2010

Open Source Technology Changes Intra-Organizational Systems Development - A Tale of Two Companies

Juho Lindman
Helsinki School of Economics, juho.lindman@hse.fi

Matti Rossi
Helsinki School of Economics, matti.rossi@aalto.fi

Pentti Marttiin
Nokia Siemens Networks, pentti.marttii@nsn.com

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

Journal:	*18th European Conference on Information Systems*
Manuscript ID:	ECIS2010-0319
Submission Type:	Research Paper
Keyword:	OSS/FLOSS, Organizational change, Institutional theory, Critical perspective

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.

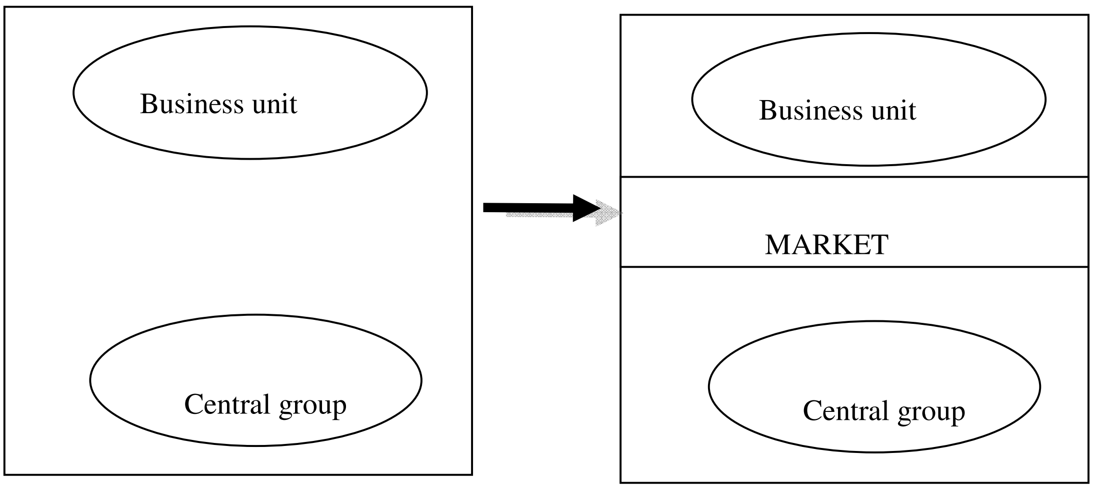

Figure 1: Change in organizations

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.