluctant to become involved if it is perceived as too much a customer-led initiative.*

(3) Marketing project to increase visibility. *Holistic approach to product commercialization and marketing whereby the customer provides overall marketing to attract other developers and adopters to ensure a vibrant and successful project.*

(4) Transparency and close project monitoring. *Given the external and public nature of OSS, close project monitoring is vital. However, the customer has to be prepared to adapt to the particular project monitoring regime in use in the OSS community and to make their processes transparent to the community.*

(5) Creating a sustainable ecosystem. *Given the "techie-to-techie" nature of interaction, trust building – vital to the creation of a sustainable ecosystem – is greatly facilitated. However, the company must match the community member loyalty by making sure staff remain on the project. Also, the importance of licensing in open source and patents and IP in proprietary software companies come to the fore and must be resolved to the satisfaction of both customer and open source community.*

Community obligations (i.e. obligations for which the community must bear responsibility):

(1) Clear and democratic authority structure and process transparency. *Given the lack of a written contract, this was seen as important, and also facilitated by the increased involvement of traditional professionals in open-sourcing. Democracy and shared responsibility are also emphasized.*

(2) Responsible and innovative attitude. *Again, the initial characterization that the community would solve problems independently became blurred as it is very much a case of collective responsibility with much input from the customer side. However, the OSS community has to deliver according to its capabilities and is also expected to be innovative and help achieving positive impact among (the customer's) customers.*

(3) Creating a sustainable ecosystem. *Community networks of OSS companies operating in an overall ecosystem in a spirit of co-opetition are becoming a common mode of delivering 'whole product' OSS offerings to customers. The model allows customers to identify high quality developers who may be employed by the customer. The community, comprised of loyal and motivate developers, exercises a strong pressure on the customer not to rotate its own developers across projects. Given the 'techie-to-techie' nature of the interaction, knowledge transfer is greatly facilitated.*

Although an OSS community indeed differ from a traditional outsourcing partner, a large part of the supplier (here community) obligations seem to apply also in an open-sourcing context. Particularly, clear authority structures that make the democratic decision making process transparent are vital. In general, the OSS development model is assumed to attract high-quality developers, and there is evidence that this is also the case. Given the community spirit associated with OSS development, it is not surprising that the OSS community is expected to contribute actively to the creation of an ecosystem manifested in deep collaboration with the customer. Open-sourcing means not just commissioning to an OSS community but also to contribute back to that community. The study thus reveals an ongoing shift from OSS as community of individual developers to OSS as community of commercial organizations, primarily small and medium-sized enterprises, operating as a symbiotic ecosystem in a spirit of

co-opetition. This also means that ongoing knowledge transfer is greatly facilitated, partly due to the 'techie-to-techie' nature of collaboration.
At the beginning of the studied project, the open source community represented an 'unknown' to the customer company, and indeed vice-versa, as IONA were quite 'unknown' to the community, and needed to achieve a position of being trusted by the open source community as capable of successfully sponsoring an open source project. The study also reveals that outsourcing to the OSS community provides ample opportunity for companies to headhunt top developers, whereby community members become also employees of the customer – hence moving from outsourcing to a largely unknown OSS workforce towards recruitment of talented developers from the open source community. To paraphrase former US Secretary of Defense, Donald Rumsfeld's torturing of the Johari Window – there are known knowns, known unknowns, and unknown unknowns. In this study we see a move from 'unknown unknowns' as neither the customer nor the community are known to each other, to a scenario of 'known knowns' as each gets to understand each other's position and build complementary skills. Effectively, this means that the open-sourcing model is not purely about offshore outsourcing, but to some extent also incorporates in-house offshore development, following the definitions introduced above (Figure 1). Essentially, the open-sourcing model consists of a company (as open-sourcing customer) and a community of individual developers and other companies with who the customer interacts. This community provides for cost savings, but also innovation potential, recruitment opportunities and new business. Interestingly, while the community is expected to behave professionally and provide transparent authority structures (thus reducing the 'unknown factor'), innovation potential is primarily to be expected from the 'unknown' part of the workforce. Hence, there are forces in the ecosystem pulling in opposite directions – cost-savings and innovation which tend towards a large 'unknown', while recruitment and trust building tend to reduce this.
Due to the newness of the open-sourcing phenomenon, this study is clearly exploratory in nature and the revised list of obligations is still tentative. More research is needed to further explore these obligations and to establish their validity and potential generalizability. A possible way forward would be to follow Koh et al.'s (2004) example and adopt quantitative research methods to this end. Before that, however, qualitative insights into additional projects would be useful in order to enrich our understanding of the open-sourcing phenomenon in a meaningful way. In any case, we believe this paper represents an innovative and important step in OSS research. As mentioned above, much OSS research has focused inwards on the OSS phenomenon itself and far less has been done on the organizational implications of OSS, and particularly on company-led OSS projects. In fact, it seems that the huge amount of freely available OSS project data on SourceForge and other project portals has led researchers to focus on easily extractable data rather than on seeking a deeper understanding through in-depth analysis of particularly important projects (again, there are of course exceptions). We hope that this paper will stimulate interest in this area so that a better understanding can be reached, not only of the OSS development model as such, but also of the business implications of that model. Particularly, more research that compares and contrasts established wisdom in traditional business settings with their open source counterparts could further advance this important research area.

10.4 Towards A Model of Governance in Open Source Service Networks: An Exploration of Social Mechanisms

J. Feller[1], P Finnegan[1], B. Fitzgerald[2], J. Hayes[1]
[1]University College Cork, Ireland
[2]Lero, University of Limerick, Ireland

Abstract

This research-in-progress investigates the governance of networks of Open Source Software (OSS) firms, which we are calling Open Source Service Networks (OSSN). These socio-digital business communities are an under-researched, but growing, phenomenon as small OSS firms collaborate to compete with larger firms. The paper illustrates that social mechanisms are a viable alternative to formal/legal inter-firm agreements under the exchange conditions that exist in OSSN. It develops a preliminary governance model and hypotheses from a qualitative analysis of one OSSN, and concludes with a plan for refining the model using a quantitative analysis of two additional OSSNs.

