5-15-2012

JUSTIFYING THE VALUE OF OPEN SOURCE

Magnus Bergquist
University of Gothenburg

Jan Ljungberg
University of Gothenburg

Bertil Rolandsson
University of Gothenburg

JUSTIFYING THE VALUE OF OPEN SOURCE

Bergquist, Magnus, University of Gothenburg, 41296 Göteborg, Sweden, magnus.bergquist@gu.se

Ljungberg, Jan, University of Gothenburg, 41296 Göteborg, Sweden, jan.ljungberg@gu.se

Rolandsson, Bertil, University of Gothenburg, 41296 Göteborg, Sweden, bertil.rolandsson@gu.se

Abstract

Over the last decade, free and open source software (FOSS) has gradually become recognized by different actors in society outside FOSS communities and increasingly integrated in corporate software development, challenging proprietary software practices and establishing new open source companies. Literature describing this transition is focusing a narrow view on the value of using FOSS, mainly understanding it as an efficient alternative to established models for software development. This is not sufficient to fully understand the uptake of FOSS into companies. In order to gain a deeper understanding of this, there is a need to articulate a wider range of different values associated with FOSS and how they interplay in the intersection of corporations and movements. To do this we propose the order of worth framework developed by French sociologist Luc Boltanski and colleagues, which focus on the arrangements of value logics as an analytical strategy to understand how values form strong or weak arrangements in processes of institutionalization. By applying the framework on key texts from the free and open source software movement as well as on a an interview study with professional software developers employed by firms, we set out to identify how values associated with FOSS become justificatory arrangements that give legitimacy to FOSS and how these arrangements change over time, from the early free software movement to the emergent uptake of FOSS in contemporary professional software development. By understanding how justificatory logics come to play and interplay, corporations that want to adopt FOSS can better manage their engagement in FOSS activities.

Keywords: Free and Open Source Software, Orders of Worth, Justification logics, Value of Open Source.

1 Introduction

Over the last decade, free and open source software (FOSS) is increasingly incorporated in professional software development contexts (Bonaccorsi and Rossi, 2003; Demil and Lecocq, 2006; Lerner and Tirole, 2002; Spagnoletti and Federici, 2011; Ågerfalk and Fitzgerald, 2008). This development has been described as OSS 2.0, progressive open source, corporate code, and professional open source, indicating an adaptation of FOSS (Fitzgerald, 2006; Gurbani et al., 2006; Dinkelacker et al., 2002). A recent research stream also outlines FOSS and its mode of software co-production as an engine for innovation (Morgan and Finnegan, 2010; Murray and O'Mahony, 2007; Osterloh and Rota, 2007; von Hippel and von Krogh, 2003).

Most literature describing this transition tends to focus on FOSS as an efficient, high quality but less costly alternative to established models for software development. However, given the major differences between community and company goals and value rationalities this focus on efficiency and economic value cannot sufficiently explain why FOSS is actually chosen. Even if FOSS has challenged traditional software industry and contested its practices, there are differences between community and company goals that professional developers have to manage when adopting cultural, economical and social practices of FOSS.

There is a need to identify how the value of FOSS has become an asset in professional software development and how different forms of values associated with both company and FOSS movement interplay with FOSS entering companies. To gain this understanding we propose the *order of worth* framework developed by French sociologist Luc Boltanski and colleagues (Boltanski and Thévenot, 2006; Boltanski and Chiapello, 2005), which focus on the arrangements of value logics as an analytical strategy to understand how values form strong or weak arrangements in processes of institutionalization. The purpose is to identify how values associated with FOSS become justificatory arrangements that give legitimacy to FOSS and how these arrangements change over time, from the early free software movement to the emergent uptake of FOSS in contemporary professional software development. The research question is: *how do professional software developers justify FOSS when introduced in their work practice*? By understanding how justificatory logics come to play and interplay, corporations that want to adopt FOSS can better manage their engagement in FOSS activities.

The paper is structured as follows. First we present the order of worth framework. After that the method section describes the data used and methodological considerations. Following that the two empirical sections are presented. First justificatory arrangements during different historical periods of FOSS are analyzed to give accounts for the value base from which FOSS is today integrated into companies. The latter part of the analysis contains the interview study with professional developers and how they integrate FOSS into their business. Discussion and conclusion completes the paper.

