Information Systems Science

Not Accidental Revolutionaries: Essays on Open Source Software Production and Organizational Change

Juho Lindman

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 13/2011

Not Accidental Revolutionaries: Essays on Open Source Software Production and Organizational Change

Juho Lindman

Aalto University
School of Economics
Department of Information and Service Economy
Information Systems Science

Aalto University publication series
DOCTORAL DISSERTATIONS 13/2011

© Juho Lindman

ISBN 978-952-60-4038-7 (pdf)
ISBN 978-952-60-4037-0 (printed)
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934 (printed)

Aalto Print
Helsinki 2011

Publication orders (printed book):
juho.lindman(at)aalto.fi

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Juho Lindman

Name of the doctoral dissertation
Not Accidental Revolutionaries: Essays on Open Source Software and Organizational Change

Publisher Aalto University School of Economics

Unit Department of Information and Service Economy

Series Aalto University publication series DOCTORAL DISSERTATIONS 0/2011

Field of research Information Systems Science

Abstract

Open Source Software research has established that OSS technology (tools and practices) holds untapped potential. Based on a systematic literature review and a research engagement over a three-year period of data gathering, my dissertation describes how organizations leverage OSS practices to produce software. Leveraging OSS can be divided into two processes: 1) inbounding (moving public assets inside a company) and 2) outbounding (publishing) OSS. I outline the structural consequences these changes in software production entail and provoke. My research question is: What is the relation between local renegotiation of the term OSS and the organizational change provoked by OSS technology?

I chose a qualitative approach to examine the case companies, informed by OSS research and institutional theory. The bulk of the data emerges from the industrial ITEA-COSI project, which focused on software commodification. I aim to provide a narrative of how the term OSS travels from the writings of enthusiasts to the daily work practices of software producing organizations. The findings underline the importance of local renegotiation of the term OSS. This renegotiation provokes structural changes in 1) the organizations that adopt OSS technology, but more widely also in 2) the industries these companies operate in.

The main contribution of this research thesis, reported in four essays, is directed at two audiences: first, at academics, to promote the idea that OSS in organizations should be researched in a sensitivized manner. This requires moving away from too simplistic institutional contexts and ”the OSS business model”. Second, it is directed at practitioners, to reduce uncertainty about the adoption of OSS technology and to help build a capacity to accept, search for, motivate and reward contribution.

Keywords Open Source, Inner Source, Outbound OSS, Inbound OSS, Organizing Vision

ISBN (printed) 978-952-60-4037-0 **ISBN (pdf)** 978-952-60-4038-7

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Pages 140 **Location of publisher** Helsinki **Location of printing** Helsinki **Year** 2011

A? **Aalto-yliopisto**

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä(t)
Juho Lindman

Väitöskirjan nimi
Esseitä avoimen lähdekoodin tuottamisesta organisaatioissa

Julkaisija Aalto-yliopiston kauppakorkeakoulu

Yksikkö Tieto- ja palvelutalouden laitos

Sarja Aalto-yliopiston julkaisusarja VÄITÖSKIRJAT 0/2011

Tutkimusala Tietojärjestelmätiede

Tiivistelmä

Organisaatiot eivät ole pystyneet hyödyntämään kansainvälisessä tutkimuskirjallisuudessa tunnistettua avoimen lähdekoodin potentiaalia. Tässä väitöskirjassa on kuvattu avoimen lähdekoodin hyödyntämistapoja organisaatioissa kolmivuotisen seurantatutkimukseen ja systemaattiseen kansainväliseen kirjallisuuskatsaukseen perustuen. Väitöskirja kuvaa ohjelmistotuotannon muutoksen ennakkoehtoja ja organisatorisia seurauksia. Tarkka tutkimuskysymys on: Mikä on avoimen lähdekoodin termin merkitystä koskevien paikallisten tulkintojen ja avoimen lähdekoodin teknologian aiheuttaman organisatorisen muutoksen välinen suhde?

Tutkimuksessa on käytetty laadullista tutkimusotetta ja sen viitekehyksenä toimivat avoimen lähdekoodin tutkimus ja institutionaalinen teoria. Suurin osa datasta on kerätty ITEA-COSI -tutkimusprojektista, joka keskittyi ohjelmistokommodifikaatioon. Tutkimuksen tarkoituksena on luoda kuvaus siitä, miten avoin lähdekoodi kulkeutuu ohjelmistogurujen kirjoituksista ohjelmistoja tuottavien organisaatioiden jokapäiväisiin käytäntöihin. Työn tulokset korostavat paikallisten neuvotteluiden merkitystä. Neuvottelut koskien termin ”avoin lähdekoodi” merkitystä aiheuttavat erilaisia muutoksia ohjelmistotuotantoon 1) niissä organisaatioissa, jotka hyödyntävät avointa lähdekoodia, mutta laajemmin 2) niillä toimialoilla, joilla nämä yritykset toimivat.

Neljässä tutkimusesseessä raportoidut tulokset on suunnattu kahdelle eri yleisölle. Ensinnäkin tutkijoille, jotta avointa lähdekoodia tutkittaisiin paikallisten tulkintojen merkitys huomioiden. Tämä vaatii sensitiivisyyttä organisaatioiden erityispiirteille. On varottava liiallista yksinkertaistamista ja ”voittavan avoimen lähdekoodin liiketoimintamallin” etsimistä. Toiseksi tulokset on suunnattu asiantuntijoille vähentämään avoimeen lähdekoodin liiketoimintakäyttöön liittyvää epävarmuutta. Tulokset kannustavat asiantuntijoita etsimään, hyväksymään, motivoimaan ja palkitsemaan kontribuutioita.

Avainsanat Avoin lähdekoodi, rajoitettu lähdekoodi, avoimen lähdekoodin julkaiseminen, avoimen lähdekoodin rajoittaminen, organisoiva tulevaisuuskuva

ISBN (painettu) 978-952-60-4037-0 **ISBN (pdf)** 978-952-60-4038-7

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Sivumäärä 140 **Julkaisupaikka** Helsinki **Painopaikka** Helsinki **Vuosi** 2011

Acknowledgements

This dissertation was made possible mostly by bad coffee and some cheap beer. Not only did they make me work harder, but also offered excuses for the exchanges that have given shape to this dissertation. Any venture of this kind would never have been finished without the help of so many individuals along the way. The number of people who have contributed to the joint effort is too large to do them justice, and any attempt to do so will ultimately remain futile. That said, the following persons have especially pushed me along the way. My apologies to those who had to be omitted to prevent this section from growing longer than all the other sections.

First of all, I would like to thank my supervisor, Professor Matti Rossi, who has patiently given me the advice and support needed to get this study wrapped up. Matti never made any remarks about my going to yet another policy meeting, geek conference or street campaign. I will miss our discussions on contemporary politics. Thank you, Matti.

Professors Juhani Warsta from the University of Oulu and Pär Ågerfalk from the University of Uppsala were kind enough to act as external evaluators of my dissertation. Their critical comments and suggestions have improved the quality of the manuscript significantly. I would additionally like to thank Pär Ågerfalk for agreeing to act as my opponent.

I have been privileged to co-author articles on OSS with many talented individuals: Risto Rajala, Juha-Pekka Juutilainen, Topi Uitto, Anna Paajanen, Pentti Marttiin and Mikko Riepula. I also want to mention Virpi Tuunainen, who has had an instrumental role in providing resources.

This dissertation work was conducted in a European context. Not only have I benefited from doctoral consortiums held in different countries of Europe (ECIS2008, CEMS-NITIM2008, AIM2008, OSS2009, UKAIS2010), but a European research project, ITEA-COSI, is the origin of most of my data. Special thanks to the Norwegian co-authors in NTNU: Thomas Østerlie, Øyvind Hauge and Sven Ziemer.

This dissertation would probably still remain unwritten without the warm welcome of the London School of Economics and Political Science ISIGgroup, where I finished vital parts of this thesis in early 2010.

From the Helsinki School of Economics (not Aalto ECON), I would like to thank some of my fellow travellers: Miira Juntumaa, Jenni Peuranto, Sami Relander, Johanna Bragge, Petri Hallikainen and everyone else in ISS. I am also grateful to several philosophers: Markus Neuvonen, Otto Bruun, Jenni Sahramaa, Sampsa Hario, Reima Launonen, Pasi Hario, Aura Kostiainen, Jouni Westling, Jukka Pajarinen, Konsta Nikkanen, Heikki Hyppänen, Tero Heikkinen, Alexander Volkov and others. Special thanks to Roni Saari for so much help with my English.

Certain individuals have kept me in contact with not-so-virtual struggles and fighting the power: Jenni Uljas, Dan Koivulaakso, Riikka Taavetti, Jorma Pikkarainen, Jussi Saramo, Veli-Matti Kuntonen, Jenni Korkeaoja, Johanna Rojola, Laura Tuominen, Jukka Peltokoski and Ritva Pitkänen. In addition, warm thanks to the good people of Electronic Frontier Finland, especially Ville Oksanen, Tapani Tarvainen and Jukka Huhta.

For funding, my thanks go to TEKES, GEBSI-doctoral school, GRAMIS doctoral school, Kauppakorkeakoulun tukisäätiö, Liikesivistysrahasto and my current employer, CKIR.

Most important of all, I want to thank my family who make all this worthwhile: Riitta Ahos, Jan Lindman and Oskar Lindman. And finally Virpi, with love.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ORIGINAL PAPERS

Paper I: Lindman, J. (Unpublished). "From buzz-word to work: Open Source Software". Submitted to an international journal. Earlier version of the paper has been published in the proceedings of UKAIS. Lindman, J. (2010). The Term Open Source Software Renegotiated. UKAIS, Oxford, UK, 23-24.3.2010.

Paper II: Lindman, J., Rossi, M. and Marttiin, P. (2010). "Open Source Technology Changes Intra-Organizational Systems Development – A Tale of Two Companies". Proceedings of the European Conference of Information Systems, Pretoria, South Africa 7-9.6.2010.

Paper III: Lindman, J., Juutilainen, J-P. and Rossi, M. (2009). Beyond the business model: Incentives for organizations to publish software source code. In Boldyreff, C., Crowston, K., Lundell, B. and Wasserman, A. (Eds.): Open Source Ecosystems: Diverse Communities Interacting, 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009, Proceedings. IFIP 299 Springer, ISBN 978-3-642-02031-5.

Paper IV: Lindman, J. and Rajala, R. (Unpublished). "Lessons on the FLOSS business learned from the Open Source Software Pioneers". Submitted to an international journal. Earlier version of the paper has been published in the proceedings of ICSOB2010. Lindman, J., Rajala, R. and Rossi, M. FLOSS-induced Changes in the Software Business: Insights from the pioneers. ICSOB, Jyväskylä, Finland, 21-23.6.2010.

PART I: OVERVIEW OF THE DISSERTATION

The first part of the dissertation leads into the theme of Open Source Software (OSS) in organizations and gives an overview of how the dissertation work was conducted. In the first chapter I position the study in the fields of OSS research and institutional theory, then outline my focus on the main objectives related to leveraging OSS in organizational contexts and finally discuss the limitations of the study. In the second chapter, I discuss philosophical perspectives concerning my choice of methodologies, outline my fieldwork and set the scope of my study. Chapter three describes the findings and contribution of the different papers. Chapter four summarizes the results and discusses their implications and further avenues for research.

1. INTRODUCTION

As one of the most often referred to icons of the digital age, the network society and the new digital commons, OSS (Open Source Software) is inherently present in current public discourse, policy discussion and research (Castells and Himanen, 2002; Zizek, 2009), often without a clear understanding of the actual software development practices and the hierarchical constraints imposed on the paid part of the development work in organizations. A plethora of anecdotal evidence suggests a conceptualization of Open Source either as a revolutionary paradigm shift (Raymond, 1999), or as nothing new. Both of these polar opposites fail to provide a basis for building explanations of what happens in a software production organization when it engages in OSS technology (tools and practices) and what are the consequences this creates.

In 1999, a group of Open Source enthusiasts, including Eric Raymond (1999) and Bruce Perens (1999), wanted to make Free Software business credible. To this end, Raymond published the book *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. The tactic of enrolling the twin audiences of enthusiast OSS developers and business managers proved successful (Fitzgerald, 2006). The drawback of the tactic was that the term OSS, in a commercial setting, remained elusive and fluid (Hauge et al., 2008; Von Krogh and Von Hippel, 2006).

As I was struggling with choosing an exact topic for my dissertation, one of my early respondents (the CEO of a very small start-up Open Source Software company) recounted in his autobiographical account about how his company had evolved: "*Originally Linux was only a tool for us as starting software entrepreneurs. But later, when we got more interested about the entire philosophy, we joined the Free Software movement. We were reading Eric Raymond's Cathedral and Bazaar in 1999 and started thinking how we could build our business model and business case on that book.*" This quote sent me thinking of how an ICT innovation travels from a book to something that restructures the software production practices of a company, and ultimately, the structure of a whole industry.

No matter how welcome, the proclaimed Open Source Software revolution never came to be. Instead, we observe a steady growth in the industrial use of OSS (Fitzgerald, 2006) and OSS-like practices for software production (Linden et al., 2009; Santos, 2008). Companies that struggle to make their software assets public obviously cannot gain their revenue from software license sales (Sharma et al., 2002; Osterwalder et al., 2005). However, mixing open and proprietary product strategies offers potential to many companies (Fosfuri et al., 2008). In addition to OSS use, companies have also changed their internal software production based on the lessons of the OSS world (Wesselius, 2008; Gurbani et al., 2010).

During 2005-2008, I participated in a joint industry-academia research project on the topic of software commodification (ITEA-COSI, 2008). As the project reached the empirical stage, I was observing, to my growing unease, quite a clear gap between the scientific term OSS, the actual local diffusion of OSS technology and the changes it provoked in organizations.

In my dissertation, I decided to investigate the complex interplay between local negotiation of the fluid term OSS and the consequent structural changes in selected companies. The focus of this dissertation is on organizational change related to the adoption of OSS technology (tools and practices) and on arrangements mediated by employment, or other, contracts. My focus is not on individual developers.

The main contribution of this research is an attempt to clear common misconceptions related to OSS especially in organizational settings. I review the popularized writing about OSS as a specific kind of critical research (Lyytinen, 1992), that is, research driven by a social motive and orientated towards certain policy changes. The term OSS is often used as a buzz-word, which means that it has several meanings. This kind of rhetorical room for manoeuvring, or even inconsistency, in a term (Astley and Zammumoto, 1992, p. 451) helps provide common reference points for those who want to actually change the software production practices of their organizations. However, there are two potential problems related to this fluidity of the term: 1) as well as help, it may hinder organizational efforts to benefit from OSS tools or practices, or 2) more academically, the use of words which have an unclear area of application or reference phenomena is not scientifically accurate.

My aim is not to try to provide the "best" business model or licensing model or hybrid model for leveraging OSS. More modestly, my intention is to point out that the situation in an organization is too complicated to be solved with one software production model. There are several different ways in which research on OSS technology can help organizations, but this research needs to be based on empirical investigation and has to value the nuances of change.

1.1 THEORETICAL POSITIONING OF THE STUDY

The aim of my thesis is to describe the changes that occur when organizations adopt Open Source Software technology in their software development. There are two alternative processes of adopting this technology: inbound and outbound OSS. Furthermore, I am interested in how these changes relate to the local renegotiation concerning the term OSS.

The issue can and has been approached from several different theoretical directions. Combining previous multidisciplinary literature concerning OSS with a particular stream of institutional theory, this study investigates how the term is carried, how it enters organizations, how the local processes of negotiation occur and what are the structural changes not only in an individual company but also in a wider organizational field.

1.1.1 Main concepts of the chosen theoretical perspectives

The term Open Source Software was coined by Raymond (1999) and Perens (1999), but draws on a long stream of discussions, going back to the question of the nature of software as a commercial product, and including, but not limited to, terms like packaged software (Xu and Brinkkemper, 2007), free software (Stallman, 2002; Williams, 2002), FLOSS (Ghosh et al., 2002), commercialization of software and the break towards commons-based peer production of digital goods (Benkler, 2006; Ostrom, 1991). The term was contested from the beginning, as it was criticized by both Free Software enthusiasts (Stallman, 2002; Stallman, 2009) and incumbent software companies (Szczepanska, 2005).

The main theoretical term is Open Source Software (OSS). Literature recognizes that OSS is "not a precise term" (e.g. Gacek and Arief, 2004). I use it as an umbrella term to refer both to the licensing method and the software artefact (including the source code). I try to use the term OSS technology when I want to emphasize the links between tools and practices in software production in Raymond's (1999) writings and in popular OSS projects on the internet. OSS practices are here understood as practices that emulate how development takes place in an OSS community (technical infrastructure enabling communication, reward structures, supporting work and knowledge transfer). OSS practices often include the use of email (and the archives thus available) as the primary communication tool, the availability of the code from a source code repository, web presence (e.g. Sourceforge), use of CVS (Concurrent Versioning System), and some kind of issue tracker. Thus I follow the OSI (Open Source Initiative) definition where "*Open Source is a development method for software that harnesses the power of distributed peer review and transparency of process*" (http://www.opensource.org). I want to re-emphasize that my definition thereby includes more than just the license of the software (for a thorough discussion of licensing, see Välimäki, 2005).

I focus on the software artefact, its source code and the processes that bring it about. This, in turn, has a direct effect on how companies and industries are formed. For example: the open sourced code of the Linux (operating system) kernel has a direct impact not only on the practices of the organization contributing source code to the joint effort, but also on its distribution and inclusion in software packages, and ultimately on the end user and any program running on the end user's computer.

OSS has its historical roots in the Free Software social movement, which utilizes non-commercial and even anti-commercial principles of collective action and distributed work practices (O'Leary et al., 2002; Williams, 2002; Benussi, 2005). Free software also has several meanings (Stallman, 2002). This is because the English word "free" can have the meaning of zero-cost, whereas what the proponents are more often referring to is free as in "freedom of speech". The term FLOSS covers all Free/Libre/Open Source Software, but I opt to use OSS since my main interest is in commercial actors, who normally use the term OSS. All in all, for the purposes of this study, any one of these definitions would be suitable.

The on-going discussion about the benefits of software commercialization and the hacker subculture dates back to the beginning of the micro-computer age and IBM's decision to unbundle hardware and software (for a thorough historical account, see e.g. Weber, 2004). Voluntary cooperation-based collective action systems in many cases involve some form of public or semipublic good (Heckathorn, 1996). Public goods offer the participants in a network collective benefits that are 1) non-excludable, in that they are available to all network partners, and 2) jointly supplied, in that partners' uses of the goods are non-competing (Udéhn, 1993). In OSS development communities, Open Source Software plays the role of a public good and underlies collective action towards voluntary cooperation, interaction and goal setting of heterogeneous developer incentives.

The software business can be divided into a primary and a secondary software industry. The primary software industry refers to commercial companies that develop, maintain and publish their software (BMBF, 2000; Rajala, 2009; Xu and Brinkkemper, 2007), while the secondary software industry means companies that include software as part of their offering (BMBF, 2000). These boundaries are far from exact, but here serve to limit the scope of this study. An additional problem comes from the fact that software companies, for example in Finland, are categorized under ICT companies. Rönkkö et al. (2007) approximate that in 2006 Finland had about 1000 software product companies, while Rajala (2009) estimates their number in 2009 to be 1355. The software business has been moving towards a service dominant logic (Vargo and Lush, 2004) and open innovation (Von Hippel and Von Krogh, 2003; Henkel, 2009). Strategic software innovations serve as barriers of entry to competitors and are valuable assets for the company (Chesbrough, 2003). This kind of proprietary innovation development leans on keeping the technological progress secret (Meyer, 2003), as opposed to a more open innovation environment, where to remain competitive means leaning on using external resources.

Concepts used to describe OSS inspired practices within one organization include Corporate Source (Dinkelacker et al., 2002) and Inner Source (Linden et al., 2009; Lindman et al., 2009). Open Source can also be considered as a sourcing strategy. Opensourcing may be defined as a governance model where software development tasks are opensourced to an unknown workforce, including approaches like liberation and commercialization (Ågerfalk and Fitzgerald, 2008). Liberation means releasing a previously proprietary software asset under an OSS license, whereas commercialization is a sequence where a company evolves around an existing open source product, such as the MYSQL database (but see also Fosfuri et al., 2008). Another way to characterize the heterogeneity of leveraging OSS in organizations is to divide the processes into inbound and outbound OSS (Fink, 2003). The idea behind these concepts is that OSS can be used as inbound, which means moving assets (software, contribution, software code etc) from the public domain into the organization. Outbounding, on the other hand, means moving assets from an organization into the public domain, for example by publishing the source code on the internet. These two processes form the core of the structure of this work.

1.1.2 Open Source Software Research

Several recent and comprehensive literature reviews on OSS show wide proliferation of the term, not only in Software Engineering and Information

Systems Science (Hauge et al., 2010b), but also in many related fields such as Psychology, Law and others (e.g. Gacek and Arief , 2004; Von Krogh and Von Hippel, 2006).

Hauge et al. (2010b) conducted a systematic literature review on the different types of OSS adoption in organization. The review was built on an extended a classification of OSS adoption proposed by Ziemer, Hauge, Østerlie and myself (2008). The results of the literature review (Hauge et al., 2010b) indicate the heterogeneity of the ways in which companies approach OSS, and shows a lack of context in the research on OSS adoption as well as highlights the lack of empirical research on OSS adoption in organizations. Only 59 papers of a total sample of 1540 journal papers on software engineering and information systems (about 4%) matching the keyword "Open Source" actually discussed OSS adoption in an organizational context.

Another notable systematic literature review was conducted on conference proceedings papers published at an international conference on Open Source Systems (Stol and Babar, 2009). Although the quality of the reviewed conference papers is naturally no match for the published journal articles, the analysis shows a similar heterogeneity and development trajectory of the field as did the review by Hauge et al. (2010b). The authors note that only 18 out of 219 (or 8.2%) of the published conference proceedings on OSS were empirical studies in an organizational context.

These two recent literature reviews imply a need for more empirical research on the interplay of OSS and organizations. Instead of replicating the reviews, I reviewed a list of 21 information systems journals (see APPENDIX 1). The list of publications considered was a combination of the OSS journals listed by Hauge et al. (2010b) under "OSS, Information Systems and Management" and the top AIS journals proposed by Rainer and Miller (2005).

In the first phase I searched these journals for the term "Open Source" from the articles' full-text versions. A total of 1337 published articles were found. In the second phase I browsed through the titles and abstracts of these articles to determine whether the article's research concerned primarily with Open Source or whether Open Source was mentioned for some other reason. If this was not obvious from the abstract, I read the entire article. This gave me 230 published articles that focused on OSS. In the third phase, I reviewed these articles to determine if the focus of the article was on empirical research in an organizational (commercial) context. The result of this exercise was along the lines proposed by Hauge et al. (2010b): only 54 of the 1337 "Open Source" articles (4%) discussed OSS in an organizational context.

I do not claim that my analysis covers the entire body of knowledge on OSS in Information Systems Science, but even this limited sample shows that there is less research interest in organizations and more research emphasis on community-driven development and on the LAMP-stack (Linux, Apache, Mozilla, Perl/Python) (Østerlie and Jaccheri, 2007). Thus, based on these literature reviews and numerous calls to investigate OSS in an organizational context (Fitzgerald, 2006; Santos, 2008), I conclude that there is a research gap. This dissertation aims to address that gap.

OSS research can be characterized in several ways. Von Krogh and Von Hippel (2006) offer one such characterization by dividing the research into three groups: 1) motivations for contributions, 2) governance, organization and the innovation process and 3) competitive dynamics. Building on their work and on my literature review on OSS in organizations, I propose the following characterization of prominent OSS research themes from the chosen organizational perspective (TABLE1).

Prominent OSS research themes					
Research theme	**Developer motivation**	**Structure of community**	**Knowledge transfer model**	**Company - community relationship**	**OSS licensing and business models**
Topical research question	How to motivate OSS developers (often working without compensa-tion)?	How are communities structured, e.g. in terms of contribu-tion and communica-tion?	How is knowledge transferred despite organization al borders?	How should companies build their relationship toward OSS communities and developers?	How should companies structure their revenue models and license their products?
Key concept	Developer	Community	Open innovation	Community management	Business model
Reference discipline	Psychology, Economics	Sociology (esp. network theory and SNA)	Economics, Policy studies	Management and Information Systems Science	Management, Law
Example studies	Lerner & Tirole (2002); Hars and Ou (2002); Mustonen (2003); Hertel et al. (2003); Ke and Ping (2010)	Koch and Schneider (2002); Lin (2004); Shah (2006); Crowston and Howison, (2006); Bach and Carroll (2010)	Kogut and Metiu (2001); Von Hippel and Von Krogh (2003); Lanzara and Morner (2005)	West, (2003); Dahlander and Magnusson (2005, 2008); Shaikt and Cornford, (2010); Mehra et al. (2010)	Hecker (1999); Välimäki (2005); Osterwalder et al (2005); Fitzgerald (2006); Rajala et al. (2006); Casadesus-Masanell and Ghemawat (2006); Bonaccorsi et al. (2006); Nagy et al. (2010)

TABLE 1: Prominent OSS research themes

Different research streams have focused on the economic and psychological incentives of the developers (Hars and Ou, 2002; Ke and Ping, 2010), structures of the communities (Crowston and Howison, 2006), more open knowledge transfer (Von Hippel and Von Krogh, 2003), orchestrating of the company-community relationship (Dahlander and Magnusson, 2008) and OSS licensing and business models (Bonaccorsi et al., 2006).

Several organizational opportunities constrain organizational OSS implementation. Due to these constraints OSS is leveraged using either "generic OSS business models" (Hecker, 1999; Osterwalder et al., 2005; Fitzgerald, 2006; Rajala et al., 2006), different hybrid models (Sharma, 2002) or management strategies (Shaikt and Cornford, 2010; West, 2003). Competition may also occur at the level of platforms (Economides and Katsamas, 2006) or standards (Bonaccorsi et al., 2006), or companies may use opensourcing as opposed to outsourcing as a means to create sustainable relations in a business ecosystem by means of co-opetition (Ågerfalk and Fitzgerald, 2008).

Literature finds several different ways to engage in OSS (Hauge et al., 2010a) in addition to its use (Ven, 2008). These include, but are not limited to, using OSS CASE tools in the organization (Toth, 2006), integrating OSS into software systems (Li et al., 2009), participating in OSS development (Dueñas et al., 2007) or providing the company's product as OSS (Santos, 2008; Frost, 2007).

Despite the growing literature, uncertainty about what it actually means to adopt OSS still prevents organizations from fully engaging in OSS and creates unrealistic expectations about it (Lacotte, 2004; Goode, 2005). OSS practices have various merits and possess radical potential to alter the software industry landscape (Fitzgerald, 2006; Hauge et al., 2008; Dahlander and Magnusson, 2008).

Prominent research combines different levels of analyses by focusing on the level of both the individual and the organization or a community of developers and even the entire economy. My interest remains on changes in software production at the level of organizations and organizational fields as elaborated in detail in the next section.

1.1.3 Organizational adoption and software production change

The four combined conceptual lenses chosen for this study are: the organizing vision (Swanson and Ramiller, 1997), entrepreneurial institutionalism (Garud et al., 2007; Greenwood and Hinings, 1996), sense-making (Weick et al., 2005) and OSS business literature (e.g. Raymond, 1999; Fitzgerald, 2006; Rajala et al., 2006). Two theoretical underpinnings which bring together these four streams of literature are institutionalism (Scott, 1995) and social constructionism (Berger and Luckmann, 1966) especially in the form developed by the social construction of technology (Bijker et al., 1987).

The backdrop of this dissertation is institutional theory, although I only use bits and pieces of it in the individual articles of the dissertation. Institutional theory is far from a monolithic tradition: different authors have developed it for different purposes (for a more thorough discussion about ”old” and “new” institutionalism, see Powell and Dimaggio, 1991; Greenwood and Hinings, 1996). I use the following definition of institutions as *“multifaceted, durable social structures, made up of symbolic elements, social activities and material resources”* (Scott, 1995, 49). Institutional structures, such as reward and communication structures, are set in motion by regulative, normative and cultural elements or pillars (Scott, 1995). Organizations that ponder any ICT innovation adoption rely on different carriers to develop an internal collective conceptual understanding (i.e. meetings, presentations, workshops, trainings shops and expositions). There is a need to understand how cultural cognitive elements (Scott, 1995) travel or are carried. Scott (2003) divides these carriers into four categories: symbolic systems, relational systems, routines and artefacts.