Keywords: Open Source Service Networks, network governance, social mechanisms

Introduction

Open Source Software (OSS) potentially enables third-party service providers to engage in level-field competition with the software creators by leveraging the open nature of the source code; thus lowering barriers to entry (Woods and Guliani, 2005), leading to the proliferation of technology-driven OSS microfirms offering specialized products and services. However, such firms cannot always meet customer requirements for end-to-end system implementation and support. Thus, as with other sectors, these firms form cooperative business networks in order to compete. These networks more closely resemble OSS development communities (characterized by IT-mediated communication and informal arrangements) than business networks in other sectors (characterized by legal/formal agreements).

This paper examines the role of social mechanisms in the governance of networks of OSS firms. The next section examines the theoretical foundation for the study. This is followed by a consideration of the research methodology; a two-phased approach combining qualitative and quantitative approaches. A governance model based on the qualitative analysis is presented. The paper concludes by outlining the work-in-progress on the quantitative analysis.

Theoretical Grounding

Our analysis of 317 peer-reviewed OSS research papers (published 1998-2005) revealed that they all studied one or more of four basic types of OSS communities (see Table 1). We found that while OSS has been investigated from a variety of disciplinary and theoretical perspectives, the dominant research theme has been socio-cultural analysis and treats OSS communities as socio-digital networks; social networks whose interactions are predominantly mediated through ICT (e.g. Berquist and Ljungberg, 2001; Gallivan, 2004; Sagers, 2004).

Research on commercial organizations (Table 1) is dominated by studies of OSS startups (e.g. Redhat) and corporate-sponsored projects (e.g. Sun/OpenOffice.org and Netscape/Mozilla.org), limiting our understanding of the variety of OSS business

models that have emerged. OSS products and services have come to closely resemble other complex offerings. Huang (2001) argues that individual organisations do not have sufficient competencies to deal with all aspects of such offerings. Additionally, Stafford (2002) argues that organisations must align themselves in IT-mediated partner networks in order to meet customer requirements in a rapidly changing market. The research literature has neglected such networks in the OSS field despite the fact that they are observable in practice.

Table 1. Socio-digital Community Types

	***Ad Hoc* Communities**	**Standardized Communities**	**Organized Communities**	**Commercial Organizations**
General Structure	Informal communities collaborating "in the wild" on small projects.	Larger projects with more formalized development and management standards.	Standardised communities with a non profit legal identity.	Standardized communities embedded in profit-seeking firms.
Goals	Driven by individual goals (Reputation, learning, etc.).	Individual goals plus quality assurance, project management, standardisation, etc. towards the overall goal of building a public good.	As standardized communities plus the need to provide legal protection for contributors and engage in with other organizations.	As organised communities plus the desire to generate share-holder value.
Digital Environment (Communication and Collaboration Tools)	Generally hosted by a third-party (e.g. SourceForge).	Self- or third-party hosted tools plus group identity web site.	Self-hosted tools plus organizational website.	As organized communities plus explicit integration with corporate development, communication and management structures.

We have identified one type of OSS business network, which represents what Clemons and Row (1992) term a "move-to-the middle"; networks of organisations that interact in order to deliver value to the end consumer. We are calling such networks 'Open Source Service Networks (OSSNs)'; defined as "networks of OSS service providers (usually third-party SMEs and Micro-firms) whose primary purpose is to enhance the ability of member firms to deliver business value in competition with larger firms and with other networks."

OSSNs typically seek to meet what Woods and Guliani (2005) describe as the challenge of "productizing" open source software by offering support, implementation, modification and related services. This is referred to as the 'whole product' by Moore (1999). Meeting this challenge is often beyond the capabilities for individual firms (cf. Huang, 2001; Stafford, 2002). OSSNs are characterized by their reliance on ICT to mediate collaboration and communication and have grown in an organic fashion strongly informed by the practice of community-based OSS projects. While being legal entities, they have avoided formal or legal contracts to govern inter-firm coordination. This type of inter-firm co-ordination has been termed 'network governance'.

Network governance is defined by Jones et al. (1997: 913) as "interfirm coordination characterized by organic or informal social systems, in contrast to bureaucratic structures within firms and formal contractual relationships between them." They identify four pre-conditions for network governance: demand uncertainty, customized (asset-specific) exchanges, complex tasks executed under time-pressure, and frequent exchanges between partners. Jones et al. argue that under such conditions, networks develop structural embeddedness, which they define, citing Granovetter (1992:35), as the extent to which a "dyad's mutual contacts are connected to one another" creating both direct and indirect ties between parties. Jones et al. assert that when high levels

of structural embeddedness are present, it enables the use of various social mechanisms to resolve exchange problems by coordinating and safeguarding exchanges within networks. These social mechanisms are:

5. Restricted Access – the "strategic reduction in the number of exchange partners in the network" (Jones et al 1997: 927)
6. Macroculture – a "system of widely shared assumptions and values … that guide actions and create typical behaviour patterns among independent entities" (*ibid*: 929)
7. Collective Sanctions – ways in which groups punish members who violate shared norms, values and goals (*ibid*: 931)
8. Reputation – "estimations of one's character, skills, reliability and other attributes important to exchanges" (*ibid*: 932)