2 Logics of Justification

In a series of texts Boltanski et al. have proposed a theoretical framework for analyzing logics of justification in societies (Thévenot, 2001; Boltanski and Chiappello, 2006; Boltanski and Thévenot, 2006), what they call “orders of worth”. The framework identifies six different logics or justificatory regimes by which societies justify different social orders. These are coherent and systematic principles of evaluation that strive for universality but are in reality more or less fragile compromises between different logics of justification. The more a value refers to “common good” the stronger it becomes as a justifying logic. A *justificatory arrangement* is a combination of logics of justification in a society during a certain time period. The more diverse and the less universal a justificatory arrangement is, the weaker legitimacy it can create. Justificatory logics are thus shared beliefs inscribed in institutions and bound up with actions in a certain time. Logics of justification are also used to legitimate changes. The logics of justification can therefore be seen as tools that may function to manage uncertainties or fragile organizational circumstances associated with the adaption to new phenomena, such as e.g. new

technologies, business models, organizational behavior etc. The six logics of justification suggested by Boltanski and Thévenot (2006) are the following:

- An *inspirational logic* is founded on a principle of grace or artistry serving what is perceived as authentic qualities of life, e.g. manifested by creativity or authenticity. A "great person" would be akin to a saint or an inspired artist.
- A *domestic logic* is founded on an established hierarchy made out of personal interdependencies, with a patriarch on top. It is justified by referring to a stable social order or tradition. An example could be a conservative family organization. A noble person would here be the father or elder, to whom respect and loyalty is due, and who in return gives protection and support.
- In a *popular logic* justification is reached through importance of being renowned, by being granted credit and esteem in the opinion of others. The value is dependent on identification and fame.
- In a *civic logic*, justification relies on being representative and on acting in accordance with a collective will. Value is created through the capacity to mobilize collectives around common interests. In this process, moral claims, and definition of identity become important. A great person is the representative of the group, who expresses its collective will.
- In a *market logic* justification depends on individuals and their ability to possess and compete. The value is related to individuals' selling and buying goods and services. This can be perceived as an egoistic practice. However, the right to possess and seize market opportunities is related to a claim that, if done fairly common good will emerge out of market transactions. The great one is the entrepreneur, the person who makes a fortune.
- An *industrial logic* justifies actions and initiatives by referring to efficiency and the scale of abilities. Contrary to the market logic, the industrial logic focuses on whether functionality and productivity is organized in a reliable way. The great one is the professional, the expert or director of a large industry.

The framework is applied as a way to identify justificatory arrangements and how they develop over time by describing the historical evolution of FOSS and identify changes in the arrangements as well as analyzing professional software developers' experiences of working with FOSS in companies. The focus is not only on which logics are involved, but also how they are configured, e.g. if and how they support or contradict each other. Different actors struggle to justify the use of FOSS by referring to justificatory logics (Stark, 2009; Jagd, 2007). Weak or even competing orders of worth will create tensions and uncertainties about the consequences of adopting FOSS (Rolandsson et al., 2011). The justificatory arrangements guide developers in how uncertainties can be handled and FOSS appreciated. Through these arrangements, it is possible to identify how FOSS can become an integrative legitimizing force and which challenges are created by e.g. tensions between social order and moral claims. Furthermore, it enables tracing the role of previous and existing logics of justification on FOSS in contemporary professional software development as resources to agree and act upon (Boltanski and Thévenot, 2006).

3 Method

The result section is based on two sets of data. The first part identifies two justificatory arrangements in the historical development of the FOSS movement. The second part of the study consists of qualitative interview data describing the current uptake of FOSS in two types of companies. Methodological strategies chosen are described below.

Two justifying arrangements that historically have been used to define the value of FOSS, was traced by gathering and analyzing canonical texts i.e. discourses that are seen as the most influential, often referred to or considered to have a major impact on the perception of a phenomenon (Kilduff, 1993; Macintosh and Baker, 2002; Introna and Whittaker, 2004). Also additional work, such as journalistic accounts and previous research was used. When collecting and analyzing canonical texts we applied what Foucault named archaeology of knowledge (Foucault, 1969), which is an analytical method to compare series of sources over time in order to capture changes in dominant modes of thinking, acting

and organizing, in this case the orders of worth. Certain events function as approximate starting points of the two time periods, each composing a distinct arrangement. The early movement justificatory arrangement appears already during the 1970s, but with the free software definition and the 1985 constitution of the Free Software Foundation (FSF) it becomes established, and hacker Richard Stallman emerges as a front figure. The second justificatory arrangement named pragmatic FOSS develops from activities in the 1990s leading to the formulation of the open source definition and constitution of the Open Source Initiative (OSI) in 1998. The empirical material is thus based on texts related to OSI and one of its front figures Eric Raymond.