Besides institutional theory, social constructionism is the other tradition this dissertation draws on. Social constructionism is a (sociological) theory about knowledge in a society (Berger and Luckmann, 1966), which claims that (social) interactions maintain and create knowledge and that institutions have a role to play in this. The social construction of technology (SCOT) is a development of social constructionism (Bijker et al., 1987; Wynne, 1988; Woolgar, 1985). Linking back to social constructionism, the rhetorical fluidity of the term OSS is the starting point of my argument. If we want to understand how the term OSS changes organizational practice, we need to track down the history of the term.

It should also be noted that there are tensions between these two traditions, which are mainly beyond the scope of this dissertation. For now, it is enough to note that Scott (1995, p. 51), for example, notes that "*choice is informed and constrained by the ways in which knowledge is constructed*". Rather than trying to resolve all the conflicting issues in this dissertation, I take these tensions as a healthy sign of the vitality of the traditions. My research thus draws both on institutional theory and social constructionism.

The theoretical framework combines different streams of literature to address the research gap on OSS and organizations. The various components of my research framework are summarized in the table below (TABLE 2).

Theoretical background of the dissertation				
Streams of literature	**Organizing vision**	**Entrepreneurial institutionalism**	**(Collective) Sense-making**	**OSS business**
Seminal works	Swanson and Ramiller (1997)	Garud et al. (2007); Greenwood and Hinings (1996)	Weick et al. (2005); Weick (1995)	Raymond (1999); Hecker (1999); Fitzgerald (2006); Rajala et al. (2006)
Focus	Content of community discourse related to OSS driving change	Organizational change	Collective efforts to understand local OSS adoption	Legitimization of organizational OSS adoption
Relation to institution-alism (Scott, 1995)	Builds on institutional theory focusing on changes in community discourse.	Builds on institutional theory which is critiqued for focusing too much on inertia. Instead, focuses more on organizational change.	Focuses on the cultural/ cognitive pillar of institutional theory.	Previously more focus on individual developers and community-led development.
Relation to social construc-tionism (Berger and Luckmann, 1966; Bijker et al. 1987)	Relies on and develops the ideas of social constructionism in the community discourse concept.	Focuses on the actors and their strategic (also linguistic) use of resources to change organizations.	Sense-making focuses more on (shared) situational negotiations in organizational setting.	Underemphasis on research embracing social construction.

TABLE 2: Theoretical background of the dissertation

No technology or the understanding of it is fixed when it enters an organizational field (Rosenberg 1994; Scott, 1995) or an organization. The vagueness of the exact nature of (ICT) innovation provides its potential adopters flexibility and seemingly common reference points (Astley and Zammumoto, 1992, 451). Even incoherence may help innovations grow (Eccless and Nohria, 1992). Over time, certain shared terms start to form a "legitimated vocabulary" concerning the new ICT innovation (Meyer and Rowan, 1977). Innovations arise in the context of a given institutional field (DiMaggio and Powell, 1983), and information about the benefits of a technology travels from one organization to another. Adoption is also linked to the career paths of the managers and developers responsible for the adoption decision – and the implementation that follows (Mathiassen, 1998).

Swanson and Ramiller's (1997) concept of the organizing vision provides a link between understanding organizational change and interactions over the (local) meaning of the currently adopted ICT innovation. The organizing vision describes how organizational diffusion and legitimization take place, focusing on the shared community discourse as the development engine. Past examples of these kinds of organizational visions include once buzz-wordish innovations such as CASE tools, client-server and the intranet (Swanson and Ramiller, 1997). More precisely, the organizing vision stands for "*a focal community idea for the application of information technology in organizations*". It can be divided into three different aspects 1) interpretation, 2) legitimation and 3) mobilization. (Swanson and Ramiller, 1997.) The organizing vision can be understood as a conceptual understanding of how a rhetorical buzz-word such as OSS is negotiated in an organizational field and what structural changes it brings about.

The organizing vision is developed within, or at least building on, the tradition of institutional theory (Swanson and Ramiller, 1997) to take into account the driving role of community discourse, which also relies on a social constructionist understanding of the ICT innovation – in our case, OSS.

Traditionally institutionalism has focused on continuity (Powell and DiMaggio, 1991; Garud et al., 2007, 960) rather than on incremental or radical organizational change (Greenwood and Hinings, 1996).

Entrepreneurial institutionalism focuses on change and on creating platforms of change (Garud et al., 2007). Thus, entrepreneurial institutionalism accompanies the organizing vision concept in explaining organizational change.

The assumption here is that the actions of organizations reflect the socially constructed understanding of those who make these decisions and act on them (Weick et al., 2005; Weick, 1995). This sense-making requires plausible images (e.g. Greenwood and Hinings, 1993; Oliver, 1991) which are created and maintained in a collective discursive manner, thereby linking the cultural-cognitive pillar of institutionalism (Scott, 1995) to the social construction of situated negotiations. Situational sense-making can accompany the organizing vision (Swanson and Ramiller, 1997), which focuses more on the negotiation (of the term OSS) in an organizational field.

This more nuanced effort in organizational studies on OSS has often been directed towards community-driven development (e.g. Hemetsberger and Reinhardt, 2009; Cornford et al., 2010) or towards coordinating a collective agency (Lanzara and Morner, 2005). OSS in organizational software production, in turn, focused early on generic business models (Hecker, 1999; Rajala et al, 2006) and only later called for more nuanced theoretical development (Fitzgerald, 2006). To summarize my argument: treating OSS as an organizing vision, theoretically rooted in in Scott's institutionalism and in social constructionism, and accompanied by organizational entrepreneurship and sense-making provide theoretical tools to explain organizational OSS adoption – and its consequences. The term OSS is renegotiated when it enters an organization. This renegotiation provokes structural changes in 1) the organizations that adopt OSS technology, but more widely also in 2) the industries that these companies operate in.

1.2 RESEARCH OBJECTIVES AND DELIMITATIONS

The main objective of this study is to help organizations leverage OSS technology. Prior to discussing this, I clarify the path of the term OSS from proponent writings to the daily work practices of an organization, the related local negotiation concerning the reorganization of software production, and the changes in organizations or organizational fields this provokes.

These outcomes are pursued by means of a systematic enquiry that ties together the term OSS and organizational change. Based on fieldwork, I then provide a longitudinal analysis of change concerning several organizations and their practices as they were adopting OSS technology.

I am interested in leveraging the organizational change caused by OSS in software production. Explicitly, the research question of this thesis can be formulated as: *What is the relation between local renegotiation of the term OSS and the organizational change provoked by OSS technology (tools and practices)*? This question can be broken down and reformulated into the following four smaller research questions:

1. *How is the term OSS renegotiated in relation to organizational change?" (*Paper I)

2. *How can implementing OSS technology be leveraged to change development practices and what are the institutional effects of these changes?* (Paper II)

3. *What are the pursued benefits [of releasing the software to the open domain]?* (Paper III)

4. *How do the open source software pioneers perceive the FLOSS-driven changes in their business? (*Paper IV)

The first two questions focus on the processes of inbound OSS and the latter two on outbound OSS. Question 3 focuses on business incentives and question 4 takes a more managerial perspective and summarizes some of the work in the other papers. The questions are formulated in more detail in the original articles.

Answering the research question is instrumental in understanding how OSS can be leveraged in commercial contexts often characterized by planning, control, hierarchy and paid development work. There is a clear tension between the actual characteristics of development work in organizations and the common understanding of OSS originally promoted by Raymond (1999) and Perens (1999).

I attempt to make several contributions. The main goal is to show why the hierarchical organizational settings and paid work change the assumptions derived from the writings of OSS enthusiasts. This will provide a basis for researchers and practitioners alike to better understand the actual phenomenon, and also its limits. Learning from the OSS practices of the pioneers helps understand the benefits that OSS can offer and determine what strategies are useful in harnessing the potential of OSS. Their experience can also benefit everyone interested in software production change.

I hope to demonstrate how the structural changes following OSS adoption are linked to the renegotiation of the term OSS. This will help to build ways to manage, or at least limit, the risks of the renegotiation processes. Some failures in communication may also be avoided. OSS is not a silver bullet that will solve all the acute problems related to software production, but it can help in questioning some of the prevailing ideas about organizational software production – for example, those related to the downsides of certain proprietary strategies which aim to black box software development.

I leave OSS licensing outside the scope of this study (for a thorough legal review, see Välimäki, 2005), since I am mainly interested in the production practices related to OSS. Some of the considerations in comparing different licensing schemes require legal expertise far beyond the more topical areas of this study.

1.3 OUTLINE OF THE STUDY

The main fieldwork phase of this dissertation took place in the course of the ITEA-COSI project, which ran during 2005-2008. Some further dissemination efforts took place after the project had already ended. The year 2009 was dedicated to finishing the research work and the early part of

2010 to writing this dissertation. The bulk of my data emerges from project interviews as described below.

1.3.1 Research design

The chosen aim favours a qualitative approach. Various streams of the literature are summarized based on their contribution to the different papers of this dissertation including the methodologies used, as indicated in the table below (TABLE 3).

Most of my data were obtained as part of the ITEA-COSI research project and consist of transcribed qualitative interview data. In the articles I have used an interpretative case study approach, a comparative case study approach, as well as narrative analysis. The respondents were selected on a per paper basis depending on the specific research question.

The respondents' identities are concealed in all the original papers, but some of the papers do include the names of the partner companies, by agreement, whereas some others do not . I have not included the names of the companies in this introduction to protect the anonymity of those companies that did not want their names to be disclosed. It should be noted that some of the cases have been analysed previously; for example, Wesselius (2008) studied inner source software development from the perspective of building an internal market, and Ågerfalk et al. (2008) analysed the DVTk open source project from the viewpoint of psychological contract theory and opensourcing.

Essay	**Theoretical background**	**Methodology**
Paper I: From buzz-word to work: Open Source Software	Organizing vision; Collective sense-making	Comparative interpretative case study (3 companies)
Paper II: Open Source Technology Changes Intra-Organizational Systems Development – A Tale of Two Companies	Organizing vision,; Entrepreneurial institutionalism	Explorative interpretative case study, respondents from different organizational groups (2 companies)
Paper III: Beyond the business model: Incentives for Organizations to Publish Software Source Code	OSS business	Interpretative case studies (3 companies)
Paper IV: Lessons on the FLOSS business learned from the Open Source Software Pioneers	Collective sense-making; OSS business	Narrative analysis, managerial interviews (5 companies)

TABLE 3: Methodologies used in the original articles

In addition to the above methodologies, I have used secondary sources such as corporate annual reports, articles in professional journals and the trade press, technical user data, and numerous interactions concerning OSS implementation and related subjects on different occasions over the three-year research project.

1.3.2 Relationships of the included articles

The first two articles of the dissertation, papers I and II, emphasize Inbound OSS, that is, software production internal to the company. In the two latter articles, papers III and IV, I outline Outbound OSS, which refers to leveraging OSS published outside organizations. Papers I and III focus more on the reasons for organizational transformation, and papers II and IV on the consequences in companies and organizational fields.

Paper I discusses the different definitions that OSS may have when leveraged inside a company. Paper II describes the consequences of these different definitions as organizations institutionalize OSS tools and practices. Paper III focuses on the different motivations of organizations for publishing their source code assets. Finally, Paper IV describes the consequences of these publishing decisions. The structure of the papers is illustrated below (FIGURE 1).

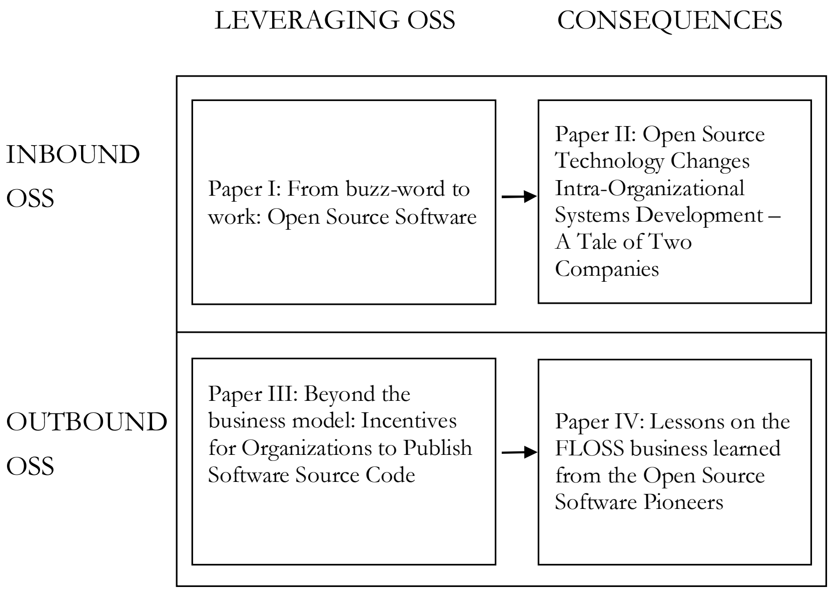
LEVERAGING OSS
CONSEQUENCES
INBOUND OSS
OUTBOUND OSS
Paper I: From buzz-word to work: Open Source Software
Paper II: Open Source Technology Changes Intra-Organizational Systems Development – A Tale of Two Companies
Paper III: Beyond the business model: Incentives for Organizations to Publish Software Source Code
Paper IV: Lessons on the FLOSS business learned from the Open Source Software Pioneers

FIGURE 1: The structure of the original papers

This figure illustrates how the papers relate to each other in the thesis, building on the separation of inbound and outbound OSS. The arrows in Figure 1 show the structural consequences of OSS adoption to the organizations and the wider organizational field in a commercial context.

2. METHODOLOGY AND SCOPE

The focus of the enquiry is on the travel of the term OSS from the writings of enthusiasts into industrial usage and the resulting changes in the organizations and industries. The following section gives an overview of the data collecting and analysis methods, the ontological and epistemological views that inform my research, and a discussion of the reliability and validity of the studies described in the articles.

2.1 METHODOLOGIES

My work does not follow one epistemological and ontological trajectory, as is often assumed to be the case between the research approaches of the individual papers included in one dissertation work. This "waterfall model" of first choosing the ontology, then the epistemology and then finally conducting the research, I would claim, is actually quite rare in the course of any empirical fieldwork related to a doctoral dissertation. I posit that certain assumptions may or may not guide the data gathering and analysis, and when the results are written up it is the outlet and the audience of each individual research paper that determine the rhetoric (Czarniawska, 1998) needed for publication on a per paper basis. The two main traditions from which I draw on are institutional theory (Scott, 1995) and social constructionism (Bijker et al., 1987), as explained in the previous chapter. Moreover, the main theoretical concepts of the organizing vision (Swanson and Ramiller, 1997), entrepreneurial institutionalism (Garud et al., 2007) and sense-making (Weick et al., 2005) in relation to previous OSS business research are a recurring theme in the individual articles.

My position is that of a pluralist (Mingers, 2004), that is, accepting and celebrating a wide variety of paradigms and research approaches on the same topic (Mingers, 2001) – in this case, OSS technology in an organizational context. I do not consider this multidisciplinarity of methodology to be a flaw, if the object of study stays (even roughly) the same during the course of the empirical work (Becker and Niehaves, 2007; Lee et al., 2003). Instead, the different methodologies complement each other and corroborate the findings per paper.

My approach to research is generally qualitative and, considering the cases, explicitly interpretative in all of the original papers (Klein and Myers, 1999; Walsham, 1995). I am interested in building theories of processes rather than variance based models (Pentland, 1999; Markus and Robey, 1988). If we assume that research between interpretative and positivistic case studies forms a continuum, then paper III does contain reasoning and the accompanying rhetoric, often associated with a positivist case study research (Yin, 1989). The paper is based on a model of a clear and identifiable causal linkage between the motivations of the software companies' managers and the publication of source code. The idea is to identify the different variables that affect this publication decision by means of an explorative, in-depth case study.

The case studies in papers I and II are more interpretative, focusing more on local renegotiation over the term OSS and the following structural change, with also a comparative element between the various cases. Paper IV takes the more autobiographical perspective of a software entrepreneur and focuses on the narratives (Rhodes and Brown, 2005) used by different managers to explain the interplay of the change in their organizations and the wider business environment.

To summarize, all of the articles take the managerial and commercial perspective of an individual company on OSS technology and focus on OSS related change.

2.2 DATA AND ANALYSIS

All of the included original papers examine how commercial companies build strategies to benefit from OSS technology. Thus the focus is on individual organizations as cases, but the articles also emphasize that changes in organizations also drive changes in industries (Pentland, 1999;

Markus and Robey, 1988). As noted, the cases were selected among the partner companies of the industrial-academic research project ITEA-COSI (2008).

I worked together with the internal partners of the research project in gathering the data. The context of the ITEA-COSI project enhanced collaboration for a better understanding of the challenges of software commodification. One of my co-authors was actually working for the global concept owner of the (source code portal) service unit described in several of the papers. Still, we could hardly be counted as insiders of the organization, as it was made explicitly clear that our engagement was purely that of researchers. Potentially the researcher role might have changed the perceived roles of the participants in the interviewing situation (Essers, 2009) or the recounted plotline of events (Lanzara, 1991). However, it is unlikely that this had any unwanted influence on the research process.

A common methodological objection to many qualitative studies is to discount the interview responses as only "the opinion of one person" or to reflect only "the opinions of a few". While there are several problems associated with this type of objection, the most obvious is that the objection implicitly assumes that the response might somehow be false. I am the first to admit that responses and respondents can indeed be biased or wrong, yet it is normally unrealistic to believe that the respondent is lying or inventing the entire story just for the researcher (Czarniawska, 1998).

To gather the primary data, I interviewed people from the different case organizations several times: there were two different sets of interviews. The appendices of this dissertation include a list of interviews and the organizational roles of the respondents (APPENDIX II) and the research protocol of the thematic interviews (APPENDIX III). All interviews were tape-recorded and the transcriptions sent back to the respondents for comments.

A wide variety of secondary data accumulated over the three-year span of the research project were used to corroborate the findings and to make sure I had understood the context of the study in the same way as the respondents. Secondary data included corporate documentation, informal discussions, trade press articles and the deliverables of the project as sources to get a clearer picture on how events unfolded.

I applied different methods of analysis on the gathered primary data. The approach was chosen on a per paper basis guided by the addressed research question. The various methods are described briefly in the following chapter and in more detail in the original papers. The primary data wielded the results presented in the following chapter. In addition to the summary of the results, there is also discussion about their implications for theory and practice.

3. REVIEW OF RESULTS

My dissertation includes four original research articles. The papers aim to provide a rich description of the change caused by OSS to answer the research question: *What is the relation between local renegotiation of the term OSS and the organizational change provoked by OSS technology (tools and practices)?* This question can be divided into four subsections: 1) *How is the term OSS renegotiated in relation to organizational change?* 2) *How can implementing OSS technology be leveraged to change development practices and what are the institutional effects of these changes?* 3) *What are the benefits pursued [of releasing the software to the open domain]?* and 4) *How do the open source software pioneers perceive the FLOSS-driven changes in their business?*

Papers I and II outline the processes of leveraging OSS inside organizations and their consequences, while papers III and IV discuss the leveraging processes that rely on the publication of software code.

The first paper discusses the different meanings of OSS and the renegotiation taking place when OSS is adopted. The second paper investigates the link between the term OSS and the subsequent organizational change, focusing especially on structures of reward and communication. The third paper aims to move beyond the business model to understanding the heterogeneous incentives that companies have for the publication of software code. The fourth paper focuses on the experiences of software entrepreneurs and the consequences of the publication decisions.

In what follows, I review the research goals, questions and implementation of each of the individual essays. I then move the focus to the contribution of the different original papers.

3.1 PAPER I: CAUSES OF INBOUND OSS

Lindman, J. (Unpublished). "From buzz-word to work: Open Source Software". Submitted to an international journal.

3.1.1 Research objectives

Paper I focuses on research subquestion 1 of this thesis: *How is the term OSS renegotiated in relation to organizational change?* The aim is to answer the question by focusing on two aspects: 1) *How is the term OSS renegotiated when it enters an organizational field?* and 2) *How does the organization adopt OSS technology (tools and practices)*? The chosen context is inbound OSS.

More explicitly, this is an investigation into local renegotiation over the meaning of the term OSS in three case companies. The renegotiation is claimed to be necessary in order to create a shared understanding of the adopted (ICT) innovation: that is, an organizing vision (Swanson and Ramiller, 1997). The paper analyses local renegotiation by examining how the term OSS is interpreted, legitimated and mobilized in the case companies.

3.1.2 Contribution

The article argues that there is both a theoretical and practical link between the renegotiation of the OSS term and structural change in the organization. One way to understand the intertwined change process is through the concept of the organizing vision. The paper also provides some non-anecdotal recipes for how organizations can benefit from OSS in their software production and operations.

The contribution of this paper is to urge researchers to be more specific in promoting OSS, by describing the nuances of the changes related to OSS adoption, but also by calling attention to the application of the term OSS as a rhetorical device to sell organizational transformation.

The main contribution is summarized below, showing the different renegotiated terms and the related organizational changes (TABLE 4).

Case	OSS term renegotiated	Interpretation	Legitimation	Mobilization
Case 1	Inner source	Changes internal software development	More distributed and collaborative development	Business units gain more control
Case 2	iSource	Hosting and joint practices inspired by OSS	Increased visibility of the software assets internally	Some developers and projects require OSS tool based on its merits
Case 3	De-facto standard	Company provides a standard for a field	Company needs to adopt OSS anyway. Drawing on external resources is cheaper.	More open and autonomous development reduces fears of partiality.
Case 4	OSS entrepreneur-ship	Company providing public source code and services on top of the software	OSS has business benefits related to price, resources and competition.	Groups of founders and customers agree on certain OSS merits.

TABLE 4: Renegotiated OSS term and the related changes

I hesitate to promote, or to criticize, the leveraging of OSS technology in general terms. There is a clear need to first engage an individual organization to find out and analyse the history of OSS in its particular organizational context in order to identify the pro's and con's of OSS adoption. Previous examples of leveraged OSS technology offer valuable lessons in this.

The findings indicate that future research on organizational OSS may benefit from a more critical review of the processes occurring under the term OSS, especially when the aim is to provide an account of how to embrace OSS to transform the software production of commercial organizations.

3.2 PAPER II: CONSEQUENCES OF INBOUND OSS

Lindman, J., Rossi, M. and Marttiin, P. (2010). "Open Source Technology Changes Intra-Organizational Systems Development – A Tale of Two Companies". Proceedings of the European Conference of Information Systems, Pretoria, South Africa.

3.2.1 Research objectives

The aim of paper II is to understand the institutional changes needed in and emerging from the process of changing software practices. The paper builds a conceptualization based on entrepreneurial institutionalism (Garud et al., 2007) and examines the changing relations between different organizational groups in the change process. The addressed research subquestion 2 is twofold: *How can implementing OSS technology be leveraged to change development practices, and what are the institutional effects of these changes?* The goal is to identify the links between 1) the emerging, yet embedded technology, and 2) the underlying institutional reward and communication structures.

Focusing on inbound OSS, the paper approaches the question by investigating two aspects, the first of which was also discussed in the first article: 1) *How does the organization adopt OSS technology (tools and practices)*? and 2) *What are the resulting structural changes in organizations and organizational fields*?

The empirical part of the paper is based on an analysis of the software production of two companies by means of interpretative in-depth and longitudinal case studies.

3.2.2 Contribution

The conclusion of the paper is that the adoption of OSS technology changes the organizational reward and communication structures and generates a wide institutional change, which represents a far more fundamental rearrangement of software production than previously assumed. The main contribution is to link the term to the structural changes that have occurred and show what these changes are like. Below is a summary of the results concerning the changes in reward and communication structures, comparing OSS technology in the classical OSS literature and renegotiated OSS technology in organizations (TABLE 5).

	Classical OSS technology	Renegotiated OSS technology
Reward structure	Mostly voluntary in task assignment, peer recognition, sometimes sponsored development.	Designated projects, contributions based on (employment) contracts and task assignments, development costs divided based on negotiation between actors.
Communication structure	Open discussion email-lists, open message boards, web-presence of projects, open documentation, open training materials.	Intranet, visibility to selected partners who share the development costs.

TABLE 5: Redefinitions of OSS technology (tools and processes)

Furthermore, respondents offered an explanation for the situation. They were inclined to explain the change as the introduction of a software marketplace, which institutes new rewards and a more accurate information and communication structure inside a company (as depicted in FIGURE 2).

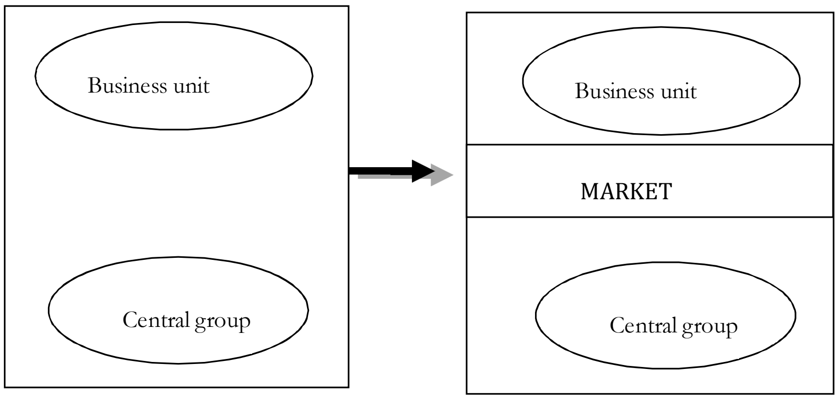

FIGURE 2: Introducing a software marketplace

The change can be seen as instituting a replacement of bureaucratic software organizations with markets inside companies. Implementation of OSS technology in the two case companies followed the neoliberal idea that markets will a priori bring about efficiency. This view seems to resonate well with the bottom-up pull of OSS practices, which empowers developers but also benefits the business units competing with each other for resources.

The main contribution of the paper is to show the potential structural consequences of implementing OSS in software production. It also urges to focus more empirical work on the actual changes in work practices concerning software production in large hierarchical organizations.

3.3 PAPER III: CAUSES OF OUTBOUND OSS

Lindman, J., Juutilainen, J-P. and Rossi, M. (2009). Beyond the business model: Incentives for organizations to publish software source code. In: Boldyreff, C., Crowston, K., Lundell, B. and Wasserman, A. I. (eds.). Open Source Ecosystems: Diverse Communities Interacting, 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009, Proceedings. IFIP 299 Springer, ISBN 978-3-642-02031-5.

3.3.1 Research objectives

This article takes a managerial viewpoint on the process of outbounding OSS. More specifically, the focus is on the different motivations for companies to publish their software source code. The paper contributes to the aspects dealt with in paper 1, but this time in an outbound context: 1) *How is the term OSS renegotiated when it enters an organizational field?* and 2) *How does the organization adopt OSS technology (tools and practices)?* To gain empirical insight from the company perspective on releasing software to the open domain, the paper aims at answering research subquestion 3 of this thesis: *What are the benefits pursued [of releasing software to the open domain]*? In examining on the "pursued benefits", the focus here is on the legitimation and rationale of adopting OSS.

The chosen research methodology is qualitative and interpretative. The objective is to clarify the decision-making situation regarding OSS adoption by identifying and elaborating the variables that need to be taken into account. Three explorative, descriptive case studies were conducted, based on interview data, to gain an in-depth understanding of the various incentives involved.

3.3.2 Contribution

The main contribution of the paper is to show that concentrating on the conventional wisdom related to generic business models and revenue streams leaves opportunities unused. The different benefits pursued by the case companies included the chance to steer an OSS community, obtain development resources, gain cost-savings, improve the quality of the software, increase the trustworthiness of the software and promote standardization in large organizational contexts. All of these incentives are poorly explained if the focus is overly on small (start-up) software companies that build their main revenue source on service and consultancy.

A further contribution is to highlight some of the challenges that publishing source code entails. The main concern of the case companies was that the released software was beneficial mainly to quite a niche market where the possibility of end user contributions is quite low. Another concern was that, this being the case, it does seem hard to build a sustainable OSS community.

3.4 PAPER IV: CONSEQUENCES OF OUTBOUND OSS

Lindman, J. and Rajala, R. (Unpublished). "Lessons on the FLOSS business learned from the Open Source Software Pioneers". Submitted to an international journal.

3.4.1 Research objectives

The main idea of paper IV is to tap into existing managerial experience by taking an entrepreneurial viewpoint on the topic. The contribution of the paper is to answer research subquestion 4: *How do the open source software pioneers perceive the FLOSS-driven changes in their business*? More specifically, the paper examines the same aspects as discussed in paper II, but now in an outbound context: 1) *How does the organization adopt OSS technology (tools and practices)?* and 2) *What are the resulting structural changes in organizations and organizational fields?*

The paper focuses on the perceived consequences of OSS adoption both to the individual organization and the wider organizational field. The software pioneers here refer to software enterpreneurs. Their viewpoint on the changes taking place in their business environment is instrumental in mapping out the consequences of OSS. Building a theory based on service dominant logic (Vargo and Lush, 2004) and open innovation (Chesborough, 2003), I adapt Ramiller and Pentland's (2009) approach to explaining change in organizations. The chosen method is qualitative analysis, using the narratives of software entrepreneurs (similar to Bartis and Mitev, 2008). I am especially interested in the actions taken by different actors as they seek to achieve particular outcomes and attain certain goals, by certain means, and within specific settings.

