Canonical vs. Community

- an Outside Study

Master thesis

by

Lasse Stissing Jensen

Jane Billestrup

Dan Kærvang

July 31th 2009

Aalborg University
Department of Computer Science
Information Systems
Selma Lagerlöfs Vej 300
DK-9220 Aalborg ø
http://www.cs.aau.dk

Title:
Canonical vs. Community
- an Outside Study

Theme:
Open Source Development

Project term:
February 1st to July 31st 2009

Project group:
d613a

Members:
Dan Kærvang
Jane Billestrup
Lasse Stissing Jensen

Supervisor:
Peter Axel Nielsen

Circulation: 5

Numbers of pages: 73

Appendix: 1 + CD

Finish date 31/07-2009

Synopsis:

This report is about the collaboration between a company and an open source community. We have examined what control the company has over the community, and what the relation is between the community and a company with commercial interests. The open source distribution examined in this thesis, is Ubuntu and the company is Canonical. Our research method is a case study, and our method for collecting data is conducting qualitative research interviews with both Canonical employees and Ubuntu community members. Additionally we have conducted an unstructured observation study at the Ubuntu Developer Summit in Barcelona May 25th to May 29th, 2009, to confirm our findings from the interviews, and gathering further empirical data. Our conslusion states among others that Canonical has a great deal of control with the Ubuntu releases, even though they might try to give the volunteer developers a chance to be heard. In the end both sides have one major goal in common: Making Ubuntu as great a distribution as possible.

Preface

This report is our master's thesis, which documents the result of our work during the spring semester 2009. The topic of this thesis is about Ubuntu and what control Canonical have over a commercial/volunteer open source project, and what the relation is between the volunteer participant and Canonical. This project marks the completion of a specialisation year in the Information Systems research unit at the Department of Computer Science at Aalborg University, Denmark. As a supplement for gathering our empirical data, we went to the Ubuntu Developer Summit in Barcelona May 25th to May 29th.

Acknowledgements: We would like to thank our supervisor, Professor Peter Axel Nielsen, for providing valuable input and references. Additionally, we would like to thank Mark Shuttleworth, the eight Canonical employees and eleven Ubuntu community members, who volunteered to being interview for this project, without them this project would not have been possible. A special thanks goes to the Canonical employees Soren Hansen, James Westby, Daniel Holbach and Thierry Carrez for all their help and for patiently answering every question we had during this process.

Lasse Stissing Jensen

Jane Billestrup

Dan Kærvang

Aalborg, July 31th, 2009

Summary

This report is the documentation of a master thesis in computer science. The motivation for this thesis is an interest in open source software, which is under going great changes in these years. Commercial interests in open source software are growing, and businesses are forming around open source software.

The focus of this study, is the examine the collaboration between a for-profit company and an appurtenant volunteer open source community.

Our research questions are as follows:

- What control does a company have over a commercial/volunteer open source software (OSS) project?
- What is the relation between volunteer participants and a company with commercial interests?

As a basis for better understanding the the issues of opensource software, literature studies were conducted of the history of open source, development in open source and business in open source, respectively. To answer our research question we conducted an explorative case study. The case was the Linux distribution Ubuntu. This is an interesting case, because it is one of the most popular Linux distribution among both users and contributors. Furthermore the commercial backing by the company Canonical and its involvement in Ubuntu to make it a viable business makes it a suitable case for this study. The case study was conducted by a total of 20 qualitative interviews with Canonical CEO Mark Shuttleworth, 8 Canonical employees and 11 volunteer contributers, respectively. Additionally we have conducted an unstructured observation study at the Ubuntu Developer Summit in Barcelona May 25th to May 29th, 2009. This was done to supplement the interview study, and gather further empirical data.

Our findings was devided into five pieces: "The Business Model of Canonical", "Ubuntu Culture", "Champion and Canonical owner", "The Collaboration in Ubuntu" and "Open Source", respectively. These findings are the discussed in relation to the background material within the history, development and busines of open source, respectively.

Our conslusion states among others that Canonical has a great deal of control with the Ubuntu releases, even though they might try to give the volunteer developers a chance to be heard. In the end both sides have one major goal in common: Making Ubuntu as great a distribution as possible.

Contents

Chapter 1

Introduction

1.1 Motivation and Background

Almost since the beginning of software development, open source software has been present in some form. In the early days of computing, sharing source code was the common practice, since the main focus was to make code that worked and not code that looked nice[FF02]. As software became more business oriented, commercial interests pulled towards proprietary licences. Keeping one's source code secret became an important part of most software businesses. In its more recent form, the open source movement has emerged as an alternative to the proprietary software licensing model. It has its roots in what is known as the *hacker culture*, where the right to modify software and share information freely are central issues. This hacker culture, and the software it has produced, provides much of the software infrastructure in use today, and open source software has been expanding onto desktop computers as well.

Open source software is still a minor part of the overall market for software, but has a strong position in certain applications, such as web servers and rendering farms [OSD]. In recent years there has been an increasing commercial interest in open source software. The notion that open source meant anti-commercial is no longer predominant. Not only have companies, governments and municipalities begun adopting open source software, but several companies are getting involved in the development of open source software, some even make open source the core of their business — Companies like Red Hat, Canonical or SugarCRM. The French government is a pioneer in using open source software. At this point open source has or will replace Microsoft in the near future in the French police departments, governments and schools [Stac].

A very interesting phenomenon is the emergence of open source projects, where a company spearheads a development effort, while a community of volunteer contributors supplement the development. This constellation unions

potentially very different interests. A company can not base its business on selling software licenses, but must find other ways to secure profits. Also volunteer contributors have to accept the company's commercial interest in their work. Management becomes a balance between keeping participants interested in the project, and allowing for some form of alignment.

How such a joining of open source and business takes place is of great interest because it potentially leads to a significant new direction for software development which will be a focus area of this study.

1.2 Focus of This Study

The subject of this study is development of open source software by cooperation between a company and a volunteer community of contributors. The collaboration established in a situation where the commercial interests of a company and the multitude of interests of a participating community is found very interesting. This study will examine how a large open source project is organised across a software company and a contributing community of volunteers. The central research questions are:

- What control does a company have over a commercial/volunteer open source software (OSS) project?
- What is the relation between volunteer participants and a company with commercial interests?

The work process, which forms the collaboration between company and community, is also examined in order to understand how these two main entities have adjusted to each other.

1.3 Report Outline

Chapter 2 presents an overview of the history of open source software. In chapter 3, open source development practices and open source licensing. In addition this chapter explains about trademarking. Chapter 4, presents the theory about open source business models. In chapter 5 our research approach is described. This includes a short presentation of Ubuntu. Chapter 6 and 7 presents and discusses our findings in an analysis. Chapter 8 and 9 conclude the thesis and present possibilities for future work, respectively.

Chapter 2

The History of Open Source Software

In this chapter we provide a historical overview of some of the most important events in the history of open source development. Our main source for this section is the book "Understanding open source Software Development" by Brian Fitzgerald and Joseph Feller[FF02]. Where no other source is given the source is this book.

2.1 The Beginning

In the 1940s the computer was primarily used for scientific problem solving. In the early 1950s computers began to spread beyond that of scientific problem solving, addressing the area of business data processing.

At this time good programs were efficient rather than well-documented, because of the limited memory capability. It was a major achievement to get a program to run at all. Any working software was shared widely.

The PACT (Project for the Advancement of Coding Techniques) established in 1953, was one of the earliest formalised examples of free sharing of software between the military and aviation industries, who were in fact competitors. The motivation was efficiency benefits, and the initiative for the collaboration was taken by the programmers in both industries who persuaded the management into this collaboration.

By 1960 the business data processing had overtaken the scientific one. At that point the business use of computers accelerated. In the US the number of computer installations increased more than twenty-fold between 1960 and 1970.

Another important event for ensuring the availability of free software, was in 1956 where AT&T was forbidden to enter non-telephony markets, such as computing. This meant that UNIX created at AT&T in 1969 could not be sold commercially. Instead UNIX was distributed to universities and

other research institutions for a nominal fee. UNIX is an operating system widely used in both servers and workstations.

2.2 The Berkeley Software Distribution

In 1977 the Berkeley Software Distribution (BSD) was established at the Berkeley University in California. BSD was based on UNIX. The BSD project was headed by Bill Joy, who later was a co-founder for SUN Microsystems. The BSD group modified and improved the UNIX system, then redistributed it to others who then contributed their own enhancements, making BSD even more powerful.

In 1984 AT&T sought to commercialise UNIX. This gave a court battle between BSD and AT&T over copyright violations, resulting in claims and counter-claims from both sides. It was resolved in the early 1990s in an out-of-court settlement. The long uncertainty about the outcome resulted in many volunteers choosing to contribute to Linux instead (which was protected by the GNU - General Public License) [Stad]. BSD later forked into FreeBSD, OpenBSD and NetBSD. According to Feller and Fitzgerald, the BSD initiative marked the beginning of a more ideological underpinning in the free software history[FF02].

2.3 The Free Software Foundation

In 1985 the Free Software Foundation (FSF) was founded by Richard Stallman. This is one of the most significant milestones in the history of open source.

The idealism around the FSF was, and continues to be, very strong. This idealism puts forward the view that closed source software is considered immoral, and they believe all software should be open source. Richard Stallman devoted his attention to creating a suite of free software products, called the GNU family. In the GNU manifesto [Stab] (1985) Stallman coined the term "free software", thus formalising a process that had been pretty ad hoc in the past.

The ambiguity of the word "free", meaning both "unfettered" and "gratis", later led to the coining of the term "open source".

The strong ideological nature of the FSF and their licenses, is typified by the common copyright phrase, *copyright - all rights reserved* to be *copyleft - all rights reversed*. Under the copyleft concept, everyone has permission to run, copy, modify or redistribute the software. The only clause is that it is not allowed to add any restrictions to the modified code either. [Staa]

2.4 Linux

In 1991, at age 21 Linus Torvalds began the Linux operating system. He modelled his system on Minix which is a Unix clone. Torvals' goal with Linux was to create an operating system for the IBM PC 386 series, and he openly sought help for this project. Torvalds' choice of name for his system was Freax, from the words "free", "freak" and an "x" for a consistency with the naming of operating systems. But he was persuaded into changing the name to Linux by Ari Lemke also from the University of Helsinki, who also offered a sub-directory on the university's' FTP site for the system. The name Linux was derived from the words "Linus" and UNIX.

Torvalds succeeded in attracting other developers for support. An estimate suggests that more than a thousand developers worldwide have collaborated on the Linux kernel development.

2.5 Apache

In February 1995 came the next milestone in open source. It was the development of the Apache HTTP Server, which is widely used all over the world. The Apache server was based on a series of patches developed for a web server developed by Rob McCool at the National Centre for Supercomputing Applications (NCSA). This server was very popular and many individual webmasters developed extensions or patches. The development was ceased when a key individual left (to form Netscape). At this point the volunteers came together to coordinate the distribution of these patches, and Brian Behlendorf provided hosting for the project.

2.6 Mozilla

The Mozilla project is said to be one of the most important in the history of open source Software. Beside the products it has produced it also had a great impact promoting corporate and media awareness of the concept. In January 1998, Netscape announced that the source code for their new browser would be made available, and the name of the product would be Mozilla. At this point Netscape had a market share of 13% and it was decreasing in favour of Microsoft's Internet Explorer. Netscape decided to make the source code available with the product. They created a special pair of licenses, The Mozilla Public License (MPL) and the Netscape Public License (NPL) for the project. The MPL is OSI approved (Open Source Initiative)[OSIa] but the NPL is not, since it contains a clause allowing Netscape (and only Netscape) to re-license third-party Mozilla development to create a proprietary product. The strategy worked, and by the end of 1998 the company was regaining market share with Mozilla, which is one of

the most popular Web browsers today. Netscape on the other hand died in 2008. The last updates was released in march 2008.

2.7 Open Source Software

Eric S. Raymond, who participated in the initiation of the Mozilla strategy, discussed the need for a long-term strategy in relation to bringing Free Software to a wider audience. In 1998, "open source" was coined as an alternative term to "Free Software". One of the people who took this decision was Bruce Perens, who had produced the *Debian Free Software Guidelines* [gui] for the Debian project which is a Linux distribution including mainly GPL-licensed software. The founding of the Open Source Initiative (OSI) [OSIa] was an extension of this activity. OSI is the organisation who made the rules called The Open Source Definition [def], which is based on the *Debian Free Software guidelines*. In 1998, Microsoft unintendedly became a main cause in the promotion of open source. This happened when internal Microsoft memos named the *Halloween Documents* was leaked to the OSS community. The document discusses the concerns of Microsoft and the threat posed by Open source software in general, and Linux in particular. This memo caused a very large increase of media interest in the open source topic. On this topic, Eric S. Raymond is quoted saying *"Wall Street finally came to us"* [FF02]

2.8 Summary

In this section we have provided a lot of information and a lot of years. To sum up the most important event we have made a timeline, see Figure 2.1, showing the most important events and years.