Hybrid companies		
Hybrid 1	Large global hardware and service oriented company	6 interviews
Hybrid 2	SME hardware oriented company	2 interviews
Hybrid 3	Large global hardware and service oriented company	3 interviews
Hybrid 4	SME Vehicle service provider	2 interviews
Hybrid 5	SME ERP developer/vendor, newly acquired by global enterprise	2 interviews
Pure-play companies		
Pure-play 1	Consultants with customer oriented approach (adapting applications, service, education, support)	4 interviews
Pure-play 2	Developer/vendor of office systems	2 interviews
Pure-play 3	Developer/vendor of Content Management System (CMS)	2 interviews
Pure-play 4	Consultants with customer oriented approach in CMS and web	4 interviews
Pure-play 5	Open source service platform and application provider	2 interviews
Pure-play 6	Open source security solutions	1 interview

Table 1. Type of companies and number of interviews performed in each company.

These historical accounts are used to contextualize professional open source developers' application of FOSS in their companies, which is conceived as an emergent third justificatory arrangement. Qualitative interviews were conducted with 30 software developers recruited from pure-play and hybrid companies (Feller and Fitzgerald, 2002). *Hybrid companies* are based on proprietary business models that gradually incorporate open source software and methods. The majority of the hybrid companies are large corporations of which most act on a global market. However, in this category also smaller firms with a proprietary history were selected. 15 interviews were done among hybrid companies. *Pure-play companies* are formed around a FOSS business model. These are typically small and medium sized (SMEs) entrepreneurial service oriented consultancy firms, with a multitude of approaches for incorporating open source software development into their work practice. We have interviewed 15 programmers employed by such SMEs. Table 1 gives an overview of the types of companies and number of interviews included in the study.

Respondents were chosen to represent a broad sample of different approaches to FOSS and software development. The interviewees included designers, coders, system developers, software engineers, software architects and to some extent project leaders. They also differed in other aspects, such as age and length of employment. Programmers working for hybrid companies with a long proprietary history frequently were qualified engineers (with university degrees) whereas those working for pure-play SMEs had a wider variety of educational backgrounds.

4 The Historical Development of Justificatory Arrangements in FOSS

4.1 Justificatory Arrangements of The Early Movement

In the early computer history there were no market for software, since it was developed directly for specific hardware (Campbell-Kelly, 2004). Programmers were used to share solutions, knowledge and source code. When a market for software then emerged, these practices of sharing were abandoned, and the source code became private company property to be carefully protected. This provoked some developers to mobilize resistance in the shape of a politically driven movement, organized as an ideologically framed commune. A number of initiatives institutionalized this resistance to proprietary software, such as the GNU project (Stallman, 1986; 2002), the Free Software definition (GNU Bulletin, 1986), the Free Software Foundation, and the GNU General Public License (GPL) that was designed to ensure that the rights of the free software definition were preserved (i.e. an inscription of the free software definition in copyright law). The "viral" character of GPL, i.e. that other software that is bundled with a GPL-licensed software must also be released under GPL, created tensions with proprietary software.

An important justification was here the *civic logic* formed on principles and rules defining free software as a common good, i.e. source code must be made available for anyone to use, alter and redistribute to secure future development of the ideas that the code entails. The proprietary practices were seen as a threat against the programmers' freedom: "fundamental act of friendship among programmers is the sharing of programs; marketing arrangements now typically used essentially forbid programmers to treat others as friends" (Stallman in Gay, 2002, p. 33).