3.4.2 Contribution

The findings indicate that the effects of FLOSS go far beyond the companies' offerings, impacting, for example, the innovation process, user involvement, resources and revenue models (TABLE 6). To capture the potential and to benefit from external resources in their innovation activity, managers should focus on taking advantage of user contribution already in the early phases of their innovation processes.

Focus	Dimensions	Impact
Goals	User involvement	FLOSS activity emphasizes user involvement in software development and delivery.
Means	External resources	FLOSS activity emphasizes access to external capabilities rather than internal resource ownership.
Actions	Innovation process	FLOSS-based software development urges software innovators to open up their innovation process.
Outcomes	Revenue models	FLOSS-based public goods change the revenue models of companies taking part in OSS development.

TABLE 6: OSS enacted changes

Some enthusiasts of OSS favour the view of source code release and engaged entrepreneurial activity as opposites. This misunderstanding may hinder an understanding of the relation between the entrepreneurial reality and source code disclosure. It also clouds how the actions of entrepreneurs, linked with the OSS paradigm, drive changes in the business environment.

4. DISCUSSION AND CONCLUSIONS

This dissertation sought to answer the primary research question *What is the relation between local renegotiation of the term OSS and the organizational change provoked by OSS technology (tools and practices)*? In addition, there were four subquestions, as listed in chapter 1.2.

In what follows, my findings are discussed in relation to these questions. First I will give an overview of the findings, then move on to the contribution of this thesis, and finally outline some avenues for future research.

4.1 DISCUSSION OF THE MAIN FINDINGS

The main contribution of this thesis are my findings related to 1) the need to move research away from oversimplistic or individualistic myths, 2) the need for more sensitivity in research, 3) the links between OSS, a service dominant logic and open innovation, and 4) the requirement to build organizations that support external input. Addressing these issues will help leverage OSS in an organization.

Reality check. The findings imply that researchers and practitioners should move away from overly simplistic myths concerning OSS in organizations. Researchers who accept the motivations of Raymond (1999) and Perens (1999) for making OSS business credible are at a risk of misunderstanding more organized software production. Overemphasis on community-driven development or the individual developer perspective prevents an understanding of the full extent of the change enacted by OSS in organizations. For example, if developer motivations in a corporate environment are explained without taking into account that most developers are paid to develop the assigned software, there is a risk that one of the main drivers for development will be overlooked.

What is even more risky is that the researcher may assume that developers are always "scratching their itches" or living in a mythical hippy-style community of love – without bothering, for example, to review even developers' mailing lists to find that, despite the community spirit, disagreements and even personal turf battles are often going on. Omitting such aspects would effectively paralyse research on OSS.

Sensitivity. Another major finding is that researchers should become more sensitive about the nuances of paid development work and hierarchies in software production. Organizations that rely on paid software development work may need to change their communication and reward structures to benefit from a more open development style. Researchers who focus too much on openness and too little on the organizational setting may fail to provide relevant research. On the other hand, too much emphasis on the production side, leaving the more open practices without due note, may lead to an underestimation of the true potential of OSS.

There is no one way to adopt OSS practices (Raymond, 1999; Hecker, 1999; Himanen and Castells, 2001; Hauge et al., 2008). Researchers promising to deliver "one size fits all" fail to account for the local renegotiation processes of the term OSS.

Open innovation and the service dominant logic. The findings discussed especially in paper IV show how entrepreneurial activity and OSS are tied together to drive changes towards a service dominant logic (Vargo and Lush, 2004) and open innovation (Vujovic and Ulhoi, 2008), not only inside actual OSS-producing companies, but also in the wider primary and secondary software sectors. Some OSS proponents tend to see engaged entrepreneurial activity and OSS as somewhat opposed to each other. This hinders an understanding of the relationships between the publication of software code and the entrepreneurial reality. In more general terms, however, this transition towards a service dominant logic is not the main topic of this dissertation.

New organizations. The findings further indicate that the term OSS undergoes changes when it enters an organization, but the organization changes as well. Such "renewed" organizations are needed to gain and incorporate new software assets (inbound OSS) or external contribution (outbound OSS).

Acquiring outside contribution requires that the company relinquishes some of the control over its software development (Shaikt et al., 2010). Careful cost-benefit analyses can help to make the risks of major source code disclosure more visible. On the other hand, as the process of adoption includes negotiations over the exact meaning of the term, it remains unclear whether all the benefits and challenges can made visible at the planning stage. However, it is certain that without an informed discussion about the consequences of OSS adoption, organizations are likely to make hasty decisions that can result in underestimating their opportunities.

4.1.1 Inbound OSS

The original papers I and II of this thesis focus on inbound OSS. My main result is to point out the link between the term OSS and the structural transformation of the organization. This link provides a more nuanced understanding of both the renegotiation of the term and the consequent change taking place in the organization.

The evolution of the term can be illustrated using three aspects of OSS adoption: 1) interpretation, 2) legitimation and 3) mobilization. The associated structural changes are related to the organization's communication and reward structures.

To summarize, inbounding OSS is not just a simple process of moving technology or transferring OSS practices, but instead, entails renegotiation processes that challenge the organization’s current practices related to software production. Internally, it seems to cause a paradigm shift (Fitzgerald, 2006) in how software production is organized. However, this shift will not come about without an engagement with the organizational context where the renegotiation of the term is to take place.

4.1.2 Outbound OSS

The focus in the original papers III and IV is on outbound OSS. Their main result is to show that the structural characteristics of software production also shape the software landscape in large organizations. Many organizations would like to combine the agility of small start-ups with the resources of a large company. OSS is sometimes seen to offer several of these benefits.

There is also a clear call not to set OSS and entrepreneurial activity as opposites, but as a new entanglement which drives the software business towards more open innovation and a service dominant logic. Thus, to understand outbound OSS, we need to analyse the reasons and legitimation of the actual code release and the structural changes in software production, but also the wider the consequences both in the primary and secondary software industries. While packaged software will likely retain its large share in the software market, there is also a growing bulk of software released and available for use under an OSS license. This OSS stack is likely to be used both as a platform for collective action as well as to reduce the costs of product and service development for different organizations.

4.2 CONTRIBUTION

This thesis makes three principal contributions. First, it systematizes the use of the terms OSS, inbound OSS and outbound OSS in the context of organizations leveraging Open Source Software. Second, it shows how these terms travel from the writings of enthusiasts to the practices of software production and undergo change in the process. Third, it reveals the scope of the resulting changes in individual companies and in a wider organizational context. These changes are mainly related to an organization's communication and reward structures and move towards a service dominant logic and open innovation in a wider context. Overall, the thesis provides organizations with conceptual tools to build on if they are interested in leveraging OSS.

4.2.1 Theoretical contribution

In addition to providing clarity on OSS in an organizational context, the main theoretical contribution of this work is to link OSS research with the emerging research streams of entrepreneurial institutionalism, open innovation and the service dominant logic. Furthermore, it seeks to move the focus of research onwards from OSS business models (as revenue models) (Fitzgerald, 2006; Rajala, 2006).

Entrepreneurial institutionalism (Garud et al., 2007) addresses the embedded agency problem that has long haunted the institutional literature. OSS as an organizing vision contributes to this literature by providing an example of a linkage between a particular concept and structural transformation.

The open innovation literature (Chesbrough, 2003; Von Hippel and Von Krogh, 2003) frequently presents OSS as an icon of how distributed development processes can work. However, open innovation often fails to address the content of the innovation, implicitly assuming that numerous different innovations can be developed in a similar fashion to OSS. This thesis calls for more clarity by pointing out that OSS in the organizational context requires more engaged research and may thus be a poor example for researchers aiming to provide an understanding of an open innovation in progress.

The service dominant logic (Vargo and Lush, 2004; Rajala, 2009) is directly tied to the changing industry dynamic in many industries. This relates to the question of exactly what part the (software) product plays in the total offering of a company. This is becoming blurred in several industries. In the case of software, the notion of packaged software is being questioned. OSS can well serve as one example which is moving this dynamic further.

Engaged Scandinavian research on Information Systems (Mathiassen and Nielsen, 2008) has had surprisingly little to say about Open Source Software. Although this is true for IS research in general, it raises questions especially since so much of the actual code contribution to the OSS stack originates from Scandinavia. This research stream calls for more Scandinavian efforts, particularly given the strong tradition of engaged, critical and organizationally orientated research in the Nordic countries. Notable exceptions to this rule are Ågerfalk and Fitzgerald (2008), Hauge (et al., 2010b), and Dahlander and Magnusson (2008).

4.2.2 Managerial contribution

My main contribution to the management literature is to engage in the discussion about what it means to leverage OSS in the commercial and organizational contexts. Practitioners have struggled for years with the issues described in this thesis. For them, I want to provide real-world examples accompanied by more theoretical work to help them to build their software production organizations and implement changes as they see fit. There is always a certain amount of local renegotiation over the term, so changing to OSS is far from a disruptive change.

Additionally, I hope to show that not all research is focused on community-driven development or individual developers. How to build relations between OSS communities and companies is and will remain important. What we need to understand is that OSS often requires a structural transformation within the organization producing the software. New institutional arrangements, even new organizational forms, may be required to harness the potential of new ways of producing software. These new arrangements are, however, always mediated by the already existing institutional arrangements. Opening up the production to a network of collaborators makes it necessary to build new capacities and systems of reward and communication. Organizations require changes, but in many cases these changes are probably incremental.

OSS is mainstream, but as all technologies, it has its limitations. Knowing these limitations as well as its strengths will help to build more sustainable models of software production.

4.3 LIMITATIONS AND FURTHER RESEARCH

This research has several limitations. Consisting mainly of explorative work rather than testing of hypotheses, my research efforts are more process-oriented than variance-based. As the idea is to provide a highly nuanced understanding of how companies use OSS in their software production, this qualitative research approach is of necessity quite limited in its number of companies, cases and interviews. This should, however, serve as a call for more quantitative research on the changing dynamics of the primary and secondary software industries.

Moreover, there is a risk of inbuilt bias in focusing mostly on entrepreneurs and developers although it is widely acknowledged that these respondents have a great influence in decision-making processes concerning organizational changes. One future avenue for research is to find more respondents from different levels of the corporate hierarchy to challenge these findings.

I do not feel confident about generalizing the results widely across different organizational fields. This is because the main aim of the dissertation was to provide a nuanced understanding and warn against overly simple models of understanding OSS.

More research is also needed on the relation between communities and organizations. However, this research cannot start from simplistic assumptions concerning communities or companies; for example, the members of a community are often, at the same time, paid developers for companies, which blurs the boundaries and distinctions.

I have not touched upon the issues of control and power (e.g, Bartis and Mitev, 2008) in an organizational setting in any detail. However, there is a clear need for a more critical evaluation of these issues both in a voluntary and a hierarchical context related to software production, especially to virtualization and the increasing shift towards entrepreneurial common to all knowledge intensive production. There is also a lot more research to be done concerning the exact consequences of large-scale OSS adoption as well as the changes, for example, in the management of software production and maintenance processes related to OSS practices.

OSS can also be studied to understand how the nature of work is becoming more distributed, virtual and based on collectively governed goods, even commons. The tension between service-based software companies, small entrepreneurial start-ups and collectively managed software commons offers a wide variety of interesting research questions – not to mention the OSS potential in the public sector, or in developing countries.

REFERENCES

Allen, R. (1983). Collective invention. Journal of Economic Behavior and Organization, 4, 1-24.

Astley, G. and Zammuto, R. (1992). Organization Science, Managers, and Language Games. Organisation Science, 3, 4, 443-460.

Bach, P. and Carroll, J. (2010). Characterising the Dynamics of Open User Experience Design: The Cases of Firefox and OpenOffice.org. Journal of the Association for Information Systems, 11, 12, 902-925.

Bartis, E. and Mitev, N. (2008). A multiple narrative approach to information systems failure: a successful system that failed. European Journal of Information Systems 17, 112-124.

Becker, J. and Niehaves, B. (2007). Epistemological perspectives on IS Research: a framework for analysing and systematizing epistemological assumptions. Information Systems Journal, 24, 4, 663-688.

Benkler, Y. (2006). The wealth of networks: How social production transforms markets and freedom. Yale University Press, New Haven.

Benussi, L. (2005). Analyzing the Technological History of the Open Source Phenomenon. Stories from the Software Evolution.
http://opensource.mit.edu/papers/benussi.pdf [retrieved 15.3.2009]

Berger, P. and Luckmannn, T. (1966). The Social Construction of Knowledge: A Treatise in the Sociology of Knowledge. Anchor Books, Doubleday, New York.

BMBF (2000). Analyse und Evaluation der Softwareentwicklung in Deutschland. The Federal Ministry of Education and Research, Germany.

Bijker, W., Hughes, T. and Pinch, T. (1987). The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology. MIT Press, Cambridge MA.

Bonaccorsi, A., Giannangeli S. and Rossi, C. (2006). Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. Management Science, 52, 7, 1085-1098.

Casadesus-Masanell, R. and Ghemawat, P. (2006). Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows. Management Science, 52, 7, 1072-1084.

Chesbrough, H. (2003). Open Innovation: How Companies Actually Do It. Harvard Business Review, 81, 7, 12-14.

Crowston, K. and Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. Knowledge, Technology and Policy, 18, 65-85.

Czarniawska, B. (1998). A Narrative Approach to Organization Studies. Qualitative Research Methods Series. Sage, Thousand Oaks CA.

Cornford, T., Shaikh, M. and Ciborra, C. (2010). Hierarchy, Laboratory and Collective: Unveiling Linux as Innovation, Machination and Constitution. Journal of the Association for Information Systems, 11, 12, 809-837.

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

Dahlander, L. and Magnusson, M. (2008). How do Firms Make Use of Open Source Communities? Long Range Planning, 41, 629-649.

DiMaggio, P. J. and Powell W. W. (1983). The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. American Sociological Review 48, 147-160.

DiMaggio, P. J. and Powell, W. W. (1991). Introduction. In: Powell W. W. and DiMaggio P. J. (eds.). The New Institutionalism in Organizational Analysis, 1–38. University of Chicago Press, Chicago.

Dinkelacker, J., Garg, P., Miller, R. and Nelson, D. (2002). Progressive Open Source. In the Proceedings of ICSE 2002, 19-25.5., 174-184.

Dueñas, J., Parada, H., Cuadrado, F., Santillán, M. and Ruiz, J. (2007). Apache and Eclipse: Comparing Open Source Project Incubators. IEEE Software, 24, 6, 90-98.

Eccles, R. and Nohria, N. (1992). Beyond the Hype: Rediscovering the Essence of Management. Harvard Business School Press, Boston.

Essers, C. (2009). Reflections on the Narrative Approach: Dilemmas of Power, Emotions and Social Location While Constructing Life-Stories. Organization, 16, 2, 163–181.

Fink, M. (2002). The business and economics of Linux and open source. Prentice Hall, New Jersey.

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

Fosfuri, A., Giarratana, M. and Luzzi, A. (2008). The Penguin Has Entered the Building: The Commercialization of Open Source Software Products. Organization Science, 19, 2, 292-305.

Frost, R. (2007). Jazz and the Eclipse Way of Collaboration. IEEE Software, 24, 6, 114-117.

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

Garud, R., Hardy, C. and Maguire, S. (2007). Organization Studies 28, 957-969.

Ghosh, R., R. Glott, R., Krieger, B. and Robles, B. (2002). FLOSS Final Report – Part 4: Survey of developers. University of Maastricht, The Netherlands. www.Infonomics.nl/FLOSS/report.

Goode, S. (2005). Something for nothing: management rejection of open source software in Australia's top firms. Information and Management, 42, 5, 669-681.

Greenwood, R. and Hinings, C. (1993). Understanding Strategic Change: the Contribution of Archetypes. Academy of Management Journal, 36, 5, 1052-1081.

Greenwood, R. and Hinings, C. (1996). Understanding Radical Organizational Change: Bringing together the Old and the New Institutionalism. The Academy of Management Review, 21,4, 1022-1054.

Gurbani V., Garvert, A. and Hersleb, J. (2010). Managing a Corporate Open Source Asset. Communications of the ACM, 53, 2, 155-159.

Hars, A. and Ou, S. (2002). Working for free? Motivations for participating in open-source projects. International Journal of Electronic Commerce, 6, 3, 25-39.

Hauge, Ø., Cruzes, D., Conradi, R., Velle, K. and Skarpenes, T. (2010a). Risks and Risk Mitigation in Open Source Software Adoption: Bridging the Gap between Literature and Practice, Proceedings of OSS2010, Notre Dame, IN, USA.

Hauge, Ø., Ayala, C. and Conradi, R. (2010). Adoption of Open Source Software in Software-Intensive Industry – A Systematic Literature Review. Information and Software Technology, Information and Software Technology 52, 11, 1133-1154.

Hauge, Ø., Sørensen, C-F. and Conradi, R. (2008). Adoption of Open Source in the Software Industry. Proceedings of the OSS 2008, Limerick, Ireland.

Heckathorn, D. (1996). The Dynamics and Dilemmas of Collective Action. American Sociological Review, 61, 2, 250-277.

Hecker, F. (1999). Setting Up Shop: The Business of Open-Source Software. IEEE Software, 16, 1, 45–51, January-February.

Hemetsberger, A. and Reinhardt, C. (2009). Collective Development in Open-Source Communities: An Activity Theoretical Perspective on Successful Online Collaboration. Organization Studies, 30, 9, 987-1008.

Henkel, J. (2009). Champions of revealing – The role of open source developers in commercial firms. Industrial and Corporate Change, 18,3, 435–471.

Hertel G., Niedner, S. and Herrmann, S. (2003). Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel. Research Policy, 32, 7, 1159-1177.

Himanen, P. and Castells, M. (2002). The Information Society and the Welfare State. Oxford University Press, Oxford.

ITEA-COSI. (2008). http://www.itea-cosi.org.

Klein, H. and Myers, M. (1999). A set of Principles for Conducting and Evaluating Interpretative Field Studies in Information Systems. MIS Quarterly, 23, 1, 67-94.

Kogut, B and Metiu, A. (2001). Open source software development and distributed innovation. Oxford Review of Economic Policy 17, 2, 284-264.

Koch, S. and Schneider, G. (2002). Effort, co-operation in an open source software project: Gnome. Information Systems Journal, 12, 27-42.

Lacotte, J.-P. (2004). ITEA Report on Open Source Software. Technical report, ITEA - Information Technology for European Advancement.

Lanzara, G.F. (1991). Shifting stories: Learning from a reflective experiment in a design process. In: Schon, D. A. (ed). The reflective turn: Case studies on practice and in practice. Teachers College Press, New York.

Lanzara, G.F. and Morner, M. (2005). 'Artifacts rule! How organizing happens in opens source software projects'. In: Czarniawska, B. and Hernes, T. (eds). Actor-Network Theory and Organizing, Copenhagen Business School Press, Copenhagen.

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

Lee, A. and Baskerville, R. (2003). Generalizing generalizability in information systems research. Information Systems Research, 14, 3, 221-243.

Lin, Y. (2004). Hacking Practices and Software Development: A Social Worlds Analysis of ICT Innovation and the Role of Open Source Software (Unpublished doctoral thesis).

Linden, F., Lundell, B. and Marttiin, P. (2009). Commodification of Industrial Software – A Case for Open Source. IEEE Software, July-August.

Lindman, J., Juutilainen, J-P. and Rossi, M. (2009). Beyond the business model: Incentives for organizations to publish software source code. In: Cornelia Boldyreff, Kevin Crowston, Björn Lundell, Anthony I. Wasserman (eds.). Open Source Ecosystems: Diverse Communities Interacting, 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009, Proceedings. IFIP 299 Springer, ISBN 978-3-642-02031-5.

Lyytinen, K. (1992). Information systems and critical theory. In: Alvesson, M. and Willmott, H. (eds.). Critical management studies, 159–180. Sage, London.

Markus, L. and Robey, D. (1988). Information technology and organizational change: causal structure in theory and research. Management Science, 34, 5, 583-598.

Mathiassen, L. and Nielsen, P. A. (2008). Engaged Scholarship in IS Research – The Scandinavian Case. Scandinavian Journal of Information Systems, 20, 2, 3-20.

Mehra, A., Dewan, R. and Freimer, M. (2010). Firms as Incubators of Open-Source Software. Information Systems Research, Articles in Advance, 1-20.

Meyer, J. W. and Rowan, B. (1977). Institutionalized Organizations: Formal Structure as Myth and Ceremony. American Journal of Sociology, 83, 2, 440- 463.

Meyer, P. (2003). Episodes of collective invention. Working paper 368, US Department of Labor, Bureau of Labor Statistics, Washington, DC.

Mingers, J. (2001). Combining IS Research Methods: Towards a Pluralist Methodology. Information Systems Research, 12, 3, 240-259.

Mingers, J. (2004). Real-izing Information Systems: Critical Realism as an Underpinning Philosophy for Information Systems. Information and Organization, 14, 2, 87-103.

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

Nagy, D., Yassin, A. and Bhattacherjee A. (2010). Organizational Adoption of Open Source Software: Barriers and Remedies. Communications of the ACM, 53, 3, 148-151.

Oliver, C. (1991). Strategic Responses to Institutional Processes. Academy of Management Review, 16, 1, 145-179.

O'Leary, M., Orlikowski, W. and Yates, J. (2002). Distributed work over the centuries: trust and control in the Hudson's Bay Company 1670–1826. In: Hinds P., and Kiesler, S. (eds.). Distributed Work. MIT Press, Cambridge, MA.

Osterwalder, A., Pigneur, Y. and Tucci, C.L. (2005). Clarifying business models: Origins, present, and future of the concept. Communications of the Association for Information Systems, 16, 1-25.

Østerlie, T. and Jaccheri, L. A. (2007). Critical Review of Software Engineering Research on Open Source Software Development. Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, Gdansk, Poland, 5.6. 2007.

Ostrom, E. (1990). Governing the commons: The evolution of institutions for collective action. Cambridge University Press, New York.

Powell, W. W. and DiMaggio, P. J. (1991). The New Institutionalism in Organizational Analysis. University of Chicago Press, Chicago.

Pentland, B. (1999). Building process theory with narrative: from description to explanation. Academy of Management Review, 24, 4, 711-724.

Perens, B. (1999). The Open Source Definition. In: Dibona, C. and Ockman, S. (eds.). Open Sources: Voices from the Open Source Revolution. O'Reilly Media, Sebastopol, CA.

Phillips, N. and Hardy, C. (1997). Managing multiple identities: Discourse, legitimacy and resources in the UK refugee system. Organization, 4, 2, 159-186.

Rainer, K. and Miller, M. (2005). Examining Differences Across Journal Rankings. Communications of the ACM, 48, 2, 91-94.

Rajala, R., Nissilä, J. and Westerlund, M. (2006). Determinants of Open Source Software Revenue Model Choices. Proceedings of the 14th

European Conference on Information Systems (ECIS 2006), 12 - 14 June, Göteborg, Sweden.

Rajala, R. (2009). Determinants of Business Model Performance in Software Firms. Doctoral Thesis. Helsinki School of Economics A-357.

Ramiller, N. and Pentland, B. (2009). Management Implications in Information Systems Research: The Untold Story. Journal of the Association for Information Systems, 10, 6, 474-494.

Raymond, E. (1999). The Cathedral & The Bazaar – Musings On Linux And Open Source By An Accidental Revolutionary. O'Reilly Associates, Sebastopol, CA.

Rhodes, C. and Brown, A. D. (2005). Writing Responsibly: Narrative Fiction and Organization Studies. Organization, 12, 4, 467-491.

Rosenberg, N. (1994). Exploring the Black Box. Technology, Economics, and History. Cambridge University Press, Cambridge.

Rönkkö, M., Eloranta, E., Mustaniemi, H., Mutanen, O-P. and Kontio, J. (2007). Finnish Software Product Business: Summary Results of National Software Industry Survey 2007. http://www.swbusiness.fi

Santos, C. (2008). Understanding Partnerships between Corporations and the Open Source Community: A Research Gap. IEEE Software, 25, 6, 96-97.

Scott, R. (1995). Institutions and Organizations. Sage, Newbury Park, CA.

Scott. R. (2003). Institutional carriers: reviewing modes of transporting ideas over time and space and considering their consequences. Industrial and Corporate Change, 12, 4, 879-894.

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

Shaikt, M. and Cornford, T. (2010). 'Letting go of Control' to Embrace Open Source: Implications for Company and Community. Proceedings of HICSS2010, January 5-8, 2010, Kauai, Hawaii.

Sharma, S., Sugumaran, V. and Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. Information Systems Journal 12, 1, 7-25.

Stallman, R. (2002). Free software, free society: Selected essays of Richard M. Stallman. Free Software Foundation, Boston, MA.

Stallman, R. (2009). Why "Open Source" Misses the Point of Free Software. Communications of the ACM, 52, 6, 31-33.

Stol, K. and Babar, M. (2009). Reporting empirical research in open source software: the state of practice. In: Boldyreff, C., Crowston, K., Lundell, B. and Wasserman, A. (eds.). Proceedings of the 5th Conference on Open Source Ecosystems: Diverse Communities Interacting, June 3rd-6th, Skövde, Sweden, IFIP Advances in Information and Communication Technology, vol 299/2009, Springer 2009, 156-169.

Swanson, B and Ramiller, N. (1997). The Organizing Vision in Information Systems Innovation. Organization Science, 8, 5, 458-474.

Toth, K. (2006). Experiences with Open Source Software Engineering Tools, IEEE Software, 23, 6, 44-52.

Udéhn, L. (1993). Twenty-Five Years with the Logic of Collective Action. Acta Sociologica, 36, 239-261.

Walsham, G. (1995). Interpretative case studies in IS research: nature and method. European Journal of Information Systems, 4, 74-81.

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

Weick, K. (1995). Sensemaking in Organizations, Sage Publications, California.

Weick, K., Sutcliffe, M. and Obstfeld, D. (2005). Organizing and the Process of Sensemaking. Organization Science, 16, 4, 409-421.

Ke, W. and Zhang, P. (2010). The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development. Journal of the Association for Information Systems, 11, 12, 784-808.

Wesselius, J. (2008). The Bazaar inside the Cathedral: Business Models for Internal Markets. IEEE Software, 25, 3, 60-66.

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

Williams, S. (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software. O´Reilly, Sebastopol, CA.

Woolgar, S. (1985). Why Not a Sociology of Machines? The Case of Sociology and Artificial Intelligence. Sociology, 19, 557-572.

Wynne, B. (1988). Unruly Technology: Practical Rules, Impractical Discourses and Public Understanding, Social Studies of Science, 18, 147-167.

Vargo, S. L. and Lush, R. F. (2004). Evolving a services dominant logic. Journal of Marketing, 68, 1-17.

Ven, K., Verelst, J. and Mannaert, H. (2008). Should You Adopt Open Source Software? IEEE Software, 25. 3, 54-59.

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, March-April.

Von Krogh, G. and Von Hippel, E. (2006). The Promise of Research on Open Source Software. Management Science, 52, 7, 975-983.

Vujovic, S. and and Ulhoi, J P. (2008). Online innovation: the case of open source software development. European Journal of Innovation Management, 11, 1, 142-156.

Välimäki, M. (2005). The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry. Helsinki University of Technology, Helsinki, Finland.

Xu, L. and Brinkkemper, R. (2007). Concepts of product software. European Journal of Information Systems, 16, 531-541.

Ziemer, S., Hauge, Ø., Østerlie, T. and Lindman, J. (2008). Undertanding Open Source in an Industrial Context. In: Dipanda, A., Chbeir, R. and Yetongnon, K. (eds.). Proceedings of the 4th IEEE International Conference on Signal-Image Technology&Internet-Based Systems

(SITIS2008), November 30th-December 3rd, Bali, Indonesia. IEEE Computer Society, 2008, 539-546.

Žižek, S. (2009). First as Tragedy, Then as Farce. Verso, London.

Ågerfalk, P. and Fitzgerald, B. (2008). Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy. MIS Quarterly, 32, 2, 385-409.