In resolving exchange problems in network governance, Jones et al. propose that coordination is supported by Restricted Access and Macroculture, while safeguarding is supported by Restricted Access, Collective Sanctions and Reputation (Figure 1). The four social mechanisms described by Jones et al. already loom large in the OSS literature (e.g. Gallivan, 2001; Mockus et al., 2002; Stewart, 2005; Szczepanska et al., 2005), albeit not in the business network context. In fact, the Jones et al. framework has been directly applied to open source projects hosted by SourceForge (Sagers, 2004). Nevertheless, while the Jones et al. framework has been widely cited in a number of fields, neither the original framework, nor subsequent applications of it, provide measures of network effectiveness. However, research on Inter-organisational Systems (IOS) has provided a more considered conceptualization of network effectiveness. In particular, De Wever et al. (2005) posit that network effectiveness can be operationalised as the ability of networks to provide member firms with sustainable access to strategic or value-generating resources. De Wever et al. (2005) propose that "the performance of inter-organisational networks requires sequentially: access to strategic resources; transfer or exchange of strategic resources in order to acquire them; and the internal use of the exchanged strategic resources," thus highlighting the interaction between network effectiveness and the absorptive capacity of member firms.

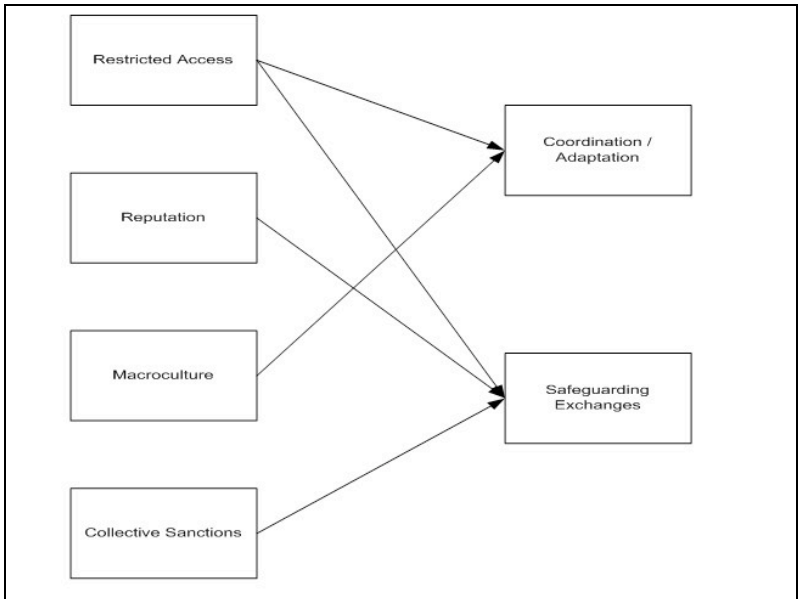

Figure 1. Social Mechanisms and exchange problems (Jones et al. 1997)

Research Objective and Method

The objective of our study is to explore the role of social mechanisms in the effective governance of Open Source Service Networks.
We have adopted a "soft positivist" epistemology (Kirsch, 2004) and seek to apply a combination of qualitative (phase 1) and quantitative (phase 2) research methods. In phase 1, we begin with the Jones et al. framework for network governance (with network effectiveness operationalised as per De Wever et al., 2005), and refine this for an OSSN context by means of an in-depth case study of one sophisticated OSSN. This is in line with Lee and Baskerville (2003) who, in addressing the issue of generalization, document the process of generalizing from theory to empirical description (research seeks to apply theoretical findings confirmed in one setting to another one). Thus, our qualitative method follows in the tradition of Eisenhardt (1989) and Madill et al. (2000); it is designed to reveal pre-existing, relatively stable and objectively extant phenomena and the relationships among them in a manner that is not limited to examining only pre-identified constructs.
In phase 2, we test our OSSN governance model by means of a quantitative study within two other OSSNs. We therefore seek to validate the model emerging from phase 1 vis-a-vis a more conventional interpretation of generalization (Baskerville and Lee, 2003). This phase is outlined at the end of this paper.

Qualitative Study: The Development of a Conceptual Model

The qualitative study was concerned with initially achieving an increased understanding of this phenomenon. Thus, an interpretivist approach which sought to inductively develop a richer understanding based on a deep analysis of a single case was deemed appropriate, as this "revelatory case" (Yin, 1994) may provide such rich insight. We begin with the Jones et al. framework for network governance in general (with network effectiveness operationalised as described above), and refine this for an OSSN context by means of an in-depth case study of one sophisticated OSSN – the Zope Europe Association (ZEA). ZEA is an international network of businesses that build software and deliver services around the application server technology called Zope, widely used for developing content management systems, intranets, portals, and related applications. We believe that ZEA represents a sophisticated exemplar of the OSSN concept.

Qualitative Study: Method

Data for the qualitative study was gathered from November 2004 to April 2006 using a number of sources including intensive workshops, formal face-to-face interviews and informal telephone interviews (Table 2). Interviews were generally of one to two-hour duration with follow-up telephone interviews used to clarify and refine issues that emerged during transcription. Interviews were complemented by comprehensive reviews of documents and presentations at the workshops.

Table 2 Data Sources

Interviews	Founder ZEA (19/11/04, 4/3/05, 3/11//05, 14/4/05, 24/4/06) CEO ZEA (19/11/04, 13/7/05, 3/3/06) Founder, Infrae (22/4/05) Director BlueFountain (11/5/05) Chief Architect, Plone Solutions (1/6/05) Owner, Zetwork (8/6/05)

	Owner, Gocept (8/7/05) Owner, Reflab (13/7/05) Owner, Zest Software (26/8/05) Owner Bubblenet (13/10/05)
Workshops with ZEA members	The Hague (19/11/04) Paris (4/3/05) Genoa (13/7/05) Skovde (3/3/06)

A primarily qualitative grounded theory (GT) approach of data analysis was adopted (cf. Strauss and Corbin, 1990; Miles and Huberman, 1994). This approach recognises that social phenomena are complex, and seeks to develop theory systematically in an intimate relationship with the data. Interview data was transcribed; generating 133 pages of field notes. These were coded according to the constructs represented by the network governance theory factors, and analytical memos were written as patterns and themes emerged.