Emanating from the roots of the hacker culture of the early sixties, we can also identify an *inspirational logic*. Programming was often seen as an art (c.f. Dijkstra, 1971). Donald Knuth described it: "The chief goal of my work as educator and author is to help people learn how to write beautiful programs" (Knuth, 1974, p. 6). This is also manifested by the culture of the MIT research groups' early experimenting with new technologies, as described by Levy (1984):

"a new way of life, with a philosophy, an ethic and a dream.[...] hackers that by devoting their technical abilities to computing with a devotion rarely seen outside monasteries they were the vanguard of a daring symbiosis between man and machine." (Levy, 1984: 39)

Hacking and playing with technology was justified as the authentic values of life. While the word hacking changed over time from "a spirit of harmless, creative fun" to "acquire a sharper, more rebellious edge" (Williams, 2002), it was still deeply linked to an inspirational logic of solving difficult problems for its own sake: "Playfully doing something difficult, whether useful or not, that is hacking." (Stallman, 2002).

During the same period we also recognize a *popular logic* of justification. This is visible in the way skilled hackers are recognized by the community based on reputation. Being recognized as a hacker was something that earned by getting respect from the community. The most admired programmers were referred to as demigods: "a person who is defined as a hacker with years of experience, a world-wide reputation, and a major role in the development of at least one design, tool, or game used by or known to more than half of the hacker community." (Jargon-file 4.3.1). Famous projects or persons could attract many contributors, creating a high value for certain FOSS initiatives.

Finally, a *domestic logic* could be traced in the early movement, resembling a tight community with closed clan like hierarchy of personal interdependence and patriarchal governance, what Raymond (1999a) later characterized as a cathedral-building style of development typical for both early free software and the proprietary software development.

The justificatory arrangement in this first time period was thus characterized by a strong civic logic built around the idea of creating a common god. This was expressed in several institutions such as the

GPL and in hacker practices based on sharing and mutual respect. Communities where to a large extent oriented away from business and market due to a critical approach towards proprietary business models.

4.2 Justificatory Arrangements of Pragmatic FOSS

During the mid-nineties a new approach to justify FOSS occurred, rooted in some developers concern with what they defined as the early movement's hostile attitude to commercial software:

"It seemed clear to us in retrospect that the term 'free software' had done our movement tremendous damage over the years. Part of this stemmed from the well-known 'freespeech/ free-beer' ambiguity. Most of it came from something worse – the strong association of the term 'free software' with hostility to intellectual property rights, communism, and other ideas hardly likely to endear themselves to an MIS manager." (Raymond, 1999a)

In order to avoid these connotations, the term open source was coined, indicating that open source is viewed as a means to an end of producing software of high quality, not an end in itself. A number of initiatives institutionalized this more pragmatic attitude. The Open Source definition explicitly states that an open source license must not "contaminate" other software (as the GPL-license). More permissive licenses were used to make it easier for open source to coexist with proprietary software (Välimäki, 2003; 2005). The Open Source Initiative (OSI) was founded in 1998 to support the new focus on technology rather than ideology. This more pragmatic nature of the movement, downplayed the ideological focus, and contributed to a wider diffusion of free and open source software, including both large software companies and new small firms dedicated to open source. Civic logic companioned by a *market logic*, which created strong tensions between members of the communities, here illustrated by the dispute between FSF spokesperson Richard Stallman (RMS) and Open Source Initiative's Eric Raymond:

"RMS's best propaganda has always been his hacking. So it is for all of us; to the rest of the world outside our little tribe, the excellence of our software is a far more persuasive argument for openness and freedom than any amount of highfalutin appeal to abstract principles. So the next time RMS, or anybody else, urges you to "talk about freedom", I urge you to reply "Shut up and show them the code." (Raymond, 1999b)

It is not philosophical or political principles, but the excellence of the software that should convince, which points to quality ideals based on an *industrial logic*. As an alternative to the domestically oriented cathedral style, where wizards were leading a tight, closed tribe of skilled hackers, Raymond proposed the "Linus Torvalds's or the bazaar style of development – release early and often, delegate everything you can, be open to the point of promiscuity" (Raymond, 1999a, p. 30).

The justificatory arrangement that previously was focused on civic logic had now become more dynamic. There is a continuing presence of both *inspirational* and *popular logic* as part of the bazaar. Skilled "hacking" is still appreciated, and inspirational value of open source software would depend on spontaneous and passionate initiatives, like Linus Torvalds initiative to develop Linux in order to learn how operating systems work (Torvalds and Diamond, 2001). The popular logic was strengthened due to the growing movement. Skilled programmers could gain reputation and fame among a high number of peers, if they succeeded to pass the peer review systems with their contributions of code. However the introduction of a market logic somewhat challenged the universal claims of a previously dominating civic logic.