APPENDIX I: SYSTEMATIC LITERATURE REVIEW

Publication	Database	All	OSS	Org
Information Systems Journal	Wiley Interscience	42	14	3
Journal of Database Management	Proquest	9	7	2
Journal of Industrial Economics	Wiley Interscience	4	2	0
Knowledge Technology and Policy	Springerlink	32	13	2
Long Range Planning	Sciencedirect	32	4	2
Management Science	Proquest	13	13	1
MIS Quarterly	Proquest	4	3	2
MIS Quarterly Executive	Misque.org	1	1	0
MIT Sloan Management Review	Proquest	7	3	3
Organization Science	Proquest	2	2	2
Research Policy	Sciencedirect	75	36	7
Communications of the ACM	ACM digital library	355	43	14
Informations Systems Research	Informs	41	10	3
Journal of Management IS	Mesharpe	4	2	0
Harvard Business Review	hbr.org	14	2	0
Decision Sciences	Wiley Interscience	11	0	0
Decision Support Systems	Sciencedirect	84	6	1
ACM Tr. on Information Systems	ACM digital library	20	2	0
IEEE Tr. on Software Engineering	IEEE Explore	164	9	2
IEEE Software	IEEE Explore	385	51	8
Information & Management	Sciencedirect	38	7	2

APPENDIX II: PRIMARY DATA

Paper	Case	No. of interviews	Position in the company
Paper I, Paper II	Case 1: Inner Source	2	Development Manager
Paper I, Paper II	Case 1: Inner Source	2	Program Manager, Medical Imaging Platform
Paper I, Paper II	Case 1: Inner Source	1	Business Architect
Paper I, Paper II	Case 2: Isource	2	Global concept Owner
Paper I, Paper II	Case 2: Isource	2	Service Manager, CMS in Business Infrastructure
Paper I, Paper II	Case 2: Isource	1	Senior R&D engineer
Paper I, Paper III	Case 3: DVTk	2	Interoperability Program Manager
Paper I, Paper III	Case 3: DVTk	2	Project Leader
Paper I, Paper III	Case 3: DVTk	1	Developer, Test Specifications
Paper I, Paper IV	Case 4: Dulo	1	CEO
Paper I, Paper IV	Case 4: Dulo	1	Marketing manager
Paper I, Paper IV	Case 4: Dulo	1	Developer
Paper IV	Case:5 Tripod	1	CEO
Paper IV	Case:5 Tripod	1	Marketing manager
Paper IV	Case:5 Tripod	1	Developer
Paper IV	Case:6 Yoga	1	CEO
Paper IV	Case7: OurDB	1	CEO
Paper IV	Case7: OurDB	1	CTO
Paper IV	Case8: Nemesis	1	CEO
Paper IV	Case8: Nemesis	1	CTO
Paper III	Case9: ESI	1	Developer
Paper III	Case10: Nokia	1	Senior Software Specialist (OSS)

APPENDIX III: OUTLINE OF THE THEMATIC INTERVIEWS

Interview questions

1. Offering

What is the target market of the software?

What kind of software do you offer?

What kinds of services are offered to complement the software?

For what does the customer pay? What does he/she buy?

What type of licensing is used (GPL, LGPL, etc.)?

Are there different offerings for different customer groups/segments?

What is the distribution method of the product?

How the end-users get the product?

How do you view OSS components? Benefits? Drawbacks?

How do you adapt to different versions of your product?

2. Resources

What are the key internal resources and capabilities possessed by the company?

What resources are needed in the innovation and product development activity?

What resources are needed in the commercialization activity?

What kinds of resources are obtained from the OSS community?

How would you characterize opportunities involved with these resources?

How would you characterize threats involved with these resources?

3. Relationships

How do you perceive the OSS community?

How does OSS community effect your decision-making (technological or business)?

Who are your key partners in the OSS community? Why?

What kinds of relationships exist with the members in the OSS community?

Who are the key commercial actors in your business network?

What activities do your key partners in the OSS community and in the business network perform?

How do you communicate with the OSS community?

How do you stimulate Community involvement?

How have you leveraged community (or plan to leverage community)?

4. Revenue Model

What are the main sources of revenue?

Who pays to you (from whom do you get the revenues)?

How (on what basis) is the product priced?

When do you get paid?

What does it cost to use OSS components?

5. Other questions or comments emerged during the interview

PART II: ORIGINAL RESEARCH PAPERS

Paper I: Lindman, J. (Unpublished). “From buzz-word to work: Open Source Software”. Submitted to an international journal.

Earlier version of the paper has been published in the proceedings of UKAIS. Lindman, J. (2010). The Term Open Source Software Renegotiated. 23-24.3.2010. UKAIS, Oxford, UK.

From buzz-word to work: Open Source Software

Juho Lindman

Information Systems Science, Aalto University School of Economics, Finland.
juho.lindman@hse.fi

Abstract. Implementing Open Source Software (OSS) technology (tools and practices) entails potential for a radical organisational transformation of software production. In order to reap the benefits, companies engage in adoption processes including re-negotiation over the local meaning of the term OSS, followed by organisational structural transformation. We claim that these processes (1. the renegotiation of the term OSS, and 2. organisational change) are intertwined. Renegotiation of the term is needed in order to create a collective cognitive view of what it means to leverage OSS locally. Furthermore, this shared understanding is a stepping stone for the legitimation of and the mobilization to change the current situation, i.e. to adopt OSS technology and the related changes. Based on a literature review of OSS in organisations, we investigate four company cases to outline what kind of renegotiation of the term occurred when companies embraced OSS technology and what were the organisational consequences. Our findings indicate that future research on organizational OSS may benefit from a more critical review of the processes occurring under the term OSS, especially when aiming to provide an account on how to embrace OSS to transform the software production of commercial organizations.

Key words: Open Source Software, Inner Source, Organising vision.

1 Introduction

This paper reports a case study of four cases on how the term Open Source Software (OSS from now on) is renegotiated in relation to an organisation's changing software production. Traditionally OSS includes either a software licensing method or development process characterized by the publication of the source code under an OSI-approved software license. We use OSS as an umbrella term to characterize a practice of opening up organisation's software production and drawing on the development practices iconised by successful OSS projects over the internet (Linux, Apache etc.). Corporate source (Dinkelacker et al., 2003) or Inner source (Linden et al., 2009) are concepts used to describe OSS inspired practices limited inside one organisation.

Research literature does agree on some of the merits of OSS practices and their potential to radically alter the software industry landscape (Fitzgerald, 2006; Hauge et al. 2008;Dahlander and Magnusson, 2008). However, research efforts have an overemphasis on community driven development and the LAMP-stack (Linux, Apache, Mozilla, Perl/Python) (Osterlie&Jaccheri, 2007). Less research effort has been directed towards OSS in organisations and especially related to organisational changes such as “inbounding OSS” (Fink, 2003;Wesselius, 2008).

This paper reports part of the results of a three-year research project using a case study of four cases focusing on understanding the organisational dynamic of OSS technology (both tools and practices) entering an organisation. We seek to answer the question: *“How is the term OSS renegotiated in relation to organisational change?”*

Answering this research question is important, because currently some examples of the adoption of OSS in software production are based on simplistic assumptions of what the local adoption of OSS means. These assumptions may misguide companies to base their decisions on partial calculations and to expect too high "return of investment" from changing their current software production practice. On the other hand, assumptions that appear too naive may also cause companies to underestimate their potential to benefit from OSS practices and thus opt-out some winning competition strategies.

The remainder of this paper is structured as follows: The next section reviews the literature on OSS in an organisational context and especially related on how now ICT innovations are carried in organisations and change their structure. The third section provides background information about the research methodology and chosen case study protocol. The fourth section analyses the cases in detail. The fifth and final part discusses conclusions, limitations and implications.

2 OSS in Organisations

The original coiners of the term Open Source Software focused on the individual developer or entrepreneur “scratching an itch” (Raymond, 1999;

Perens, 1998). Later OSS theorist, coming from a wide variety of fields have used OSS as an icon for a network society (Castels, 2005), political economy (Weber, 2004), as an example of commons-based peer production solving some of the inefficiencies of creating a marketplace (Benkler, 2006), as an example of a cellular form of post-capitalist new order (Bauwens, 2005) or as the new networked digital commons (Zizek, 2009). To summarize, originally OSS researchers were interested on individuals and drew on their own experiences on producing software and later the focus shifted to the (more overtly political) level of societies, sometimes without theoretical or practical experience of software production (and notably from outside the Information Systems community). I am interested in what is happening in the mediating layer, that of an organisation.

Any technology and understanding of its applicability is unstable when it enters the organisational field (Rosenberg 1994) or an organisation. The vagueness of the exact nature of innovation provides its potential adopters flexibility and seemingly common reference points (Astley&Zammumoto, 1992, 451). Even incoherence helps innovations grow, be redefined and to benefit from organisational experiments (Eccless and Nohria, 1992). Over time, certain shared specialized terms and phrases start to form a "legitimated vocabulary" concerning the new ICT innovation (Meyer and Rowan, 1977).

Definitions of OSS in the public press have been directed at the twin audiences of commercial companies (Raymond, 2001;Scacchi, 2007) and of OSS enthusiasts. It is not surprising that literature finds that "OSS is not a precise term" (Gacek&Arief, 2004, 35). This probably holds true also for the wider organisational field (Scott, 1995) as the innovations arise in the context of an institutional field (DiMaggio&Powell, 1983). Information about the benefits of a certain technology travels from one organisation into another. Adoption is also linked to the career paths of the managers and developers responsible for adoption decision - and implementation that follows (Mathiassen, 1998).

There are several organisational opportunities that constrain OSS implementation. Due to these constrains, companies leverage OSS by implementing one of several "generic OSS business models" (Hecker, 1999;Osterwalder et al., 2005) or using different hybrid models (Sharma, 2002) or management strategies (Shaikt and Cornford, 2010). When organisations ponder adoption they rely on different carriers to develop conceptual understanding (for example meetings, presentations, workshops, trainings shops and expositions). In this paper, the focus is on how cultural cognitive elements (Scott, 1995) travel or are carried. Scott (2003) divides these carriers into four categories: symbolic systems, relational systems, routines and artefacts.

Traditionally institutionalism has focused on continuity (Garud, et al, 2007, 960), rather than incremental or radical organisational change (Greenwood and Hinings, 1996). In understanding change, one way to create explanations is to link macro states to the individual actors and then to how their behaviour creates new macro states at a later time (Hedstrom and Swedberg, 1998). One such link between micro-mechanisms that link macro-states is sensemaking (Weick, 2005). Similarly, Swanson&Ramillers' (1997) concept of organising vision provides a link between understanding organisational change and interactions over the (local) meaning of the currently adopted ICT innovation. Organizing vision describes how organisational diffusion and legitimisation takes place, focusing on the shared community discourse as the development engine of change. Past examples of these kind of organizational visions include once buzz-wordish innovations such as CASE-tools, client-server and intranet (Swanson&Ramiller, 1997). I am interested in this diffusion process and choose to view OSS as an organizing vision. My focus is on understanding the link between local renegotiation and changes that occur in organisations.

The fluidity of the OSS term is the starting point of the argument in this paper. The concept of organizing vision stands for a "*…a focal community idea for the application of information technology in organizations*" (Swanson and Ramiller, 1997). An organising vision can be divided into three different aspects, forming the core of my analyses, 1) interpretation, 2) legitimation and 3) mobilization (Swanson and Ramiller, 1997).

3 Methodology and Cases

The focus of this paper is on human interaction in an organisational context (Alvesson and Karreman, 2000), which led to use a qualitative approach and to adopt the protocol of interpretative case studies (Klein and Myers, 1999; Walsham, 1995). The actual field research was conducted as part of a European research project (ITEA-COSI) spanning the range of over three years (2005-2008). The main topic of the research project was the commodification of software, especially in embedded environments. All the case companies were observing some of the benefits of OSS, but were unsure on how to adopt the practices in their own software production. As the project progressed, the researchers became concerned of the fact that the organisations participating to the project seemed to have quite varied viewpoints on OSS and especially on how they managed their changing organisation. Three out of four cases were chosen from among the project partners. It should be noted that cases one and three were from the same company, but from totally different units. The fourth case was chosen from one of the previous research projects, where the author had already established cooperation. Short descriptions of the case companies are listed in below (Table 1). Longer versions of the companies are described in the analyses.

	Short description
Case 1: Inner Source	Implementing distributed collaborative software development practices internally
Case 2: Isource	Implementing more open project hosting and OSS practices internally

Case 3: Dvtk	Controlling the development of an OSS tool
Case 4: Dulo (Pseudonyme)	Releasing OSS for collaborative development

Table1: Short case descriptions

The bulk of the data emerged from semi-structured thematic interviews, which were complemented by formal and informal industry-academia project meetings and joint work on the project deliverables. I gathered information on the history of the cases, the organisational changes occurring over time and current challenges. The topic of the thematic interviews was the change in software production and OSS practices especially in relation to the current organisational opportunities and business models. The interview protocol collected first the basic information about the respondents and the history of the described cases. Business model change was the second topic under investigation. I posed questions about the resources, networks, offering and the revenue models of the organisations. Finally, time was left for discussion on the emerged topics.

The data is partly longitudinal, i.e. interviews occurred at different phases of the project and I interviewed some of the respondents several times over the duration of the project to see whether the description of the process changed over time. I interviewed three people from four organisations and repeated the interviews for two people in three organisations. The interviews were about one hour long. In total, we thus had 18 interviews and over 22 hours of recordings (see Table 2). The interviews were conducted informed about the possible biases caused by both the interview situational roles (Essers, 2009) and the fact that most of the interviews described the past and thus also recreated the events (Lanzara, 1991). The main threat was that the author would be regarded as an insider, which might cause issues about the findings. The author's role as a university researcher was made clear to the respondents to address this threat. Respondents agreed that the project findings would be made public.

In order to make the data more polyphonic I interviewed respondents from different organisational positions. For each case, one respondent was selected from an internal software service unit, one from an internal business unit and one from a developer/user perspective. The respondents per case, number of interviews and respondent positions in the companies are listed below (Table 2).

Case	Number of interviews	Position in the company
Case 1: Inner Source	2	Development Manager
Case 1: Inner Source	2	Program Manager, Medical Imaging Platform
Case 1: Inner Source	1	Business Architect

Case 2: Isource	2	Global concept Owner
Case 2: Isource	2	Service Manager, CMS in Business Infrastructure
Case 2: Isource	1	Senior R&D engineer
Case 3: Dvtk	2	Interoperability Program Manager
Case 3: Dvtk	2	Project Leader
Case 3: Dvtk	1	Developer, Test Specifications
Case 4: Dulo	1	CEO
Case 4: Dulo	1	Marketing manager
Case 4: Dulo	1	Developer

Table 2: Descriptions of the respondents (total 18)

In addition to the thematic interviews, I used secondary data accumulated over the three year span of the research project including corporate documentation, informal discussions, trade press articles and the deliverables of the project as sources to get a clearer picture on how the events unfolded.

4 Analyses and Findings

The transcribed interviews were emailed back to the respondents for comments. All the respondents corroborated that they had been quoted correctly. The interviews also served as data sources for the published project deliverables, so there was an on-going discussion about the meaning of the results, which also informed the analyses in this paper.

In the analyses I focused on how the respondents talked about the term OSS, about the three aspects of organizational vision and organisational change: interpretation, legitimation and mobilisation (Swanson and Ramiller, 1997).

I tabled these aspects and found that there seem to exist several different meanings for the application of the term OSS. Thus I focused the analyses on three different aspects and derived three questions to guide the analyses i.e. to understand better the relation between how the negotiation of the term is intertwined to the organisational change below (Table 3).

Aspects	Analyses	Questions
Interpretation	Focus on how case respondents made sense of the situation	What is OSS technology in the organisation?

Legitimation	Focus on the accepted reasoning for OSS use in the organisation	What are the reasons used in explaining why organisation adopted OSS technology?
Mobilisation	Focus on groups that are promoting and gaining the benefits	What are the benefits of OSS technology to different actors?

Table 3: Research approach. Aspects following Ramiller and Swanson(1997)

The focus of the analyses was directed on the shared community discourse about the organising vision (i.e. OSS) and how the organisation changes during the negotiation. Interpretation was approached by analysing how respondents made sense about the situation. Legitimation was related to the reasoning behind the “generally accepted” notions related to OSS. Mobilisation was linked to what groups were promoting change and their perceived benefits. It was unsurprising that it was often difficult to separate these three aspects when respondents talked about the subject.

4.1 Case 1: Implementing Distributed Collaborative Software Development Practices Internally

The case units’ company is currently employing approximately 31,000 people and offering a portfolio of technical medical equipment systems and the related software. The customer base consists of medical professionals and patients. The company's products have a large installed base and interoperability requirements make changes to existing software expensive. The case describes changes from a centralised software group into a more distributed development setting while promoting visibility of the software assets.

Interpretation. In the earlier setting, software components were developed in a central software group and then integrated into products in different business units. Different business units had unaligned roadmaps, which made it difficult to forecast the necessary workload for the central software group and thus forced business units to wait for the excess capacity of the central group to provide them the, often hurried, software assets. *[...] previously we were always waiting for platform groups to do things for us. [...] If you wanted something done, you would queue up with all the other groups who would need something from the platform. And depending on how our demands were appreciated by the platform group, it would be in the next release or it would never get into release. –Manager*

To solve this issue, business units started to contribute to developing the software assets and to build up their own expertise and then contribute to the shared portfolio. The change was described as moving to an Inner source way of working. *“Internally we have a supply chain, where software components, in an Inner Source fashion, are being supplied to system groups”. –Manager*

Legitimation. *“[...]say you have multiple groups with the past of developing the software on their own island. Now, we suddenly say that we have to work together [...] It is an idea that needs some selling. It is not automatically done. It is not happening in one go. You can talk about it, we need to do this, we need to work for Inner Source.” –Manager.* OSS technology (tools and practices) was seen a way to make this change more legitimate to the managers and developers. *“In our [case] community is not Open Source but much more Inner Source. I mean, what our model is used internally within [our company]?” –Program Manager.*

Already from the beginning, OSS changed meaning, becoming Inner Source, which meant the term was renegotiated to fit the organisational needs of the company. *“On a number of aspects we have a quite formal relationship. We have steering groups, we make agreements, we make project plans, we give commitments to project plans.” –Program manager.* The application of the OSS term variant was now one of a wider visibility of source code between the central group and the business units, but also new division of labour and costs between the units. Work was divided according to the idea that the central group would be responsible for the platform and business units for the developed add-ons, customised and configured the software. The development cost were divided according to contracts between central unit and business units. *“Yeah, it’s an agreement per year, upfront principles. We agree for next upcoming budget year and then on monthly basis we get our investment money”. –Program manager, Central group.*

Mobilisation. The change was managed in a rather top-down fashion and the benefits were captured in many cases in business units rather than the central unit.*“[The community way of working] is most beneficial in business units where there are a lot of different groups who can contribute or use the contribution. It is a way of improving the components which can be used by others, or we think that others can contribute.” –Business architect.* The benefits of a more distributed setting would result in know-how being closer to the business units and thus for the business units to have more say in how the software was developed.

At the time of the interviews, the company was building a new system to allocate the development and maintenance costs between units in a more “just” way. The previous method of “component tax” was not considered adequate and the company was building an internal software marketplace to share the assets in a better way. This might also serve organisational needs to determine the cost of the different software assets, should they be, for example, later outsourced.

It is not difficult to track down how the term OSS changed as it became work practice in an organisation – and how this renegotiation also changed the organisation. The Raymond’s ideals of meritocratic contribution and voluntary task assignment morphed into contracts between business units, budget negotiations and steering group meetings. *We don’t really have a community as such. Of course we have a community in a sense that a lot of* *people working…but it is not an active [OSS] community in a way that you probably mean it. –Manager.*

4.2 Case 2: Implementing more Open Project Hosting and OSS Practices Internally

The case 2 company is a mobile communications company employing about 60000 people. It's customer base consists mostly of operators. The company focuses on the production and maintenance of telecommunication network equipment. The iSource source code portal was created in 2003. The portal enables the use of OSS practices and tools within the case organisation. It includes version control tools (Subversion, CVS), issue tracker, mailing lists (Mailman), forums, and file management.

Interpretation. Originally iSource was a response to the growing need to address the issues related to hosting and the reuse-support of software assets. *"But it's very difficult to find source code that you really can reuse. And if you take it somewhere, will it really work? If you do it by yourself you know how it works. [...] there must be really clear interfaces that you can trust."* -*Senior R&D engineer* iSource portal was intended to (1) build on the familiarity the developers already had with OSS tools and practices, (2) streamline and standardize a transparent set of tools and software development practices, (3) enhance collaboration across units made possible by the wider visibility of the source code. *"Everything that we are putting into iSource is ours. It can be used inside the company freely, but nobody else can use it." –Service Manager*

Currently, iSource's active projects are counted in the hundreds and active users in the thousands. The projects attract global participation. Business units value the version control, quick set-up and the possibility to support agile projects. There are several ways to use the portal. Common examples of use include (1) a transparent version control tool, (2) an internal reuse marketplace for software asset showcasing, and (3) a set of tools supporting collaborative software development practices.

Legitimation. When examining after the fact how the events had unfolded, it is clear that iSource was meant to test the benefits of open source inside one large organisation. *We are using these version control tools from [OSS world] and also other tools [...] but when it comes to the software that we are selling then it is Inner Source. The most important thing is to share the information what we have implemented inside our company. It is then easy to see if someone has already implemented a feature and it would also be easy to take these features into other projects." –Service Manager.* The interpretation of the tool was based on it enabling access to certain good sides of OSS, namely reuse and collaboration. It was legitimated as a proven light-weight portal with certain benefits to those projects and developers who were willing to use it. *[...] but the end customer is not really interested in how we deliver the product, so they are not interested if we are using some Open Source, iSource or whatever, they just want to have good quality product. -Senior R&D engineer.*

Also price and familiarity were mentioned as reasons to favour an OSS inspired solution. *OSS it is of course cheaper for us to use (CVS and Subversion). Engineers that are coming in are quite familiar with OSS tools since they have used them in universities or in OSS projects.” - Senior R&D engineer*

Mobilisation. The mobilisation was also based on voluntarism, but an internal service unit was created to give iSource institutional credibility and to promote it internally. This unit has a steering group, budget and it serves a key user network consisting of iSoure users all around the case company. The business units fund the service unit based on the use of the iSource portal. At the time of the interviews, SLAs (Service Level Agreements) were negotiated with the business units and iSource.

The organization was a light-weight environment compared to some of the more rigid development tools in use at the company. Business units have the autonomy to select whether to use iSource or other internal or external services for their projects. This resulted in a service being driven bottom-up from business units and by individual developers. “*We need to have some named persons behind this service who have the money to pay for the service internally. This has been difficult for a global service, because the interest is coming bottom up. There is no top management layer making decisions to use to the tool.” –Global Concept Owner*

If the answer to the question of what is OSS is determined based on solely on the software license, then iSource is not OSS. However, it is clearly inspired by OSS technology. Technical infrastructure that supports the development process is an almost exact copy of the development process used in OSS communities on the internet. Instead of OSS, the respondents tended to talk about inner source and so we see a development similar to the first case. Under negotiation the term morphed from a published source code and community driven open collaboration into internal collaboration based on development and maintenance costs divided between business units, with some added visibility of the source code. *I think that our community is only a set of projects. –Global Concept Owner.* At the same time, a new software production organisation reflecting certain open source practices inside organization was formed. But, as one respondent noted, unlike in Raymond’s Cathedral and Bazaar “*Ultimately, we are paid by work hours” - Senior R&D engineer*

4.3 Case 3: Controlling the Development of an OSS Tool

The Case 3 company (as noted in Case 1) employs approximately 31,000 people. The customer base consists of medical professionals and patients. The company is developing a DICOM (Digital Imaging and Communications in Medicine) validation toolkit named Dvtk. The DICOM standard makes interaction between different medical hardware and software possible and the validation toolkit is used to test the compliance. A rather autonomous business unit of about 2.5 people was dedicated to the development of the testing tool.

Interpretation. DVTk software was originally launched and developed in 2000 and was originally a proprietary software package developed by two different companies as proprietary software. However, the proprietary license failed, due to fears of partiality. The problem was addressed by publishing the source code of DVTk under an LGPL-license as OSS. The term OSS offered interpretation for the publication of the software and a starting point for a collaboration activity, which aimed to gather outside contribution to the development of the testing tool. *I think the most important benefit [community brings] is that users use the application and provide feedback about bugs they find. - Project Leader*

Legitimation. OSS provided legitimation by showing how the logic of collaboration works in some cases outside a company. The company was moving in a similar direction although it had encountered problems in attracting outside contribution. *Currently the most decision making is still done on…on non-Open Source of way of working. –Project Leader*

The user community could be divided into two groups: those who were the company's customers or using the testing tool and those who were not. The feature requests of the first group seemed to have a priority. *[Development] is driven by the requests of new functionality from our user community. If there is someone who really is expecting, or is a need for certain solution, then at that moment we built new alpha release." –Project Leader* Currently most of the communication with the users happened via the projects web-site, which was also maintained by the same business unit. "The w*ebsite is where the community comes together. They can ask questions on the forum, they can download the software, they can download [software]. If they find a problem they submit problem reports. –Software engineer*

Mobilisation. The organization was mobilised with a promise that if certain steps were taken, the project would gain outside contribution and thus better quality and lowered development costs. These steps included organizational issues like moving and creating development discussion channels to the voluntary developers and creating incentives for outsiders to participate. It should be noted that this process was still ongoing and currently almost all development work took place in the original companies that had established the network. *We don't have anyone else outside these companies that is constantly working on (our product] –*software engineer

The business unit had a steering group, which decided on the strategic issues off-line. *"We have within the project a steering committee in which the main contributors come together. In the steering committee it is decided, what we put related to functionalities in the next beta release." -Project Leader* The funding for the unit came from the different business units, who were benefiting in sales from the offered testing suite.

The publication of the source code led to several changes in how the software production was organised. The change was a change of role from a provider to a joint-developer participating in the community providing the software. The company aimed to ultimately shift most of the development and maintenance to an active user community. The developer community is mainly driven by the original initiators and contributors who serve as gatekeepers.

The term OSS was used to describe software production which changed, during the negotiation, from OSS as a voluntary based open joint collaboration to the direction of a project with a clear management rationale and developed mainly by company employees. The source code is published, though it is mainly provided, hosted and controlled by company employees.

4.4 Case 4: Starting Own Start-Up Company Based on OSS Development and Consulting

Dulo (pseudonym) is a small Finnish FLOSS firm that specializes in developing knowledge management and collaborative learning software, related training, and consultancy. Founded in 1998, company has its headquarters in Helsinki. The company employed three at the time of the interviews, but has grown since. Dulo’s revenue model is based largely on service contracts with public organizations. These contracts cover software updates, new feature development, support, and training.

Interpretation. The main software product of the company is licensed under OSI-approved license and thus the founders did not hesitate to proclaim Dulo as a 100% OSS company. What about the motivation? The founders thought “*starting a company would be a good way to earn some money as enterpreneurs. We started early by providing training (for example Microsoft Office) to schools.” –CEO*. Originally the company was engaged in hobbyist web design and selling of educational services to schools.

The original choice to base company’s revenue model on OSS and focus on public sector schools happened almost without elaborate planning. “*We had been using Linux internally and the server solutions we offered were based on Linux. Our LAN-services included some of our own technology. We got interested about learning environments and to offer more services to schools and thus got into that business.” –Marketing manager*. It should be noted that although the source code is published, most of the expertise remains in Dulo, whose developers also make most of the development decisions.

Legitimation. There are several different justifications for the choice of the licensing scheme and thus the revenue model. “*We decided to make our own [OSS] product. Originally Linux was only a tool for us as starting software entrepeneurs. But later, when we got more interested about then entire philosophy, we enrolled to the free software movement. We were reading Eric Raymonds Cathedral and Bazaar in 1999 [about OSS] and started thinking how we could build our business model on top of that.” – CEO* The respondent clearly identifies the link between Raymond’s book and how they built their business.

“*We considered if we could release it as OSS. But even in the beginning we saw that we could not compete with the big players who were also moving into eLearning.” –CEO*. The big competitors, increasing development cost and gaining credibility in the marketplace all pushed towards drawing on a larger pool of external resources. Respondents agreed that they probably lost some licensing fees because of the license decision, but "*license fees form a very small part of the actual price of the product or the total cost of ownership." –Marketing Manager*.