In the following section we will describe the development practices of open source development, including a description af the open source License GNU GPL (GNU - General Public Licenses), and trademarks.

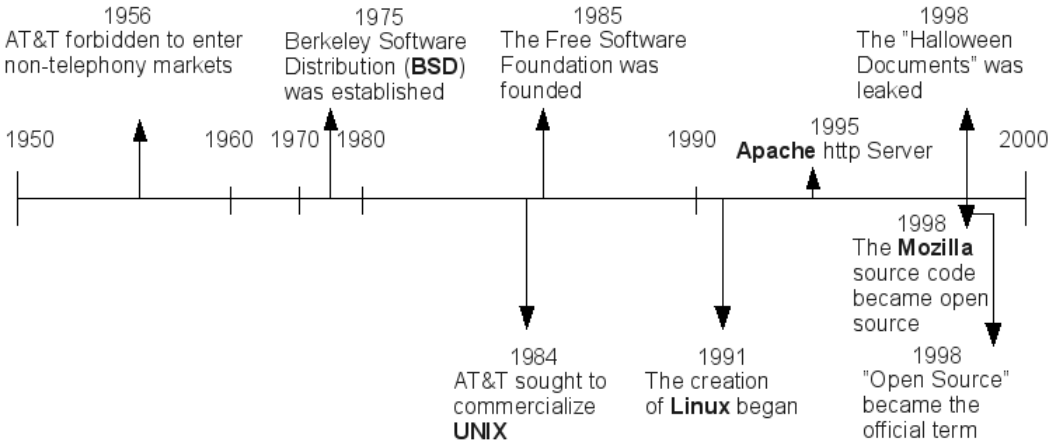

Figure 2.1: Timeline

Chapter 3

Open Source Development

In this section we describe some important open source development practices. We will also give an overview of open source licenses and a description of a trademark and what it means regarding open source software and its use.

3.1 Development Practices

Open source software (OSS) is defined by the licence under which it is distributed. The open source initiative (OSI) have specified an open source license. These are known as the open source definition [def]. Beyond its license, OSS is typically also characterised by a certain development practise.

Most open source projects start with a single person or a small group, who begins developing software. For a project to grow larger, it must attract more developers. A unique property of most open source software is this possibility for any interested developer to participate. Even when a project has attracted many participating developers, it is often a smaller group, who does most of the work. This is seen in case studies of the Apache and Mozilla projects [MFHA02], as well as in surveys among developers [Maa04]. This leads to the notion of a "core group" responsible for the majority of the time contributed to a project.

Open source development is also characterised by participants working distributed, and without many of the coordination means of traditional software development. Elaborate designs and time plans are often not used in open source development. Development tends to be incremental. Discussions about design are carried out through e-mail, online fora, IRC or Instant Messenger. In this seemingly chaotic setup, the tools supporting such distributed work is considered very important. Holck and Jørgensen suggest, that the use of version control tools, and the process surrounding them, offsets the need for traditional means of coordination [HJ03].

Most large and successful open source projects have provided what is

characterised as infrastructure software [FF00], that is: operating systems, browsers, web servers and data base management systems. In other words software that provides a foundation for other applications. Such software is general and generic, and that makes it easier for developers to gain a shared understanding of how to build it.

A very important part of open source development is which open source license the code is released under. In the following section we will describe the GNU General Public License, which is a very widely used open source license, and the concept of trademarks in the context of OSS.

3.2 Open Source Licenses

There are 66[OSIb] open source licenses approved by the Open Source Initiative. To get a license approved it has to go through a license review process [OSIc] to make sure that all software and licenses labeled Open Source, conforms to existing community norms and expectations.

GNU General Public License
The most well known open source license is the GNU General Public License, also known as the GPL. Next a quote from the GPL to describe its purpose:

> "When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.[Stad]"

This means that with software under this license, you can use it, change it, sell it or what ever you wish, but you cannot make it closed source. The source code must always be available giving others the same rights to your software as you have to their software and your derived software must also be under the GPL license.

There are two derived versions of the GPL license, the LGPL[LGP] (Lesser General Public license) and the AGPL[AGP] (Affero General Public License). The LGPL is a modified, more permissive version of the GPL. It was originally intended to be used for some software libraries. Since the LGPL is more permissive than the GPL it is allowed to redistribute software under the LGPL, under the GPL, but the other way around is not allowed. The greatest difference between publishing under the GPL and the LGPL is that, if you combine something under the LGPL with something

else in a larger project not under the LGPL, you do not have to license the whole software package under the LGPL. This is a contrast to using software under the GPL, since this license is more strict than the LGPL license. The AGPL license is a similar license, but focused on networking server software. It is as strict as the GPL, the difference is that the AGPL also covers the use of software over a computer network. This is done by requiring that the complete source code is made available to any network user of the AGPL-licensed work. For instance a web-application.

3.3 Trademark

To give an understanding about what is allowed to do with open source software, it is also important to know what a trademark means. In this section we will give a short and very simplified description of a trademark. A trademark can typically be a name, word, phrase, logo, symbol, design, image, or a combination of these elements. Having something under a trademark means that no one else can use that name, word etc. without permission[tra]. There are three types of trademarks: unregistered trademark, unregistered servicemark and registered trademark. We will only focus on the registered trademark, because this is the only kind refered to in this thesis. When looking at trademarks from an open source perspective, this means that even though other people or companies might have the right to use, re-distribute or sell others open source software, they do not have the right to use the trademark of this product, without permission.

3.4 Summary

In this section we have given a description of the open source development practices. Including how an open source community is formed and how they communicate. We have given a description of the open source license GPL and its derived versions. Also we have tried to give an understanding of which rights follows a trademark, and which are free with open source software. This section was mainly used to give the reader a broader understanding of some of the phrases used mostly in our analysis. We have assessed that these terms are essential to the readers understanding of our thesis. In the next chapter we will describe how businesses can use open source, and how they can make money or save money on open source software.

Chapter 4

Open Source Business Models

When talking about open source "business models" we are essentially talking about the

> "why, how and what for a business to generate revenues and achieve profit objectives" [Ber]

which means that we are talking about how a company expects to make money. In this section we give descriptions of the business models used in open source companies. The difference between open source companies and proprietary software companies is that in an open source company the code it self has no value, since everything is free and in the open. Therefore those companies have to find other ways to make money.

4.1 The Business of Open Source

An interesting development in OSSD is the increasing involvement form corporate entities. This is seen in empirical research of projects on the Sourceforge[1] portal by Bonaccorsi et al., where 32.33% of the examined projects had some form of firm participation[BLMR07]. Also, surveys of open source communities have shown a significant minority of developers being paid to contribute[LT03],[HO01]. Fitzgerald describes a development in open source, towards something more commercially oriented. That is, the development cycle of OSS is increasingly influenced by strategic consideration of companies pursuing a broadening field of open source business strategies. As open source spreads to more horizontal product domains,[2] input from companies with such domain knowledge is of growing importance[Fit06].

[1] An online portal hosting open source software projects.
[2] Domains characterised by highly specialised knowledge.

4.1.1 Reasons for Closing Source

Raymond describes the difference between the "use value" and the "sale value" of a software product as follows:

> The "use value" of a program is its economic value as a tool — a productivity multiplier. "The sale value" of a program is its value as a saleable commodity. [Rai01, 2]

When reading this citation it gets clear that according to Raymond, the software that is made for internal use has no sale value, meaning that this type of software is pure expenses. He also claims that more than 75% of what programmers get paid to do is making software with "use value" and no "sale value"[Rai01]. If this statement is correct, it means that about 75% of all software could be made open source without companies losing money, since it has no "sale value", and also the company will gain on opening the software since others can maintain, debug and extend the software without cost to the original company. An argument often met against open source is the fear of revealing something confidential. Raymond claims that such arguments are not an argument against open source but against bad design, which is a concern present in proprietary software as well. Another fear against open source is the fear of crackers[3] and intruders. He does not see open source as a problem regarding crackers. He sees the use of bad algorithms as the problem, and describes the solution as the following:

> Security is an aspect of reliability; only algorithms and implementations that have been thoroughly peer-reviewed can possible be trusted as secure[Rai01, 9].

4.1.2 Open Source Business Models

Raymond describes nine open source business models. These business models are all related to software with a "use value" and without a "sale value", because you do not lose anything by making "use value" software into open source software. The nine business models will be explained in the following sections.

4.1.3 Use-Value Funding models

The first two models Raymond define as Use-value Funding models are cost sharing and risk spreading .

[3]The word "cracker" is used to not confuse the use of the word "hacker" within the open source community. In the open source community a "hacker" is a skilled programmer and a "cracker" is a computer criminal.

Cost sharing

If a company only needs some software for internal use and therefore the product has no sale value, cost sharing is a good way to keep expenses down. Eric S. Raymond used the Apache web server as an example, saying that if someone need a web server they have three possibilities: Either buy a proprietary web server, roll their own or join the Apache group. If choosing to get a proprietary web server, it will cost money every time something need to be added or updated. Building your own web server will provide all the features you want and you can update it when you want, but you will have to pay for it in development time -and the company might have a problem if the developers leave. If choosing to join the Apache project, there will be many hours saved in developer time, and the company will gain some advantages as if they made their own web server, since it can be shaped as the company wishes, and to that adding a debugging effect from the community and the availability of skilled developers. In the end this cost sharing model can give the company a better product for a lower cost, as long as it is software without any "sale value".

Risk spreading

As with the cost sharing model Risk spreading require that the software has no "sale value". Eric S. Raymond used Cisco as an example. The example is that Cisco needed a print-spooling system. But since the developers would not be at Cisco forever the system might get out of sync. Therefore they argued to make the system open source. After doing this a community was growing around the system, taking care of the maintenance of the system.

4.1.4 Indirect Sale-Value Models

The last seven models Raymond define as Indirect sale-value models.

Loss leader or market positioner

In this model the open source product does not give revenue directly, instead it is used to created interest in other related proprietary software. An example is that Netscape made Mozilla open source to regain market share on the Internet browser market in the late 1990's.

Widget frosting

Open Source software from hardware manufactures, like drivers, configuration tools up to operating systems: The hardware manufacturer makes no money on the software that support the hardware. The open source community is there to enhance the product value of the hardware. Sun

Microsystems have several examples of products under this business model, like Netbeans.

Give away the recipe, open a restaurant

In this model the open source software is used to create a market position in selling services, around some open source product.

Accessorising

This is mostly a by-product for a company owning the trademark, like selling t-shirts, books and magazines.

Free the future, sell the present

The software is sold as closed source, but has en expiration date. After this date the product will be made open source. This model ensures costumers that a product will be maintained even if the company behind it dies, and it makes sure that the costumer can customise the product themselves after a short period of time.

Free the software, sell the brand

Raymond sees this business model as a speculative one. A company is speculating in others wanting to use their open source code to create another product. Like when SUN open sourced some of the code of Star Office, this was used to create OpenOffice. But SUN still had the rights for Star Office. The original company is making money by offering validation tests and statements.

Free the software, sell the content

This last business model Raymond also see as a speculative one, The software is still free and open source but sell subscriptions to the content. An example could be for instance an open source game where people could subscribe to getting more levels.

4.2 Organisational culture

When looking into the collaboration, between a volunteer community and a company working on the same project, there must be an organisational culture within such a project. In order to be able to take this into consideration, during our research we need to define organisational culture. For this purpose we adapt Edgar H. Scheins formal definition on organizational culture as

> A pattern of shared basic assumptions that the group learned as it solved its problems of external adoption and internal integration, that has worked well enough to be considered valid and, therefore, to be taught to new members as the correct way to perceive, think, and feel in relation to those problems. [Sch04]

4.3 Summary

This section gives a short description of when to, and when not to make software open source, and gives a description of the nine business models that Raymond defines for making money around open source software. We will use these concepts later in the analysis of the business model. In the following section we will describe what methods we are planning to use for analysis in a later section.