A problem that has been identified in relation to qualitative research is what is termed multiple realities; the unavoidable fact that the understanding of reality is based on an individual interpretation of the data, and that different individuals may interpret the same data in different ways (Kaplan and Duchon, 1988). This problem was addressed as follows. First, the grounded theory method of data analysis prescribes rigorous coding and memoing processes which provide a traceable, documented justification of the process by which research conclusions are reached; thereby providing an audit trail of the process (Guba and Lincoln, 1981). Second, venting (Goetz and LeCompte, 1984) was used as results and interpretations were discussed with ZEA representatives and fellow researchers.

Qualitative Study: Results and Discussion

ZEA illustrates the preconditions for network governance as predicted by extant research. The availability of contracts is subject to what the ZEA Founder calls ‘valleys and peaks’ and the OSS domain experiences constant changes in knowledge and technology, leading to a requirement for information dissemination amongst firms. These factors produce what Jones et al. term *demand uncertainty*. The small size (typically <10 people), geographic/linguistic limitations, and specialised knowledge of member firms limit the size, location and complexity of the contracts that they can bid for. The network helps overcome demand uncertainty by allowing companies pool resources to compete for larger/global contracts. In competing on the basis of a ‘whole product’ (Moore, 1999), the network allows partners offer a full range of value-chain activities rather than concentrating exclusively on their own specialities. *Task complexity* is evident as producing the ‘whole product’ for a wide range of markets and customers requires the inputs of specialists across a range of business functions. *Customised exchanges high in human asset specificity* are recognisable as member firms collaborate to produce a customised product/service that is not easily transferred. Inter-firm routines are learned emergently through the collaborative process rather than by prior agreement. Due to geographical distance, ZEA members *frequently* collaborate and exchange knowledge in a digital environment. Ongoing interactions between member firms and the mutual sharing of partners, clients and contacts with the wider OSS community all provide evidence of s*tructural embeddedness.*

The ZEA Founder explained that for inter-firm coordination, social mechanisms "are the only thing. This is an opt-in system. A large component of the perceived benefit is the reputation improvement from being in the network. Social mechanisms really underpin that". Our analysis revealed that, in line with extant research on such social mechanisms, restricted access, reputation, macroculture and collective sanctions enabled the coordination and safeguarding of exchanges between ZEA members. However, our analysis revealed that the functioning of these mechanisms and how they relate to coordination/adaptation and safeguarding exchanges followed a different pattern to that proposed by Jones et al. Following the work of Koh et al. (2004), we provide sample interview evidence in Table 3-8 to illustrate how the social mechanisms function. This functioning reflects the fact that ZEA inherited many of the cultural values and social mechanisms present in the wider OSS community, but is also a commercial professional services network.

Table 3: Restricted Access to the OSSN safeguards exchanges by:	
Reducing transaction costs	*Transaction costs occur because people email me if they run into friction in the system; I manually have to deal with it. I think it would be impossible to have an order of magnitude increase using the same systems that we have right now. But that is almost exactly the way OSS communities are; so many of them fail when they get their eleventh contributor because the people who started the project don't really know how to manage ten contributors.* (ZEA)
Reducing the danger of opportunistic behaviour	*Each applicant we evaluate has to accept a mutual respect culture. We have had cases where we felt they would be 'disruptive' and so we said 'no'.* (ZEA)
Increasing identification amongst partners	*The partners are all very small firms, mainly led by technologists. They need a place to complain about business things, to ask questions, to gain experience etc. You can't really do that out in the open because you don't really want to share business failures with the entire world.* (ZEA) *I'm big on personal meetings. They help me to get to know and trust some members better.* (Zest)
Not diluting benefits	*I think there are enough partners. I think it will be hard to evenly distribute work so every company can earn back its membership fee.* (Zest)

Table 4: Restricted access to the OSSN improves coordination /adaptation by:	
Facilitating the establishment of routines for working together	*We're still at an early enough stage that we are able to work together on a person-to-person basis…There was a meeting in Munich at which ten of the companies showed up for a two-day seminar on how we were going to go after bids, collecting all the material, references, corporate profiles, etc.* (ZEA)
Reducing co-ordination cost through increased visibility	[Co-ordination cost is] *helped by the transparency in the community...The best barometer is the way people act in the community. If they are a competent developer then you can see them send mails to the mailing lists, do check-ins and file bug reports, write papers for conferences etc. There are a multitude of avenues available to show their competence. If someone is invisible from all those avenues, that says something*. (ZEA)
Adopting deliberate membership requirements facilitating a component-based approach to work allocation	*There is no single point of failure; you can swap out components or companies. If one company is not talented enough or do not have the domain knowledge to do this particular thing we normally have another company that has worked with those kind of things. It's very agile and flexible*. (Plone Solutions)
Giving members a voice in decision making	*We also have the ability to steer…the whole product idea was something that we could approve or reject* (Reflab) *We talk a lot…which gives me the chance to give my opinion and maybe shape part of the process, but not officially*. (Bubblenet)
Reducing variances that parties bring to the exchanges (facilitates mutual adjustment).	*About 50% of the problem is solved by working the same way we worked [in OSS communities]. The companies have a set of tools, a way of working together, a common culture and ways of communication that we use on ZEA projects* (ZEA)
Establishing routines for working together	*We have standing contracts, customer references etc on file. So the friction in assembling a team gets lowered. The next part is to have a common methodology, a common way of thinking about a problem, assigning work, tracking results, reporting bugs*. (ZEA)