Mobilisation. In the case of a small start-up company, it is easier to make dramatic changes in the business model, especially if the software product is new. There is a less established user-base and hierarchies than in more established companies. The number of people who need to accept the changes is quite small and consists of enterpreneurs, other founders and of course of customers paying for the software product. "*More ideological reason is that we have gotten into this Open Source spirit. As a software developer, you cannot close out ideological views, although business comes first. We would not have gotten involved and continued building our business on OSS, if we would not have believed in it!" –CEO.* Another group whose reasons and benefits related to OSS were discussed were the software company's customers. *Not all of our customers share the OSS spirit. Some just think: Hahaa, we are getting this for free! But well, that is not the point actually. Someone still of course has to make the effort to make the free software. The point is not to make everything free. –Marketing Manager*

The case shows an example of how Raymonds (1999) ideas of OSS travel and are carried over to the company's software production. But even in the case of a small start-up company we can identify some renegotiation over the term and clear link between the term and how the software production organisation is changed. There is a clear idea that the company controls the development, draws on external resources, and ultimately manages the expertice to develop the software. When we compare how this small company views OSS as opposed to some of the other cases, we can clearly see that they have a common origin, but their application is very different.

5 Discussion

The four empirical cases show how the term OSS is carried to organisations. When entering an organisation, there is local renegotiation over the term. We claim that this renegotiation is linked to the resulting changes in the case organisations. When the OSS term is carried into software production organisations practice, we can see the resulting organisational changes as indicated in the table below (Table 4).

Case	OSS term renegotiated	Organisational changes
Case 1: Inner Source	Inner source	Distributed software maintenance in the business units and increased co-development
Case 2: Isource	iSource	Source code portal, internal service unit, OSS technology used internally

Case 3: Dvtk	De-facto standard	Separate business unit developing and maintaining the test software
Case 4: Dulo	OSS entrepreneurship	New software company, OSS published software product, Network relying on outside contribution and competitive edge based on the know-how.

Table 4: The link between renegotiation and organisational change

Furthermore, we claim that the engine of this change is a collective cognitive view, an organising vision. In order to explain what happens in organisations, our analyses used the three aspects of organising vision: interpretation, legitimisation and mobilisation. The case findings are summarised in the table below per case basis (Table 5).

Case	OSS term renegotiated	Interpretation	Legitimation	Mobilisation
Case 1: Inner Source	Inner source	Changing internal software development	More distributed and collaborative development	Business units gain more control
Case 2: Isource	iSource	Hosting and joint practices inspired by OSS	Increased visibility of the software assets internally	Some developers and projects require the tool based on its merits
Case 3: Dvtk	De-facto standard	Company providing a standard to a field	Company needs to do it anyway. Drawing on external resources makes it cheaper.	More open and autonomous development reduces fears of partiality.
Case 4: Dulo	OSS entrepreneurship	Company providing public source code and services on top of the software	OSS has business benefits related to price, resources and competition.	Groups of founders and customers agree on some of the merits.

Table 5: Cases summarised

For an academic audience, the results indicate that the renegotiated term and organisational change are intertwined, especially if we start from the end results and track the path of the term. Analysis of the cases shows how OSS has several local meanings in organisational software production and that the process of the adoption of OSS can follow different paths. In other words, the word is fluid. This fluidity is probably for a reason: it is easier to negotiate on the meaning of an ICT innovation, if its meaning is a bit vague.

However, analytical scientific work benefits from taking a more critical approach towards the term 'OSS technology' when enacted in a certain organization and how it directly relates to the changing software production. OSS should not be defined in simplistic terms in academic discussion.

To practitioners the paper shows empirically that organisations can approach OSS technology in different ways. One of the key issues seems to be that the success of the diffusion of technology is related to the renegotiation of the term itself. Thus we hesitate to promote, or criticize, the leveraging of OSS technology in general terms, without first engaging the organisation to find out what OSS technology could actually mean to an organisation. The examples of leveraged OSS technology offer valuable lessons.

This paper has several important limitations. The analysis part is limited to the process of renegotiation of the term OSS and describes the organisational transformation quite briefly. This is in part because showing in detail how the organisational change unfolds is, by necessity, quite difficult, especially if we are following how one buzz-wordish ICT innovation carried into use in huge multinational environments characterised by hierarchical constrains, paid development work and constant flux. There still remains work to do in describing what the structural changes are in detail, and how the software production practices and company revenue streams actually change with more open development.

Another avenue for future research would be to focus more on contractual arrangements in loose hierarchies and organisations. This is because, ultimately, the IS community’s socio-technical and Scandinavian engaged perspective (Mathiassen and Nielsen, 2008) is lacking in many OSS related discussions and thus our field is not heard when it should be. OSS discussions are often driven by personal software development experience on one hand and macro-level (societal) understanding of the phenomenon on the other. Heated (political) discussions about the benefits and drawbacks of OSS for a given society do not drive the discussion forward as well as careful analyses of the changed software practices in commercial organisations, especially if these generalised personal experiences or sometimes anecdotal macro-level explanations are accepted at face value to define OSS to be somehow opposed to more closed development practices.

Acknowledgments

Acknowledgements omitted for reviewing purposes.

6 References

Alvesson, M., and Karreman, D., “Varieties of discourse: On the study of organizations through discourse analysis,” *Human Relations*, 53, 2000, pp. 1125-1149.

Astley,G., and Zammuto, R., “Organization Science, Managers, and Language Games,”*Organisation Science*, (3:4), 1992, 443-460.

Bauwens, M., The Political Economy of Peer Production. http://www.ctheory.net/articles.aspx?id=499, 2005.

Benkler, Y., The Wealth of Networks, Yale University Press, 2007.

Castells, M., The Rise of the Network Society, (2nd ed.), Blackwell, 2010.

Dahlander, L., and Magnusson, M., “How do Firms Make Use of Open Source Communities?,” *Long Range Planning* (41), 2008, pp. 629-649.

DiMaggio, P., and Powell W., “The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields,” American Sociological Review 48, 1983, pp. 147-160.

Dinkelacker, J., Garg, P., Miller, R., Nelson, D., Progressive Open Source. In the Proceedings of ICSE 2002, 19-25.5.2002, pp. 174-184.

Eccles, R.G. and N.Nohria, Beyond the Hype: Rediscovering the Essence of Management, Harvard Business School Press, Boston, 1992.

Essers, C., “Reflections on the Narrative Approach: Dilemmas of Power, Emotions and Social Location While Constructing Life-Stories,” *Organization*, 16:2, 2009, pp. 163–181.

Fink, M., The Business and Economics of Linux and Open Source. Prentice Hall, New Jersey, 2003.

Fitzgerald, B., ”The Transformation of Open Source Software,” MIS Quarterly, 30:3, 2006, pp. 587-598.

Gacek, C., and Arief, B., “The Many Meanings of Open Source,” *IEEE Software*, 21:1, 2004, pp. 34-40.

Garud, R., Hardy, C., and Maguire, S., *Organization Studies* 28, 2007, pp. 957-969.

Greenwood, R., and Hinings, C., “Understanding Radical Organizational Change: Bringing together the Old and the New Institutionalism,” *The Academy of Management Review*, (21:4), Oct., 1996, pp. 1022-1054.

Hauge, Ø ., Sørensen, C.-F. , and Conradi, R., “Adoption of Open Source in the Software Industry,” In Eds. Russo, B., Damiani, E., Hissam, S.A., Lundell, B., and Succi, G. Open Source Development Communities and Quality Working Group 2.3 on Open Source Software, volume 275 of IFIP International Federation for Information Processing, Springer, 2008, pp. 211-222.

Hecker, F., “Setting Up Shop: The Business of Open-Source Software,” *IEEE Software*, (16:1), 1999. pp. 45–51.

Hedstrom, P., and Swedberg, R., “Social mechanisms: An introductory essay,” In P. Hedstrom, R. Swedberg (Eds), Cambridge University Press, Cambridge, 1998, pp. 1 - 31.

Klein, H. K. and Myers, M., “A Set of Principles of conducting and Evaluating Interpretative Field Studies in Information Systems”, *MIS Quarterly*, 23:1, 1999, pp. 67-94.

Lanzara, G. “Shifting stories: Learning from a reflective experiment in a design process” In: SCHON, D. A. (Ed), *The reflective turn: Case studies on practice and in practice*. Teachers College Press, New York, 2001.

Linden, F., Lundell, B., and Marttiin, P., ”Commodification of Industrial Software – a case for Open Source,” *IEEE Software*, July/August, 2009.

Mathiassen, L., “Reflective systems development,” *Scandinavian Journal of Information Systems,* (10:1+2), 1998, pp. 67-118.

Mathiassen, L., and Nielsen, P. A., “Engaged scholarship in is research—the Scandinavian case,” *Scandinavian Journal of Information Systems,* (20:2), 2008. pp. 3-20.

Meyer, J. W., and Rowan, B., “Institutionalized Organizations: Formal Structure as Myth and Ceremony,” *American Journal of Sociology,* 83(2), 1977, pp. 440- 463.

Østerlie, T., and Jaccheri, L., A., “Critical Review of Software Engineering Research on Open Source Software Development,” In Proceeding of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, Gdansk, Poland, 5.6. 2007..

Osterwalder A., Pigneur, Y., and Tucci C., “Clarifying business models: Origins, present, and future of the concept,” *Communications of the Association for Information Systems*, 16, 2005, pp. 1-25.

Raymond, E., The Cathedral & The Bazar - Musings On Linux And Open Source By An Accidental Revolutionary, O’Reilly Associates, Sebastopol, CA, 1999.

Rosenberg, N. Exploring the Black Box. Technology, Economics, and History. Cambridge University Press, Cambridge, 1994.

Perens, B., “The Open Source Definition,” in *Open Sources: Voices from the Open Source Revolution* (Eds. Dibona, C., and Ockman, S,), O’Reilly Media, Sebastopol, CA, 1999.

Scacchi, W., “Free/Open Source Software Development: Recent Research Results and Methods,” in Zelkowitz, M. V. *Advances in Computers*, Academic Press. vol. 69, 2007, pp. 243-269.

Scott, W.R., Institutions and Organizations, Sage, Newbury Park, CA, 1995.

Shaikt, M., and Cornford, T., "'Letting go of Control' to Embrace Open Source: Implications for Company and Community," in the proceedings of HICSS2010, January 5-8, 2010, Kauai, Hawaii.

Sharma, S., Sugumaran, V. and Rajagopalan, B., "A framework for creating hybrid-open source software communities," *Information Systems Journal*, 12:1, 2002, pp. 7-25.

Swanson, B., and Ramiller, N., "The Organizing Vision in Information Systems Innovation," *Organization Science*, 8:5, 1997, pp. 458-474.

Walsham, G., "The emergence of interpretivism in IS research," *Information Systems Research*, 6(4), 1995, pp. 376–394.

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

Weick, K., Sutcliffe, M., and Obstfeld, D.,"Organizing and the Process of Sensemaking," *Organization Science*, (16:4), 2005, pp. 409-421.

Wesselius, J., "The Bazaar inside the cathedral; Business models for Internal Markets," IEEE Software, 25 (3), 2008, pp. 60-66.

Žižek, S., First as Tragedy Then as Farce. Verso, London, 2009.

Paper II: Lindman, J., Rossi, M. and Marttiin, P. (2010). "Open Source Technology Changes Intra-Organisational Systems Development – A Tale of Two Companies". Proceedings of the European Conference of Information Systems, Pretoria, South Africa, 7-9.6.2010.

OPEN SOURCE TECHNOLOGY CHANGES INTRA-ORGANIZATIONAL SYSTEMS DEVELOPMENT – A TALE OF TWO COMPANIES

Lindman, Juho, Helsinki School of Economics, Runeberginkatu 22-24, 00101 Helsinki, Finland, juho.lindman@hse.fi

Rossi, Matti, Helsinki School of Economics, Runeberginkatu 22-24, 00101 Helsinki, Finland, matti.rossi@hse.fi

Marttiin, Pentti, Nokia Siemens Networks, Linnoituskatu 6, 02600 Espoo, Finland pentti.marttii@nsn.com

Abstract

This paper explores how two organizations have changed their software development practices by implementing Open Source technology. Our aim is to understand the institutional changes needed in and emerging from this process. The paper develops a conceptualization building on the insights of entrepreneurial institutionalism and concentrating on the changing relationships of organizational groups in the areas of reward and communication. We identify the links between the 1) emerging yet embedded technology and 2) the underlying institutional reward and communication structures. In terms of contribution, we propose to move the Open Source 2.0 research agenda forward by concentrating empirical work on the nuances of institutional change that open source brings forward in large hierarchical organisations.

Keywords: Open Source, Entrepreneurial Institutionalism, Organizational Change.

1 INTRODUCTION

In this paper we study the institutional transformation created by the implementation of Open Source Software (OSS) technology (practices and tools) within traditional development organizations. By OSS technology we don't mean the license of the development software, but the common infrastructural tools used in OSS communities. The tools include concurrent versioning systems, issue trackers, email-driven and archived communication, and web presence, which all support software development practices similar to OSS in creative commons, but in our cases within a single organization.

The authors were involved in a research project on software production structure change in two large international organizations. During the project we observed that previous research on how open source technology is institutionalized failed to account for the process we were part of. OSS literature often assumes a "bazaar" of development in a virtual organization characterized by loose control, openness and community orientation. However, inside a big organization, where contributions came from employees or subcontractors the phenomenon appeared to be quite different. The companies introduced OSS practises and fostered the creation of communities, because it helps to create quality products. This was believed to be caused by looser structure, more open documentation, feedback from the user community and the introduction of agile practises. These development arguments were corroborated by business arguments of partial outsourcing to the developer community, cost savings from using common (sometimes external OSS) platforms and the possibility of creating industry standards through wide availability of the finished products.

The identified phenomenon is important because open source technologies are, 1) adopted in large organizations based on only partial understanding of the nature of the institutional change they enable, drive, or even necessitate, and 2) are not adopted in organizations because their consequences are seen to include unnecessary or unknown risks. We believe that building a conceptualisation based on extensive field work will enable better evaluation of these technologies and their contextual appropriateness.

Therefore our research questions are:

- How can implementing OSS technology be leveraged to change development practises?
- What are the institutional effects of these changes?

To answer these questions, we analyse two implementations of OSS technology within large corporations. Our goal is to build a conceptualisation of what happens in a hierarchical systems development organization when OSS technology is adopted[1]. Informed by the institutional theory on enrolling group interests, we seek to identify the inertia caused by old institutional forces and the changes in reward structure and the developer and manager mindset needed to realize the benefits of more open development. Furthermore, we try to identify the incentives needed to institutionalize the new practises.

This paper is organised as follows. In the second section we review relevant literature on OSS technology. In the third section we develop a conceptualisation informed by institutional theory and

[1] One of the main reasons for companies to adopt OSS technology is their interest in improving software reuse and re-development. At the same time companies are adopting distributed and virtual teamwork practises and changing their software development processes from waterfall to iterative, thus adopting agile practises (about traditional, agile and open source practises in Barnett, 2004). These two changes favour the adoption of OSS tools, but failed to address the challenge of reuse.

especially entrepreneurial institutionalism to explain the transformation. The fourth chapter is about the research approach used. Case findings then demonstrate the links between the embedded technology and the communication and reward structures. In the final section we conclude how OSS technology is leveraged in the case companies' systems development and what the accompanying institutional changes are.

2 REVIEW OF LITERATURE

OSS is used more and more as an integral part of all kinds of products (Scacchi, 2007). The use of Open Source Software -inspired (OSS) development processes is gaining foothold in large commercial organizations. OSS is traditionally defined as software licensed under an OSI certified software license (Raymond, 1999; Välimäki, 2005). OSS practices are practices that emulate how development takes place in an OSS community (technical infrastructure enabling communication, reward structures, supporting work and knowledge transfer). OSS practices often include the use of email (and the archives thus available) as the primary communication tool , availability of the code from a source code repository, web presence (for example Sourceforge), use of CVS (Concurrent Versioning System), and some kind of issue tracker.

OSS has gained industrial credibility as a development style based on distributed and global practices. OSS development is often characterized by a modular software architecture, distributed global development teams, meritocracy, voluntarism, often elaborate decision making mechanisms, and the technical and legal openness of the code which enables code inspection, bug reporting, and maintenance (Fitzgerald, 2006; Fink, 2003). In the first phase of OSS commercialisation companies were interested in ways to directly benefit from the revenue stream created by OSS. Now, in the second phase commercial actors are reviewing ways to leverage OSS products and practices in hierarchical organisations (Fitzgerald, 2006). The main difference between traditional (closed source) and OSS development is that latter can sustain communities as the source code is available. The source code might belong to its developer or the community in a way that prevents traditional software license sales (Dahlander&Magnusson, 2005). However, the availability of the source code outside the organisation is not a prerequisite on implementing practices similar to OSS inside a company (for example, Fitzgerald, 2006).

Organizations are struggling to balance the possibilities of using OSS to the challenges of maintaining OSS systems. The use of information goods created based on voluntarism and not controlled by the providers poses fundamental questions about the sustainability of the solutions. OSS has been successfully implemented in different organizations (Hauge, 2008; Lundell, 2006; Ghosh, 2002). Research on OSS has contributed on boosting OSS business viability by providing “generic business models” (Hecker, 1999) or even “the OSS business model” (Raymond, 1999). While benefiting the understanding of the phenomenon, these research efforts were directed to the heterogeneous OSS research audience consisting of academics, enthusiasts and business people (Ziemer et al., 2008).

Inner source (Linden et al., 2009; Lindman, 2008) and corporate source (Dinkelacker, and Garg, 2003) are words used to describe OSS practices limited inside companies. Often the implementation of OSS starts with these tools, but as “tools are not only tools” their productive application might require fundamental changes in software development (Sharma, 2002). Inside a large organization (Wesselius, 2008) or in a business-to-business setting (Fink, 2003) the fundamental differences between OSS and traditional software are smaller than inside small software companies. The license and corporate policies and processes define how software is acquired, procured, installed, used, maintained and discarded. Furthermore, company guidelines, contracts and/or licenses also define how software is developed, remuneration acquired and benefits divided (Välimäki, 2005).

3 CONCEPTUAL FRAMEWORK

3.1 Institutional theory

Institutional theory views institutions as “*multifaceted, durable social structures, made up of symbolic elements, social activities, and material resources”* (Scott, 2001, p. 49). Institutional structures, such as reward and communication structures, are set in motion by regulative, normative and cultural elements or pillars (Scott, 2001). Institutional theory (Powell&Dimaggio, 1991) has been accommodated to explain change (Greenwood&Hinings, 1996), even though it has been criticized for not mainly focusing on “convergence” (similarity) (Buckho, 1994).

Institutional theory underlines organizations “relationship” between its normative context and the groups' (stakeholders) varying interests inside the organisation. Functionally different groups in organisations are not neutral towards each other, but instead the groups' technical boundaries are reinforced cognitively (Greenwood&Hinings, 1996). Usually groups inside organisations compete for the allocation of resources and aim to transform the division to their benefit. Institutional theory has used the concept of translation to demonstrate the link between meaning and power (Czarniawska, 1996). Translation supposes that different actors are enrolled in order to make changes. Enrolling actors is based on a premise that practices are negotiated locally and become institutionalized as their meanings become shared in organization and across wider organizational fields (Zilber, 2006, 283).

Our approach suggest that while normally the actors and proponents of organizational change truly subscribe to OSS inspired values for the better, “the OSS spirit”, they are also renegotiating the exact meaning of OSS to fit the organisational context (Ziemer et al., 2008). Thus OSS is not a “mere buzzword”, but a justification to an organizational change, an organizing vision (Swanson&Ramiller, 1997). The exact meaning of adapted OSS is renegotiated and implies changes in the allocation of resources and the division of work between units.

3.2 Entrepreneurial institutionalism

Research in institutionalism which focuses on individual and shared agency is called entrepreneurial institutionalism. It is a response to the call for institutional theory to focus more on agency and organizational change (Garud et al., 2007). Work on institutions has traditionally focused on continuity (Garud, et al, 2007, p. 960). In contrast, work on entrepreneurship has focused on change. Inside institutional theory, this contrast of structure and agency has been identified as the paradox of embedded agency (Seo& Creed, 2002, 226; DiMaggio&Powell, 1991). One solution to this paradox is to view structures as platforms for change rather than constraints (Garud& Karnoe, 2003).

Any new technology is a change in status quo with winners and losers. The process of understanding different interest of the different groups to enrol becomes essential to understanding how institutions evolve. The meaning of organizational visions (Swanson&Ramiller, 1997) is renegotiated within boundaries of a certain language community and draw on local discursive resources.

OSS technology is an organizational tool that stresses local issues regarding software production in the context of a certain organization. OSS also provides ways of addressing these issues. It can be seen as a metaphor used in an organisation making sense of its changing business environment to be able to operate in it (Weick, 1995). OSS often offers a promise of a more agile development approach, more contribution, more open discussion and less hierarchy in software development. In short, it poses certain justification, reasoning and enrolment opportunities to a decision-maker faced with difficult decisions concerning reorganization or introducing a new organizational innovation (Van de Ven, 1993).

We draw on the institutional entrepreneurship lens to identify how the meaning of OSS technology changed during implementation and how our two organizations evolved when OSS technology was institutionalised. We aim to provide insight on the process on OSS technology institutionalisation and the accompanying underlying changes. In order to explain the institutionalisation of OSS technology we focus on two structures in the companies: the reward structure and the communication structure. We do not claim that these are separate entities, but interwoven sides of the same structure.

We chose the different organizational groups to highlight their different interest and incentives in the process. The different selected groups (stakeholders) whose interest need to enrolled are 1) the technology provider unit (the central group), 2) the technology user unit (business unit), and 3) the developer/users.

4 RESEARCH APPROACH

The nature of our research problem, human behaviour and interaction, led us to use a qualitative research approach (Seaman 1999; Klein and Myers 1999). We chose a case study approach (Wynn 2001), and adopted the principles of interpretive case studies (Klein and Myers 1999). The two cases were selected among the partner companies of the ITEA-COSI project. ITEA-COSI was a joint academic and industrial project focused on software commodification.

As the main data collection method, we applied semi-structured thematic interviews. We interviewed 3 persons per case organisation in two occasions over a 2-year period. We stopped interviewing after the 10th interview. The first half of the interviews was gathered in 2006 and the second round of interviews was conducted in 2008. Each interview lasted about one hour and focused on the different elements of OSS implementation inside the companies.

The interviewed people represented three different organizational groups, one person from the service provider group, one from the service user group and additionally one from developer/user group. We chose managerial respondents from the business and central groups to gain an understanding of the management rationale for introducing OSS technology. The developers were included to bring in the user viewpoint, although we speculated that the user viewpoint would not yield contrasting accounts concerning the reward and communication structures.

One of the researchers works in one of the case companies and is able to reflect on the organizational context. We also used secondary data obtained in the course of the industry research project such as project descriptions, manuals, portal usage data, documentation and visits to the sites to familiarize ourselves with the setting.

We analysed the interviews by first recounting the organizational history and change as described by the respondents. We circulated the transcribed interviews back to the respondents, so they could correct themselves should they have been misinterpreted.

The systematic analyses were based on recurring themes and pattern matching of themes between different interviews and categorizing the data according to the themes (Strauss and Corbin 1990). We focused on the themes of how the respondents talked about 1) instituting new technology, 2) changes in the communication media and the reward structures between units and individuals, and 3) changes on the different ways the respondents described their group involvement. The authors extracted all the instances where the respondents talked about our themes and report the findings in this paper.

We classified the findings into three areas: 1) how OSS technology is renegotiated to fit the organizational context and how OSS infrastructural tools are used inside companies, 2) how the respondents saw the change between business units and central unit, and 3) how the respondents described the reward and communication structures as both a platform and driver of change.

5 ANALYSIS OF THE CASES

5.1 Philips Inner Source

The offering of Philips Medical Systems (PMS) consists of a wide variety of medical systems, for example X-ray technology, ultrasound, magnetic resonance and information management. The factory preinstalled software is customised and configured, but not sold separately. PMS normally maintains the software for 10 years, which often leads to a large installed base and makes large changes very complicated. PMS is maintaining and developing a large software base including a set of software components reused in all business units.

Historically components were developed in a central software group (Wesselius, 2008). In this configuration it was difficult to manage the different development activities and unaligned roadmaps. Lack of required domain knowledge in the central group made asset reuse difficult.

To solve these two issues, the business units started to contribute to developing new software assets. This would enable the business unit with the best domain knowledge to develop the software and then contribute it to a shared portfolio. Business units would not have to wait for the central group to develop the (often rushed and high priority) asset. OSS technology (tools and practices) was introduced in PMS to legitimate the change.

The division of work was based on the idea that the central group was responsible for the common platform and business units developed add-ons, customised and configured the software. Components are distributed via intranet, email, ftp and CD. Business units choose the components for use, customization and configuration. Different groups offer services to each other (for example support and maintenance) based on agreements between internal customers. Developed software was also made available to other business units. One of the main benefits of a common platform is that it would avoid duplicate work and promote the reuse of software. Co-development activities with business units and central group were favoured in order to benefit from organizational learning.

There were also certain risks involved mainly dealing with the distributed setting. The central group would become more dependent on not only one business unit schedule, but several at the same time. The overall quality would be more difficult to control, if business units would only make stand-alone add-ons. Business unit incentives were also un-aligned as it seems that there is no guarantee that units would actually contribute back and not only use the outcome. This applies also to the maintenance of the software asset and balancing the maintenance between business units. The scenario where one business unit is putting a lot of resources and effort on development and maintenance, but all the business units would use the outcome was considered problematic.

The communication was aimed to be developed as explicit as possible and share information with all the interested parties. Co-development activities required informal discussions between developers, but broader issues were decided in formal settings such as steering groups and operational teams. There were also formal architect meetings and monthly platform group meeting all interested parties could participate in. Information was also posted on the intranet and PMS mailing-lists. A back-channel of communication were so called marketers, who were selected per business unit to promote inner source and gathered in case of problems. Development work is somewhat controlled by steering groups and operational meetings, but mainly development is driven by business groups which need some new functionality.

Philips is building a new system to divide the development costs. The old model was based on centralised component development and component-tax where the central group did not have profit targets. The central group performed maintenance of the components. Component tax was evaluated based on component development and maintenance activities and on an agreed upon roadmap on a yearly basis. Based on the relative amount of component usage and the size of the unit's external sales, the estimated costs are then distributed over the business units. Users of old component versions paid more for maintenance to stimulate use of the most recent versions and to reduce the total burden of maintaining many old versions.

When moving to an inner source approach, old component-tax model does not work properly since it does not promote making contributions to the shared component base. A business unit that contributes a reusable component has to make an extra effort to make the component reusable. Business units have profit targets and investing resources to make components reusable is conflicting with these targets. It wasn't clear which group was expected to perform maintenance for the contributed component or allocate the maintenance resources. If the contributing business unit has to do the maintenance, this will again add costs to the unit. However, making the central component group responsible for maintenance would require this group to build competences for maintaining software components developed by other groups. The central group would be enlarged and take away the domain experts from the business units.

5.2 Nokia iSource

Nokia is the world leader in mobile communications. It is a publicly held company with listings in five major exchanges and in 2004 (prior to the merger of its Network unit with Siemens to form Nokia Siemens Networks or NSN) it's net sales totalled EUR 29.2 billion. iSource is a corporation wide source code portal that enables agile, fast cycle, multi-site software development (Lindman, 2008).

iSource originates from the free version of SourceForge that has been later upgraded to GForge. The web portal integrates a set of tools for use by projects including version control tools (Subversion, CVS), issue tracker, mailing lists (Mailman), forums, and file management. Today both Nokia and NSN have their own corporation wide instances of iSource. Altogether, active users are counted in thousands and scaled by 5 when including passive users.

The main idea behind iSource was to provide a portal enabling visibility of software and the source code inside the company. The goals were to increase individual engineers' awareness of software developed inside the company, and to boost innovation by avoiding the problem of re-implementing the wheel. The Inner Source concept was launched to tackle the challenges of supporting reuse and further cultivation of software assets.

A corporation wide iSource -service was established 2003 by the Nokia IT department to support infrastructure and to promote the portal tool. Service level agreement was made between the IT department and the business units. IT department takes care of the iSource application (including hardware and software, server installations, backups, maintenance etc.) based on the agreed service level agreement with business units. The service costs are shared to business units based on the amount of active users. The user base has been increasing with the help of bottom-up information sharing and leaving passive users out of service costs. Application development, that is, integration of new components, tool upgrades, and other customizations, has been release based, and it has lately turned to follow agile practises (Vilkki, 2009). The budget for application development is renegotiated yearly.

iSource support has been organized based on ITIL model (OGC, 2002a; OGC, 2002b). The iSource service provides basic self-training material or buys training courses from third parties. Overall learning to use iSource relies heavily on the inner community support and nominated persons (key users) that are experts and serve as a first point of contact for users. A steering group decides on the development contents. Members of the steering group are core developers from different business groups. Development is release-based and a project is established for a new release.