Chapter 5

Research Approach

This chapter will cover the selection of our research approach and present the chosen research approach in detail. In order to make the process transparent such that it is clear how this research has been conducted, the techniques used in the research will be described and discussed.

5.1 Selection of Research Strategy

According to Robert K. Yin one of the most important steps when conducting research is to select a proper research strategy, which fits the research question[Yin94]. The research questions can be categorised into different types of questions: "who," "what," "where," "how," and "why.". The different types of questions are reflected in the different types of data which are needed to provide a satisfactory answer. As depicted in table 5.1, this again affects which research approach is the most suitable choice.

Research questions in the "what" category can be of exploratory or quantifiable character, i.e. "what are the means of making the lottery fun to play?" is seeking an explanation, which can be used for shaping hypotheses and suggestions for more extensive investigations. The question "what are the effects of the new lottery rules?" is more of a "how much" question, which can be used to list some criteria and measure them. For the second kind of question, surveys or archival studies would be obvious choices, since surveys can easily be designed to provide data on different influences and their effect. Depending on the data available an archival might be able to show changes in sales, revenue, profit and prices paid. Experiments or case studies could be carried out as another solution, but the scale needed for answering the question makes archival analysis or surveys the most preferable strategy. Questions like "who" and "where" is somewhat similar to the second type of "what" questions, and the obvious strategy here will often rely on data which can be quantified. As opposed to questions favouring the quantitative research strategies, research questions like "how" and "why"

strategy	form of research question	requires control over behavioral events?	focuses on contemporary events?
experiment	how, why	yes	yes
survey	who, what, where, how many, how much	no	yes
archival analysis	who, what, where, how many, how much	no	yes/no
history	how, why	no	no
case study	how, why	no	yes

Table 5.1: Relevant Solutions for Different Research Strategies [Yin94]

are usually more interested in explaining phenomena and their context. I.e. if you ask the question "why did the factory workers go on strike?", a case study will be more appropriate than a survey, as it allows for an understanding of the situation in a higher level of abstraction, and because it can describe the context of workers culture, company leaders, union influence and political conditions. Yin states that case studies has an advantage when:

> "a "how" or "why" question is being asked about a contemporary set of events, over which the investigator has little or no control." [Yin94]

As described in our research question in section 1.2, our focus of interest with this study is to obtain a thorough understanding of what control a company has over a commercial/volunteer OSS project, and how the relationship works between the volunteer community and a company with commercial interests. Since both of our research questions are of the "what" type and treats contemporary events, of which we have no control, a case study is the most suitable research strategy for us. The selected approach corresponds to what Cunningham classifies as an intensive case study [J97], which is suitable for understanding organisations through descriptions, interpretations, experiences or events. Intensive case studies does not imply specific methods or techniques for gathering data, but relies on different evidences from the case. More specifically the research has the characteristics of an interpretative case, which in some terms will be a case example, demonstrating different events and approaches from within the case.

Having stated our research question and decided the research strategy, we

need to select one or more suitable cases. We need one or more cases which can give us valid and reliable data for our research. It would be interesting to have more OSS projects within the scope of the research. However, the combination of time and resources available for this project does not justify such an extensive study. If we are able to select a case which is a good representative of similar OSS projects we should be able to retrieve valid data with only a minor loss of reliability. If we were to span over two, three or more projects, we would only be able to scratch the surface and not gain the necessary in-depth understanding needed to explain the mechanisms in the collaboration to a satisfactory extent. As a result the scope is limited to a more narrow but in-depth investigation of an OSS project which must be able to provide a valid an reliable source of data.

5.2 The Chosen Case

Our choice fell on the Ubuntu project since it contains both a commercial main contributor and a voluntary community providing contributions. The Ubuntu distribution has since its first release rapidly grown to be among the most popular Linux desktop distributions[BTL+09], and the Ubuntu project, furthermore has been able to attract a substantial number of contributors in short time. This makes the Ubuntu project a well suited case for our study. This said, there are other well suited OSS projects, where openSUSE and Fedora probably are the OSS projects which resembles Ubuntu the most.

5.2.1 History of Ubuntu

The Ubuntu project was started on initiative by the South African entrepreneur Mark Shuttleworth, and took shape in April 2004. It made its first release in October 2004 [Ubu09]. From the beginning the project was financed by the company Canonical, which Mark Shuttleworth established especially for the Ubuntu project. Through Canonical he invested money to hire developers, and establish an infrastructure to support the project. An important part of the infrastructure provided for the projects is the Launchpad platform for the collaboration with developing Ubuntu. In 2005 Canonical founded the Ubuntu foundation, which aim is to fund support of Ubuntu in case Canonical would go bankrupt. This gives users a warranty that their Ubuntu installation will be supported in the time announced.

Since the first release of Ubuntu 4.10 in 2004 there has been another nine releases with an six month release cycle. The only release which did not follow the six month cycle was the release of Ubuntu 6.06 which were the first Long Term Support (LTS) release. The LTS release is a release intended for enterprises promises three years updates on the desktop release and five year on the server release. LTS versions of Ubuntu is released every

second year, making the next LTS release the Ubuntu 10.4 which should arrive in April 2010. Prior to the undertaking of every new release Canonical arrange the Ubuntu Developer Summit which is a week long conference with employees and the top contributors from the voluntary community. UDS has no entrance fee and everyone who is interested can attend.

In 2005 the Ubuntu community grew rapidly and started contributing in various different ways. Today the increasingly number of users and contributors has over time made it possible to release independent and self-supporting distributions i.e. Kubuntu and Xubuntu. These release uses Ubuntu as platform but using KDE or Xfce as the graphical desktop environment instead of the default Gnome environment provided with Ubuntu.

The word "Ubuntu" is African and means *"Humanity towards others"*. In the spirit of this name Ubuntu has a "code of conduct"[coc] with six ground rules that everyone in the Ubuntu community have to follow. Those rules are made to make sure that everyone feels welcome when joining the Ubuntu community and to make sure that the atmosphere surrounding Ubuntu is positive. The code of conduct has the following rules: "Be considerate", "be respectfull", "be collaborative", "when you disagree consult others", "when you are unsure, ask for help" and "step down considerately".

5.3 Collecting the Data

With the selection of our case in place it is now time to select the most suitable techniques for collecting the data. The data needed for our analysis must be able to provide us with a deeper understanding of the processes involved in the development of Ubuntu and the collaboration between Canonical and the community plus the business model of Canonical. Our understanding of these topics are at this point rather limited and uncertain. This makes quantitative surveys impractical to us. First due to the risk of asking the wrong questions as a result or our limited insight and secondly due to the little opportunity for the respondents to elaborate their answers. The choice fell on interviews which provides the possibility to adapt our questions in order to pursue interesting new topics that might appear during an interview, and for the respondents to elaborate important subjects. In between interviews it is possible to reflect over the conducted interviews and adjust questions and topics to suit the new understanding. In this way we are able to rapidly adapt, if there are indications that we have missed something vital, or if our initial understanding of the project seems to be far from what is experienced in the interviews. According to Järvinen this overlap between data collection and analysis can give good opportunities to adjust and speed up the process of collecting and analysing data in the start phase of the data collection[Jä04].

Using qualitative techniques for data collecting often yields large amounts

of data, for which treatment cannot be automated on a computer, i.e. transcribing qualitative interviews is unlikely to present the possibility of an automated sorting or reduction of the data set. However, transcribing the interviews can help achieving understanding of the subject, which is our main goal. To reach that understanding based on the data and be able to extract the essential information it is necessary to identify topics and categorise the data accordingly. In this section we will present our techniques for collecting data. The data source in this study is a group of active participants in the Ubuntu community. The participants are either employees at Canonical or voluntary contributers. The data will be collected through interviews and verified and extended through observation. This section will discuss different types of interviews and outline the chosen technique. After covering the subject of interviews we treat the observation technique used at Ubuntu Developer Summit in Barcelona, May 2009 (UDS).

5.3.1 Research Interview

There are two types of research interviews. The qualitative and the quantitative research interview [KB08]. Even though the difference between the two interview forms might seem quite significant, it mostly comes down to the quantitative interview employing numerical measurement, where as the qualitative does not [Bry08]. Going a bit deeper into the theory there are further differences. Those are outlined in table 5.2. The three contrasts in the table might be basic but are also fundamental.

	Quantitative	Qualitative
Principal orientation to the role of theory in relation to research	Deductive; testing of theory	Inductive; generation of theory
Epistemological orientation	Natural science model, in particular positivism	Interpretivism
Ontological orientation	Objectivism	Constructionism

Table 5.2: Fundamental differences between quantitative and qualitative research strategies [Bry08]

A quantitative research can be:

> "constructed as a research strategy that emphasises quantification in the collection and analysis of data and that:
>
> - entails a deductive approach to the relationship between theory and research, in which the accent is placed on the testing of theories;

- has incorporated the practices and norms of the natural scientific model and of positivism in particular; and
- embodies a view of social reality as an external, objective reality". [Bry08, page 22]

By contrast qualitative research can be:

> "constructed as a research strategy that usually emphasises words rather than quantification in the collection and analysis of data and that:
>
> - predominantly emphasises an inductive approach to the relationship between theory and research, in which the emphasis is placed on the generation of theories;
> - has rejected the practices and norms of the natural scientific model and of positivism in particular in preference for an emphasis on the ways in which individuals interpret their social world; and
> - embodies a view of social reality as a constantly shifting emergent property of individuals' creation". [Bry08, page 22]

Even though table 5.2 makes the differences between qualitative and quantitative research distinguishable, the reality is not that straight forward. There are examples of qualitative research has been employed to test rather than to generate theories [Bry08].

Since the purpose of our study is, to explore and understand the collaboration in the Ubuntu project, the use of qualitative research is chosen. The focus is not numbers and frequencies, but descriptions of relations and procedures. Because of that, the remaining will focus on the qualitative research interview.

5.3.2 Qualitative Research Interview

The qualitative research interview has been conducted since the mid 1970s, and has been a dominant strategy for conducting social research [Bry08].

The purpose of the qualitative research interview is "obtaining descriptions of the life world of the interview person, with a view to interpret the meaning of the described phenomenon". [KB08]

According to Kvale and Brinkmann[KB08] an interview study consist of seven stages: thematization, design, interview, transcription, analysis, verification and reporting. The following pages will give an exposition of these stages.

Thematization

Thematization is a clarification of concepts, a theoretical analysis of the chosen theme and a formulation of the research questions. Central to the structure of an interview are the words; *What*, *why* and *how*.

What is used to derive an understanding of the chosen subject.

Why should clarify the purpose of the study.

How will be used for locating the different types of interview and analysis techniques, and decide which one is suitable.

It is important to choose the right method, but before this is possible, the purpose has to be established. This means, that when designing the interviews you have to know the content and aim of the study, to choose the right method. Hence the questions *what* and *why* have to be answered before *how*.

Design

At the design stage you plan and improve the methodical procedures, aiming towards acquiring the desired knowledge. At this stage the number of test persons and their representativity is chosen. All later stages should be taken under consideration, since many of the problems and errors, that can occur later in the process, can be avoided with a more elaborated design phase. Also the quality of the interviews can be elevated, if the purpose and subject is considered from the beginning. Ethical guidelines also has to be complied at this stage. That is, the test persons have to give their consent, and confidence has to be secured. Also it has to be considered, if the interview situation can have any consequences for the person interviewed.

Interview

There are several types of professional interviews. Examples include: job interview, judicial interrogations, therapeutic interviews or research interviews. The interview we will focus on, is a semi-structured research interview, because this type of interview gives the interviewer the possibility, to pursue answers and stories from the interviews.

It is important to remember, that an interview is not a conversation between equal partners, since the interviewer is defining the situation, introducing conversation subjects, and controlling the questions by the use of further questions. The preparation is the most important factor to the quality of the interview. The central questions in an interview is, as mentioned earlier, “what”, “how” “and why”.

Before conducting the interview, the interview technique and method for analysis should be considered. An interview should always begin with a briefing where the situation and context is defined to the interviewee. That is, elaborating the aim of the interview, the recording device, and answering

any questions the interviewee may have. The interview should be followed by a debriefing, where the interviewee can get more information about the study. After every interview the interviewer should allot ten minutes for reflecting, and recall what has been learned in the last interview.

An interview guide should be created. It can either contain subjects, or precisely formulated questions for the interview. In a semi-structured interview there will be a list of subjects and proposals for questions, but it is not crucial that they are used during the interview. The interview should proceed more or less as a conversation, but with a specific aim, and a different structure. The questions should be short and simple. It is important that the interviewer can sense the meaning of the spoken, and that the interviewer has an interest is the subject.

