Chapter 1

Introduction

Background

Ever since Netscape Communications announced their plans to make their Web browser an open source product by establishing the Mozilla project in January 1998, an increasing amount of software companies have been pledging to open source software development (Demil & Lecocq, 2006). This is an interesting observation, as it is initially hard to understand how open source software development can be profitable on the software market. The unique features of open source software involve granting the user permission to study, change, improve and redistribute the software in a modified or unmodified form, and is often termed "free software". Open source software can therefore be understood as a public good accessible to everyone (Hippel & Krogh, 2003; O'Mahony, 2003). Nevertheless, several scholars have shown various means of appropriating returns with open source software (Bonaccorsi, Lorenzi, Merito, & Rossi, 2007; Dahlander & Magnusson, 2005; Demil & Lecocq, 2006; Lin, 2006; O'Mahony, 2007a), and an increasing amount of business models surrounding open source software are emerging. The escalating commercial interest in open source software development is most likely a result of the success of the open source model, as detailed by Steve Weber (2004).

Twenty years ago the position of open source software in relation to commercial software development was most uncertain, but today open source software is a major part of the mainstream information technology economy. Nearly 40 percent of large American companies use Linux in some form (ibid). The Apache software holds 65% of the web server market. The email transfer and management software Sendmail powers about 4 out of 5 mail servers in the

world. Increasingly, open source software products such as these are running major enterprise applications for large and small corporations alike (ibid, p.6). Furthermore, it does not look like the open source movement is going to die out any time soon; SourceForge.net is the world's largest open source software development web site, hosting more than 100,000 projects and over 1 million registered users[2]. Although open source software on many occasions has proven to be technically superior to proprietary alternatives, the answer to its success is probably more sociological than technical. Open source software is often developed in a public, collaborative manner, as opposed to software that is developed in a proprietary fashion where the program source code is patented and remains within the developer organization. The latter is the most common form of commercial software development today. The open source software development process is based on voluntary participation and voluntary selection of tasks (Weber, 2004), and attracts talented programmers who's participation is primarily based on intrinsic motivation (Hippel & Krogh, 2003). Furthermore, the Internet enables a vast amount of programmers to participate simultaneously on the same project. This makes it possible to mobilize a large amount of "manpower", which also can raise the quality of the product. Eric Raymond, an influential programmer and writer on the open source movement, explains this logic with what he calls Linus' Law: "Given enough eyeballs, all bugs are shallow" (Raymond, 2001). The organizational model of open source can easily seem somewhat chaotic with the lack of formal authority and price mechanisms to provide governance, but it is no anarchy. There are several alternative mechanisms in place to deal with coordination and complexity in projects, for example through norms, licensing schemes, sanctioning, individual incentives and leadership practices that prevent the communities and projects from falling apart (see Weber 2004 for a extensive description of these macro-foundations).

The development in the software market the last decade has shown that "being open source" may bring success, and we are now witnessing a range of different shapes and forms of open source initiatives blending in various ways with proprietary software and for-profit companies. Many large established software companies such as HP, SUN, IBM, Oracle and Novell (to name a few) have positioned themselves over the last years towards embracing open source technology and methodology. On the one hand, previously proprietary companies such as these are supporting existing autonomous open source projects, by contributing manpower and software code improvements. This has been a typical form of collaboration between the commercial and the open source software domain for some time now. In addition, some incumbent software firms have recently taken a step further than simply participating in open source communities and using open source software. Several proprietary software companies have chosen to open up their own software projects in an attempt to *create surrounding open source communities* to support their

[2] http://sourceforge.net/docs/about

own development. This is what O'Mahony and West describe as *firm-sponsored open source communities* (O'Mahony, 2007a; J. West & O'Mahony, 2005), which is the main theme for this thesis.

Motivation

While autonomous open source software communities have received a great deal of empirical and scholarly attention within the last decade, the research on firm-sponsored communities is fairly limited. The exceptions are found in (O'Mahony, 2007a; Joel West & Gallagher, 2004; J. West & O'Mahony, 2005). This thesis therefore aims at exploring this topic in greater detail, and provide a better comprehension of this organizational model of collaboration. In Siobhan O'Mahony's research, she finds that a major difference between firm-sponsored and autonomous communities is that the former need to handle a tension between openness and control in their product development (2007a). Given that developers earn the trust and prove their competence, the autonomous communities can basically offer the same opportunities for everyone to gain rights to commit code and part-take in strategic decisions. For firms, it is more problematic to offer this kind of openness to external community members, due to their need for control of the product upon which their business model relies. *This relationship is equally present in the case of this study*. I am therefore motivated to find out more about how firm-sponsored open source communities are *managed* in practice.