The Inner Source process was not in the scope of the service provider and thus such a corporation wide process never existed. The tool was first adopted by leading edge research projects and later by platform projects of the business units. The major drivers have been version control (namely Subversion) and a quick set-up for working with collaborator companies. Also, it is popular among agile projects that tend to select lightweight tools suitable for their purposes. Decisions to use iSource may be made on bottom-up per project basis, and this has been common among research projects. However, adoption in platform projects with legacy tool infrastructure has required a management decision.

iSource case is tool driven corporate wide approach for business units and platform programs Although iSource is now adopted company wide Inner Source practises are scattered each platform program following its own approach. Each unit and program can decide whether and how they use it. There are at least three ways of using iSource. It can be used 1) as an inner source server, where business units can put their project assets and outputs available, so everybody in the company has access to them, 2) as a version control (CVS or Subversion) tool 3) as a set of tools for collaboration and setting up collaborative projects.

6 DISCUSSION

6.1 The meaning of OSS technology is re-negotiated locally

On examining the cases in our study, it seems that OSS technology has become institutionalized in both organizations. New tools have gained acceptance and provided inspiration and familiarity to the developers. Both case companies use OSS tools and processes as a way to promote software projects inside the organization.

At the same time, the meaning of OSS tools seems to have changed to enroll the different stakeholders. In retrospect we can see a process of renegotiating the meaning of OSS to suit the organizational context. The adopted practices do not resemble OSS as understood by the "classical OSS movement": being based on voluntarism, peer-recognition and public discussion. Instead, institutionalized OSS technology supports designated projects based on work contracts. Costs are made visible and their sharing between units is based on agreement between units. The results are summarized in table 1.

	Classical OSS technology	Renegotiated OSS technology
Reward structure	Mostly voluntary in task assignment, peer-recognition, sometimes sponsored development.	Designated projects, contributions based on (employment) contracts and task-assignment, development costs divided based on negotiation between actors.
Communication structure	Open discussion email-lists, open message boards, web-presence of projects, open documentation, open training materials.	Intranet, visibility to selected partners who share the development costs.

Table1: Redefinition of OSS technology (tools and processes)

Promotion of OSS technologies was a way of “selling” the organizational innovation to the affected parties by aligning the change process to fit the agendas, serve interests and translate the interests of three key groups: business units, the central unit and developers. As a result, the organizational changes needed for new software development process seem to have been accomplished successfully in these organizations.

6.2 Changes in practices

Both case companies use OSS tools and processes as a way to promote software projects inside organization. It seems that respondents were inclined to explain the change as an introduction of a software marketplace (instituting new rewards and more accurate information and communication structure) inside company as (depicted in Figure 1). The idea of using components from other parts of the organization seemed to be easier to accept for developers, when it was done through the introduction of the open culture that is associated with OSS development by the developers.

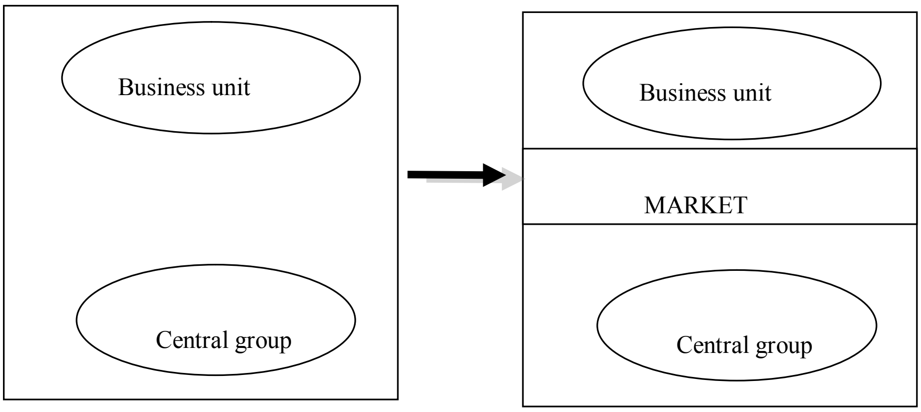

Figure 1: Change in organizations

Both organizational changes can be viewed as instituting replacement of bureaucratic software organizations with markets inside companies. Implementation of OSS technology in these two companies follows the neoliberal idea in which markets will *a priori* create efficiency. This view seems to resonate well with the bottom-up pull of OSS practices, which empowers developers, but also benefits the business units competing for resources with each other.

Concerning the reward structure and organization of production, Philips Medical Systems changed its component tax into a system of rewarding co-operation between the business units and central group. There was a call to change resource allocation. Business targets were set according to the new organizational form. There was also a shift from one central software group into more competitive development setting and thus a need to change the organizational remuneration processes accordingly. At Nokia the implementing of the iSource service can be identified as the institutionalization of OSS development inside the organization with the consequence of restricting access to the source code to within the company.

The reward structure in Philips followed an externalized service provision logic, whereas NSN's iSource moved towards a centralized iSource service. When launched, Nokia iSource was seen as a tool to support small projects in addition to heavy-weight software solutions for software project and configuration management. In PMS, one of the goals was to decentralize software production by giving the business units more responsibility in the process. Philips had previously had problems with component tax: it did not incentivize the central units and business units correctly. It can be argued that the use of OSS as a leverage to introduce the change helped to institutionalize it. NSN's problems were related to the reuse of software assets. These previous lessons helped to introduce new organizing vision of software development through OSS technology (Table 2.)

The communication structures of the case companies changed towards greater openness inside the companies. Philips decentralized the communication structure. Nokia's iSource enabled community building in the source code portal and increased visibility across internal organizational boundaries (Table 2.)

Changes in practices	Philips: Inner source	NSN: iSource
Reward structure	Service provisioning externalized Component tax introduced Funding of maintenance	Service provisioning centralized Problems of reuse in the past
Communication structure	Development decentralized	Community enabled development Innovation across internal organizational boundaries

Table 2: Institutional forces in the cases

These moves result in more competition about the resources between business units. Success in competition can give rewards to the business unit and thus incentivize a more efficient behavior. If managed poorly, the downside might be more siloed production resulting from increased competition. In our case we could not find clear evidence to support this proposition: instead the new development platforms and OSS practices increased communication channels across organizational units and among individual workers inside the organization. This can be seen as fulfilling one of the stated goals of OSS: increasing the sharing of information inside the organization.

6.3 Conclusion

We conclude that 1) the adoption of OSS technology changed the reward and communication structures implementing a wide institutional change and that 2) this implementation represents a far more fundamental rearrangement of software production than was previously thought.

Our cases do not primarily concern technical changes, but deep organizational ones, in which the embedded technology plays a leading role by reinforcing the new institutional arrangements. Our contribution is to show how institutional theory can be used to understand the changes in reward and communication structure and the existence of different groups and the enrolment of interests. Both of our cases serve as good examples of how the balance of power inside an organisation is changed by creating "a market" inside a big international organisation and how this change may facilitate increased visibility, communication and contribution. It can be argued that the new institutional arrangement would not have been possible without the simultaneous introduction of OSS and the market. Thus the institutional forces outside the company both forced the change (market orientation) and made the new organizational arrangement possible (wide acceptance of OSS among the developers).

There are two main limitations to this study. The two cases serve as descriptions of successful implementations rather than universal models of implementing OSS technology. These two companies are very big players that have the capacity to do intra-firm software development, and thus instigate an institutional change in their respective industries or organizational fields. Both companies have adapted OSS technology processes by limiting the openness of the source code. These actions call for questions about the side effects of the limitations that fall outside of the scope of this study. More research is called for to understand heterogeneous OSS practices in different organizations - and the underlying changes they impose on organizations.

ACKNOWLEDGMENTS

The authors would like to extend their thanks to the ITEA-COSI project.

REFERENCES

Barnett L, (2004). Applying Open Source Processes In Corporate Development Organisations, Forrester Research. (http://www.forrester.com/rb/Research/ applying_open_source_processes _in_corporate_development/ q/id/34466/t/2)

Buckho, A.A. (1994). Barriers to strategic transformation. In Shrivastava, P., Huff, A,, and Dutton, J. (Eds.) Advances in strategic management, (10), 81-106. Greenwitch, CT: JAI Press.

Czarniawska, B. and G. Seron. (1996). Translating Organizational Change. NewYork, De Gruyter

Hecker, F. (1999). Setting Up Shop: The Business of Open-Source Software. IEEE Software 16 (1), 45-51.

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

DiMAggio, P and Powell, W. (1991). Introduction. In Powel, W. and Dimaggio (Eds.) The new institutionalism in organizational analysis. 1-38. Chicago, University of Chicago Press.

Dinkelacker, J., Garg, P., Miller, R. and Nelson, D. Progressive open source. In Proceedings of ICSE 2002, 177-184.

Fink, M. (2003). The Business and Economics of Linux and Open Source. Prentice Hall PTR.

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

Garud, R., Hardy, C., and Maguire, S. (2007). Organization Studies 28, 957-969.

Garud, R., and P. Karnøe, (2003) Bricolage vs. breakthrough: Distributed and embedded agency in technology entrepreneurship. Research Policy 32, 277–300.

Ghosh, R. A. (2002). Free/Libre and Open Source Software: Survey and Study. FLOSS Final Report. Available at http://www.flossproject.org/report/index.htm

Greenwood R., and Hinings, C.R. (1996). Understanding radical organizational change: bringing together the old and the new institutionalism. Academy of Management Review. 21 (4), 1022-1054.

Hauge, O., Sørensen, C-F., Conradi, R. (2008). Adoption of Open Source in the Software Industry. Proceedings of the 4th International Conference on Open Source Systems. 7-10 September 2008, Milan, Italy. 211-221

Klein, H. K. and Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, MIS Quarterly, 23 (1), 67-94.

Lindman, J., Rossi, M., and Marttiin, P. (2008). Applying Open Source Development Practices Inside a Company. 4th International Conference on Open Source Systems. 7-10 September 2008, Milan, Italy.

Linden, F., Lundell,B., Marttiin, P. (2009). Commodification of Industrial Software - a Case for Open Source. IEEE Software, July/August, 2009.

Lundell, B., Lings B., and Lindqvist E. (2006). Perceptions and Uptake of Open Source in Swedish Organisations. In Proceedings of the 2nd International Conference on Open Source Systems. June 8-10, Como, Italy.

Powell, W.W., and DiMaggio P.J. (Eds.) (1991). The new institutionalism in organisational analysis. University of Chicago Press, Chicago.

Raymond, E.S. (1999). The Cathedral and The Bazaar – Musings on Linux and Open Source by Accidential Revolutionary. O'Reilly&Associates, Sebastopol, CA.

Seo, M.G., Creed, W.E.D. (2002). Institutional contradictions, praxis and institutional change: A dialectical perspective. Academy of Management Review. 27, 222-247.

Seaman, C. B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering 25 (4), 557-572.

Scacchi, W. (2007). Free/Open Source Software Development: Recent Research Results and Methods. In Zelkowitz, M. V. Eds., Advances in Computers, 69, 243–269. Academic Press.

Scott, W.R. (2001). Institutions and Organizations, 2nd ed.. CA, Thousand Oaks.

Sharma, S., Sugumaran, V. and Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. Information Systems Journal 12 (1), 7-25.

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

Swanson, B., and Ramiller, N. (1997). The Organizing Vision in Information Systems Innovation. Organization Science, 8 (5), 458-474.

Office of Government Commerce. (2002a). ITIL –Planning to Implement Service Management – CD v2.0. The Stationary Office. Norwich, UK.

Office of Government Commerce (2002b). ITIL –Service Delivery – CD v2.0 & Service Support167– CD v2.1. The Stationary Office. Norwich,UK.

Van de Ven, A.H. (1993). Managing the Process of Organizational Innovation in Huber, G.P. and Glick, W.H. (Eds.). Organizational Change and Redesign: Ideas and Insights for Improving Performance. Oxford University Press, New York.

Weick, K. (1995). Sensemaking in Organizations, Sage Publications.

Wesselius, J. (2008). The Bazaar inside the Cathedral: Business Models for Internal Markets. IEEE Software, 25 (3), 60-66.

Wynn, E. (2001) Möbius Transitions in the Dilemma of Legitimacy in E. M. Trauth (Ed.), Qualitative Research in IS: Issues and Trends, 20-44. Idea Group Publishing, Hershey, PA.

Vilkki, K. “Impacts of Agile Transformation”, Flexi Newsletter 1/2009, pp.5-6. http://www.flexi-itea2.org.

Välimäki,M. (2005). The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry. Helsinki University of Technology, Helsinki, Finland.

Yin, R.K. (1994). Case Study Research, Design and Methods. 2nd ed. Sage Publications, Newbury Park.

Ziemer, S., Hauge Ø., Østerlie T., and Lindman J. (2008). Understanding open source in an industrial context. Proceedings of SITIS 2008, Bali, Indonesia.

Zilber, T.B. (2007). Stories and the discursive dynamics of institutional entrepreneurship: The case of Israeli high-tech after the Bubble. Organization Studies. 28 (7), 1035–1054.

Paper III: Lindman, J., Juutilainen, J-P. and Rossi, M. (2009). Beyond the business model: Incentives for organizations to publish software source code. In: Boldyreff, C., Crowston, K., Lundell, B. and Wasserman, A. I. (eds.). Open Source Ecosystems: Diverse Communities Interacting, 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, June 3-6, 2009, Proceedings. IFIP 299 Springer, ISBN 978-3-642-02031-5.

Beyond the business model: Incentives for organizations to publish software source code

Juho Lindman[1], Juha-Pekka Juutilainen[2], and Matti Rossi[1]
1 Helsinki School of Economics, Information Systems Science, PO Box 1210, 00101 Helsinki, Finland. {}@hse.fi,
Web page: http://www.hse.fi
2 Accenture, P.O. Box 1109, 00101 Helsinki, Finland
j.juutilainen@accenture.com,
Web page: http://www.accenture.com

Abstract. The software stack opened under Open Source Software (OSS) licenses is growing rapidly. Commercial actors have released considerable amounts of previously proprietary source code. These actions beg the question why companies choose a strategy based on giving away software assets? Research on outbound OSS approach has tried to answer this question with the concept of the "OSS business model". When studying the reasons for code release, we have observed that the business model concept is too generic to capture the many incentives organizations have. Conversely, in this paper we investigate empirically what the companies' incentives are by means of an exploratory case study of three organizations in different stages of their code release. Our results indicate that the companies aim to promote standardization, obtain development resources, gain cost savings, improve the quality of software, increase the trustworthiness of software, or steer OSS communities. We conclude that future research on outbound OSS could benefit from focusing on the heterogeneous incentives for code release rather than on revenue models.

1 Introduction

Traditionally OSS is seen as being developed in a distributed setting by a loosely-knit community of heterogeneous developers who contribute to a software project without always being employed or paid by an institution [10]. The development model has resulted in reliable, high quality software products that have a short development cycle and decreased development costs. Many voluntarily started OSS products have outperformed commercial software with similar functionalities. Successful examples include Apache web server, MySQL database, and Linux operating system. Interest towards the OSS phenomenon has grown among companies wanting to replicate these OSS success stories [6]. To this end, organizations have leveraged OSS in their operations, boosted their offering [20], and built their business on new business and revenue models [9]. On the supply side, fundamental changes have occurred in the development process, reward mechanisms, distribution of development work, and revenue models that govern how profit is gained [6]. On the demand side, the buy or build alternatives that are traditionally available to organizations have been supplemented with OSS [6].

In addition to using OSS, some companies have released products under OSS licenses or even initiated completely new OSS projects [5]. We have chosen to focus our research effort on understanding this process, coined outbound OSS. Earlier literature on outbound OSS has focused on the revenue stream of the OSS business [19, 9, 12, 17]. While we agree on the importance of a viable company sustaining a guaranteed revenue stream, the heavy emphasis of the earlier literature on the revenue model might have caused some of the other incentives of the organizations' OSS release to be overlooked.

In this paper, we take the viewpoint of the manager making sense of the changing software landscape rather than the viewpoint of the OSS enthusiast. The aim is to gain empirical insight from the company perspective on releasing software to the open domain and thus our research question is: What are the benefits pursued?

2 Background

There has been a paradigm shift concerning software: companies no longer necessarily consider software products as a source of competitive advantage or as the main source of revenue. Conversely, their actions seem to imply that by releasing the source code they gain more than by keeping it secret. Matt Asay, Novell's director of OSS strategy claims that 99.99 % of the products in the world's economy are commoditized [7]. This means that most of the products do not contain anything unequaled. According to Perens, 90% of the software in any business is not differentiating [1, 18]. In most software products, only a small part (5-10%) is differentiating and the remainder is common to the domain. Ultimately every offering that a company delivers to its customers gets commoditized over time [5]. This means that customers are not willing to pay as much for the commodity components and therefore companies should concentrate on creating new and higher value for them [5]. Developing commodity components in-house is not feasible, because they do not provide any additional value. More value is created, if companies concentrate on developing differentiating components and acquire commodity components through subcontracting, by using commercial-off-the-shelf products (COTS), or by utilizing OSS.

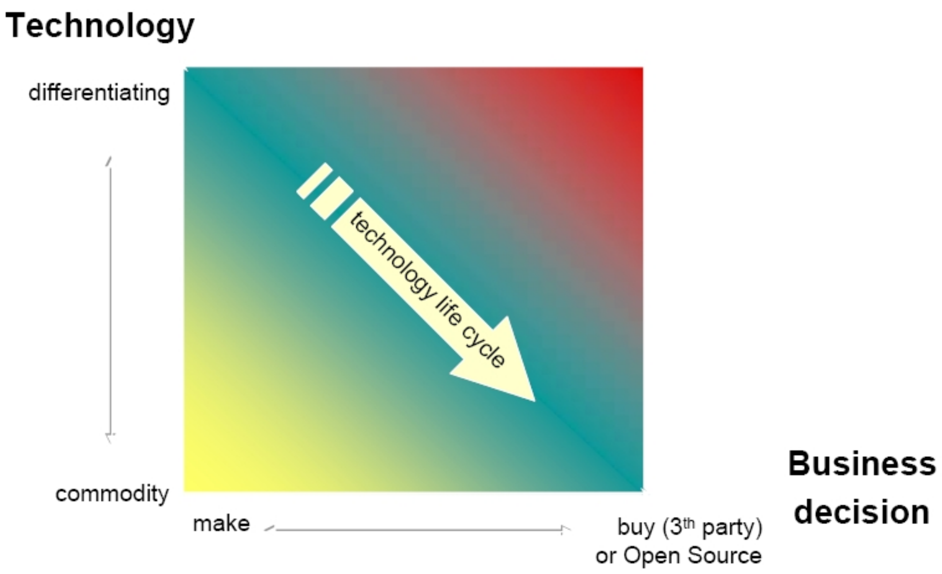

Fig. 1. Commoditization of software (Source: http://www.itea-cosi.org)

Outbound OSS approach refers to taking software that is currently sold under a proprietary license and moving it under an OSS license [5]. The opposite process is called inbound OSS, where a company utilizes previously available OSS code and practices inside their own organization [5]. Outbound OSS approach can be characterized as the license-centered approach where a company initiates an OSS project by either releasing the source code of an existing solution to a community as OSS, or initiating an OSS community to develop a new software product [2]. The released source code will then be the basis for the future development of software. West and O'Mahony would call this outbound OSS approach a spinout project because software is first developed internally and later on released to the public under an OSS license [21]. IBM's Eclipse project is one successful example of the outbound OSS approach. After spending more than 40 million dollars on the development of Eclipse, IBM released its source code. By utilizing the outbound OSS approach, there were expectations that IBM could gain development help from other companies, lower the development costs, gain credibility, and gain a better position to compete on the market [23]. Another, not so successful example of source code release would be the Mozilla Netscape browser, where developers needed years of work to make the previously proprietary code feasible after it was published [18].

The outbound OSS approach offers several means through which a company can improve its position on the market. Companies often offer complementary services on top of free software and thus revenue is generated from the sales of the services.

A company can pursue cost-reductions and better time-to-market by working collaboratively with the community [5]. The outbound OSS approach can help to reduce development costs if the company succeeds in attracting OSS developers to participate in the development [2, 3]. If the collaboration succeeds, the company can get development resources and be able to improve the product. OSS communities are well-known for having low tolerance for poor contributions, which helps to guarantee good quality [5]. In addition, through frequent releases and with the help of a large community, bugs can be found and fixed quickly [19]. Earlier literature implies that security and reliability can be increased through an OSS-based development because OSS products get tested with the help of a global user community [11]. Finally, by getting involved in OSS projects companies can incorporate OSS ideas into commercial software, spot talented programmers for hiring purposes, and also attract programmers who want to work in an intellectually challenging environment [13].

Outbound OSS approach can also aim for a larger user base and increased feedback. By releasing software as OSS, it is possible to attract new users because the software is free of charge. If there is a commercial counterpart with similar functionality, many users will likely choose the OSS product because it is free. Company can thus gain market share from its competitors and even be able to boost the sales of some related products or services [22]. Thus, the outbound OSS approach can be a powerful method especially if the company has strong competitors [14]. It is also a useful approach in an industry that is dominated by a monopoly [16]. The same reasoning applies to a situation where a company has lagged behind its competitors [5]. Source code release can speed up the diffusion of the product since there are no costs involved in obtaining OSS [2]. Thus, the outbound OSS approach lets companies that could never challenge their competitors on their own, challenge them with the help of an OSS community [15]. The outbound OSS approach can in particular help small companies with limited resources if they succeed in attracting voluntary developers to help in the software development [3]. The releasing company may gain better competitive position with the help of an active development community. Releasing a low cost alternative also puts pressure on the competitors to lower their prices [2]. Taking part in OSS projects might also arouse interest in the general public and improve corporate image [3].

The outbound OSS approach can help in diffusing new technologies. Approach can be useful if a company has a core infrastructure technology that is an enabler to other products and solutions in the company's portfolio [5]. OSS could then be used as a method to make the company's technology pervasive, or adopted as a standard. OSS development is a useful way to promote standardization [22]. Compatibility is a challenge on the software and hardware markets where there are a vast number of different manufacturers and products. Therefore large companies like IBM want to become active participants in the OSS development and to shape it in their interest [22]. On the other hand, by embracing and supporting OSS projects companies can pre-empt the development of a standard around a technology owned by a powerful rival [13]. Finally, OSS has an effect of encouraging collaboration and it can be used as a way to work with partners and competitors on very large projects, sometimes even involving customer at earlier stages of development [5].

Much of the potential success of outbound OSS will depend on the efforts of people who are willing to work for free [9]. That is why companies need to attract software specialists who are willing to participate in OSS development. However, many voluntary software developers will not participate if they are not treated fairly and provided with freedoms and other intangible "payments" [9]. Thus, in order to succeed in the outbound OSS approach, companies may have to invest considerable amounts of time and money [4].

3 Methodology

Our aim is to show the different benefits companies pursue with the use of an outbound OSS approach. Our selected approach is qualitative and interpretative as we aim to clarify the relevant variables and to understand how companies make decisions about pursuing benefits with outbound OSS [8]. We used three exploratory descriptive case studies and interviews of the company respondents. To be able to formulate a comprehensive view of the outbound OSS approach, in-depth data collection and analysis was needed. In terms of systematic data collection, a series of formal face-to-face semi-structured interviews was conducted. Since the aim was to lay emphasis on the depth, nuance, complexity, and comprehensiveness of the data, interviewing was considered to be the most appropriate method for data collection. Interviews were designed in a way that if a later researcher follows similar procedures when conducting the case study, they should arrive at the same findings [24].

The interviews were conducted as a part of the ITEA-COSI-project. Our selected partners were Philips Medical Systems, Nokia Networks, and European Software Institute (ESI). The selected cases can be seen as typical instances of the phenomenon under study. Five interviews were conducted: three at Philips and one at Nokia and one at ESI. The interviewees were selected so that it would be possible to form a holistic view of the utilization of outbound OSS approach in the case companies. It was desirable that each interviewee would have a comprehensive view of business, close relations to the OSS community, and a broad understanding of how the OSS approach has impacted on the company. Open questions were chosen to make sure that the answers would be constrained as little as possible. The questions were sent to the interviewees in advance so that they were able to get acquainted with them before the interview. Before and during the actual interview, the interviewees had the possibility to ask for clarifications concerning the questions. During the interview, some of the questions were explained more precisely to guarantee that all the interviewees would understand them in the same way. Some follow-up questions were also posed and clarifications given when necessary. The interviews were conducted in an iterative manner, so it was accepted that responses to certain questions could stimulate new awareness and interest in particular issues, which could then require additional questions to be posed to the interviewee. The estimated time of the interviews was one hour.

The data analysis occurred in three phases. First, the data gathered through the interviews was transcribed. The transcription was conducted by word-for-word basis to guarantee the accuracy of the answers and to avoid misinterpretations. After transcription, all the transcribed interviews were sent to the interviewees so that they were able to read them through and clarify their answers if needed. Only one interviewee clarified some answers. Following this, in a second phase the data was elaborated. The objective was to find relevant information from each case and to develop a rich understanding on the incentives of companies' outbound OSS approach. Finally, in the third phase the results were analyzed and the incentives of the outbound OSS outlined

4 Cases

4.1 Philips Medical Systems – DVTk

Philips Medical Systems (PMS) manufactures products for the health care industry. Its product portfolio covers for example medical imaging, ultrasound, health care IT, defibrillation, and monitoring modalities. Philips Medical Systems and its partner company created in 2000 a validation application for the medical communication protocol DICOM (Digital Image Communication in Medicine). The application was called DVTk (Dicom Validation Toolkit) and it was distributed within Philips and was also freely downloadable from the Philips Internet pages. After several years of co-development, Philips Medical Systems and its partner company decided to release the DVTk as OSS in June 2005. DVTk is licensed under the LGPL, the source code is available at the SourceForge website and the software is freely available for download.

The DVTk tool itself is free so it does not generate any direct revenues. The long term goal of PMS is that with the help of a user community the quality of DVTk is improved and this will eventually reduce the service and support costs of the tool. The main reason for releasing the source code of the DVTk was to create an independent leading tool for the DICOM validation and service tools. Since the application was earlier closed, the results of validation with DICOM were not always trusted by other organizations. By releasing the application as OSS and by providing the opportunity to review and contribute to the code, trustworthiness of the application was expected to increase. Users could trust the software more because they were able to see that there are no hidden features and see how the product is implemented. In addition, there was an aim to rationalize the software development by releasing the source code of DVTk. Prior to releasing as OSS the distributed development between different sites and between different organizations impacted the efficiency of the work. The development of the application was running on different isolated source control environments to prevent different developer companies from accessing each other's contributions.

Another reason for opening the code was the intention to create a larger and more active community that could use DVTk, report on bugs, and also help in the development. DVTk application was frequently downloaded even before the code was released, but often the feedback was not very useful. By releasing software as OSS, there was expectation in PMS to have more feedback from the users. In addition, it was expected that PMS could involve more companies in the development of DVTk and this way to reduce development costs.

4.2 Nokia Networks – Benchmark

Nokia Networks is one of the leading telecom equipment providers in the world. It merged in 2007 to form Nokia Siemens Networks. The data was gathered before the merger, so we use the name Nokia Networks when referring to this company. Nokia Networks provides network infrastructure, communications and networks service platforms, as well as professional services to operators and service providers. These solutions include both software and hardware. Nokia Networks uses and integrates OSS products (e.g. Linux) into their products, but software that is ultimately offered to the market is not OSS. Nokia Networks does not currently directly contribute much to OSS projects, but would benefit from some influence on the direction of the development. There have been efforts at Nokia Networks to influence OSS communities by participating in the creation of specifications like OSDL Carrier Grade Linux (CGL) requirements specifications, but the results have not had the desired effect. Our case was aimed to create a benchmarking tool for the selected OSS projects. Earlier Nokia created Network Database Benchmark which is used for measuring the Home Location Register (HLR) type of performance of databases. In our case Nokia Networks was preparing Control Plane Benchmark.

Nokia Networks' goal is that Control Plane Benchmark would highlight possible deficiencies in OSS projects and cause developers to steer projects in the direction Nokia Networks would like them to go. Nokia Networks perceives OSS communities and components as a future-proof solution because commercial companies are getting smaller all the time and their long-term existence is uncertain. The respondent considers OSS communities as a more sustainable option sometimes for software development than commercial companies.

Nokia Networks does not have much official interaction with OSS communities. The communities are often suspicious of big companies and are not especially interested in the products that Nokia Networks provides. Thus communication with OSS communities is mainly through individuals who work in Nokia Networks and are also part of an OSS community. However, these people are not representing Nokia Networks when they are involved in the communities. Nokia Networks has some projects and initiatives to form a closer relationship with OSS communities, for example, a portal to manage its OSS projects and to promote Nokia Networks' involvement in OSS projects. Nokia hosts, contributes to, and sponsors multiple OSS projects. Nokia is, for instance, a strategic developer in the Eclipse Foundation. Nokia Networks is also one of the 20 companies that support Open Source Development Lab (OSDL). With the other members in OSDL, Nokia has developed a kind of future roadmap for Linux distributors. Nokia Networks' aim is to create vision and guidance to enhance Linux and to meet the needs of both the data center and carrier grade market segments.