Table 5: Reputation within the OSSN safeguards exchanges by:	
Acting as a prerequisite for participation	*Getting in the network is only the first part. If you want to do transactions with others in the network, you need to show yourself to be competent and professional.* (ZEA)
Rewarding good behaviour	*Getting a reputation in the community is very important; other companies will send me projects that they cannot handle*. (Bubblenet)
Ensuring that there are economic consequences of reputation	*My first contracts were subcontracts from other Zope companies. That gave me enough presence in the community to switch to my own contracts, but not ones as small as before. Being part of Zope Europe has helped a lot.* (Bubblenet)

Table 6: OSSN Macroculture improves co-ordination / adaptation by:	
Fostering a culture of network agility	*We want to define this OSS business model idea. Instead of having a cathedral model of Accenture, we want to have multiple players in multiple countries. We can move things around as new trends/specialities emerge.* (ZEA)
Ensuring new members fit in to network's culture.	*We're going to focus on people who've already decided they're interested in OSS, not people who don't understand value and values. The prime consideration is to make sure that the fabric that holds this experiment together doesn't get torn* (ZEA).
Creating a sense of mutual interest (good Karma)	*We have not been taking ZEA projects because there are other companies that need the work more. It's partly that we are busy and partly that we are trying to be nice to other people; they're good people and they should have more work than they currently do*. (Plone Solutions)
Preventing fear of lock-in	*The part that seduces me is that someone who joins can leave easily. This is very important because it helps in reducing people's fears; this means they work better together.* (Bubblenet)
Reducing information / knowledge asymmetries (including those with customers)	*It's very important that we coordinate, share best practice, learn from each other, leverage our collective experience.* (BlueFountain) *The customer is a participant not a recipient. We want to engage with their in-house staff and teach them how to become OSS developers; how to participate in the community, ask questions etc., even show them how to contribute back.* (ZEA)

Table 7: OSSN Macroculture safeguards exchanges by:	
Balancing commercial interests with network objectives	*Most of us are in this for the lifestyle rather than purely to make money; that's difficult to formalize* (Bubblenet) *You get respect by helping other people; you earn the right to ask more questions and favors.* (Zest)
Taking an ethical approach to commercial issues	We tell our customers we do three things: transparency, integrity and honesty. If there's a problem we'll always put our hand up. (Bluefountain) We wouldn't leave a client hanging just because it didn't suit us to go back to that deployment. It's a value thing. (Plone Solutions)
Creating and environment of trust	*There's a trust vector when it comes to organising and assigning work that people can make money on. That person has to be perceived as being neutral and competent* (ZEA)

Table 8: Collective sanctions safeguard exchanges because of the threat of:	
Damaged reputation within the network	*We had cases where people said yes to a ZEA project but were actually overbooked…this affects their reputation with in the network* (ZEA)
Exclusion from the Network results in damaging reputation in the OSS community (outside the network)	*Whenever you have a commons, somebody is going to be the shark and try to hijack the commons, and have a winner take all mentality. You can fight back in the courts, but that takes time and money. If you get kicked out of the network that's just as bad as getting sued, as word will get around to your potential customers.* (ZEA)

The relationship between social mechanisms and coordination/adaptation and safeguarding exchanges (Figure 2) is the same as extant research, except that we found macroculture safeguards exchanges rather than just facilitating

coordination/adaptation as predicted by Jones et al. Additionally, our analysis reveals how social mechanisms reinforce each other. The need for this type of analysis was called for by Jones et al. as a way of extending their framework. *Access* to the ZEA Network is restricted based on a firm's *reputation* and performance in the OSS community as well as previous interactions with member firms. It is not just about keeping numbers low, it is a strategic restriction; firms have to fit in with the *macroculture* and needs of the network, and exhibit a commitment to ZEA's success. ZEA's *macroculture* is typical of values associated with OSS communities; the involvement of users (customers) as equal partners, the visibility of actions, etc. However, this macroculture is evident not just as a shared set of values (including beliefs, language, etc) as proposed by Jones et al., but also includes the mechanisms by which these ideas are expressed and shared amongst members. This is a richer view of the cultural process and is in line with the thinking of Hannerz (1992) on cultural complexity in information societies. Our analysis also reveals a reinforcing relationship between *collective sanctions* and *reputation* as members fear damaging their reputation safeguards exchanges.