There are different kind of questions to be used in a semi-structured interview, those will be described next.

Preliminary questions: The opening question can set the agenda for the rest of the interview. This question could be; "Try talking about", "Remember a situation where", "Can you describe a situation like that".

Follow up questions: These questions can be used to elaborate on the interviewee's answers. It can be a direct question from the interviewer, or a nod, a "hmm" or simply silence. Also repeating the most important words, can be a suggestion to the interviewee to keep talking or elaborating.

Probing questions: With a probing question the interviewer will pursue answers from the interviewee, and then probe the content. An example of a probing question could be "Can you tell me more about that?" or "Can you give another example?".

Specifying questions: This type of questions gives the interviewer the possibility to use a more operationalized question, such as "What was your thoughts about that?" or "Did you experience that yourself?".

Direct questions: Direct questions will primarily be used later in an interview. At a time when the interviewee has given his or her own descriptions and important aspects related to the theme. An example could be "When you talk about...".

Indirect questions: An indirect question cannot be interpreted without the use of further questions. An indirect question can both refer to the position of others, but also to make the interviewee give his own opinion. An example might be "How do you think others see...".

Structured questions: It is the interviewer's responsibility to maintain the structure of the interview, and decide when a subject should be dropped. This means, that the interviewer can interrupt the interviewee, if he or she is talking about a subject, that is not relevant to the research. The interviewer can interrupt by using a phrase as "And now I would like to talk about another subject".

Silence Not talking will give the interviewee time to think, and give important information.

Interpretative questions Different kinds of interpretative questions can be used. An answer can be rephrased: "You mean that...". A question can be clarifying: "Is it correct that you feel...?". It can also be a direct interpretation of what the interviewee is saying: "Is it correctly understood, that your anxiety originates from...?". Due the geographical distributed location of interviewees we have have chosen to conduct the interviews via VoIP [1]. Prior to the interviews we have prepared a interview guide (appendix A) to ensure that we will cover all the desired topics.

Transcription

In order to stick as close to the original interview during analysis it is important to transcribe the interviews, so the risk of unconscious filtering of the data is minimised. It is important to keep in mind, that transcriptions are not original data, but artificial constructions of oral to written communication. Also if two people are transcribing the same passage, it will never be the same, because of the difference in weighing of pauses, gestures etc. Also emotional aspects as laughter, sighing, giggling and so on can be perceived differently. This is also why it is important, that the people transcribing, have made some common rules, to make the transcription as homogeneous as possible.

Analysis

Analysing the data is almost as important as the collection of data itself, and with the large amounts of rather unstructured data that qualitative methods often produce. The goal here is understanding this is best achieved by finding order and coherences[Jä04]. If the study covers several cases this would typically involve write-ups for each case. Our study however only covers one case, but with several different topics. We will use a procedure where each of our initial topics are covered. The transcriptions will be traversed identifying statements regarding the various topics, and if other topics that we were not initially aware of emerge, all of the transcriptions we go through review once more looking for statements regarding that topic. The identified statements on the different topics will be combined into individual documents and condensed into our understanding hereof. This gives us an overview and understanding, which we can afterwards use to reach a more complex understanding of the topics in correlation and summarise this into our understanding of the case.

[1] Voice over IP/internettelefoni

Verification and Validation

Even though validation is described separately, it is actually an ongoing process through all stages. This gives a continuous quality control, instead of a detached inspection at the end of the process. In the following validation questions for an interview study is described.

- **Thematization** - The validation depends on the study's theoretical preconditions.
- **Design** - If the design and applied methods are appropriate for the purpose, the validity stands.
- **Interview** - The validity of the interview depends on the quality of the interview and the credibility of the interviewee. It also depends on the interviewer's continuous control of the data gathered.
- **Transcription** - The validity depends on the linguistic configuration of the transcription.
- **Analysis** - In the analysis the validation depends on the way the questions are analysed, and if the interpretation is sound.
- **Reporting** - Validation in reporting depends on whether the final report gives a valid exposition of the most important findings.

Reporting

The last sequence is about documenting the results in a sensible and credible manner. The report is the end product of a longer process, and it should document the main purpose, choice of method, results and implications. The easiest way to make a report as readable as possible, is to take the final reporting into consideration from day one. This is most easily done as follows:

- **Thematization** - The end product has to be taken into consideration from day one to make it easier to write the report.
- **Design** - Systematic documentation of the design procedure as a foundation for writing about the applied method.
- **Interview** - Ideally the interview should be of a form, that can be easily communicated to the readers of the report.
- **Transcription** - The readability of the interviews should be kept in mind while the interviews are being transcribed.
- **Analysis** - The presentation of the results and the analysis of the interviews should be embedded in the writing process.

- **Verification** - The decisive factor is how the study, will be reported.
- **Reporting** - Working towards the final report from day one, will help in the process of making a readable report with well-documented results.

As stated earlier we need ensure that our analysis and interpretations are sound. this can be achieved through clarifying interviews and observations where, if sound, the understanding should correspond to the observed reality. Hence the next section will cover observation as a technique and how we have used it to verify our results.

5.3.3 Observation

To verify our findings from the interviews and to gain further insight in the processes in the Ubuntu project we will observe the Ubuntu Developer Summit (UDS) in Barcelona, May 2009, where the road map of Ubuntu 9.10 is planned. This gives us the possibility to get first hand experiences with the process of planning a new release and observe how the Canonical employees and community members decide the changes and new features for an upcoming release.

Observation Typologies

Observation is typically a phenomenological data collection technique. Observation works best if it is used with another methodical praxis i.e. conducting interviews, and will often, like in our study, only be an integrated part of a methodic praxis, meaning that it will not stand alone as the only used technique [KK99, p. 45-46]. The typologies of observation studies can be split into four different typologies; structured observation in an artificial environment, unstructured observation in an artificial environment, structured observation in a natural environment, and unstructured observation in a natural environment. These can be seen in table 5.3.

A laboratory test is defined as a test taking place in an artificial environment, where unpredictable incidents and unintended impacts on the observed are being minimised. Observation in a natural environment on the other hand takes place in a context that already existed before the observer stepped in. In this type of research the observer is there on the terms of the field. Which means that the observer is aware of the fact that unpredictable and unintended incidents may occur. The line between a structured and unstructured observation is much more blurry, than the line between a laboratory test and an observation in a natural environment. Structured and unstructured observations can better be described as two extremities, where the structured observation will produce quantitative data. In the unstructured observation the observer is not looking for something specific,

		Degree of structure in data gathering	
		Structured	Unstructured
Degree of structure in the observed fiel	Artificial Environment	I Structured Laboratory test	II Unstructured Laboratory test
	Natural Environment	III Structured Observation	IV Unstructured Observation

Table 5.3: Typologies of an observation study

but is looking more generally and exploratory in the field. Our observation does clearly take place in the natural environment and we have chosen to make an unstructured observation so that we remain open minded to new impressions, however keeping our the result from the interviews in mind. Notes are taken preserve the parts that we find important. Compared to taking complete notes of everything this allows for freedom to participate discussion and ask questions that can aid our goal.

Degree of Participation

Another thing to consider besides the typology, is the degree of participation from the observer, the study will be conducted with[KK99, p. 99-111].

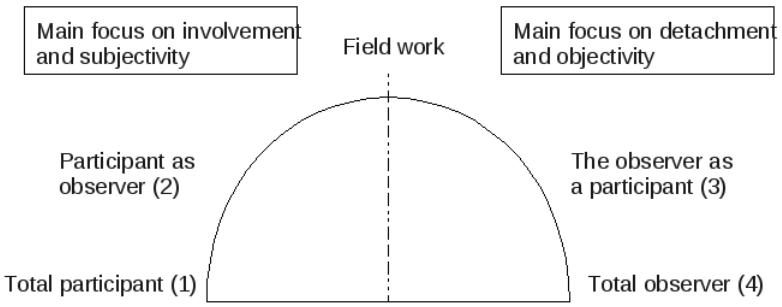

Figure 5.1: Classification of roles in field work

Figure 5.1 shows that there are four degrees of observation. One extreme is total participation, and the other is total observer. In the case where the observer is a total participant, the true purpose and identity of the observer

will not be known to the people that are being observed. The observer will observe the subjects in their natural environments and act as an equal. In this kind of observation the observer will be in an ethical dilemma toward the unknowing participant. Which is also why this kind of observing should be avoided unless it is the only option to obtain the wanted results.

When the observer act as a total observer, he will not have any interaction with the subjects. Like in the case where the observer acts as a total participant, the subjects will not know that they are being observed. In the case of total observer there is no risk that the observer might "go native" and get too involved with the subjects to remain objective[KK99]. There is, however, a risk that the observer will reject the subjects' views, and stick to his own conviction. The observer could talk to the subjects at another time to get them to explain a given behaviour, to give this kind of observation substance. The next variation is the observer as a participant. In this case the subjects are aware that they are being observed. The communication between the observer and the subjects is often short and sporadic. There is a risk that the observer misunderstands his subjects or the other way around. The observer might not realise this before it is to late or not realise it at all. Since the interaction is brief the observer might not really gain access to the social interactions and structures of the group. Both the total observer and the observer as a participant are seldom used in social studies, because the observer is quite distanced from his subjects.

The last role an observer can assume, is participant as an observer. In this case the subjects are aware that they are being observed. It is important that the observer both interacts with the group, and bond with some of its members, to gain accept and access to the group through those subjects, called informants. The problem with this kind of observation, is that the observer might not get access to all interesting members of the group, since the informants choose who to introduce the observer to. This could mean that the observer is being kept from some parts of the group.

We have chosen to undertake the role of being "The observer as a participant". This means that we have pay attention to opportunities get in contact with new people.

The Stranger

Being both a participant and an observer at the same time, is a dilemma. The dilemma could be that the observer might get so involved that he "goes native" or end up distancing himself from the group. If we refer back to figure 5.1 the distance might be an issue in the case where the observer is a "total observer", and proximity might be an issue when the observer use "total participation". This dilemma should be handled by balancing proximity and distance [KK99, p. 72]. To make sure to keep this balance between proximity and distance it is suggested to take the role of "the stranger"

[KK99, p. 72]. According to Krogstrup, Georg Simmel has stated that to obtain this balance, the observer should undertake the role of a stranger when doing field work. "The stranger is ascribed to a specific form of interaction, but also an objectivity, related to the distinction between proximity and distance" [KK99, p. 73-74]. Simmel describes this distinction as the following:

> "The stranger comes in random contact with all elements, but is not organisational bound to establish a friendship, or to the locality. The stranger is also not bound to specific rules or the dispositions of the group. The stranger will meet this with a distinct objective attitude, which does not acquire a lack of participation, and also accommodate both proximity and distance, indifference and participation. The objectivity is also defined by freedom. The stranger is not bound in a way that may make his perception, understanding and assessment of the data biased."[KK99, p. 73-74]

This should prevent the observer from "going native" which means loosing the distance to the field, which could entail loosing objectivity of the study.

Our Study

In our study we have chosen to make an unstructured observation in a natural environment. This was conducted at the Ubuntu developer summit(UDS) in Barcelona May 24th to May 29th 2009. This observation took place after all interviews had been conducted. The observations at UDS is used as a quantification of what we learned during the interviews. The role we took was "participant as an observer". Even before UDS we had made contact with several people we knew would be at UDS. This gave us a chance to be a part of the group and to gain further relations to the people surrounding them. At the UDS there was approximately 300 people which also made it easy to not go native, meaning that we could keep a distance to the participants but still blending in. During the UDS we attended sessions, but with up to fourteen simultaneous sessions we could not attend them all. We made written notes when we discovered something of interest, but did not write complete notes of everything in order the be more vigilant during the sessions. We chose individually which tracks we wanted to follow, meaning that we sometimes followed the same track. If there were sessions of particular interest to the study we would attend those. At the UDS we got a chance to see how the teams worked together, and how the community worked together with the Canonical employees. Both the community members present at the sessions and the community members participating via online IRC[2] and live audio streaming which were broadcasted live from

[2]Internet Relay Chat client

the sessions.

5.4 Research Design

5.4.1 Analysing Data

During the period of the interviews there is an ongoing evaluation of the interviews where it is evaluated whether the interviews provides sufficient information on the topics. If not, adjustments are needed in order to explore newly identified topics or simply to elaborate more on topics of which we have vague understanding. Prior to the analysis the interviews are transcribed in order to ease the handling of the data. The process starts with identifying categories of interest, and categorise statements regarding that specific topic. The participation at UDS will be used to verify our findings in the interviews and to extend our hypothesis where they are confirmed and to abandon those which cannot be verified.