4.3 European Software Institute – V-Manage

European Software Institute (ESI) was launched as an initiative of the European Commission, with the support of the Basque Government and European companies working in the field of information technology. ESI's main activity is based on helping the software industry to produce software of a higher quality, on time, and at a lower cost. ESI offers consultancy and training services as well as technological support. One of the services that ESI offers to organizations is consultancy for implementing a software product line. The purpose of this consultancy service is to achieve a high level of reuse in all products. ESI provides organizations a disciplined methodology and a suite of tools, called V-Manage, for developing software for embedded systems. Now ESI is planning to utilize the outbound OSS approach and to release the source code of V-Manage. V-Manage helps organizations to develop software especially for software product lines and it is mainly offered to small and medium sized companies.

ESI's service consists of a software called V-Manage and a consultancy service. At the moment, the main source of revenue for ESI is the consultancy service consisting of training, support, and maintenance. Currently, V-Manage is proprietary software licensed to the customers of the consultancy service, but ESI is investigating whether they should license it with an OSS license. In the future the revenues will be generated through the sales of consultancy services. There is an expectation in ESI that opening the code would increase other companies' interest towards the application and eventually increase revenues through the sales of consultancy services. However, it is not expected that obtaining development resources from external parties would result in lower costs. Instead, extra development resources are seen as a way to boost the popularity of V-Manage.

ESI has the aim of providing extension points to V-Manage so that external developers can extend the tool by means of plug-ins. This enables customers and possibly a development community to customize the application according to their own needs and add new features. ESI is planning to release the source code of the extension points and plug-ins and keep the platform proprietary. This way ESI could retain core parts of the V-Manage as closed. The source code of plug-ins would be released under a license that assures that all the modifications and derivative works are distributed and made available under the same license. Initially ESI is planning to use LGPL. By means of this new approach, ESI aims to get software development resources from external partners who are willing to develop the application through extension points. The releasing of the source code could result in an active development community. However, the amount of potential development help is still rather uncertain because the application is very specific so it is not likely to attract a large number of developers. Because of the special nature of the tool, it is expected that developers will more likely be companies than individuals.

5 Incentives for openness

Probably the best known classification of different OSS revenue models is the one presented by Hecker [9]. Hecker's revenue models concentrate mainly on the cash flow between the company and its customers. However, our empirical findings demonstrate that companies also have incentives other than revenue for utilizing the outbound OSS approach. Actually, the only case in our data which can be categorized according to Hecker's classification is ESI's V-Manage. ESI's approach is consistent with Hecker's support seller model where revenues are generated from selling associated services. By means of the outbound OSS approach, ESI aims to increase the popularity of V-Manage and to boost its revenues through the sales of consultancy services. However, the source code of V-Manage is currently not opened and likely will not be opened at all.

It was evident that the case companies perceive the commercial potential of the outbound OSS approach. Companies have various incentives for releasing the source code of their software. These different objectives also have influence on how outbound OSS is applied in practice. Outbound OSS approach is considered to be suitable for companies whose main business is not the software itself. This implies that a company does not necessarily risk its business by releasing the source code. Instead, revenues are generated for example through the sales of different services. Below are the different incentives categorized in a table format (Table1).

Table 1. Incentives per case company.

PMS	Nokia Networks	ESI
	Steer OSS community	Steer OSS community
Obtain development resources		Obtain development resources
Gain cost-savings		
Improve the quality of SW		Improve the quality of SW
Increase trustworthiness of SW		
Promote standardization		

ESI's strategy seems to be that by opening parts of V-Manage companies may become more interested in the tool because they are able to customize it to their own needs and ultimately ESI would generate revenue by consultancy services. Instead, the objectives of neither Nokia Networks nor Philips Medical Systems are directly related to generating revenues through OSS. PMS' goal is to rationalize the software development, create a de-facto standard, and to try to form an active development community. Through the outbound OSS approach, PMS aimed to gain external development resources and improve DVTk. The PMS respondent also maintained that OSS can increase the trustworthiness of DVTk because everyone is able to see how it is implemented. Nokia Networks' objectives notably differ from the goals of PMS and ESI. Nokia Networks tries neither to generate revenues nor gain development resources through the outbound OSS approach. Nokia Networks is developing benchmarking tool to be used by OSS communities. This tool is then released as OSS. The aim of Nokia Networks is it could then leverage the OSS communities through these tools.

It seems that the case companies have very different objectives when they chose the outbound OSS approach. It seems that ESI is the only company having a revenue incentive to release the source code. However, it is evident that financial reasons play a role also with Philips Medical Systems and Nokia Networks. In PMS it is considered that the DVTk project may have an indirect impact on total revenues of PMS. PMS's goal is that by improving the DVTk the service and support costs will decrease. Nokia Networks aims to gain cost savings if they succeed in steering OSS communities because the company will get software products that are implemented according to Nokia Networks' needs.

6 Conclusions and implications

The objective of this paper was to investigate incentives for commercial companies to release software source code. Revenue models were not the primary concern for any of the case companies. The role of revenue models was considered, but the decisions were not incentivized by direct revenue streams.

Although commercial actors are coming into terms with releasing source code they need to tackle practical concerns. One of the main problems was that companies' OSS products are specialized to niche markets that fail to attract a large population of developers. Another challenge is that companies were willing to utilize OSS resources, but they do not always have plans to compensate for the acquired benefits. The outbound OSS approach also highlights some challenges that a company can confront after the source code is released. Based on our analysis, it seems that these challenges are mainly related to collaboration with OSS communities and maintenance of the code base. Voluntary OSS developers will only participate in software development if they find the project interesting. Thus, gaining contributions from the OSS community is not certain. If the software is very specialized and does not interest the general public, the company might confront difficulties in attracting developers. The company also has to be aware that the community's objectives and timetable in software development will most likely differ from the company's own goals. In order to succeed, the company should create a strategy on how it is going to attract developers, motivate them to participate, and steer them so that the company's objectives will be reached.

It should also be noted that the cases in the paper are at very different stages of their OSS activities, and as such are unlikely to give direct applicable solutions to other companies. They do serve as empirical account of what the incentives for commercial companies are, and hopefully help to refocus research beyond revenue models to the multitude of different company incentives.

Acknowledgements

The authors thank the ITEA-COSI project.

References

[1] ITEA-COSI-project. http://www.itea-cosi.org/ [Accessed 14.11.2008]

[2] M. AlMarzoug, L. Zheng, ,G. Rong, and V. Grover. Open Source: Concepts, Benefits, and Challenges. *Communications of the Association for Information Systems*, 16:37, pp.756-784, 2005.

[3] A. Bonaccorsi, and C. Rossi. Comparing Motivations of Individual Programmers and Firms to Take Part in the Open Source Movement: From Community to Business. *Knowledge, technology and policy*. 18:4, pp.40-64, 2006.

[4] L. Dahlander, and M. Magnusson. Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy*, 34:4, pp. 481-493, 2005.

[5] M. Fink. *Business and Economics of Linux and Open Source*. Prentice Hall, New Jersey, 2002.

[6] N. Fitzgerald. The Transformation of Open Source Software. *MIS Quarterly*, 30, 3, pp. 587-598, 2006.

[7] G. Goth. Open Source Business Models: Ready for Prime Time. *IEEE Software*, November/December, pp.98-100, 2005.

[8] D.E. Gray. *Doing Research in The Real World*. Sage Publications, California, 2004.

[9] F. Hecker, Setting Up Shop: The Business of Open-Source Software. IEEE *Software*, 16, 1, pp. 45-51, 1999.

[10] G. Hertel, S. Niedner, and S. Herrmann. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy* 32, 7, pp.1159-1177, 2003.

[11] S. Krishnamurthy. A managerial overview of open source software. *Business Horizons*, 46, 5, pp.47-56, 2003.

[12] de P.B., Laat. Copyright or copyleft? An analysis of property regimes for software development. *Research Policy*, 34, pp.1511-1532, 2005.

[13] J. Lerner, and J. Tirole. The Open Source movement: key research questions. *European Economic Review*, 45, 4-6, pp. 819-826, 2001.

[14] J. Lerner, and J. Tirole. Some Simple Economics of Open Source. *Journal of Industial Economics*, 50, 2, pp. 197-234, 2002.

[15] M.L.Markus, B. Manville, and C.E: Agres. What Makes a Virtual Organisation Work –Lessons From the Open Source World? *Sloan Management Review*, 42,1, pp.13-26, 2000.

[16] S.C. O'Mahony, Dissertation: The emergence of a new commercial actor: community managed software project, 2002.
http://opensource.mit.edu/papers/omahony.pdf [Accessed 14.11.2008]

[17] A. Osterwalder, Y. Pigneur, and C. Tucci. Clarifying business models: Origins, present, and future of the concept. *Communications of the Association for Information Systems*, 16, pp. 1-25, 2005.

[18] B. Perens, The emerging economic paradigm of Open Source. *First Monday* 10 (special issue 2: Open source), 2005.

[19] E.S. Raymond, *The Cathedral and the Bazaar*, 2000.
http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/
[Accessed 14.11.2008]

[20] R. Rajala, J. Nissilä, and M. Westerlund. Revenue Models in the Open Source Software Business. In *Handbook of research on open source software – Technological, Economic, and Social Perspectives*, K. St.Amant, and B. Still (Eds.), New York, Hershey, 2007, pp. 541-554.

[21] J. West, and S. O'Mahony. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. Proceedings of the *38th Annual Hawai'i International Conference on System Sciences*, Waikoloa, Hawaii, 196c-196c, 2005.

[22] T. Wichmann. Firms' Open Source Activities: Motivations and Policy Implications. Free/Libre Open Source Software: Survey and Study, *FLOSS Final Report*, Berlecon Research GmbH, 2002. http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf [Accessed 14.11.2008]

[23] D. Woods, and G. Guliani. *Open Source for the Enterprise: Managing risks, reaping rewards*, O'Reilly Media, 2005.

[24] R. K. Yin. *Case Study Research: Design and Methods*, 2nd edition, Sage Publications, California, 1994.

Mentioned OSS projects

DVTk http://www.dvtk.org/

Network Database Benchmark *http://hoslab.cs.helsinki.fi/homepages/ndbbenchmark/*

Paper IV: Lindman, J. and Rajala, R. (Unpublished). "Lessons on the FLOSS business learned from the Open Source Software Pioneers". Submitted to an international journal.

Earlier version of the paper has been published in the proceedings of ICSOB2010. Lindman, J., Rajala, R. and Rossi, M. FLOSS-induced Changes in the Software Business: Insights from the pioneers. ICSOB, Jyväskylä, Finland, 21-23.6.2010.

Lessons on the FLOSS business learned from the Open Source Software Pioneers

Abstract

In addition to opening the software innovation processes, the Free/Libre Open Source Software (FLOSS) activity has affected the entire software business. It has induced the business models of software firms to be increasingly grounded on user-centric and service-oriented offerings. However, the literature has not paid sufficient attention to how entrepreneurs in small software firms perceive the FLOSS induced changes in their businesses. This study investigates software entrepreneurs' perceptions of these issues. The findings were grouped into four categories: changes in the innovation process, user involvement, the use of resources and revenue models in software firms.

Keywords: Open source software; entrepreneurship; open innovation; service-dominant logic; narrative methodology

Introduction

The Free/Libre Open Source Software (FLOSS) phenomenon has received increasing attention among information systems (IS) researchers as an important aspect of the information economy and an essential consideration for all software companies (Fitzgerald 2006). FLOSS has two distinct features. Firstly, it is connected with licenses that provide existing and future users with the right to use, inspect, modify, and distribute modified and unmodified software to others (Raymond, 1999; Von Krogh & Von Hippel, 2006). Secondly, it has created a new practice of collaborative innovation in numerous FLOSS communities (Kogut & Metiu, 2001; Dahlander & Magnusson, 2008). Years of development in these communities has resulted in high quality mainstream applications, such as the Linux operating system and the Apache Web Server (Raymond, 1999; Mockus et al., 2002).

In the early inquiries into the fielfd of FLOSS, researchers focused their attention mainly on illustrating how companies could organize their businesses around new revenue models (e.g., Raymond, 1999; Hecker, 2000). The research focus then shifted to an analysis of how established companies could leverage FLOSS (Dahlander & Magnusson, 2005; Dahlander & Magnusson, 2008; Scacchi, 2007) and how research should reorient toward these FLOSS-induced changes in the software industry (Fitzgerald, 2006; Von Krogh & Von Hippel, 2006; Xu and Brinkkemper, 2007). In summary, prior research has suggested that user participation in software development changes value creation and value capture throughout the entire software industry. Hence, the FLOSS activity represents an important phenomenon that deserves further investigation.

Previous empirical research into the impact made by FLOSS among small and medium-sized enterprises (SME) in the software business is rather exiguous. Studies on entrepreneurship (e.g., Hannan and Freeman 1984; Baum and Oliver 1992) have illustrated that conducting business in a turbulent environment is especially challenging for a SMEs, due to their the liability of smallness, newness and resource scarcity. These characteristics constrain their innovation activity. In addition, SME's lack the resources necessary to master business operations on a large scale.

FLOSS includes interesting opportunities for many entrepreneurs in regard to the benefits of accessing resources in the innovation communities.

However, little is known about the changes FLOSS activity induces in software businesses and the actions the FLOSS entrepreneurs have taken to meet these changes. This study aims to fill this void in the literature by improving the understanding of the entrepreneurial FLOSS practices characterized by moves towards open innovation.

We scope our investigation to the primary software industry and focus on companies that actively take part in open source software development. Taking an entrepreneurial viewpoint, we pose the following research question: *How do open source software pioneers perceive the FLOSS-driven changes in their businesses?* We address this question empirically through a qualitative inquiry. Based on an analysis of narratives emerging from interviews of software entrepreneurs and senior managers at software firms, we group our findings into four categories that describe the changes identified from the decision-making perspective and manifest the firm-level responses to these changes.

This paper firstly discusses the research background and the underpinnings of open and free software development and their effects on software innovations. We use these concepts in our investigation of the effects of FLOSS on small and medium-sized software firms that engage in open source software development. We then present the research setting, narrative methodology, empirical cases and their analysis. Finally, we discuss the findings and conclude the paper with suggestions for further research on the topic.

Background for research

The “FLOSS phenomenon” describes a new paradigm for the management of innovation in the software industry. Using this approach, firms work with external partners to commercialize their internal innovations and obtain a source of external innovations that can be commercialized. Currently, FLOSS has established positions in several market segments, ranging from operating systems, middleware, and end-user products, such as media players, office suites, and games (Von Krogh & Spaeth, 2007). Innovation processes have traditionally been seen as an intra-organizational activity based on the development, manufacturing, and subsequent marketing of an innovation. However, the innovative activities associated with software development are becoming increasingly blurred and interconnected with extra-organizational actors and processes.

Most research on the domain has focused on successful FLOSS projects (Radtke et al., 2009) and open source (OS) development approaches, rather than responding to the growing interest in FLOSS among companies (Osterlie & Jaccheri, 2007). Prior research on making the use of FLOSS commercial has primarily focused on pointing out that managers must take care when adopting FLOSS (see Ven et al., 2008; Fitzgerald, 2009). Doing so for the wrong reasons can harm the organization, whereas not adopting FLOSS might leave considerable opportunities unused (Ven et al., 2008).

While the development of FLOSS is becoming increasingly popular, recent analyses have shown that it has incorporated and accelerated subversive changes throughout the software industry (Vitari & Ravarini, 2009; Elpern, 2009) and has had an extensive impact on the economy and society (Von Krogh & Spaeth, 2007).

The emergence of open innovation

The benefits of open innovation are widely accepted in the software development community (e.g., Von Hippel & Von Krogh, 2003; Henkel, 2008). In its broadest sense, software innovation refers to research and development (R&D) activities that involve intellectual capital, physical products, and processes in software production (Vujovic & Ulhoi, 2008). Chesbrough (2003) observed that strategic innovations have typically been regarded as a company's most valuable competitive assets, which also serves as a barrier to entry by competitors. This kind of proprietary innovation development and competition is characteristic of closed innovation processes, where technological progress has generally been kept secret to earn expected profits (Meyer, 2003). Conversely, in the open innovation environment, a company's ability to remain competitive increasingly relies on utilizing accessible resources in the continuous development of new and superior products and services in a business environment characterized by growing instability. To this end, companies utilize knowledge to improve their products, services, and processes. This enables them to remain competitive and continue being innovative (Vujovic & Ulhoi, 2008). The innovation capacity within the FLOSS community relies, in many respects, on openness and teamwork, as well as on decentralized linkages and knowledge flows.

FLOSS has its historical roots in the Free Software social movement, which utilizes non-commercial and anti-commercial principles of collective action, as well as distributed work practices (O'Leary et al., 2002; Williams, 2002; Benussi, 2005). The principles of the open innovation model can be traced back to the end of the 19th century, when open innovation in the United Kingdom (UK) iron industry (Allen, 1983) and the United States (US) steel industry (Meyer, 2003) overcame the previous domination of proprietary innovations. The term open source software was originally coined to make the potential development and licensing method more credible to corporate actors (Raymond, 1999). This strategy turned out to be successful (Fitzgerald, 2006).

The emergence of FLOSS facilitates access to resources and the use of capabilities as the source of competitive advantage among software vendors. Service-dominant logic (Vargo & Lush, 2004) describes a significant transition in business in terms of the use of resources. It considers resources in the development and delivery of offerings as operand resources (those in which an operation, or act, is performed) and operant resources (those that act on other resources).

FLOSS development depends, to a great extent, on resources that are external to a firm. That is, the operand resources include, for example, the physical resources required to make services available to customers, while the operant resources, such as the requisite skills, knowledge, and capabilities, represent the intangible resources of the parties engaged in the collaboration. In FLOSS businesses, resources are accessed through collaborative relationships between two or more parties, or, as suggested by Dahlander and Magnusson (2005), in a company-community relationship. In these relationships, resources are accessed and exchanged through activities embodying all of the tasks required to develop and implement the software and its related services.

Effects of open innovation on software offerings

Along with the mushrooming of FLOSS, open innovation activity (Chesbrough, 2003) has surfaced as a powerful value creation engine; however, it will not enable firms to capture value without a viable business model (reference withheld for blind review). Fitzgerald (2006) argues that the emergent forms of FLOSS have a strong commercial orientation in product development, delivery, and support processes labeling these forms as OSS 2.0. Di Bona et al. (1999) and Kogut and Metiu (2001) have investigated the use of FLOSS and its implementation in organizations. In addition, Henkel (2008) illustrated that despite fears to the contrary, corporate participation in open source software development did not lead to the harmful sharing of information.

In many cases, voluntary cooperation-based collective action systems involve some form of public or semipublic good (Heckathorn, 1996; Monge et al., 1998). According to the definitions by Olson (1965) and Udéhn (1993), public goods offer participants in networks collective benefits that are (a) non-excludable, in that they are available to all network partners, and (b) jointly supplied, in that partners' uses of the goods are non-competing. In FLOSS development communities, open source software plays the role of a public good and underlies collective action towards voluntary cooperation, interaction, and the combination of the participants' interests.

There is an ongoing discussion regarding the principles that software companies should adopt when leveraging FLOSS (Dahlander & Magnusson, 2005). Some of these authors have viewed FLOSS as a social phenomenon based on a gift culture (Raymond, 1999). Others debunk the implied one-way relationship by observing that giving a gift always includes and enacts a social relationship between the actors (Ljungberg, 2000). The nature of the relationship is crucial when assessing the motivations of commercial and non-commercial actors and the viability of a given business (Dahlander & Magnusson, 2005).

FLOSS products are sometimes competitive with proprietary software, but the research reviewed herein has not endorsed the view that FLOSS should always be viewed as complementary. On the one hand, FLOSS is not automatically mimicking proprietary functionality (i.e., in operating systems such as MS-Windows and Linux). Such mimicking situations may often lead to incremental (Teece et al., 1997), as opposed to disruptive, innovations (Tushman & Anderson, 1986). On the other hand, economic incentives do not favor developing proprietary software that mimics previously available FLOSS functionality.

Methodology

Research design

This paper focuses on a new and emerging topic, focusing on how entrepreneurs in small software firms perceive the FLOSS induced changes in their businesses, in which the research conducted to date is meager. Therefore, the present study relies on multiple sources of evidence and benefits from the prior development of theoretical propositions. A narrative approach is considered a feasible research strategy in this study, as it is well suited to investigate a phenomenon within its real-life context. Narratives permit researchers to tap into the richness of the phenomenon by producing ideographic representations of participants' experiences (White, 1981). Analyzing these experiences opens a window into organizational reality (Ford & Ford, 1995). This is the case in the present study, which aims to provide answers to the research question "How do software entrepreneurs perceive the FLOSS–driven changes in their business?"

We aim to derive an answer to this question through an inductive analysis of entrepreneurs' perceptions. To clear the epistemological status of the narratives, we focus on the perceptions of the software entrepreneurs and acknowledge that these perceptions are partially constructed after the fact and for the researchers (Lanzara, 1991). However, these are not reasons to overlook the experience of the respondents, as their narratives are crafted in legitimate logic (Czarniawska, 1998).

Narratives enable communication with a researcher (Coffey & Atkinson, 1996). The language selected by the individuals reflects how they see the world, reclaim their identity, and experience their environment. White (1981) argued that narratives are of illustrating a person's viable beliefs. From this information, values can be transmitted. Moreover, they reveal issues for analysis upon which researchers then aim to articulate the situation at a level different from that expressed in the author's narrative.

The narrative approach coaches the members of a certain organization to frame their understanding of social reality and their place in it in a discursive manner (Phillips & Hardy, 1997). The resultant narratives are specific and coherent, representing their creative descriptions of the world (Rhodes & Brown, 2005). In addition, they are anchored in locally available discursive resources and enable organizations to be treated as socially constructed phenomena (Berger & Luckmann, 1966) that are sustained by social, political, and symbolic processes (Pfeffer, 1981).

Unlike other forms of qualitative data analyses, the narrative analysis focuses on narratives in their entirety. Here, the aim is to search through large units of coded signals to determine the underlying structure and content (Rice & Ezzy, 1999). The narrative approach uses stories loaded with meaning as a tool to explain a phenomenon (Burr, 1995). These narratives are not just "objective" reflections of reality, as they also shape the events they describe (Czarniawska, 1998). They create social order and imbue stories with values (Burr, 1995). There is an assumption that by analyzing how stories are told and what they say and do not say, we can discover the hidden meaning behind the world they describe (Burr, 1995).

There are several noteworthy examples recommending how to use the narrative approach to study phenomena. These examples are very similar to the one applied in this study (e.g., McDaniel, 2004; Szczepanska et al., 2005; Brown & Jones, 1998; Alvarez & Urla, 2002). In this study, the narratives of the actors are used to explain companies' actions, as they reflect the social worlds of the decision makers. Business decisions are based on how these agents make sense of the situations; in other words, how they interpret their circumstances and act (Weick et al., 2005). This sense making requires rationalization based on plausible images created during the process (Weick et al., 2005). The identity of the organizational actors formulates the interpretation and enactment of narratives, whereas the categories of sense making are derived from the organizational context (Weick et al., 2005).

A common methodological objection to the narrative approach is to discount the responses as only "the opinion of one person" or to reflect only the "opinions of the few." While there are several problems associated with this type of objection, the most obvious is that the objection implicitly assumes that the described narrative might somehow be false. We are the first to admit that narratives can indeed be biased or wrong, yet it is normally unrealistic to believe that the respondent is lying or inventing the entire story just for the researcher (Czarniawska, 1998)

To summarize our theoretical stance, following Bartis and Mitev (2008), we combine constructionism with narrative methodology to examine how the socially constructed meaning of FLOSS is related to the entrepreneurial viewpoint on the changes in the software firms' business environment.

Empirical inquiry

We selected five FLOSS software companies (see Appendix 1) to determine how their managers perceived and described the ongoing changes in their environment. The selection criteria included whether the companies used FLOSS components and products as part of their product and service offerings. The method used for data collection included semi-structured interviews in person and an extensive set of secondary data on the case companies. The respondents were senior managers within these companies. The informants were responsible for the company's strategic decisions. Given the research questions, senior managers were seen as viable sources of information. We supposed that their ability to make sense of the competitive environment was guiding the company's decision-making process, and that their narratives reflected the changes apparent in the software business.

We conducted interviews with each of the respondents from the selected firms over a 5-year period (2004-2008). The number of interviews with each respondent ranged from one to four. To gain a rich understanding about the organizations in their contexts, we interviewed the entire staff then employed by Tripod (3 persons), Yoga (1 person) and Tulip (5 persons). Conversely, OurDB and Nemesis were larger companies, so we limited our discussions to those with the CEOs and CTOs of those companies. The average duration of the interviews was 2 hours, with durations ranging from 1.5 to 3 hours. Thirteen in-depth interviews were considered sufficient to obtain enough material for the analysis, since the interviews touched upon similar themes in separate companies.

Following Essers (2009) and Visweswaran (1996), the interview situation was designed to take into account the roles of the actors. The function of the interviews was to provide a voice to the respondents and thus tap the richness of their experience. The interviewers knew all of the respondents previously, which enabled the interview atmosphere to be comfortable and relaxed. Notably, it was unlikely that this familiarity would have any negative effects on the quality of the interviews. The sessions took place in a quiet environment and were tape-recorded. Prior to the interview, the respondents were informed that the researchers were interviewing people who have experience in both open source and software businesses, and that the interviews would gather their views on the software business environment and FLOSS.

All interviews had a similar structure and included demographic questions. Moreover, questions consisted of the respondents' definition of FLOSS software and FLOSS communities, the competitive environment, the company's offerings, business benefits, industry entry, publicity, and the "heroes" and "villains" of FLOSS. The final part of the interviews centered on the future possibilities offered by FLOSS. In addition, some time was reserved to allow for further clarification.

Questions were posed in a non-leading way to ensure that the narratives would be recorded as they occurred to accurately reflect the voice of the respondents. The interviewers aimed to make the interview situation as informal as possible. Tape-recording was necessary to ensure that the nuances of the respondents' narratives were captured, although it probably introduced some unnecessary formality into the interview environment.

Analysis of interviews

For investigator triangulation purposes, the transcribed interviews were analyzed by two researchers. This analysis was conducted to gain a comprehension of the content of the narratives and to structure it into particular dimensions. After identifying the dimensions, two independent researchers coded the interview data accordingly. In the first phase of the analysis, the researchers grouped the narratives into three different areas: 1) narratives about the history of FLOSS, 2) narratives that describe how FLOSS is defined, and 3) narratives that describe changes in business patterns. We noted when the respondents were discussing the issues and tracked the relationships between these groups. It soon became apparent that the third group included the greatest number of observations, so we analyzed it in more detail, after which the focus of the present paper began to emerge.

Drawing on the definition of narrative by Pentland (1999), Ramiller and Pentland (2009) defined a narrative as:

"A story involves actors undertaking actions intended to accomplish certain goals by certain means, within specific settings, leading to particular outcomes."

In our analysis, we adopt the approach noted by Ramiller and Pentland (2009) and focus on the actions, goals, means, and outcomes of FLOSS pioneers and their effects on the software business. That is, in the narratives obtained from the selected FLOSS practitioners, we were particularly interested in the actions taken by the actors in their accomplishment of certain goals, by certain means, and within specific settings, as they seek to achieve particular outcomes.

In the second part of the analysis, we coded our transcribed interviews following the approach advanced by Ramiller and Pentland (2009) and grouped them into four primary categories: actions, goals, means, and outcomes. To improve the transparency of the analysis and the validity of the findings, we entered into a process of reflecting the findings between two independent researchers. This process became especially helpful when the researchers had divergent views regarding the group to which a certain part of the narratives belonged. These differences were ultimately resolved in all instances that caused disagreement.

Table 1: The focus of the narratives and the identified dimensions

Focus	Dimension
Goals	User involvement
Means	External resources
Actions	Innovation process
Outcomes	Revenue models

In addition to the analysis, we identified the organizational impacts of the FLOSS–driven changes in the business of companies taking part in open source software development. We were especially interested in the organizational responses these changes evoked. The focus of the analysis and the dimensions identified are listed in Table 1; they are explained in detail in the following section. We tracked the measures being taken by the entrepreneurs to address these issues. The identified dimensions of the impacts of FLOSS on the business of the case companies include changes in the innovation process, user involvement, external resources, and revenue models.