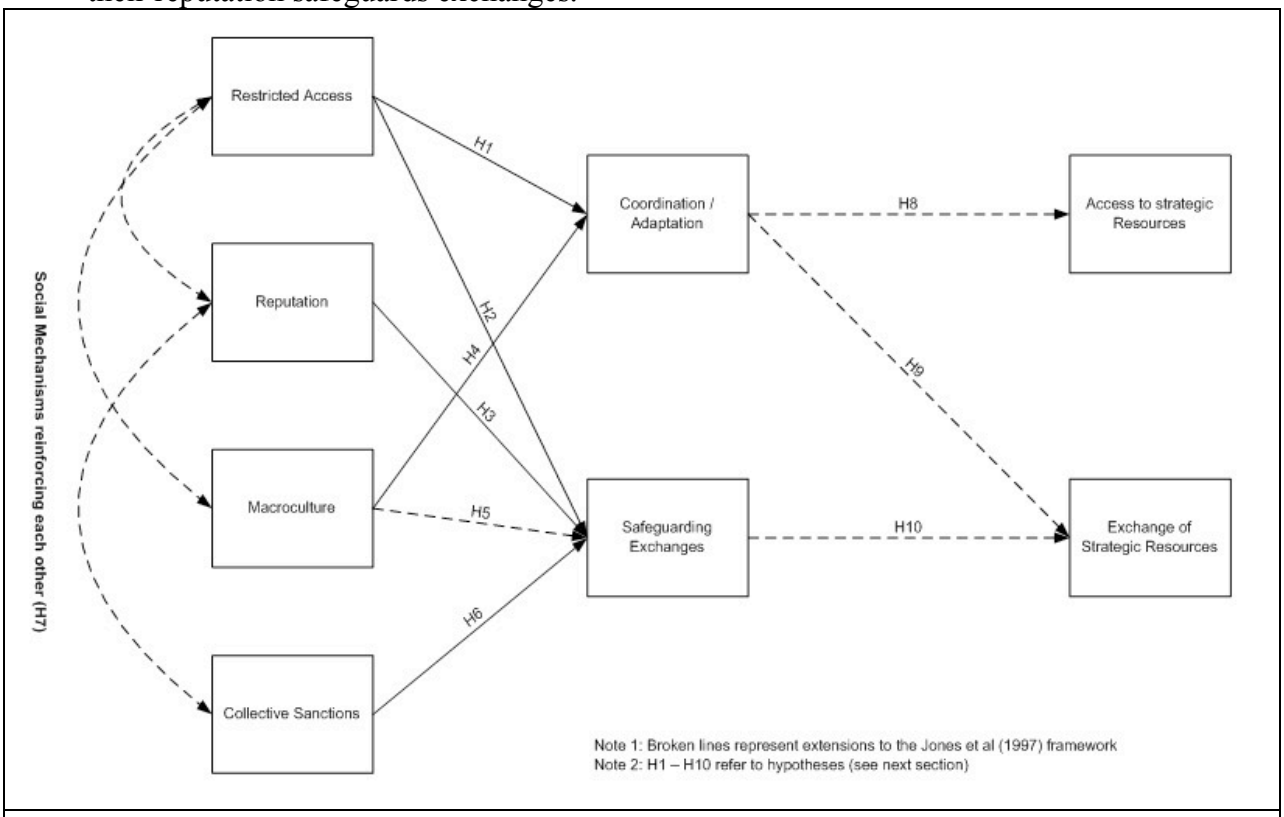

Figure 2: Model of Network Governance in OSSNs

Members consider the ZEA network effective as it allows them to access and share strategic resources. They cited the ability to; access and leverage the skill base of other firms; share customer contacts; compete for larger contracts; share business experiences/expertise (e.g. project management) and enjoy an enhanced reputation as a result of membership. According to the Founder, such resource sharing allows ZEA to "take the whole product and make it offerable by anyone in the network. It has so many benefits on profitability it's just amazing". Our analysis revealed coordination/adaptation to be essential for firms to offer the 'whole product'; thus affecting access to and exchange of strategic resources. In addition, the exchange of strategic resources was dependant on such exchanges being safeguarded. However,

our analysis revealed that the internal use of many strategic resources was regarded as being more dependent on individual firms than on the network. Thus, it is not considered a network effectiveness factor by members, and, contrary to De Wever (2005), is excluded from Figure 2

Work-in-Progress

The qualitative study allowed us to develop a conceptual model of governance in OSSNs building on the Jones et al framework by (a) illustrating the interaction between social mechanisms and (b) expanding the concept of network effectiveness. We have represented our conceptual model as a series of hypotheses. These are:

H1: Restricted access facilitates co-ordination/adaptation by facilitating the establishment of routines for working together (H1a), reducing co-ordination costs through increased visibility (H1b), adopting deliberate membership requirements facilitating a component-based approach to work allocation (H1c), giving members a voice in decision making (H1d), establishing routines for working together (H1e), and reducing variances that parties bring to the exchanges (H1f).

H2: Restricted access safeguards exchanges by reducing transaction costs (H2a), reducing the danger of opportunistic behaviour (H2b), increasing identification amongst partners (H2c), and not diluting membership benefits (H2d).

H3: Reputation within the OSSN safeguards exchanges by acting as a prerequisite for participation (H3a), rewarding good behaviour (H3b), and ensuring that there are economic consequences of a firm's reputation (H3c).

H4: OSSN Macroculture improves co-ordination/adaptation by fostering a culture of network agility (H4a), ensuring new members fit into the culture of the network (H4b), creating a sense of mutual interest (good karma) (H4c), preventing fear of lock-in (H4d), and reducing information / knowledge asymmetries (including those with customers) (H4e).

H5: OSSN Macroculture safeguards exchanges by balancing commercial interests with network objectives (H5a), taking an ethical approach to commercial issues (H5b), and creating and environment of trust (H5c).

H6: Collective sanctions safeguard exchanges because of the threat of damaged reputation within the network (H6a), and the possibility of exclusion from the network resulting in damaged reputation in the OSS community (outside the network) (H6b).

H7: Social mechanisms mutually reinforce each other as follows; restricted access and macroculture (H7a); restricted access and reputation (H7b); reputation and collective sanctions (H7c).

H8: Coordination/adaptation improves network effectiveness by providing access to strategic resources.

H9: Coordination/adaptation improves network effectiveness by facilitating the exchange of strategic resources.

H10: Safeguarding exchanges improves network effectiveness by facilitating the exchange of strategic resources.

Our work-in-progress is testing this model using a quantitative approach. We are finalising the operationalisation of the variables and the design of the survey

instrument for administration in two OSSNs; Consortium Italicum Ratione Soluta and ObjectWeb. Data gathering will take place during May and June 2006.

References

Bergquist, M. and Ljungberg, J. "The Power of Gifts: Organising Social Relationships in Open Source Communities" *Information Systems Journal* (11:4) December, 2004 pp.305-320.

Clemons, E. K., and Row, M. C. "Information Technology and Industrial Cooperation: The Role of Changing Transaction Costs", *Journal of Management Information Systems*, (9:2) Fall 1992, pp.9-28.