5.4.2 Initial Selection of Topics

Three main subjects inspired by literature review.

- Cooperation and management between Canonical, and the volunteers in the Ubuntu Community
- Cooperation and management internally at Canonical
- Business model of Canonical

5.4.3 Topics After the Interviews

Having completed and transcribed the interviews we skimmed through the data and brainstormed in order to search for new topics. This resulted in the identification of several new and specific topics:

- Company culture
- Products and activities
- Business model
- Development model
- UDS
- Ubuntu collaboration
- Open source

- The product Ubuntu
- Mark Shuttleworth

5.4.4 Topics after Ubuntu Developer Summit

After going through our session notes from UDS we found no new topics, but we got a better understanding of some topics - i.e. Mark Shuttleworth's role.

5.4.5 Achieving overview before the analysis

The transcriptions are divided into employee and volunteer interviews. All statements and session notes regarding the defined topics are sorted into topic-specific documents containing all the collected data on that topic. Each topic-specific document summarises the different opinions and outlines the main opinions and/or differences.

After sorting the data it was decided to recategorise the topics to the following:

- The business model of Canonical
- Ubuntu Culture
- Champion and Canonical owner
- The Collaboration in Ubuntu
- Open Source

Based on the summaries of the analysis and the treated literature we will present and discuss our understanding of the treated topics in relation to our research questions.

Chapter 6

Analysis

This chapter will cover the various processes we have been through in the analysis of the collected data. From the first initial preliminary categorysation to the in-depth traversing review of the interviews, to the final summarising of the chosen subjects.
In the first part of the analysis we will look at Canonical's business model. This will be done both with the use of Raymonds theory about open source business model combined with the data we have collected through the interviews with both Canonical employees and Ubuntu community members.

6.1 The Business Model of Canonical

6.1.1 Ubuntu is Free

The most noticeable thing about the business model of Canonical, is that Ubuntu is free software. For Canonical this means, that there is a lot of what they work on, that will never be profitable, since all the work they put into making the distribution has no way of making money for Canonical. In the community the question has been if Ubuntu would remain free in the long run. Mark Shuttleworth, who is the founder of Ubuntu and the owner of Canonical has made the following statement:

> "Ubuntu is free and will always be free." [Mark Shuttleworth]

As stated earlier the word "free" can refer both to the price and to the freedom of open source. When Mark Shuttleworth is using the word "free" we believe, that he is referring both to the price of Ubuntu and that Ubuntu is open source. That means Ubuntu will always remain free of charge, and with an open source code.

6.1.2 The Content of the Business Model

When looking at the business model into more detail, we can see that there are many different aspects of the business model of Canonical.

> "Well our business model it is very wide, but our main model is providing service. [IP4]

Knowing Canonical will never be able to make money on the code itself, it is interesting to find out how they do intend to make money. Based on Raymond's nine business models, we believe that Canonical is spreading their business model over four of Raymond's nine models. Those are: "risk spreading", "loss leader or market positioner", "give away the recipe - open a restaurant" and "accessorising", respectively.

Risk Spreading

We believe that Launchpad is a good example of this business model. Launchpad is a code hosting and software collaboration platform. It is used to find new tasks to solve, for the next Ubuntu distribution. Launchpad has been closed source, but was recently released as open source. It is used by several different open source projects. It was created for the development of Ubuntu. We believe this is risk spreading because Canonical can share the maintenance of Launchpad with a community after making it open source.

Loss Leader or Market Positioner

Ubuntu is the product that in this model is giving no direct revenue. Instead it is used to sell software that works in Ubuntu. Canonical have created several of this kind of software for Ubuntu. An example could be Landscape. Landscape is a system management and monitoring service, which make it possible to manage multiple Ubuntu machines through a web-based interface. Ubuntu itself has no value for Canonical, the value comes with the possibilities that Canonical get from Ubuntu, by selling related software.

Give Away the Recipe, Open a Restaurant

Ubuntu is also used by Canonical to sell services to especially other companies that for instance want to get a costumized version of Ubuntu, with support. Or just want to get support for Ubuntu. Canonical offers five different types of services.

Custom Engineering Services
Canonical offers to make a customised version of Ubuntu, to make it fit to a given company.

Support Services
Canonical offers support by phone, email or the web, both standard (9 to 5) or advanced (24/7).

Certification
Canonical offers certifications meaning that hardware manufacturers have the possibility to make sure their hardware is fully compatible with Ubuntu.

Training
Canonical offers training in using Ubuntu. This can be value able for the companies that wishes to switch from another desktop or server solution to Ubuntu.

Application Packaging
Canonical offers to package software that someone else wishes to put into Ubuntu.

Accessorising

Canonical offers a lot of different accessories in their online store. For instance they offer wearables and other Ubuntu accessories, Ubuntu CDs, DVDs and software.

6.1.3 The Interviewees Relationship to Canonicals Business Model

Our experience from the interviews regarding the business model is that answers differed between employees and community members. Even within the group of employees the answers differed. Some employees knew a lot about the business model of Canonical. This was the people who had to do with it in their daily work. The employees who did not come in contact with partners in any way did not know much about it. Except that they are selling services. The following quote describes most of the answers we got when asking employees about Canonicals business model:

> "Well our business model it is very wide, but our main model is providing service." [IP4]

Most employees knew there was more to the business model than selling services, but did not know exactly what. We believe that the reason employees does not really care how Canonical is making money, except for the ones working close to it, is mainly because of Mark Shuttleworth. It is widely known that Mark has got a lot of money and is willing to spend a lot on Ubuntu. One employee even stated that:

> "Mark Shuttleworth is made of money" [IP7]

. This means that the employees do not have to care about whether Canonical is making money or not, at this time at least.

On the community side most interviewees replied they did not really know anything about Canonicals business model. If some of them did they were mostly guessing. In the end the answers we got was close to this quote:

> "Business model? I can't really speak about that, at all." [IP14]

Most community developers does not know much about Canonicals business model. We interpret this as not important to them to know in their work for Ubuntu. A community member mentions at one point that he found out Canonical is not a public limited company, because he wanted to buy some shares in Canonical. This means that the community members have no reason for knowing Canonical's business model. They cannot make any money themselves on it, and at this point they do not need to fear that Canonical will not be there for Ubuntu, in the near future. Especially because of the Ubuntu foundation mentioned in section 5.2, meaning that if Canonical suddenly cannot support Ubuntu, an amount of 10 million dollars will be released to make sure to keep Ubuntu going for several releases.

One interview subject describes that there are mostly two types of business models in open source development. The old model that is, giving a product for free and selling services and support for that project. The new model on the other hand is about having some of the product open source and other parts proprietary, like for instance Red Hat having Fedora as open source and Red Hat Enterprise as a proprietary product. He believes that Canonicals business model belong to the old kind as can be seen in the following extract from the interview.

> "Canonical is in the old model, which is mostly about capture market share and giving all your products for free ... the very few that will require support services or will pay for getting a customised version. This is the ambitious model, something where you invest a lot in the brand and you expect it to be successful enough so with small amount of services you can still make the company live." [IP5]

If we compare this statement with Raymond's theory about business models, and our view on Canonical's business model(s) through this, it is obvious that we do not agree that Canonical only uses the old business model. What we do believe is that Canonical originally sold services around Ubuntu, but now have expanded their business model and lately have incorporated other aspects like the ones this interviewee describes as "the new model".

6.1.4 How the Business Model Affects the Employees

The business model affects the employees, when a costumer might be involved. Means that what a paying costumer wants trumph what ever the employee was working on at the time.

> "well from time to time I get pulled away from my normal development to work on or fix a bug that a paying customer has

> reported in which case that usually trumph my daily whatever I was working on for the day." [IP8]

This means that the employees are under direct influence of the business model, or Canonicals partners. Which also describes why the employees know that the business model of Canonical is mostly about selling services. Some experience this side of the business model pretty often.

6.1.5 How the Community Affects the Business Model of Canonical

The community has an indirect influence on the business model:

> "The community counts a lot because sometimes we have to decline things for customers because its just not the Ubuntu way to do things. For instance sometimes the costumer wants to put something that breaks the GPL. And we could do that, but we just say no. Because it is not the way Ubuntu works." [IP4]

We can now establish that it does actually cost something, having a community. The price is that Canonical are bound by the rules of the GPL and some unspoken rules from the community, to make the effort of keeping the community happy. There are several examples of Canonical failing this, but we will get back to that in the analysis of the collaboration between Canonical and the community.

On the other hand the community does a lot of work for Canonical and the business model:

> "You have much more community members doing advocacy stuff, doing install party's, doing Ubuntu local user groups, than you have developers. The community having this spirit, trying to create this atmosphere of everyone is happy and we are great and its very important in the business model."[IP5]

This means that even though Canonical might make less money because of the community they also gain a lot from it, especially when it comes to publicity, and support to ordinary users.

6.1.6 Pros and Cons About Canonical's Business Model

Especially the employees brings up some pros and cons with the business model of Canonical. For one thing, even though the business model might work well for Canonical, it is not a given that it will work well for other companies. This might depend on whether another company is willing to invest as much money as Mark Shuttleworth has invested into Ubuntu. The business model also require acknowledgement from investors that this might be a long term investment;

> "It is a good business model, the only problem with this kind of business model is it takes quite a long time to actually pick up. So would be very hard for us to keep up this way if we hadn't Marks money to back us up." [IP4]

We believe that the reason that Canonical is expanding their business model to include some proprietary products like Landscape and Ubuntu One is to make more revenue. Meaning that the business model about only provide services, servers and support is not enough to make a company like Canonical profitable on its own. But moving towards offering this kind of software might give Canonical some problems with the community. An employee stated the following:

> "There wouldn't be so much community members if we chose another model. You cannot see so much community in projects, that keep part of the product closed." [IP5]

This statement is made from the fact that he believes that the business model of Canonical is only about providing services for Ubuntu, but since it is not this means that Canonical might run into the problems that this employee is stating. The discussions about Ubuntu One has already been ongoing for a couple of months now, mostly with community members speaking from one side and Canonical employees speaking from the other.

6.1.7 Summary

Using the definitions by Raymond, the business of Canonical has been described as four different models. The "Free the recipe, open a restaurant" of selling services around Ubuntu, being the primary strategy.

The responses from interviewees on the business model of Canonical, has been characterized and compared with Raymond's models. Generally few, both employees and volunteers, showed much knowledge of the business side of things. Most volunteer focus entirely on their own partition of Ubuntu, and within Canonical the business related activities are allocated to specialized personel.

Furthermore it is described how certain Canonical employees' daily work can be directly affected by requests from important customers. The business of Canonical is also adapted to take care of licences issues related using open source licences like GPL, which restricts the types of agreements Canonical can make.

6.2 Ubuntu Culture

This section will present our understanding of the organisational culture of the Ubuntu project based on the interviewes and our observations at UDS.

6.2.1 Code of Conduct

One of the characteristics of Ubuntu is the code of conduct which is a set of behavioural guidelines which both the community members and employees are expected to respect. The aim of the code of conduct is to increase the members awareness of how they communicate, and encourage them to be considerate towards each other. We see this as a way of seeding a specific culture, and ensuring that it remains as initially intended. It is a clear cut example of a written set of norms. Most of the interviewees feelt that the code of conduct made the Ubuntu community a nicer place to be, and several interviewees had experience with being in other open source communities before, where they did not experience the same open and positive attitude as in the Ubuntu community. One of the interviewees expressed it as:

> "We have the code of conduct, it rarely gets mentioned ... there is a different tone in discussions, than I've been used to in many different places within the open source world."[IP 7]

The general opinion is that the vast majority of the community and the employees appreciate the code of conduct.

6.2.2 Coordination and Communication

The Ubuntu project consists of participants scattered all around the globe, with most of the developers working from home. This will be reflected in the culture especially in terms of communication channels and coordination measures. The coordination of a large and geographically distributed project as the Ubuntu project, requires a lot of communication. So how is communication. carried out within the Ubuntu project? According the interviewees the means of communicating are mostly:

- IRC
- Mailinglists
- VoIP
- Phone

According to the interviewees, most of the daily communication is done via IRC or by mailing lists. In important cases, or when the members know each other well, the communication might be by phone or VoIP. Physical meetings are more rare, and some developers only, if at all, meet at the UDS. Working with people located all over the world introduces some problems due to the span across time zones, as expressed by interviewee number six:

> "Scheduling calls for a global company is really difficult. Especially when you have to talk to someone on Munich, Singapore and Chicago. To have a phone call, someone has to be up in the middle of the night.[IP 6]"

In addition to this, several of the interviewees states that the development at times can be quite stressful due to the odd work hours.