5 Emerging Justificatory Arrangement in Contemporary Firms

Previous sections give a background that makes it possible to describe contemporary integration of FOSS in professional software development from the perspective of which logics are chosen and how they are justified when meeting existing justification arrangements in the companies.

5.1 Inspirational Logic Reframed

In many of the studied companies the inspirational logic is used to justify FOSS, with references back to previous justificatory logics in the FOSS movement. It is common that developers justify their use of open source by referring to how it enhances their engagement and help them seize control over and improve the quality of software, as well as enable them to make use of their 'artistic capability' to realize visions. However, there are changes in how this logic is framed. Instead of finding visions and inspiration rooted in being part of a movement struggling against proprietary software for a higher cause, FOSS is justified as a tool that makes the professional programmer more independent and capable at work. A software developer in a pure-play company argues:

"I enjoy sharing and helping others [in the Joomla community]. If they share with me I want to share with them. That is different from just buying software and using it. This is a way of life. It is fun to work with; both at home in private and here at work. You learn new things." (Open source web company programmer)

Inspirational justification is present but reinterpreted from being based on movement ideals of authenticity, to embrace individuals' professional engagement and empowerment as a professional programmer. Open source is justified by its capacity to liberate the individual – a good cause – that makes both work and leisure more challenging and fun. Hence, it was seen as important that firms should not undermine open source as a liberating force for the developer when appropriating it into a professional context.

5.2 The Decline of Domestic Logic

It is striking how the respondents downplay the importance of gurus from the early stages of the FOSS movement when they justify their engagement. Those who worked with FOSS at these companies stressed that they were not "religious" about ideas of open source values and practices. The early movement was criticized as rather domestic, and there was little loyalty towards FOSS for "historical" reasons. Avoiding proprietary solutions was not seen as a goal in itself. If needed by the customers, proprietary solutions can be used:

"There is a tension between the GPL [Gnu General Public License] and business which has consequences for what we can do and what we want to do. At the end of the day the company must earn money to survive. Richard Stallman has a very idealistic view of the world, which is admirable. But if one considers it from a business perspective one realizes that it is not feasible in practice." (Open source service platform and application provider)

It should be emphasized that there was a historical knowledge about previous justification regimes among the respondents, and contributing to FOSS was generally seen as a morally good thing. Still, a more pragmatic approach was justified mainly by also referring to market logic.

5.3 Reinterpreted Popular Logic

The popular logic central to the initial open source movement related to the reputation that was gained when programmers invented new and exciting projects, solved generic problems or got their contributions of code through the peer-review system and into the main code branch. Popularity built on respect attributed by the community. The more popular a project, the more status gained for the project owner as well as for the individual contributing programmer. The ideological leaders of FOSS were all famous for their contributions to the codebase of the commons.

According to the interviewed programmers the companies had transformed this popular logic in different ways. One strategy was to hire programmers with community track record and use this as a marketing strategy when packaging offerings to customers. This way of understanding popular logic transformed the justifying value base from "community respect" to "market success". The argument behind this reasoning is that an open source company's probability to succeed with a software

development project increases with the programmer's success as a FOSS developer. Hiring programmers greeted by the community will gain both the company and its customers.

A similar line of reasoning was found among programmers who used this reinterpreted value logic to trade competence gained in a community for competence required in the professional life by using their open source projects as showcases when applying for positions as professional software developers.

5.4 Market Logic Challenging Possessions

Several of the studied companies had the public sector as target for their business proposals, and they justified their business by being a resource helping the public sector to become more in control of their strategic investments. Being specialists on open source they could assure that the public sector could reach its goals of achieving a more open and transparent software strategy. They used the idea of open source as a different way of working where the customer would have the control over the product, which normally is not the case with proprietary software. A programmer in a pure-play company argue:

"It's a nice and clean way of distributing software that let the customer decide what to do with their code. It is also fair because no one can cheat on the customer and deliver crap. Since it is open everyone can see what's inside. Also we can collaborate to find bugs and stuff like that. Generally people like to help." (Open source web company)

A difference in relation to how Boltanski and Thevénot (2006) outlines the market logic is that justification for these firms does not rely on claims of possession but on openness as the main asset for competing, which contrasts the proprietary approach where openness normally would be regarded as a threat against competition.