De Wever, S., Martens, R., and Vandenbempt, K. "The impact of trust on strategic resource acquisition through interorganizational networks: Towards a conceptual model," *Human Relations* (58:12) December 2005, pp. 1523-1543.

Eisendardt, K.M. "Building theories from Case study Research", *Academy of Management Review* (14:4) December, 1989 pp.532-550.

Feller, J. and Fitzgerald, B. *Understanding Open Source Software Development*, Addison-Wesley, London, England, 2002.

Gallivan, M.J. "Striking a Balance between Trust and Control in a Virtual Organisation: A content Analysis of Open Source Software Case Studies" *Information Systems Journal* (11:4) December, 2004 pp.277-304.

Goetz, J.P. and LeCompte, M.D. *Ethnography and Qualitative Design in Educational Research*, Harcourt Brace Jovanovich, New York, NY, 1984.

Granovetter, M. "Problems of Explanation in Economic Sociology," in: *Networks and Organizations: Structure, Form, and Action*, N. Nohria and R.G. Eccles (eds.), Harvard Business School Press, Cambridge, MA, 1992, pp. 25-56.

Guba, E. G., & Lincoln, Y. S. *Effective Evaluation: Improving the Usefulness of Evaluation Results through Responsive and Naturalistic Approaches*. Jossey-Bass, San Francisco, CA, 1981.

Hannertz, U. *Cultural Complexity: Studies in the Social Organisation of Meaning*. Columbia University Press, NY, 1992.

Huang, C. "Using Intelligent Agents to Manage Fuzzy Business Processes", *IEEE Transactions on Systems, Man, and Cybernetics—PartA: Systems and Humans*, (31:6) November 2001, pp.508-523.

Jones, C, Hesterly, W, and Borgatti, S. "A General Theory of Network Governance: Exchange Conditions and Social Mechanisms," *Academy of Management Review* (22:4) December 1997, pp. 911-944.

Kaplan, B., and Duchon, D. "Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study," *MIS Quarterly* (12:4), December 1988, pp. 571-588.

Kirsch, L.J "Deploying Common Systems Globally: The Dynamics of Control" *Information Systems Research* (15:4), December 2004, pp. 375-395.

Koh, C, Ang, S., and Straub, D.W. "IT Outsourcing Success: A Psychological Contract Perspective" *Information Systems Research* (15:4), December 2004, pp. 356-373.

Lee, A.S. and Baskerville, R.L. (2003). "Generalizing Generalizability in Information Systems Research," *Information Systems Research* (14:3), September 2003, pp. 221-243.

Madill, A., Jordan, A. and Shirley, C. "Objectivity and Reliability in Qualitative Analysis: Realist, Contextualist and Radical Constructionist Epistemologies. British Journal of Psychology (91:1) February 2000, pp.1–20.

Miles, M.B., and Huberman, A.M. *Qualitative Data Analysis* (2nd ed.). Sage Publications, Newbury Park, CA, 1994.

Mockus, A., Fieldding, R.T. and Herbsleb, J.D. "Two Case Studies of Open Source Software Development: Apache and Mozilla" *ACM Transactions on Software Engineering and Methodology* (11:3) March, 2002 pp.309-364.

Moore, G. *Crossing the Chasm,* Harper-Perennial, New York, NY 1999.

Sagers, G "The Influence of Network Governance Factors on Success in Open Source Software Development Projects. *Proceedings of theTwenty-Fifth International Conference on Information Systems.* Agarwal, R and Kirsch, L. (eds.) Washington DC, December 2004, pp.427-438.

Stafford, T "Trust, transactions, and relational exchange: Virtual integration and agile supply chain management", *Proceedings of the 8th Americas Conference on Information Systems*, Association for Information Systems, August 2002, pp.2365-2371.

Stewart, D. "Social Status in an Open Source Community" *American Sociological Review* (70:5) May 2005, pp. 823-842.

Strauss, A. and Corbin, J. Basics of Qualitative Research: Grounded Theory Procedures and Techniques. Sage Publications, Newbury Park, CA, 1990.

Szczepanska, A.M., Bergquist, M. and Ljunberg, J. "High Noon at OS Corral: Duals and Shootouts in Open Source" in Feller, J., Fitzgerald, B., Hissam, S.A and Lackhani, K.R. (eds.) *Perspectives on Free and Open Source Software*. MIT Press, Cambridge, MA. 2005 pp. 431-446.

Woods, D., and Guliani, G. *Open Source for the Enterprise*, O'Reilly Media, Sebastopol, CA, 2005.

Yin, R.K. *Case Study Research, Design and Methods*, Sage Publications, Newbury Park, CA, 1994.

References

Agarwal, R. (2000) "Individual Acceptance of Information Technologies," *Framing The Domains of IT Management: Projecting the Future Through the Past*, R. W. Zmud (Ed.), Cincinnati, OH: Pinnaflex Press, pp. 85-104.

Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B., and Ó Conchúir, E. (2005) "A Framework for Considering Opportunities and Threats in Distributed Software Development," in *Proceedings of the International Workshop on Distributed Software Development (DiSD 2005),* Paris, 29 August 2005: Austrian Computer Society, pp. 47-61.

Anderson, N., and Chalk, R. (1998) "The Psychological Contract in Retrospect and Prospect," *Journal of Organizational Behavior*, 19(S1), pp. 637-647.

Ang, S., and Slaughter, S. A. (2001) "Work Outcomes and Job Design for Contract Versus Permanent Information Systems Professionals on Software Development Teams," *MIS Quarterly*, 25(3), pp. 321-350.

Argyris, C. (1960) *Understanding Organizational Behaviour*, London: Tavistock Publications.

Baskerville, R., and Pries-Heje, J. (1999) "Grounded Action Research: A Method for Understanding IT in Practice," *Accounting, Management and Information Technologies*, 9 (1), pp. 1-23.