6.2.3 Ubuntu Developer Summit

The next aspect of the culture which we will focus on, is about the scheduling of tasks and assignments in the Ubuntu project. According to the interviewees there are not made formal estimations on the projects in terms of, for instance, expected man hours, but the people involved in the project makes a guesstimate, and has regular catch-up meetings during the development. Is is pointed out, that the developer responsible for a project usually have a good feeling of how much work is required, making it a qualified estimate.

Despite the seemingly absence of formal estimations of the projects, it is discussed at the UDS, which features seems feasible during the next release, and during most of the sessions, notes are taken using the Gobby collaborative text editor, which enables all of the participants at a session, to take notes in the same document during the session. During our observations of the session at UDS, we noticed that it's very different how important the Gobby document is considered to be. The tendency is, that the documents get more attention at sessions lead by Canonical employees, than at sessions run by volunteer community members. At some sessions lead be community members, it took twenty minutes before the person running the session, noticed that no document had been created, and hence no notes had been taken. At another more informal session, it was considered whether a document should be made or not.

6.2.4 Development Model

When we asked the interviewees, what development model. The majority of the answers where that they don't really follow any specific model, and that it is mostly based on the agreements among the developers, and the progress is monitored through regular status meetings. The closest we came to an actual model, was that statements, that it would be an agile development model. So in this case, it sounds like interviewee number four's statement on the topic, gives an indication:

> "...it is quite controlled chaos." [IP 4]

This said there are indications that the many stages of rapid alpha and beta releases encourages to what could look like sprints, which is known from the agile software developing approach Scrum.

6.2.5 Reputation

It was clear to us, that reputation of employees and community members play a key role in the culture of Ubuntu. For instance Launchpad users get karma points for their profile based on the work that they do. This system is structured such that the more tasks you solve, and the more important they are - the more points you get for you profile. Then the rest of the community members can see how good you are. A contributors reputation in the community is defined through what he does and how does it. Interviewee number fourteen states precise and clearly:

> "You are treated based on how you act and what you do." [IP 14]

The reputational demands are the same whether you are a volunteer in the community or a Canonical employee.

> "you don't get any special right from working for Canonical. You still have to prove yourself." [IP 3]

According to the interviewees, they work hard to achieve their reputation there are volunteers doing up to 60 hours of Ubuntu related work a week. Among the employees one developer has been working up to 90 hours a week.

6.2.6 Employees and Volunteers

According to most of the interviewees, there is very little difference between the volunteers and the employees, where the main difference is that employees has the possibility to dedicate more of their time to Ubuntu, which makes it's easier for them to know what it happening within the project.

Canonical Culture

More of the employees sees themselves as equals with the community members, and Canonical employee and interviewee number three expresses it with:

> I03: "I think that most Ubuntu developers, that are Canonical employees, participate within Ubuntu as any other developer..."

Community Culture

When it comes to the voluteer contributors, there is a slight difference in the way, that they percieve their position. Whereas the employees position themselves as equal to the volunteers, the volunteers refrain from positioning themselves, but only concludes that it is difficult to tell who is an employee,

and who is a community member. An example is the following statement from one of the volunteer contributors:

> "it is not clear to me who works at Canonical and who doesn't. Its just all the people on the list, and everyone's opinion seems valid." [IP 13]

The volunteers respect the opions of both voluteers and employees as equal, and make no distinction.

One thing is the equality between the volunteer contributors and employees. Another is when it comes to Canonical as a company, were there tends to be some scepticism from the community, against changes Canonical pushes into Ubuntu without the endorsement of the community. Interviewee number five states it as:

> "Most people are doing obstruction, to whatever they perceive that Canonical pushes, rather than something that should be done in Ubuntu anyway." [IP 05]

An example of such resistance against forced changes, is the notification bubbles added in Ubuntu 9.04. The notification bubbles came in rather late in the release cycle, and many community members felt that it was forced by Canonical, and that it didn't have the support of the community. This leads us to our understanding of the organisational hierarchy of the Ubuntu project, it's different governing bodies, and how they resolves various disputes.

Governing Bodies

At the top of the hierarchy, founder and owner, Mark Shuttleworth, has the final say in every decision or dispute, if the the other instances can't reach an agreement. The ability to veto something, is however used very seldomly. The way it often works is that Mark Shuttleworth present and argues his view on the matter, and the other parties tends to accept his view. Most disputes are solved by the involved parties themselves but if they cannot reach an agreement, their dispute might reach the Technical Board, the Community Council or the MOTU council depending on what the dispute is about. As the name suggest, the Technical Board deals with the technical decisions, such as package selection, library dependencies and the like. The community council takes the social aspects of the community, and ensures that the community members respects the code of conduct. The community council is also the governing body, responsible for approving new teams and projects. MOTU council is a bit different from the the technical board and the community council, since it applies only to the MOTU part of the community. In many ways, it serves many of the same purposes as the community council, by resolving disputes in MOTU, and by the supervising

the MOTU community. In this way many decisions and disputes are handle without ever being elevated to the top level.

6.2.7 Summary

The code of conduct is a document stating the desired way to interact in the Ubuntu community. It asks community members to be considerate of how they communicate. It is considered one of the reasons, that the Ubuntu community is often characterized as a more welcoming community ,than some other open source communities.

Communication within Ubuntu is mostly done by IRC, since everyone is at home, without physical contact to other community members. In order to have the community meet face to face, the Ubuntu developer summit is held at the start of each new release cycle. Here both employed and volunteer community members plan the next version of Ubuntu.

The Ubuntu community rewards effort by peer recognition. A system of karma point is established to emphasise this.

6.3 Champion and Canonical owner

The Canonical CEO Mark Shuttleworth is the founder and owner of Canonical. In the following the background of Ubuntu and Canonical is presented, and the curent position of Mark Shuttleworth in Ubuntu is decribed.

6.3.1 History

While obtaining a Business Science degree in Finance and Information Systems at the University of Cape Town, Mark Shuttleworth got involved with the open source projects Debian and Apache. Mark Shuttleworth later started the certificate authority company "Thawte", which he sold in 1999 to competitor Verisign.

After realising his dream of space travel, starting both a venture capital company and a non-profit foundation to improve education, Mark begun planning what would become Ubuntu. In April 2004 he assembled a group of open source developers to work on a new Linux distribution. These first members of the Ubuntu community had all been hand picked by Mark Shuttleworth, based on their previous work and reputation within open source development. Mark Shuttleworth named the new Linux distribution Ubuntu. It is a concept from South African culture, which has been translated as "Humanity towards others". It expresses a spirit of sharing and collaborating, that is also found in the culture of open source software.

The company Canonical was founded in 2004 to employ and pay developers to work on Ubuntu. To hire the most qualified members of the global

open source community, Canonical became a virtual company, where the employees work from their homes.[HHB09]

6.3.2 Position and Control

Mark Shuttleworth is the one person everyone seems to know within the Ubuntu project. When asking about the organisation of Ubuntu, he is the most frequently mentioned person. It is well known among all the interviewees, that Mark Shuttleworth is the top figure of Ubuntu. This is also true among those, who do not have a detailed understanding of the leadership of Ubuntu.

> "Mark Shuttleworth fly pretty much right at the top and then I think there is a technical board of a few people around him and then under that there's the different teams" [IP19]

With many who know the name Mark Shuttleworth, but not much about him, could have given him a special status in the community.

> "I only know a little bit about him[Mark Shuttleworth] and some of those Canonical stars" [IP12]

The size of the Ubuntu community, and the busy schedule of Mark Shuttleworth unavoidably creates a certain distance to a large part of the Ubuntu community and the CEO of Canonical. That can make Mark Shuttleworth appear as an illusive figure atop the Ubuntu community. In any way he is the face of Canonical to external partners, as well as the volunteer contributors within Ubuntu. To the interviewees from Canonical, Mark Shuttleworth is still an authoritative figure, but he is not surprisingly described as a colleague.

Mark is portrayed as the visionary of Ubuntu by both volunteer contributors and Canonical employees.

> "He[Mark Shuttleworth] has got a really cool vision of what he wants Ubuntu to be, and I think that's inspiring" [IP8]

> "Mark Shuttleworth he will state the name of the next Ubuntu release. He will review the goals, that he wants to accomplish, and after that, various developers starts making blueprints in launchpad to try achieving those goals." [IP14]

Mark Shuttleworth champions the issues he finds are important for the next release. At the UDS in Barcelona he opened the first session on the server track with his ideas for the coming release, and then left the room to leave the server team to finish the session.

It is clear that Mark Shuttleworth have a great deal of influence on the direction in which Ubuntu develops. Mark Shuttleworth is also the final

step in a chain of escalation of disputes. Issues that can not be resolved otherwise can ultimately be decided by Mark Shuttleworth.

> "We have Mark Shuttleworth, our big boss and chieftain, and if he thinks something, well in principal he has the right to veto everything, but Mark he is sly enough to know when to use that right, and when best not to" [IP7]

The last part of that statement is an interesting example of the balancing act to direct Ubuntu in a particular direction. At the UDS Mark Shuttleworth referred to his stepping to end a discussion as a "heavy club to hit with". He further explained, that he often let things play out for a while to see if something constructive would occur, and only when that does not seem to happen will he take action ending the discussion. This did happen at the Ubuntu Developer Summit at least one time, during a short session, where the speaker made some statements the Ubuntu Community and employees did not agree with. After a few minutes Mark ended the discussion by saying:"this is not the right time or place to have this discussion", which people accepted, and the discussion was over.

Even though Mark has a lot of ideas himself and has got a great influence on the features in Ubuntu, it takes a lot of people to keep Ubuntu up and running, and going forward. Both Canonical employees and the volunteer community are needed for achieving this, and especially that they manage working together on Ubuntu.

6.3.3 Summary

Mark Shuttleworth is the visionary of Ubuntu. He is considered an inspiration among many in the Ubuntu community. His opinion carries great weight, but not all approves of his great influence, and leadership of Ubuntu requires much balancing from Mark Shuttleworth.

6.4 The Collaboration in Ubuntu

A very important part of Ubuntu is that there are many people involved in the making of Ubuntu. All these people come with very different interests and agendas. This means that clashes can and will occur especially between Canonical and the community.

6.4.1 The Relationship Between Canonical and the Community

Its important realising that Canonical is into Ubuntu to make money on it. Even the volunteers have different resons for contributing. Some are doing

it for the fun of it, and others are being paid by a company or are doing it for their own company.

> They [the community] have their own schedule and of course they have their own set of priorities which in some in many you know might be commercial more than community driven. [IP 15]

This means that there are many very different reasons for contributing, and people sees Ubuntu from very different angles. This can give some problems within the community. Canonical have to make sure that the distribution is the best possible, even though the release management team have to approve something for the next release after the release deadline.

> Somethimes we have to rush some features in, we have to bend the rules to get the features done. [IP 5]

This have given some trouble with the community. That Canonical can break the rules when many others cannot. But according to several interviewees some community members can also break the rules if they are trusted community members. Canonical are working hard to not end in situations where breaking the rules are necessary, because they know it will make people from the community very unhappy. They also know that the best way to avoid these situations is just not breaking the rules. It seems like it is very different how community members are handling Canonical breaking the rules. Some community members, states that they do not think Canonical would break the rules unless it was really necessary, but apparantly others feel that Canonical have got to much power over Ubuntu, and therefore really dont like when Canonical is breaking the rules.

> Some people, even if they're very involved in Ubuntu and stuff, they think Ubuntu is good, but they don't always think Canonical is good. Because they have an idea that we are up to something. That we have a secret plan, which off course is evil, because such plans are always evil. [IP 7]

W believe that the reason some people are not that impressed by Canonicals involvement in Ubuntu is because their view of open source is very idealistic, in the way that open source should be free and there should be no money involved at all. On the other side they might feel that Ubuntu is the best distribution for them.

6.4.2 Recent Problems

In the latest Ubuntu release cycle version 9.04 released april 2009, Canonical introduced a new team, the user experience team. This team brocke the

rules for the release schedule and released some new features pretty late in the release cycle. The features were called notification bubbles. Both the new features and the way they were introduced made a lot of people especially from the community very unhappy. When askin about this exact problem in our interviews, the joint oppinion from most interviewees was that what people was really mad about probably had more to do with that the community did not feel they had anything to say about the new features, and less to do with the features it self.