However, in firms with a previous proprietary tradition a more complex relation to market logic could be identified. Firms that developed their business around patent portfolios naturally had strict arrangements regulating what the programmer was allowed to do in terms of using open solutions. Developers in this company therefore went through a formalized process governed by the patent department before they were allowed to use open source code in their software development.

5.5 Civic Logic Transformations

The civic logic that is identified in companies reminds partly of the early movement. Open source is seen as an asset strengthening the development of a "good society" that can solve user needs, a democratic approach to software development found in public sector discourse on FOSS. Tensions could occur between civic and market logics if the customer would not subject to a community's long-term vision:

"We always write agreements with the customer about the fact that everything we do in order to give them better functions etc. will be part of the community in the end. However, this also means that we as core developers have a big responsibility to review and accept the stuff that is written. […] We never develop any customer specific solutions, we always develop stuff that may easily be configured in a way that suits many different organizations." (Open source web company)

The company was heavily involved in developing the system they sold and their goal was that every customer should adhere to this logic and see their own needs as subordinate to the needs of the community. This civic logic touches upon ideas on citizens' rights and the public sector's independence from proprietary interests. Civic logic is used as a way to justify that customers must respect the community's agenda.

5.6 Open Sourcing the Industrial logic

It is not surprising that the interviewed programmers invoke an industrial logic since it justifies efficiency, quality and the need for expertise, ideals that also justify industrial values. However, as shown, industrial logic is not new in FOSS justificatory arrangements. Expertise and profound testing procedures also stand as guarantors for the efficiency of community based peer-produced high quality software. In the interviews this theme is elaborated. The developers do not just see themselves as efficient and result oriented experts, they also combine this with inspirational logic imported from the community based open source movement. Thus, an important argument is that the flexibility provided by FOSS has made it possible to reach goals faster and with higher quality than in-house developed proprietary software, with increased efficiency and decreased cost as a result.

"Our only true advantage really is that our customers who trust us know that we always deliver. They can just ask a question, for example "we want a system that has these functions, is there anything out there?" And then we can go out in the world to see if we can find something to start build upon. Really, that is what I love about this job; it is so easy to show something that immediately can be discussed, which is a major advantage when you are into user driven software development. You don't have to write specifications." (Open source web company)

As the quote implies open source software development is perceived as an engine for pushing industrial efficiency and innovation marked by high technological standards even further by engaging in co-creation of software. However, even if open source is associated with high quality and seen as an innovative potential in most firms, a more concerned approach is identified in companies with a proprietary tradition. Instead of using FOSS in close collaboration with the customer, it is viewed as a useful mean for solving advanced problems in well-defined project groups. In accordance, the industrial logic that emerges, especially in hybrid companies, embraces an image of developers working as scientists in industrial labs. Hence, an instrumental approach to open source as a tool for industrial research and development focusing company needs can be identified, where developers work with concrete problems in their labs and only indirectly contribute to technological progress compared to direct involvement in the FOSS agenda.

6 Discussion

The studied historical periods and the contemporary set of interviews with 30 professional software developers will now be discussed in relation to the order of worth framework developed by Boltanski et al. (Thévenot, 2001; Boltanski and Chiapello, 2005; Boltanski and Thévenot, 2006) from the perspective of how professional software developers justify FOSS when introduced in their work practice. The result shows that developers create a justificatory arrangement that draws from, abandons or reinterprets previous FOSS justificatory logics. Table 2 summarizes the three identified arrangements.

Civic logic is the most continuous over time followed by popular and industrial logic. Civic logic is reinterpreted in each period. In the early movement the value logic was based on the mobilization of hackers to fight emerging proprietary software development. During the phase of pragmatic FOSS civic is reinterpreted and openness becomes a more generic and inclusive value. In the emerging value logic of contemporary software development in companies civic is connected to the idea of an innovation that serves "a good cause", which has been formulated in different ways in previous periods. Except for the early movement where an inspirational logic dominates, the industrial value logic continues quite stable – also in terms of content – throughout the studied periods, based on principles of efficiency, quality and reliability in software development. The popular value logic by which credit and esteem in the opinion of others form the value base is recognized in all periods but has undergone interesting transformations. In the early movement reputation mainly was a matter for the internal community. This value base becomes more general and wide in the pragmatic justificatory arrangement. Different approaches were found among the developers, from a deep involvement in

community reputation building to business orientation and policy making. Here we also find a new approach where popularity can be a merit that is highly valued and sometimes be traded between community and company. This means that the popular value logic can inhibit several possible interpretations, which makes it resilient.