The case

The case for this study is the American software company Novell (est. 1983) and the openSUSE project (initiated in 2005). Novell has a history as one of the largest proprietary software companies within the domain of operating systems. Towards the end of the 90's their potent operating system 'NetWare' gradually fell into obsolescence, and the company was pressured to explore alternative technologies to replace it. At the end of 2003 Novell acquired the open source company Ximian and the German software company SUSE Linux. The latter had been developing a commercial Linux distribution since 1994. The SUSE Linux operating system became the replacement for NetWare, and has made Novell a major company devoted to open source software development. Although Novell still develops a large amount of proprietary software, Linux is now the backbone of their technology portfolio. The new strategy threw Novell into an extensive internal and external transition of the company. For one of my informants, just switching the desktop on the workplace computers was a major change in the company that he remembers well (interview #1, 18.05.07). In half a year the entire organization switched from licensed Microsoft products to OpenOffice and Linux on all workstations within the company – affecting salesmen to engineers. The business organization is illustrated in figure 1 below. The Linux development in the company is done within the research and development

team (R&D) in the Open Platform Solutions unit. This is where most of the research for this thesis is done. Most engineers in the R&D team are located in Nuremberg, Germany, in the offices of the former SUSE Linux company, although the team also has employees in Cambridge/MA, Prague, Provo, Bangalore, Beijing and in home offices. In this thesis I will refer to this sub-organization as *Novells' Linux R&D*. The other units in the figure represent the other (mostly proprietary) software sectors in Novell.

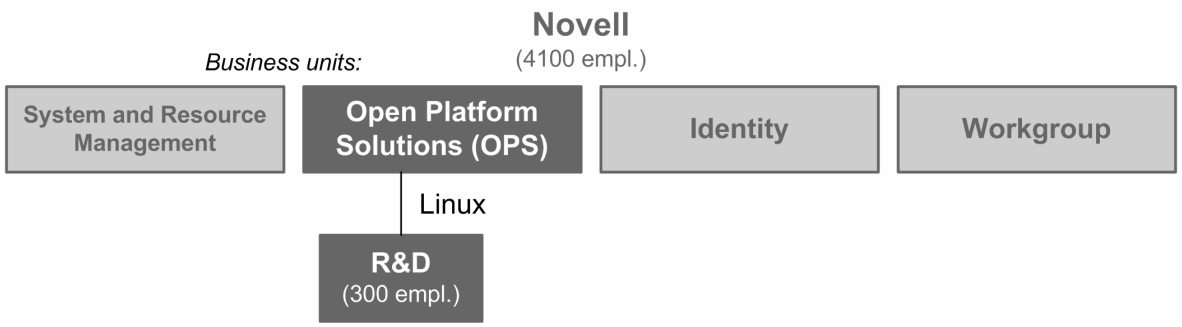

Figure 1. Novell business units and location of Linux development

In august 2005 Novell initiated a project to open up their product development process. Although SUSE Linux had always been an open source product, the development of this software was previously a closed process within the company. The openSUSE project therefore sought to bring development of the Linux product to the outside of the company's boundaries. Today, Novell's Linux R&D work in collaboration with external developers and contributors in the **openSUSE community**, which is Novell's firm-sponsored open source community in O'Mahony's terms. Since 2005, Novell now promote two Linux distributions[3]: the **openSUSE distribution** which is developed in collaboration with the external community, and the commercial **SUSE Linux Enterprise** products which are *based on* the openSUSE software. Novell sells licenses and offers support for the latter, and base many of their other licensed proprietary products upon it.

The community-developed software and the enterprise Linux products are tied closely together. For this reason Novell are struggling with releasing any control and devolving any decision-making rights to external community members. This situation may present a paradox since the motivation among individual community members to participate often relies on their ability to actually influence the software development. Therefore, most informants I spoke with at Novell and in the external community claim that the current status is not a desirable long-term situation. However, the relationship between Novell and the openSUSE community is constantly evolving,

[3] See the glossary at the start of the thesis for an overview of some common some open source terminology that is used in this thesis.

and even at the time of writing several developments have progressed since the data for this thesis were gathered. As O'Mahony notes, we do not know how this tension between openness and control will transform over time (2007a), and the evolution in Novell is therefore a very interesting case to study.

Research questions

By establishing the openSUSE community, Novell have on the one-hand extended their own organization. At the same time they are keeping the external developers at arms-length by not granting them any direct authority to commit code to the system. This situation poses some interesting questions about what organizational phenomenon we are witnessing: Can we understand Novell and the openSUSE community as one and the same organization, or as two separate ones? Is it even possible to regard the openSUSE community as an independent entity? The first question I will pursue in this thesis concerns investigating the nature of this organizational model.