> Many people, especially Kubuntu users, were not too happy about this, because we really were not able to express our opinions about the notifications beforehand. And we pretty much felt, that Canonical was using their power to get things done, which many people dont like. [IP 14]

Even Canonical employees stated that they did not think this situation was handled very well, and was not sure that implementing the notification bubbles for this release was the right choice to make. The discussion about the notifications ended when Mark Shuttleworth stated that the change was final, and that it would not be changed back.

> I don't really like the change, but Mark says it is final and he has some valid points for it, like Ubuntu should lead, not follow. [IP 11]

Especially the point about Ubuntu should lead and not follow, might be the reason that the notification bubbles did get implemented for this release. Meaning that Canonical might have felt it was worth the heat from the community, to get in front with this kind of features.

6.4.3 Distribution of Tasks Between Canonical Employees and Community Members

Since Canonical is not the owner of Ubuntu, but a contributer though a large contributer, this means that the tasks Canonical are choosing to work on, regarding Ubuntu, are the feature they wish to get in the next release. The Canonical employess does not have much say in what they are working on, those decisions are made on the manager level. The community on the other hand are free to take what ever tasks they want.

> My manager can quite happily assign a task to me, because he's my manager, but he couldnt assign it to a community member. [IP 2]

Canonical employees are working on what tasks that Canonical have decided are important for them. Since the community members are volunteers no one

can make sure that they finish an important task before a deadline. This means that if a community member takes on a task and for some reason does not finish it, the feature will not make it into the next release. What community members are often working on, are features they or maybe their employer have a special interest in,

> The community members have a clear preference in what they want to do, they usually, especially community members in my team, they are usually working for their own companies or they have their own personal agenda, they need some features in ubuntu, and they are working clearly on that. [IP 5]

If a community member does not have anything specific they want to work on, they do have the possibility to pick up tasks or fixing bugs. These tasks can be found on Launchpad.

6.4.4 Ubuntu Developer Summit

The Ubuntu Developer Summit (UDS) is the kick-off for a new release of Ubuntu, and the primary means of planning and coordination. It takes place about one month into each release cycle. Every second time it is located in Europe or North America respectively. The latest was held in Barcelona from May 25 to May 29.

Due to the distributed nature of Canonical as well as Ubuntu at large, the UDS is one of the few times the Ubuntu community meets face to face. The summit is open to everybody, who wish to take part in shaping the next release of Ubuntu. Canonical employees are expected to participate, and get all expenses covered by Canonical. Anyone can chose to participate at their own expense, but Canonical will invite certain volunteer Ubuntu contributors, and sponsor their stay.

The purpose of bringing all of those Ubuntu developers together, is to discuss ideas for the upcoming release of Ubuntu. In preparation to the summit, input from bug reports, users, developers and Canonical teams are collected to see what everybody wish to see happen in the next release. The suggestions are then arranged in a schedule for discussion at the UDS. The Canonical employees attending the summit meet the week before the summit to socialize and prepare. Sessions are often lead by a Canonical employee. For practical reasons the summit is organised in a number of different tracks. At the Barcelona UDS there were seven tracks divided among 14 rooms at the same time. The tracks were “Community”, “Desktop”, “Foundations”, “Kernel”, “Mobile”, “QA”[1] and “Server”. In total there were 335 one hour sessions throughout the week, and 20 short plenary sessions. The schedule is adjusted during the summit, to allow for follow-up discussions of issues,

[1] Quality Assurance

which were not covered in a single session. In the same way some sessions turns out to be less important, and are then removed from the schedule.

The sessions in each track are very different. Sessions in the community track could be about how to best include newcomers in the Ubuntu community, where sessions in the kernel track will often concern low level discussions of hardware interaction. To help the participant at the summit find others, who share their interest, the ID badges worn by all, could be marked with colour coded labels, signaling ones interests.

In addition to the Ubuntu community, representatives from upstream projects and Canonical partners also take part in the UDS. Since this is one of the few times the developers of Ubuntu are together in one place, this is a chance for everyone to expand their network in the community. A few sessions were lead by Canonical partners discussing ways to collaborate with the Ubuntu community. The UDS is also a great way to strengthen the cohesion between Canonical employees and community members alike, both through the sessions and social events.

The UDS is clearly an important event for Canonical, since it is one of the few chances for its employees to socialize. It is however a very busy week. The time is precious, because it is the only time the Canonical teams and volunteer members of the community have this kind of opportunity, to exchange opinions and ideas. The UDS is a vital part of the established collaboration in Ubuntu. The blueprints created based on the discussion forms the basis for the coordination of resources.

Even though UDS is an open event, practicality dictates, that only a small subset of the Ubuntu community will be present. That means the majority of the community is cut off from the direct access to participate in setting goals for the next Ubuntu. This is not in compliance with the ideals of open source development. To alleviate the issue, audio from all sessions, and video from some, is streamed online. This creates a line of one-way communication. In some sessions the chat service IRC was used to create a return channel. It was however only in a few sessions this happened. The outcome is that even with the audio or video streaming, influence at the UDS is primarily attainable for those, who are physically present.

6.4.5 Summary

The different backgrounds and interest of the people forming the Ubuntu community presents challenges in finding a common way to develop the distribution. A recent issue about the introduction af a notification system, shows how Canonical's interest to push Ubuntu in a certain direction on its own terms, can lead to unhappiness in the community. Even though volunteer and employed contributors often work under very similar conditions, the difference will become apparent when business concerns requires action from employees, or in the level of freedom a volunteer has to focus on a very

limited scope.

The Ubuntu developer summit is the main planning event for each version of Ubuntu. It is a very complex and busy event, where a broad scope of all that is Ubuntu is discussed. Despite great effort to make the event available to the larges possible part of the community, the nature of event limits the value of remote participation in this part of Ubuntu.

6.5 Open Source

Being an open source project, Ubuntu is set to exist under certain conditions. Canonical does not own the software, and can only direct the focus of its own employees, not the community at large. In principle all decisions are made by consensus. Exceptions from this principal can jeopardise community support for the project. Access to the source code means, that anyone can take the software and customise it as they see fit. Furthermore the open access to source code allows for open participation in a collective effort, to work on Ubuntu as a common good. The open source nature of Ubuntu becomes a frame for both control and participation in the project.

6.5.1 The Community's View on Open Source

There is no indication of a strong ideological support for open source among the interviewees, but there is a positive attitude towards open source. Open Source is mentioned as a part of the positive feeling of taking part in a shared effort.

> "I contribute to it because I use it every day so I am happy just to improve it for myself and other users" [IP12]

This feeling of goodwill from doing something, which can benefit others is repeated by several others. So is the first part of the above statement. That taking part in an open source project makes it possible to make improvements to suit ones own needs. This is what is sometimes called "scratching your own itch". A property of open source development expressed in a statement like the following:

> "one of the most interesting things about open source is how, you can get involved by your selfish desires, like you just wanna fix something ... with other things something might annoy you, but there's nothing you can do about it" [IP19]

The ideology of open source is not important to the contributor, but the practical consequences of open source development is.

When talking to volunteer community members about contributing to Canonical's business without compensation, many mention contributing to

an open source project, as a reason to accept this. Because the work is contributed to a common good, Canonical's possible earnings are not seen as injustice.

6.5.2 Canonical's view on open source

For Canonical as a commercial stakeholder in Ubuntu, its control is limited because Ubuntu is Open Source Software. Canonical does not have ownership of Ubuntu, and most adapt its financial operation as described in Section 6.1. The daily work routines are also adjusted to suit the distributed and community oriented development practise as described in Section 6.4. An interesting example of how Canonical participates in the open source community, is that employees of Canonical have to earn their right to access software repositories, as any other participant in the Ubuntu project.

This shows that Canonical takes the role of its employees as members of the community seriously. They value the support of the community, and wish to participate on equal terms. In that way the Ubuntu community resembles other large open source communities, with the exception that Canonical employees can work full time on Ubuntu.

Canonical is however not purely dedicated to open source. The coordination tool, Launchpad, was developed by Canonical as a closed source project. It is part of the infrastructure that is put in place to support the development of Ubuntu, and has only recently been made open source after requests from the community. Launchpad was created to be a single point of coordination for all open source projects. Providing the source code could result in the deployment of a large number of similar tools servicing different projects in a fragmented manner. In the case of Launchpad, Canonical took a pragmatic approach to achieve their goal. Ultimately releasing launchpad as open source again shows a will to please the community, but only after launchpad had become an established tool for a number of open source projects.

6.5.3 The Open Source Ecosystem

Ubuntu is part of a larger collection of open source projects. Much of the software included in Ubuntu, is developed within other open source communities. Such external development projects are referred to as upstream development. The contributors working on Ubuntu integrates the upstream software in their distribution. That helps the software, to be exposed to a larger user base, which often means bug reports and feature requests becomes available in greater numbers. The increased testing and end user input can help to improve the software. That way the Ubuntu project can contribute something back to the other projects, which provide software used by the users of Ubuntu.

Another Linux distribution called Debian is a very important upstream contributor to Ubuntu. Each Ubuntu release is based on the current unstable version of Debian at the time of each development cycle's beginning. A returning session at the UDS, is an evaluation of the relations with the Debian community. At the Barcelona UDS the notion was, that previous scepticism towards Ubuntu had declined, and experience from working with Ubuntu is now viewed as positive in the Debian community. Because Debian is a purely community driven project, the collaboration between Ubuntu and Debian is largely based on personal relations among contributors.

The structure of Ubuntu further complicates the issue of control over the project. Canonical can communicate directly with the people driving upstream development, but have even less mandate in those projects, than in the Ubuntu project.

6.5.4 Summary

As a fundamental factor in what Ubuntu is, its open source nature defines the business, the possibility for participation and ultimately the community, that is Ubuntu.

Canonical's means for controlling Ubuntu are limited compared to a traditional commercial software project, but enhanced by the committed additions by the volunteer developers. The Ubuntu community is largely driven by an intent to make Ubuntu better for one self and the community at large. Canonical has committed to an open source strategy, and are working to keep business and community together.

Chapter 7

Discussion

As stated in Chapter 1, the purpose of this study is to examine the questions: "What control does a company have over a commercial/volunteer open source software (OSS) project?" and "What is the relation between volunteer participants and a company with commercial interests?"Based on the information obtained through the analysis, those question will now be answered.

7.1 Ubuntu as a Part of Open Source History

As it has already been described in the literature, the trend for most successful open source software, has been that it is so-called infrastructure software. This is the case for much of the open source software describe previously in this report. Ubuntu can be said to be part of a development in another direction. Based on the heritage of the open source software, which today make up vital parts of the Internet, which makes the distributed development process of Ubuntu a possibility. The Ubuntu desktop at least is an attempt to take open source software in to the mainstream use, with a focus on end users and ease, which previous open source successes didn't make a priority.

7.2 Development in Ubuntu

The development effort in Ubuntu is similar to what is found in many open source projects. Communications is based on Internet technologies. IRC is the primary means of communication. What is interesting is that both the volunteer and the employed developers work under similar condition, since Canonical does not house its developers in a common office, but have them working from home. Both employed and volunteer developers work in a distributed environment. In that way the employees have access to the same communication channels as any volunteer.

The leadership in Ubuntu is centered around Canonical and its employees. Canonical employees are organised in teams, which are responsible certain functional parts of the Ubuntu distribution. These teams cover the central and most critical parts of Ubuntu. The leadership provide by the leads in the Canonical teams has a great influence on the direction in which Ubuntu is developing. This can be seen as a more organised extension of the core group found in many community driven projects. Each team is in a way the core group within its domain. Volunteer developers can take part in the work, but the main effort lies with the Canonical team. Mark Shuttleworth can be characterised as a so-called benevolent dictator. He is the one with the final say. If conflicts can not be solved, he is the one who can step in. Mark Shuttleworth is different from the leader of a non-commercial community project. Mark Shuttleworth is involved with Ubuntu to make a business. It is difficult to say for sure, but Mark Shuttleworth could have strong finansual motives not to be benevolent.