Justification logics	**Justificatory arrangement of early movement**	**Justificatory arrangement of pragmatic FOSS**	**Justificatory arrangement in contemporary firms**
Inspirational	Hacking as technical artistry and problem solving as an end in itself	Hacking as technical artistry and problem solving as an end in itself	Hacking transformed from art to scientific software development in the lab (hybrid)
Domestic	Loyalty in closed, tight communities with informal hierarchical structures	Openness strongly enforced, justified by leaders recognized by the community	Mostly absent (hybrid). Fragments of faithfulness to early movement leaders and values (pure play)
Popular	Importance of reputation and fame among peers in the community	Importance of reputation an fame in wide contexts of communities in business	Reputation and esteem derived from FOSS competence is a highly valued merit (pure play and hybrid)
Civic	Mobilizing hacker movement to defend the value of openness against emerging proprietary interests in software development	Openness valued as a good, coexisting with proprietary software	Firms promoting the value of "civic innovation" to develop the good society through the market (pure play)
Market	Market logic is absent	Mobilizing openness to develop competitive software	The value of competing with openness on the market (pure play and hybrid)
Industrial	Industrial logic is absent	The value of mobilizing the crowd to enhance large scale industrial quality and efficiency	The value of openness to enhance efficient software innovation and production (pure play and hybrid)

Table 2: Summary of justificatory arrangements in three periods of FOSS.

The domestic justification logic, which during the two first arrangements is strong and based on an appreciation of tight social relationships, clearly disappears as FOSS enters contemporary firms. The inspirational logic first show continuance but becomes reinterpreted by the interviewed developers, from artistry to valuation of inspiration as a contributor to scientific problem solving in development labs. Market, finally, as justification logic is not visible in the early movement's community oriented FOSS. The debate between the Free Software Foundation and the Open Source Initiative signifies the introduction of a justification based on market logic, which to a large extent is in tandem with a civic logic of justification promoting the role of openness to cope with market failure in the software development sector, and thus improve the market.

By comparing the historical development of justification logics related to FOSS it is possible to identify how value logics structure the integration of FOSS in the studied companies. The results show that different justificatory logics become resources for the developers. We found three approaches for how such resources were used: *inheriting*, *abandoning* and *transforming* value logics. Building a justificatory arrangement for FOSS can thus be viewed as on-going justification work in organizations (Jagd, 2011) were different logics are negotiated to form an arrangement that can become successful in that particular context.

Boltanski and Thévenot (2006) argue that arrangements can be weak or strong. Weak arrangements consist of contradicting value logics. When logics support each other a stronger alliance occur. In the data presented value logics shift over time in a way that for some areas create strong alliances. As shown the interplay between inspirational and industrial logic, and between the civic and market logic, turn them into strong justificatory logics in contemporary firms. The inspirational logic identified enabled the studied developers to realize their ability for artistic and visionary problem solving. In a similar way, the market logic innovates new ways of approaching the market to compete with free software and open standards. Furthermore, the civic logic has paved the way for a more autonomous customer and user perspective, supporting a market logic that emphasizes common good rather than proprietary strategies.

The practical implication of the analysis is that an organizational design that successfully can incorporate FOSS must be able to understand and manage orders of worth in a way that builds strong arrangements where justificatory logics can come to play and interplay in various ways. Previous FOSS history works as a resource for such endeavors. In the contemporary business driven arrangement where market and industrial logics are dominating, the historical core of FOSS still seem to be important to the justification work done in firms. The study thus depicts an interesting interplay between these logics, forming a new ground for business in the intersection of movements and corporations.

7 Conclusion

In this paper we have shown how the order of worth framework can be applied to understand more of how justificatory arrangements structure FOSS implementation in software organizations. By using the order of worth framework the study shows that several justification logics overlap and become resources when valuing FOSS in professional software development. This understanding can inform corporations that want to adopt FOSS to gain strategic advantages beyond mere efficiency. We also believe that the framework could be applied to analyze open and distributed development efforts in a wider sense, but this needs to be further researched.

References

Boltanski, L. and E. Chiapello (2005). The New Spirit of Capitalism. Verso, New York.