1. What is the (theoretical) distinction between the sponsor firm and the sponsored community?

Herein, we need to find theories that are able to describe and pin-point the characteristics of Novell in relation to the openSUSE community. For this task I have selected Niklas Luhmann's theory of *autopoietic social systems*. This interesting theory has some peculiar aspects that make it particularly suited for our case. First, a counter-intuitive aspect of autopoietic social systems is that they do not consist of humans, but only their *communications*. Furthermore, the meaning of autopoietic is that the social system is continuously and *recursively recreated* by communicative events based on previous communications. Lastly, the sub-class of autopoietic systems that Luhmann terms *organizational* systems are distinguished from other social systems in that they consist of a special form of communication: *decisions*. By contrast Luhmann defines an *interactive* system as a more basic social network of communications, such as a board meeting or perhaps an academic discourse progressing over several years. In this thesis I will argue that these characteristics are able to distinguish Novell as an organizational system from the interactive openSUSE system, and that both systems are simultaneously evolving and altering their own boundaries towards each other.

With Luhmann's theory, we will be able to see how Novell and the openSUSE community are *separated* apart from each other. The relationship between these two systems are affected by the tension between openness and control, described above. The next step we need to take is to go deeper into investigating the exact nature of this relationship, and in particular how they are *held together* in collaboration.

2. What is the nature of the relationship between the organizational and the external interactive system, and how are the two systems bound together?

Given the differences in the commercial tradition of software development and the open source software model, multiple inherent conflicts could be present in Novell's open source adventure. If we also take into account the mentioned tension between openness and control in Novell's development model, many obstacles and difficulties complicate this collaborative attempt. I am therefore interested in seeking out the mechanisms, forces and arrangements that hold things together. In the thesis, I argue there are three elements that ensure a tight coupling between the two systems. By drawing upon the work of Susan Leigh Star (1989), I first discuss how several development tools and models serve as important *boundary objects* between the two systems, enabling joint development on a common product. Secondly, *shared communication channels* are vital in creating transparency and providing access through the boundaries of the systems. Thirdly, the *marginal people* whom have roles in both systems are crucial for balancing the needs of both of them. Based on this knowledge, I will turn to discuss some future scenarios for Novell and the openSUSE project towards the end of the discussion in this thesis.

The discussion of the case of Novell is as much an attempt to explore and test theories as it is an effort to describe and explain this empirical reality. Similar to much off-the-shelf software, social theories rarely solve the problem without adjusting, modifying and expanding them. In this study, this has been the third of my driving questions.

3. How may the case of Novell strengthen our understanding of the theory of autopoietic systems and the theory of boundary objects, separately and in combination?

In this thesis I argue that Luhmann's theory might fall short of explaining how social systems are linked together, and that the theory of boundary objects may be particularly suited at demonstrating system connectivity. Similarly, the case of Novell shows that Star's boundary objects are not only able to connect individuals and social groups, but also social *systems.*

A primary contribution to the theory of boundary objects is a distinction between what I describe as *supportive-objects* and *target-objects*, where the former serve as *means* and the latter as *ends* of collaboration (the target of the cooperation). Target-objects, in particular, have some very important characteristics that are crucial in explaining how heterogeneous actors are aligned in the development of a common product, and addresses an aspect I find lacking in Star's theory. The missing element to her theory is something that may explain the overall unity that binds actors together in collaboration, at a higher level than the pure mechanics of negotiating conflicts and aligning interests. By drawing upon Emile Durkheim's classic sociology, I try to show that target-objects such as the openSUSE object can serve as symbols and are charged with emotional

energy that unite the actors even when they are apart. I argue that despite their differences, the collaborative groups in fact inhabit they same social world in a form of *organic solidarity*. This is important for explaining how Star's boundary objects can hold common meaning and bind the separated groups together in the first place. Moreover, target-objects also contain some of the properties of epistemic objects (Knorr Cetina, 1997, 2001; Rheinberger, 1997), in that they hold an inherent motivating quality since they represent the end-goal and result of the collaboration. The collaborators are all stuck to the object because they *want* and desire to participate in its evolution. I argue that the motivating qualities of the object itself is largely neglected in the open source literature's explanation of individuals motivation to participate on open source projects. This understanding should therefore be appended to this body of research.

Structure of the thesis

The structure of the thesis follows a somewhat different order than the one shown above. It is always a challenging pedagogical task to present a story where theory and empirical reality are entangled together. In this thesis they will nevertheless be discussed separately before they are merged towards the end. First, I will present the theoretical foundation for this thesis in chapter 2, where we will look at the research on open source firms and communities, Star's theory on boundary objects and Niklas Luhmann's autopoietic social systems. Thereafter I will discuss the methodology I have used to gather data for this thesis in chapter 3. In chapter 4, we will wander into the world of Novell and openSUSE where I will present my empirical findings, before we start the analysis and discussion of this reality in combination with our theories in chapter 5. In the final chapter I will outline some of the main issues in this thesis and discuss where further research may go.

Let's start!