Carmel, E., and Tjia, P. (2005) Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce, Cambridge, NY: Cambridge University Press.

Carmel, E. (1999) *Global Teams: Collaborating Across Borders and Time Zones*. Prentice-Hall, Upper Saddle River: NJ.

Chatterjee, D., Grewal, R., and Sambamurthy, V. (2002) "Shaping Up for E-Commerce: Institutional Enablers of the Organizational Assimilation of Web Technologies," *MIS Quarterly*, 26(2): 65-89.

Dinh-Trong, T., and Bieman, J. M. (2004) "Open Source Software Development: A Case Study of FreeBSD," in *Proceedings of the 10th International Symposium on Software Metrics*, IEEE Computer Society.

Feller, J., and Fitzgerald, B. (2002) *Understanding Open Source Software Development*, London, UK: Addison-Wesley.

Fichman, R. G. (2004) "Going Beyond the Dominant Paradigm for IT Innovation Research: Emerging Concepts and Methods," *Journal of the Association for Information Systems*, 5(8).

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

Fitzgerald, B., and Kenny, T. (2003) "Open Source Software in the Trenches: Lessons from a Large Scale Implementation", in *Proceedings of 24th International Conference on Information Systems*, Seattle, December 2003, pp. 316-326

Gacek, C. and Arief, B. (2004) "The Many Meanings of Open Source," *IEEE Software*, 21 (1), 34-40.

Gallivan, M. (2001) "Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework," *Data Base*, 32(3), pp. 51-85.

German, D. M. (2004) "The GNOME project: A Case Study of Open Source, Global Software Development," *Software Process: Improvement and Practice*, 8(4), pp. 201-215.

Ghosh, R. (2005) Understanding Free Software Developers: Findings from the FLOSS Study, in Feller, J. Fitzgerald, B., Hissam, S and Lakhani, K. (eds.) *Perspectives on Free and Open Source Software*, MIT Press, Cambridge.

Goetz, J., and LeCompte, D. (1984) *Ethnography and Qualitative Design in Educational Research*, Academic Press, Orlando.

Gorman, M. (2003) *A Design, Implementation and Algorithm Analysis of a Virtual Memory System for Linux,* (personal communication).

Guba, E. (1981) "Criteria for assessing the trustworthiness of naturalistic inquiries," *Educational Communication and Technology*, 29, pp. 75-92.

Hann, I., Roberts, J., Slaughter, S., and Fielding, R. (2002) *Why Do Developers Contribute to Open Source Projects? First Evidence of Economic Incentives*, Paper presented at the 2nd Workshop on Open Source Software Engineering, Orlando, FL.

Kaplan, B., and Duchon, D. (1988) "Combining Qualitative and Quantitative Methods in IS Research: A Case Study," *MIS Quarterly*, 12(4), pp. 571-587.

Koh, C., Ang, S., and Straub, D. W. (2004) "IT Outsourcing Success: A Psychological Contract Perspective," *Information Systems Research*, 15(4), pp. 356-373.

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

Lerner, J., and Tirole, J. (2002) "Some Simple Economics of Open Source," *The Journal of Industrial Economics*, 50(2), pp. 197-234.

Marshall, C., and Rossman, G. (1989) *Designing Qualitative Research*, Sage Publications, California.

Millar, C., Choi, C. J., Russell, E. T., and Kim, J. B. (2005) "Open Source Communities: An Integrally Informed Approach," *Journal of Organizational Change Management*, 18(3), pp. 259-268.

Miranda, S. M., and Kavan, C. B. (2005) "Moments of Governance in IS Outsourcing: Conceptualizing Effects of Contracts on Value Capture and Creation," *Journal of Information Technology*, 20(3), pp. 152-169.

Mockus, A., and Herbsleb, J. D. (2002) "Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source?" in *Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering*, pp. 19-25, http://opensource.ucc.ie/icse2002/MockusGerbsleb.pdf

Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002) "Two Case Studies of Open Source Software Development: Apache and Mozilla," *Transactions on Software Engineering Methodology*, 11 (3), pp. 309-346.

Moore, G (1999) *Crossing the Chasm*, Harper, NY

Patton, M. Q. (1990) *Qualitative Evaluation and Research Methods*, 2nd ed., Newbury Park, CA: SAGE.

Pavlou, P. A., and Gefen, D. (2005) "Psychological Contract Violation in Online Marketplaces: Antecedents, Consequences, and Moderating Role," *Information Systems Research*, 16(4), pp. 372-399.

Piccoli, G., and Ives, B. (2003) "Trust and the Unintended Effects of Behavior Control in Virtual Teams," *MIS Quarterly*, 27(3), pp. 365-395.

Raghu, T. S., Jayaraman, B., and Rao, H. R. (2004) "Toward an Integration of Agent- and Activity-Centric Approaches in Organizational Process Modeling: Incorporating Incentive Mechanisms," *Information Systems Research*, 15(4), pp. 316-335.

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

Scacchi, W. (2002) "Understanding the Requirements for Developing Open Source Software Systems," *IEE Proceedings–Software*, 149(1), pp. 24-39.

Strauss, A., and Corbin, J. (1998) Basics of Qualitative Research: Grounded Theory Procedures and Techniques, 2nd ed., Sage, California.

United Nations (2004) World Investment Report 2004 – The Shift Towards Services, New York and Geneva.

Von Hippel, E., and Von Krogh, G. (2003) "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science," *Organization Science*, 14(2), pp. 209-223.

Wheeler, D. (2004) *Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* Online: http://www.dwheeler.com/oss_fs_why.html

Yin, R. K. (1994) *Case Study Research: Design and Methods*, 2nd ed., Thousand Oaks, CA: SAGE.