Among the volunteer developers in open source communities, it is the ones who commit and delivers good quality, who are trusted and rewarded influence. The Ubuntu community is the same. Canonical makes its employees take part in the community on the same terms as any volunteer developer. Only after having shown to be worthy employed developers earn the privileges to change the code in Ubuntu. Once an employee have become a part of Ubuntu, the dedication a full time effort allows for, is likely to help them become among the most trusted and privileged. They will be further aided in that effort by the extensive network, an employment at Canonical provides.

The Ubuntu project is different from many others in the choice of tools. The proprietary Lauchpad platform, developed in-house by Canonical, is completely essential for the way Ubuntu works today. Canonical owns the infrastructure, on which all Ubuntu development relies. The software and the hardware.

The volunteer members of the Ubuntu community would appear very similar to those of other communities. An emphasis on considered interaction through the code of conduct, seems to have a positive effect on the comfort for the community. The Ubuntu community expresses a positive attitude towards open source, and the spirit of collaboration and sharing. In addition to the pleasant feel for opensource, there is also a very positive attitude towards all things Ubuntu. The work by the Canonical community team, could very well be a great part of that.

7.3 Business in Ubuntu

Examining Canonical's business model shows a focus on selling services on the basis of Ubuntu. The key for Canonical to make its service business

a success, is how popular Ubuntu can become. The demand for any of its services, as well as for sale software products, is relative to the number of potential clients, who runs or would like to run Ubuntu. Even without a single Ubuntu user paying Canonical for anything, a large user base can help convince hardware producers, that they need to be working with Canonical. Looking at Raymond's open source business models, the strategy Ubuntu is following is not new. The large start-up capital available to Canonical by Mark Shuttleworth is properly the most interesting aspect of Canonical business operation, making it a unusual endeavor.

7.4 Canonical's Control of Ubuntu

Ubuntu is, as many other open source projects, characterised by a system, where influence is rewarded to those who show commitment and quality in the work they contribute. The more time a contributor is willing to spend working on Ubuntu, the better are the condition for establishing a network, being able to keep taps on the activities in the community and make a name for one self. Because the employees at Canonical are paid to work on Ubuntu full time, they are more likely to be the ones, who has the most time available to contribute to Ubuntu. Furthermore the Canonical employees work as part of a team, which means they are likely in a better position to establish a network among Ubuntu developers, and they will be known names to other Canonical employees.

To ensure that critical work for each release is taken care of, employees of Canonical are typically assigned those tasks. They can be considered more reliable, since they are on contract to dedicate time to Ubuntu development. This also means, that volunteer contributors are rarely the ones working on critical new additions. Volunteer developers are generally driven by their own interest. Interest can vary, and the volunteer developers have been described as having a more narrow scope in their contributions, compared to the employed developers, who need to be more focused on entire solutions.

Because of the time invested, an established identity in the community and a spot on a Canonical team, it is the employees, who become leaders in most efforts in Ubuntu. Canonical provides the infrastructure, which makes the applied work processes possible. By doing so, they are an important part of determining the rules of how work is done, and by who.

This leads to a situation, where the most influentially, best known and most valued contributors are on Canonical contracts. Depending on exactly what degree of independence the individual employee have in how to interact with the community, this is a situation in which Canonical have a great deal more influence, than one might assume a company, not owning its core product, would have. This is influence to drive new initiatives, and it is not necessarily at they expense of volunteer contributors change to

drive their initiatives. The establishment of several software repositories, where some are under the supervision of volunteer community members, allows Canonical to dominate the parts of the distribution, which they find important. At the same time initiatives from volunteer developers can live their own lives in less restricted repositories.

7.5 The Relation Between Employees and Volunteers

Chapter 8

Conclusion

The purpose of this study was to get answers to the two following questions about open source development:

- What control does a company have over a commercial/volunteer open source software (OSS) project?
- What is the relation between volunteer participants and a company with commercial interests?

To be able to answer these questions we conducted a case study with the use of the data gathering method qualitative research interview, and additionally an unstructured observation study conducted at the UDS. We have interviewed 20 people, nine people from Canonical including Mark Shuttleworth and eight volunteer contributors. After conducting the interviews and observing at the UDS, by means of a brainstorm, we ended with five bases on which we would analyse our data.

We believe that Canonical has a lot of control over the Ubuntu release. They host the developer summits and they decide which volunteer contributors will get sponsored for these summits. The first month of a new release cycle Canonical employees are planning what features they and Mark Shuttleworth would like to see in the next release. After UDS, Canonical employees will work on the tasks that Canonical found was important features for the next release, and only a few volunteer contributors are working directly with the developing teams on the Canonical tasks. Most volunteer contributors will work on fixing bugs etc. or will be working on something set by their own agenda. The Canonical employees and most volunteer contributors acknowledge that Canonical runs most of the show about Ubuntu. Only a few open source idealists sees this as a problem, as long as Canonical is not breaking the rules surrounding the Ubuntu release cycle. When Canonical is breaking those rules a lot of people are not happy, and the discussions seems endless. This might result in a statement from Mark Shuttleworth saying that a change is final, and then the discussions dies or get more quiet. In

the end we do believe Canonical is running the show. But they do try to set up a more democratic environment around the volunteer contributors, and only use their power if it seems like the best solution for Canonical. The volunteer contributors and Canonical have one major interest in common: Making Ubuntu as successful as possible. The volunteer contributors must feel there is a reason for staying and contributing to Ubuntu, otherwise they probably would have left already. A very important factor is Mark Shuttleworth. He is an icon, and a star, even though he is also the one who can veto everything. He is playing a very important part, trying to make sure that his and Canonicals interests are nurtured but also remaining popular with the community.

Bibliography

[AGP] URL: `http://www.opensource.org/licenses/agpl-v3.html`. Available at 22/07-2009.

[Ber] URL: `http://www.mkbergman.com/?p=115`. Available at 22/07-2009.

[BLMR07] Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Cristina Rossi. Business firms' engagement in community projects. empirical evidence and further developments of the research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*, page 13, Washington, DC, USA, 2007. IEEE Computer Society.

[Bry08] Alan Bryman. *Social Research Methods*. Oxford, third edition, 2008.

[BTL+09] Ladislav Bodnar, Dr Zhu Wen Tao, Susan Linton, Chris Smart, and Robert Storey. http://distrowatch.org. web-page, June 15th 2009.

[coc] URL: `http://www.ubuntu.com/community/conduct`. Available at 30/07-2009.

[def] Open source definition. URL: `http://www.opensource.org/docs/osd`. Available at 06/03-2009.

[FF00] Joseph Feller and Brian Fitzgerald. A framework analysis of the open source software development paradigm. In *Proceedings of the twenty first international conference on Information systems*. Association for Information Systems, 2000.

[FF02] Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison Wesley, 1 edition, 2002.

[Fit06] Brian Fitzgerald. The transformation of open source software. *MIS Quarterly*, 30(3), 2006.

[gui] Debian social guidelines. URL: `http://www.debian.org/social_contract#guidelines`. Available at 06/03-2009.

[HHB09] Benjamin Mako Hill, Matthew Helmke, and Corey Burger. *The Official Ubuntu Book*. Prentice Hall, fourth edition, 2009.

[HJ03] Jesper Holck and Niels Jørgensen. Open source development: Coordination by means of continuous integration. In *12th International Conference on Information Systems Development: Constructing the Infrastructure for the Knowledge Economy*, 2003.

[HO01] Alexander Hars and Shaosong Ou. Working for free? - motivations of participating in open source projects. In *34th Hawaii International Conference on System Science*. University of Southern California, 2001.

[J97] Barton Cunningham J. Case study principles for different types of cases. *Quality and Quantity*, 31(4), 1997.

[Jä04] Pertti Järvinen. *On Research Methods*. Openpajan, Finland, 2004.

[KB08] Steiner Kvale and Svend Brinkmann. *InterViews: Learning the Craft of Qualitative Research Interviewing*. Sage Publications, 2008.

[KK99] Søren Kristiansen and Hanne Kathrine Krogstrup. *Deltagende observation - introduktion til en forskningsmetodik*. Hans Reitzels Forlag, first edition, 1999.

[LGP] URL: `http://www.opensource.org/licenses/lgpl-license.php`. Available at 22/07-2009.

[LT03] Josh Lerner and Jean Tirole. Some simple economics of open source. *The Journal of Industrial Economics*, 50(2), 2003.

[Maa04] Wolfgang Maass. Inside an open source software community: Empirical analysis on individual and group level. In *4th Workshop on Open Source Software Engineering at 26th International Conference on Software Engineering (ICSE04)*. University of St Gallen, 2004.

[MFHA02] Audris Mockus, Roy T Fielding, James D Herbsleb, and Development Apache. Two case studies of open source software, 2002.

[OSD] Linux at the movies - keynote at open source days, 2008. URL: http://www.opensourcedays.org/2008/agenda/sessions/GabriellePantera_and_RobinRowe.shtml. Available at 03/10-2008.

[OSIa] URL: http://www.opensource.org. Available at 06/03-2009.

[OSIb] URL: http://www.opensource.org/licenses/alphabetical. Available at 22/07-2009.

[OSIc] URL: http://www.opensource.org/approval. Available at 22/07-2009.

[Rai01] Eric S. Raimond. The magic cauldron. In *The Cathedral & the Bazaar - Musings on Linux and Open Source by an Accidental Revolutionary*, 2001.

[Sch04] Edgar H. Schein. *Organizational culture and leadership*. Jossey-Bass, third edition, 2004.

[Staa] Richard Stallman. Copyleft. URL: http://www.gnu.org/copyleft/copyleft.html, note = Available at 06/03-2009.

[Stab] Richard Stallman. Gnu manifesto. URL: http://www.gnu.org/gnu/manifesto.html, note = Available at 06/03-2009.

[Stac] Richard Stallman. Gnu public license. URL: http://news.cnet.com/French-parliament-dumping-Windows-for-Linux/2100-7344_3-6138372.html. Available at 27/07-2009.

[Stad] Richard Stallman. Gnu public license. URL: http://www.gnu.org/copyleft/copyleft/gpl.html. Available at 06/03-2009.

[tra] URL: http://en.wikipedia.org/wiki/Trademark. Available at 23/07-2009.

[Ubu09] http://www.ubuntu.com/community/ubuntustory. web-page, june 30th 2009.

[Yin94] Robert K. Yin. *Case study research: design and methods*. Sage Publications, second edition, 1994.

Appendix A

Interview Guide

General questions

- What is your name?
- What is your education?
- How many years of experience do you have working in software development?
  - How many years working with open source development?
- Where do you work? At home(country?)/Canonical?
- How long have you been working at Canonical?
- What is your position at Canonical? - department and primary function

Cooperation and management between Canonical, and the volunteers in the Ubuntu Community.

- How are the roadmap of Ubuntu made? What influence does Canonical have in this matter?
  - Could you give one or more examples of how suggestions or changes are evaluated before being accepted or rejected in the roadmap?
  - How is it decided which tasks are developed and which aren't?
- How are tasks estimated and prioritised after the roadmap is made?
  - Who are involved in this process?
- How are tasks distributed between the developers?
  - How is it decided which tasks the volunteers of the community should solve and which the Canonical employees should solve?
- How is it ensured that tasks are solved at the scheduled deadline?
  - How is it handled if a task can't make the deadline?
- What development model would you say is used in the development of Ubuntu?
- Can you think of any differences in the way employees and volunteers participates in the development?
- Have you experienced any problems in the collaboration with the volunteer community?
  - What was done to resolve those problems?
  - (Have you heard of any problems between employed developers and the community?)

Cooperation and management internally at Canonical

- How is the Ubuntu project structured in terms of organisational units?
  - How do Canonical employees fit into this organisation?
- Who do you cooperate with in Canonical and the community, and by which means?
  - How many people do you frequently communicate with? From which departments and on which topics?
- Please, give some examples of the daily decision process. What decisions do you make, and who else are involved?
  - Are there any general guidelines for decision processes?

Business model of Canonical

- How would you describe the business model of Canonical?
  - How does the volunteer community fit in this business model?
- How would you say that the business model affects your daily work?
  - Is the economic prospect of Canonical something you consider in your daily work
  - Have you experienced, that decisions in your daily work, are made to suit the business model?

(Er der specielle hensyn at tage til communitiet pga. Forretningsmodellen, og omvendt.)

- How does Canonicals collaboration with the Ubuntu community influence the business model?
- What influence do you think the business model has on the community?
- If any influence – could you give one or more examples?
- Can you think of any major differences or similarities in the business models of Canonicals collaboration with the Ubuntu community compared to other similar Open Source Projects?

Finishing questions

- Do you have anything further you would like to say?
- Do you have any questions about our study?