Feedback of the respondents

To improve the validity of the results, the interviews were transcribed and lodged into an interview protocol and sent to the respondents for their comments. The respondents were explicitly asked for their comments and input on the protocol. By doing this, the authors ensured that they had not misinterpreted the respondents and that the respondents agreed that the research had been conducted according to high standards. All respondents sent comments and agreed that they had been quoted correctly. The respondents also reflected on the results.

The respondent from Yoga stressed that as a platform, open source enables finding niches in the markets. This is because the necessary investments in this case were lower compared to other solutions thanks to the available FLOSS platform. The respondent wanted to underscore that it is not necessary for a start-up software company to conquer an already saturated market. One respondent from OurDB (pseudonym) stated that he would have stressed some things differently, but that there was nothing ultimately incorrect in the way he had been quoted, or the way in which the research was conducted. Moreover, the respondent from Yoga stated that he did agree with most of the findings, although he had not thought about the issues in their analyzed contexts. The respondent in Tripod accepted the interpretation and did not see anything wrong with the conclusions, although he admitted that some of his opinions had changed since the interview.

Tulip respondents sent comments related to the clarification of the units role, but did not protest the findings.

Discussion

Drawing upon the narratives conducted in the discussions with the management of our case companies, we identified fundamental changes in their business in regard to the software development processes, user involvement, and the use of external resources. Finally, we identified the shifts in revenue models as firm-level responses to these changes.

The narratives the respondents' provided to describe their businesses included a mixture of scientific theories and tried-and-true business practices. Nevertheless, they clearly illustrated how the respondents view their business environment and upon which framework they base their decision-making strategies. The narratives of the senior managers were autobiographical, emotionally loaded, and even ideological, to some extent, as one might expected from committed software entrepreneurs, especially since their business model choices were linked to their ideological ones.

"In addition to the business rationale behind our decisions, there are of course the ideological reasons. …we wouldn't be doing this; we wouldn't have started our own company if we wouldn't have believed in FLOSS. We believed in it, and still believe, because it has wonderful merits." (CEO, Tripod)

FLOSS induces user involvement in software development

The theme that FLOSS offered increased customer involvement and was raised in several stories offered by the software industry pioneers. Integrating customer feedback and requirements to the software was seen as one of the main opportunities and challenges. The respondents confirmed that FLOSS development was organized differently from proprietary development and thus is able to respond faster to customer needs.

The interviewees agreed that FLOSS enables and invites user participation in software development. There were differences in the numbers of possible contributions among the different software product categories, but users played a clear role as a contributor to the software project, product, and service.

"We would never have gained 5 million users to our database product without acting according to the principles of the open source software community. Since we first released our software under an open license, we have gathered feedback … development ideas, problem descriptions and solutions … and responded to all possible initiatives from the user community to develop the product with the skillful individuals using the product." (CEO, OurDB)

"At OurDB we love users who never pay us money. They are our evangelists. No marketing could do for us what a passionate user does when he tells his friends and colleagues about the software. Our success is based on having millions of evangelists around the world. Of course, they also help us develop the product and fix bugs." (CEO, OurDB)

Yoga's manager referred to "classical OS development" and stated that he had contributed to several FLOSS projects, according to his personal needs and preferences. This is because ready-made software did not include the necessary functionality.

"The main idea of FLOSS is working together to create tools everyone wants to use individually." (Entrepreneur, Yoga)

"We develop and use FLOSS, but in fact you might also call this a consortium approach that uses FLOSS for supporting our activities." (Project manager, Tulip)

The managers of Nemesis continued:

"Our solutions are made for the customers, not for ourselves. We want to build a working solution, but we want the customers to sit down with us, so we can do it on the users' terms. --- We believe that it is not enough for us to provide open source software. In our opinion, customer should also have open access to the actual work process. --- Not only through external communication, but also in internal collaboration. We want to get the customers' messages heard." (CEO, Nemesis)

Our respondents assumed that customers had the necessary capability and willingness for the elicitation of their requirements, and that this information should guide the development of software products. In this vein, customer expectations determined which solutions would become commercially viable. The significant role played by such expectations was taken for granted; the respondents assured us that customer expectations drive actual customer behavior and their identification is thus salient to software vendors. Customer requirements were categorized differently by the various interviewees. For example, some talked about technological requirements for certain functionalities, while others discussed business requirements, such as the maintenance, security, training, and adaptability of the software. In some other application domains, the global reach, high level of penetration, and the possibility of developing the product according to tastes were seen as guarantees that the company could answer customer requirements better.

Open source development relies on external resources

Due to the search and use of external resources and capabilities, the borders between networked companies and their environments are becoming blurred. One of the main advantages of FLOSS is the external contribution made by users and developers. Harnessing this innovation potential would allow the production of software and services that would be more tailored to users' needs.

The interviewees were like-minded in that external competencies are becoming increasingly important in terms of competition. According to the respondent from OurDB, external resources are of key importance to the company's success:

"The vast community of [our OurDB product] users and developers is what drives our business." (CEO, OurDB)

"We have five million server installations in use worldwide. Around them there are small 'software ecosystems.' There are books and articles written, lectures held, courses taught, and applications developed around our products. This community of volunteers is our most important asset. Yet, it is difficult to define." (CEO, OurDB)

With a large user group, you can attain a higher product quality, as a larger number of people use the software in different situations and provide feedback. It also limits your development costs, as you will receive some of the software from others. Sometimes greatest ides come from outside -ideas that you never had though of. In this vein, users can widen your view (Project manager, Tulip)

Moreover, a respondent from Tripod described the context in which resources are publicly available, but then stated that the capability to make use of these resources to capture value potential is essential. In this way, even though the original developers share the code in the FLOSS community, clients gladly rely on their knowledge in applying the code in the development of applications, consultancy, training, and maintenance of the software.

To conclude, the ability to utilize external resources and capabilities is recognized as one of the key factors in remaining competitive in the software business. As public goods, FLOSS-based platforms, components, and applications shift the focus from the development of proprietary innovations to the use of the goods and knowledge that is publicly available.

OS-based software development urges opening the innovation process

The openness of innovation activity is a key theme in commercial FLOSS development. This is evident in the respondents' accounts of companies' innovation processes. The responses from our study depict a fundamental difference between open and closed innovation paradigms.

"It is possible to create this kind of a joint project only if you let people see that their response has some effect on the software. ---

There was a lot to do with our software before it was ready, but we opened in a very early stage. We were able to give plausible promise and thus received a lot of valuable feedback. This resulted in a quite different end product." (Respondent from Tripod)

Hence, the quality of innovation outputs is an essential reason to engage in open innovation. The open innovation form embodies working together with numerous partners and various members of the FLOSS community. In such an innovation model, the feedback loops are short, while the software results from continuous improvement.

Our narratives underscore that through FLOSS activity, firms open their innovation processes to benefit from the knowledge and the innovation capacity of diverse OSS communities. In this way, they aim at the inclusion of several developers and users to benefit from their innovation capacity for shorter lead-times, shorter times to market, and ultimately, better product quality.

FLOSS-based public goods change the revenue models

A vital consideration in FLOSS activity is how it changes the means of value capture in software businesses. During the interviews, our respondents tended to discuss services, rather than products. This can be attributed, to some extent, to the fact that the companies provide services, not products. Yet, it is also apparent that the interviewees saw the future of the software business as being in services, rather than in commoditized products.

The respondents agree that proprietary software cannot compete successfully for long in the same market as a complementary FLOSS product. There are several reasons for this. The most obvious reason seemed to come through in a statement by the CEO of OurDB:

"---the business will have a fierce price war, where profits disappear. ---

Our narrators verbosely discuss their revenue strategies in the FLOSS business. They have experimented both with the ones that are directly product or service-related and those that indirectly benefit from the large user base.

"In the beginning we did not focus on profits at all. Instead, we focused on boosting the use of the software. --- The vast community of users and developers is what drives our business. Then we sell our offerings to firms – those who need to scale and cannot afford to fail. The enterprise offering consists of certified binaries, updates and upgrades, automated DBA services, 7x24 error resolution, etc.

-Enterprise software buyers are tired of complex pricing models (per core, per CPU, per power unit, per user, per whatever the vendor feels like that day)—models that are still in use by the incumbents. -You pay by service level and the number of servers. No nonsense, no special math." (CEO, OurDB)

"-Basically our revenue streams are very broad and far-fetched. So, any reduction of problems in our service reduces cost and is actually an increase in our profit" (Project manager, Tulip)

The narratives regarding the pricing principles of FLOSS revealed that the sales prices of the software were not very interesting to the informants; this result was not anticipated. However, the narratives underscored the total cost of ownership, including all transaction costs and lock-in costs. The potential, or even analytical, difference in the purchase price was not seen as being crucial; instead, the informants agreed that it was only a small part of the total cost of ownership, and thus, the delivery price was not a key factor in selecting software. However, the respondent from Yoga claimed that the strength of FLOSS is that, in many cases, it is the most cost-efficient solution.

"Some proprietary software companies communicate – and in some cases quite correctly – that the total cost of ownership is less for proprietary products. In some cases it is, but in some others, the price favors, very clearly, open source software." (Entrepreneur, Yoga).

One of the strengths the incumbent companies have and will apply is that of switching costs. Once a company buys something from one supplier, it becomes more costly to purchase that item from some other vendor the next time. Incumbent companies often try to lock out entrants, even by giving away some of their software to undermine rival companies. Our Tripod interviewee described this process relative to his own open source software company:

"We tried to create our own market with FLOSS and [by] selling service. Our competitors responded by starting to give away their software." (CEO, Tripod).

Networking effects seem to play an important role in the FLOSS revenue models, as it is possible to evade a lock-in to one company, as long as there are several FLOSS vendors operating on the same platform. Assume that a software product is evolving, there are several developers that sell customization and one company that coordinates the project. If a company does not purchase from the coordinator, but instead purchases from some of its rival suppliers, then, according to our respondent from Tripod, "how can he be sure that he can use the customized features in the future versions?"

However, some of the respondents pointed out that all proprietary software products will not disappear, since FLOSS products require special circumstances – at least a number of interested users and motivated developers to form a community.

"Open source will erode the great margins of the widely used software packages and will force the providers of those packages into [the] service business." (CEO, OurDB).

Our informants are united in that when software is distributed freely, the traditional revenue sources are waned, and firms are compelled to develop novel revenue models that may be based on services and only indirectly bound to the distribution of software licenses.

Conclusions

Along with the emergence of the open forms of innovation, there is a growing amount of software that is available for free. In addition, enthusiastic developers participate in numerous FLOSS projects on a voluntary basis. Therefore, it is apparent that FLOSS communities represent a means by which firms can increase the total amount of resources they can draw upon in their innovation processes. FLOSS development also paves the way for new and virtual methods of collaboration between software vendors and users. One of our informants provided us with a statement about the way FLOSS activity has changed the software industry:

"I think the architecture of participation that is embedded in the open source philosophy is a superior innovation method. And it is not limited to software. Look at Wikipedia. Software developers were the first ones to adopt it. The simple fact that everything you create is open for scrutiny by anyone else is a strong incentive to produce good stuff from the beginning. And the meritocracy of open source leads to faster innovation and thereby better innovations. It is a Darwinian system where over time, the best solutions will emerge." (CEO, OurDB)

The narratives indicate that through participation in FLOSS development, firms reframe system boundaries, and complement their traditional means of software development with new techniques for activating and involving external resources. Successful FLOSS companies are active in creating and making use of innovations developed beyond their organizational boundaries. The distributed nature of such innovation places additional demands on firms that aim to benefit commercially. This calls for the development of novel revenue models.

Our findings concur with the recent study on FLOSS development conducted by Vujovic and Ulhoi (2008) and a study on its impact on software business by Fitzgerald (2009). While we agree with Fitzgerald (2006) on the nature of software business transformation, we identify four changes that occur in the software business. These changes are summarized in Table 2.

Table 2: Identified FLOSS-induced changes in the business of companies engaged in open source software development

Focus	Dimensions	Impact
Goals	User involvement	FLOSS activity emphasizes user involvement in software development and delivery.
Means	External resources	FLOSS activity emphasizes access to external capabilities, rather than internal resource ownership.
Actions	Innovation process	FLOSS-based software development urges software innovators to open up their innovation processes.
Outcomes	Revenue models	FLOSS-based public goods change the revenue models of firms taking part in OSS development.

Table 2 summarizes the findings concerning the impact of FLOSS on the business of companies that take an active part in open source software development. Four dimensions emerge from the analysis: 1) the impact of FLOSS on the innovation process of the firms; 2) user involvement; 3) the use of external resources in software firms' operations; and 4) the revenue model as part of the firms' business models.

The findings justify the following proposition in regard to the effects of FLOSS within companies' software development practices:

Proposition 1: FLOSS-based software development urges software innovators to open up their innovation processes in a way that calls attention to user involvement in software development and delivery. In doing so, FLOSS activity emphasizes access to external capabilities, rather than internal resource ownership.

In addition, the findings give rise to the following proposition regarding the effects of FLOSS development on the business of firms that participate in open source software development:

Proposition 2: FLOSS-based public goods change the focus of competition in the software business from product-centric to service-centric operations.

The second proposition means that FLOSS has an impact on the primary software industry as a whole, as it wanes the traditional sources of revenue, and compels firms to develop new revenue models primarily based on service. Grounded on these empirical observations, this work has some important theoretical and practical implications.

Theoretical implications

Some proponents of FLOSS favor viewing source code disclosure and engaged entrepreneurial activity as somewhat opposed to each other. This view may hinder the understanding on the relationships between the publication and free availability of the software code and the entrepreneurial reality. Furthermore, it clouds how the entrepreneurial experience linked with the FLOSS development paradigm drives changes in the innovation environment of software firms. Therefore, the FLOSS phenomenon, as it is captured in this study, calls for the investigation of the behavior of individuals making the decisions to engage in the FLOSS activity at both the firm and industry levels. FLOSS companies have opened up their innovation processes and built their software using open components and products.

In particular, the findings imply that it is important to understand the effects of FLOSS on individual companies on the entire software industry, not only on single development projects and individual participants' motives and behaviors. Our work underscores the need to investigate software business management from an entrepreneurial perspective. In this vein, the present study contributes to the theoretical discussion on open innovation by extending its prevailing socio-technical perspective.

Managerial implications

This study highlights the ways FLOSS affects the business of firms that choose to participate in FLOSS development. Our findings indicate that these effects go far beyond the firms' offerings. These include the effects on the innovation process, user involvement, resources, and revenue models. Therefore, to capture the potential and benefit from the external resources in their innovation activity, managers should focus the users of their software already at the early phases of their innovation processes to benefit from users' contributions. Moreover, user involvement may foster commitment to the company's offerings. Other software firms can also benefit from readily available FLOSS components.

Our findings also illustrate that the business models of companies who do not endorse FLOSS will face changes. If some companies provide their software as "public goods", the managers of all firms providing competing offerings must rethink their revenue models. Congruent with existing studies on FLOSS business models, our findings underscore that the new revenue models are increasingly grounded on services. These FLOSS-induced changes can affect all firms in the software industry and are not limited to companies engaged in FLOSS activity.

Limitations and avenues for future research

We limited the scope of this study to the primary software industry and chose to view the business environment from a managerial perspective. As a result, there is an obvious bias slanted towards favoring FLOSS from the managers. We do, however, believe that our analyses of the FLOSS changes would have be significantly poorer had we erased the perspective of a FLOSS entrepreneur when conducting our investigation on the FLOSS–induced changes in the software industry.

For scholars interested in FLOSS and the software industry, the findings provide several avenues for further research. For example, there is a need to investigate the constraints of FLOSS in different business models or hybrid models based on both closed software processes and FLOSS. In addition, the present study calls for more empirical research on software development practices, user requirements, and the methods employed to add value to users.

We claim that there is also a need to explore the actual changes further in both individual companies and the software industry as a whole. More specifically, our explorative study calls for more research on the changes that the FLOSS activity induces in the operating environment of all software companies, not only those that are active FLOSS pioneers or their competitors. In addition, there is a need for further investigation into the changes that FLOSS brings to the secondary software sector.

References

ALLEN, R C (1983) Collective invention. *Journal of Economic Behavior and Organization*, **4**,1-24.

ALVAREZ, R and URLA , J (2002) Tell Me a Good Story: Using Narrative Analysis to Examine Information Requirements Interviews During an ERP Implementation. *The DATA BASE for Advances n Information Systems,* **33** (1), 38-52.

BARNEY, J (1991) “Firm resources and sustained competitive advantage,” Journal of Management, 17, 1, 99-120.

BARTIS, E and MITEV, N (2008) A multiple narrative approach to information systems failure: a successful system that failed. *European Journal of Information Systems* **17**, 112-124.

BAUM, J A C and OLIVER, C (1992) Institutional embeddedness and the dynamics of organizational populations. *American Sociological Review* **57**, 4, 540-559.

BENUSSI, L (2005) Analyzing the technological History of the Open Source Phenomenon. Stories from the Software Evolution http://opensource.mit.edu/papers/benussi.pdf [retrieved 15.3.2009]

BERGER, P and LUCKMANN, T (1966) The Social Construction of Knowledge: A Treatise in the Sociology of Knowledge. Anchor Books, Doubleday, New York.

LJUNBERG, J (2000) Open source movements as a model for organizing. *European Journal of Information Systems,* ***9**, 208-216.*

BROWN, A and JONES, M (1998) Doomed to Failure: Narratives of Inevitability and Conspiracy in a Failed IS Project. *Organization Science*, **35:1**, pp. 73-88.

BURR, V (1995) An introduction to Social Constructionism. Routledge, London.

CHESBROUGH, H (2003) Open Innovation: How Companies Actually Do It?. *Harvard Business Review*, 81::7, 12-14.

COFFEY, A and ATKINSON, P (1996) Making Sense of Qualitative Data Analysis: Complementary Strategies. Sage, Thousand Oaks, CA.

CZARNIAWSKA, B (1998) A Narrative Approach to Organization Studies. *Qualitative Research Methods Series*, **43**. Sage, Thousand Oaks, CA.

DAHLANDER, L and MAGNUSSON, M (2008) How do Firms Make Use of Open Source Communities?. *Long Range Planning* **41**, 629-649.

DAHLANDER, L and MAGNUSSON, M (2005) Relationship between open source software companies and communities: Observations from Nordic firms. *Research Policy*, **34**, 481-493.

DI BONA, C OCKMAN, S and STONE, M (eds.) (1999) Open Sources: Voices from the Open Source Revolution. O'Reilly & Associates, Sebastopol, CA.

EISENHARDT, K and MARTIN, J (2000) “Dynamic capabilities: what are they?” Strategic Management Journal, 21, 1105-1121.

ELPERN, J (2009) “A Framework for Understanding the Open Source Revolution. *International Journal of Open Source Software & Processes*, 1, 3, pp. 1-16.

ESSERS, C (2009) “Reflections on the Narrative Approach: Dilemmas of Power, Emotions and Social Location While Constructing Life-Stories,” Organization, 16:2, 163–181.

FITZGERALD, B (2006) “The Transformation of Open Source Software,” MIS Quarterly, 30:4, 587-598.

FITZGERALD, B (2009) “Open Source Software Adoption: Anatomy of Success and Failure. *International Journal of Open Source Software & Processes,* 1, 1, pp. 1-23.

FORD, J D and FORD, L W (1995) “The role of conversations in producing intentional change in organizations,” The Academy of Management Review, 20, 541-70.

HANNAN, M T and FREEMAN, J (1984). Structural Inertia and Organizational Change. *American Sociological Review*, **49**, 2., 149-164.

HECKATHORN, D (1996) “The Dynamics and Dilemmas of Collective Action,” American Sociological Review, 61(2), 250-277.

HECKER, F (2000) “Setting up shop: the business of open-source software,” Available at: http://www.hecker.org/writings/setting-up-shop [retrieved 15.3.2009]

HENKEL, J(2008) "Champions of revealing—the role of open source developers in commercial firms," Industrial and Corporate Change, 18:3, 435–471.

KOGUT, B and METIU, A (2001) "Open-source software development and distributed innovation" Oxford Review of Economic Policy, 17:2, 248–264.

LANZARA, G F (1991) "Shifting stories: Learning from a reflective experiment in a design process" In: SCHON, D. A. (Ed), The reflective turn: Case studies on practice and in practice. Teachers College Press, New York.

MCDANIEL, T R (2004) “A software-based knowledge management system using narrative texts,” Doctoral dissertation, University of Central Florida, Orlando, FL.

MEYER, P (2003) "Episodes of collective invention," working paper 368, US Department of Labor, Bureau of Labor Statistics, Washington, DC.

MOCKUS, A Fielding R, and Herbsleb J (2002) “Two case studies of open source software development: Apache and Mozilla,” ACM Transactions on Software Engineering and Methodology, 11:3, 309-346.

MONGE, P R FULK, J KALMAN, M E and FLANAGIN A J (1998) “Production of Collective Action in Alliance-Based Interorganizational Communication and Information Systems,” Organization Science, 9:3, 411-433.

O’LEARY, M ORLIKOWSKI, WJ and YATES, J (2002) “Distributed work over the centuries: trust and control in the Hudson’s Bay Company 1670–1826,” In: Hinds P., and Kiesler, S. (Eds.), Distributed Work. MIT Press, Cambridge, MA.

OLSON, M (1965) “The Logic of Collective Action,” Harvard University Press, Cambridge, MA.

OSTERLIE, T and JACCHERI, L (2007) “A Critical Review of Software Engineering Research on Open Source Software Development,” In the Proceedings of the 2nd AIS European Symposium on Systems Analysis and Design, Gdansk, Poland, June 5, 2007.

PENTLAND, B T (1999) “Building process theory with narrative: From description to explanation,” Academy of Management Review, 24:4, 711-724.

PFEFFER, J (1981) “Power in Organizations.” Pitman, Marshfield, MA.

PHILLIPS, N and HARDY, C (1997) “Managing multiple identities: Discourse, legitimacy and resources in the UK refugee system,” Organization, 4:2, 159–186.

RADTKE, N JANSSEN, M and COLLOFELLO, J (2009) “What Makes Free/Libre Open Source Software (FLOSS) Projects Successful? An Agent-Based Model of FLOSS Projects”, International Journal of Open Source Software & Processes, 1:2, 1-13

RAMILLER, N and PENTLAND, B (2009) "Management Implications in Information Systems Research: The Untold Story," Journal of the Association for Information Systems, 10:6, Article 2, 474-494.

RAYMOND, E S (1999) “The Cathedral and the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary,” O’Reilly & Associates, Inc., Sebastopol, CA.

RHODES, C and BROWN, A D (2005) “Writing Responsibly: Narrative Fiction and Organization Studies,” Organization, 12:4, 467-491.

RICE, L and EZZY, D (1999) “Qualitative Research Methods: A Health Focus,” Masako Ono-Kihara Japan.

SCACCHI, W (2007) “Free/Open Source Software Development: Recent Research Results and Methods,” in Zelkowitz, M. V. Advances in Computers, vol. 69, pp. 243-269. Academic Press.

SHAPIRO, C and VARIAN, H (1999) “Information rules: A Strategical Guide to the Network Economy,” Harvard Business School Press, Boston, Massachusetts.

SZCZEPANSKA A M, BERGQUIST, M and LJUNBERG, J (2005) “High Noon at OS Corral: Duels and Shoot Outs in Open Source Discourse,” in: J. Feller, B. Fitzgerald., S. A. Hissam, and K. R. Lakhani. eds. Perspectives on Free and Open Source Software. MIT Press, Cambridge, MA.

TEECE, D J PISANO, G and SHUEN, A (1997) “Dynamic Capabilities and Strategic Management,” Strategic Management Journal, 18, 7, 509-533.

TUSHMAN, M L and ANDERSON, P (1986) “Technological Discontinuities and Organizational Environments.” Administrative Science Quarterly 31, 439-465.

UDÉHN, L (1993) “Twenty-Five Years with the Logic of Collective Action,” Acta Sociologica, 36, 239-261.

VAN DER LINDEN, F LUNDELL, B and MARTTIIN, P (2009) "Commodification of Industrial Software: A Case for Open Source," IEEE Software 26:4, 77-83.

VARGO, S L and Lush, R F (2004) “Evolving a services dominant logic,” Journal of Marketing, 68, 1-17.

VEN, K VERELST J and MANNAERT, H (2008) “Should You Adopt Open Source Software?” IEEE Software. May/June 2008, 54-59.

VISWESWARAN, K (1996) “Fictions of Feminist Ethnography,” Oxford University Press, Delhi.

VITARI C and RAVARINI A (2009) A Longitudinal Analysis of Trajectory Changes in the Software Industry: The Case of the Content Management Application Segment. *European Journal of Information Systems*, **18**, 249-263.

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, March–April 2003.

VON KROGH, G and SPAETH, S (2007) “The open source software phenomenon: Characteristics that promote research,” The Journal of Strategic Information Systems 16:3, September 2007, 236-253.

VON KROGH, G VON HIPPEL, E (2006) “The Promise of Research on Open Source Software,” Management Science, 52:7, 975-983.

VUJOVIC, S and ULHØI, J P (2008) “Online innovation: the case of open source software development,” European Journal of Innovation Management, 11:1, 142-156.

WEICK, K SUTCLIFFE, K M and, OBSTFELD, D (2005) “Organizing and the Process of Sensemaking,” Organization Science, 16:4. 409-421, Jul/Aug 2005.

WHITE H (1981) “The value of narrativity in the representation of reality.” in On Narrative (Mitchell W.J.T. ed.), The University of Chicago Press, Chicago, 1-24.

WILLIAMS, S (2002) “Free as in Freedom: Richard Stallmans Crusade for Free Software,” O´Reilly, Sebastopol, CA.

XU, L and BRINKKEMPER, S (2007). Concepts of product software. European Journal of Information Systems, **16**, 531-541.

Appendix 1: Cases

Case 1: Tripod (pseudonym) is a small Finnish FLOSS company that specializes in developing collaborative learning and knowledge management software, related training, and consultancy. Its revenue model is largely based on service contracts with public organizations. These contracts cover software updates, new feature development, support, and training. The company was founded in 1996 and has its headquarters in Helsinki, Finland. The company had three employees at the time of the first interview in 2005. In the beginning of 2010, the company employed seven people.

Case 2: OurDB (pseudonym) is a large Finland-based international firm that specializes in relational database management systems and related services. Development of the company's main product began in 1995. The company's revenue model is based on dual licensing, which means it provides both FLOSS and proprietary versions of its software. Software development is managed by the company's staff, but it depends upon the innovation capacity of a worldwide community comprised of millions of users and thousands of co-developers.

Case 3: Yoga (pseudonym) is a small Finnish entrepreneurial firm that focuses on consultation and FLOSS development. The firm specializes in combining relational databases and e-mail management tools. It is based in Helsinki, Finland. At the time of the interview, the company employed only the founder, who is an expert in e-mail management software and FLOSS project management. Its revenue model is based on selling software development services and consultancy.

Case 4: Nemesis (pseudonym) is a small Finnish company that specializes in FLOSS-based Web services, especially content management systems, related services, and online support. The company was founded in 2001. Its revenue model is based on selling per-hour coding work. The software development is managed by both the company's own staff and an active developer community.

Case 5: Tulip (pseudonym) is a small unit belonging to a branch of a large multinational corporation. It produces FLOSS software systems for the interoperability testing of the company's main product lines. The parent company is based in the Netherlands and consist of three persons. The main source of their funding comes from the parent company and a partner company, as both are expecting gains from supporting their main product and reducing interoperability problems. Unit launched its main product as an OSS to gain credibility as an independent and neutral test tool, but also to attract outside contributions to the development of the software. Currently, the founding companies are responsible for almost all of the development.

Aalto-DD 13/2011

Open Source Software research has established that OSS technology (tools and practices) holds potential. Based on a systematic literature review and a research engagement, this dissertation describes how organizations leverage OSS practices to produce software. It provides a narrative of how the term OSS travels from the writings of enthusiasts to the daily practices of organizations. The findings underline the importance of local renegotiation of the term OSS. This renegotiation provokes structural changes in 1) the organizations that adopt OSS technology, but also in 2) the industries these companies operate in. The main contribution of this doctoral dissertation is to promote the idea that OSS in organizations should be researched in a sensitivized manner. This requires moving away from too simplistic institutional contexts. Another contribution is to reduce uncertainty about the adoption of OSS technology and to help build a capacity to accept, search for, motivate and reward contribution.

9 789526 040370

ISBN: 978-952-60-4038-7 (pdf)
ISBN: 978-952-60-4037-0
ISSN-L: 1799-4934
ISSN: 1799-4942 (pdf)
ISSN: 1799-4934

Aalto University
Aalto University School of Economics
Information Systems Science
aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

DOCTORAL
DISSERTATIONS