Boltanski, L. and L. Thévenot (2006). On Justification: Economies of Worth. Princeton University Press, Princeton.

Bonaccorsi, A. and Rossi, C. (2003). Why Open Source Software can Succeed. Research Policy 32 (7), 1243-1258.

Campbell-Kelly, M. (2003). From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry. MIT Press, Cambridge, MA.

Demil, B. and Lecocq, X. (2006). Neither market, nor hierarchy or network: The emergence of bazaar governance. Organization Studies 27 (10), 1447-1466.

Dijkstra, E.W. (1971). EWD316: A Short Introduction to the Art of Programming. T. H. Eindhoven, The Netherlands, Aug. 1971.

Dinkelacker, J., Garg, P.K., Miller, R. and Nelson, D. (2002). Progressive Open Source, In Proceedings of ICSE'02, May 19-25, 2002, Orlando.

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

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

Foucault, M. (1969). The Archaeology of Knowledge. Routledge Classics, London.

Gay, J. (Ed.) (2002). Free Software, Free Society: Selected Essays of Richard M. Stallman. GNU Press, Massachusetts.

GNU's Bulletin (1986). Available at http://www.gnu.org/bulletins/bull1. Accessed March 27, 2012.

Introna, L, and L. Whittaker (2004). Truth, Journals and Politics: The Case of MIS Quarterly. Kaplan, B. et al. (Eds.) Information Systems Research. Relevant Theory and Informed Practice. Springer Science.

Jagd, S. (2007). Economies of convention and new economic sociology: Mutual inspiration and dialogue. Current Sociology, 55 (1), 75-91.

Jargon File 4.3.1. Accessed May 3 2011. This version is now archived. It can be viewed at: http://www.elsewhere.org/jargon.

Kilduff, M. (1993). Deconstructing organizations. The Academy of Management Review, 18 (1), 13-31.

Knuth, D. (1974). Computer programming as an art. Communications of the ACM, 17 (12), 667-673.

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

Lessig, L. (2002). Open Source Baselines: Compared to What? Hahn, R. W. (Ed.) Government Policy toward Open Source Software. AEI-Brookings Joint Center for Regulatory Studies, Washington.

Levy, S. (1984). Hackers: Heroes of the Computer Revolution. Anchor Press/Doubleday, New York.

Machintosh, N.B., and Baker, C.R. (2002). A literary theory perspective on accounting. Towards heteroglossic accounting reports. Accounting, Auditing & Accountability Journal, 15 (2), 184-222.

Morgan, L. and Finnegan, P. (2010). Open innovation in secondary software firms: an exploration of managers' perceptions of open source software. Data Base, 41.

Murray, F. and O'Mahony, S. (2007). Exploring the foundations of cumulative innovation: Implications for Organization Science. Organization Science, 18 (6), 1006-1021.

Osterloh, M. and Rota, S. (2007). Open source software development – Just another case of collective invention? Research Policy, 36 (2), 157-171.

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

Raymond, E.S. (1999b). Shut up and show them the code. Linux Today, June 28.

Rolandsson, B., Bergquist, M. and Ljungberg, J. (2011). Open source in the firm: Opening up professional practices of software development. Research Policy, 40 (4), 576-587.

Spagnoletti, P. and Federici, T. (2011). Exploring the interplay between FLOSS adoption and organizational innovation. Communications of the Association for Information Systems, 29 (15), 279-298.

Stallman, R.M. (1986/1982). The GNU Manifesto. Available online: http://www.gnu.org

Stallman, R.M. (2002). On Hacking. http://stallman.org/articles/on-hacking.html.

Stark, D. (2009). The Sense of Dissonance. Accounts of Worth in Economic Life. Princeton University Press, Princeton.

Thévenot, L. (2001). Organized complexity: Conventions of coordination and the composition of economic arrangements. European Journal of Social Theory, 4 (4), 405-425.

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), 209-223.

Välimäki, M. (2003). Dual licensing in Open Source Software industry. Systemes d´Information et Management, 8 (1), 63-75.

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

Williams, S. (2002). Free as in freedom. Richard Stallman's crusade for free software. O'Reilly, Sebastopol.

Ågerfalk, P.J. and Fitzgerald, B. (2008). Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy. MIS Quarterly 32 (2), 385-409.