NTNU
Innovation and Creativity

A Survey of Industrial Involvement in Open Source

Andreas Røsdal

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Reidar Conradi, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The students should perform a literature study related to open source, and industrial involvement in open source. Based on this literature study, the students should create a survey in the context of the COSI project. This survey should aid the work of describing current OSS involvement, and OSS related processes.

Assignment given: 13. January 2006
Supervisor: Reidar Conradi, IDI

Abstract

This thesis presents results of an explorative survey of industrial involvement in open source. The survey is performed in collaboration with participants from the European ITEA project, COSI[1] (Co-development using inner & Open source in Software Intensive products).

The survey was performed to explore aspects of industrial involvement in open source, as industrial management of open source project, industrial use of open source components, related development, and communication processes, and industrial relationships to the open source community.

The survey is based on a study of open source literature, structured interviews of IT industry personnel, and a questionnaire. Three Norwegian companies participated in the interviews, which were used to explore different interesting issues, and to provide input and feedback to the questionnaire. The questionnaire collected answers from industrial partners in the ITEA COSI project.

Our research has an explorative research goal, and five research questions. Each research question sheds light on an interesting topic, and their answers will be presented accordingly. These answers are in form of both of qualitative and quantitative answers, but primarily descriptive information.

It has been found that companies participate in open source related development with the roles, open source owner, open source participant, inner source participant, and user of open source components. These roles are explained, together with the answers to the five research questions. We start with a presentation of the motivations behind use of open source components, and the selection processes used to find, and evaluate these components. Then, experiences from companies which have started their own open source projects will be discussed. Next, we describe the use of software development methodologies from open source, within companies. We present results related to the impact open source have on company internal development processes. Finally, industrial participation in open source projects by companies will be analyzed.

The answers to the research questions form parts of the contribution of this thesis. Some of these results are used to create guidelines for open source management, and open source component selection. We have also proposed twelve hypotheses, based on our findings. These hypotheses will work as a basis for future studies together with other results and our research design. Future studies may include distribution of the questionnaire to a larger and more representative sample, focus on fewer issues, and use of other research methods to shed light on these issues.

Keywords: Open Source, Open Source Development Practices, Inner Source Software Development, Open Source Component Selection, Industrial Open Source Management, Industrial Involvement in Open Source, Structured Interviews, Survey

[1]Project web site: `http://itea-cosi.org/`

Preface

The assignment for this thesis is given by the Department of Computer and Information Science (IDI) at the Norwegian University of Technology and Science (NTNU). The context of the work is within the European ITEA project, COSI (Co-development using inner & Open source in Software Intensive products). The COSI project wishes to investigate industrial involvement in open source, and industrial use of open source software, and open source development practices.

This thesis has been written during the spring of 2006, and it is part of the authors' MSc in Computer Science. Andreas Røsdal has completed TDT4900 Master Thesis, and Øyvind Hauge has completed TDT4735 Software Engineering, Depth Study.

Our work consist of a literature study, and a survey of industrial open source involvement, hence the title, *A Survey of Industrial Involvement in Open Source*. The survey has been performed with the assistance of the COSI partners. They have given us the chance to interview them, and they have responded to our questionnaire. The literature study, and the results from the survey, will be used as an input to deliverables in the COSI project.

We would like to thank our supervisors Professor Reidar Conradi and Dr. Carl-Fredrik Sørensen for their guidance and advice towards the completion of this thesis. We would also like to thank the partners in the Norwegian COSI project, for allowing us to work with such an interesting topic. In particular the people who welcomed us warmly at Keymind, Linpro, and eZ systems, especially Audun Jensvoll, Thomas Malt, Vidar Langseid, and Aleksander Farstad. We would also like to thank all the respondents who took the time answer the questionnaire, and ICT Norway for letting us use their survey tools.

Trondheim June 21, 2006

Øyvind Hauge   Andreas Røsdal

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction

The size and complexity of modern software is constantly increasing. Glass states that "for every 25% increase in problem complexity, there is a 100% increase in complexity of the software solution" (Glass 2002, 19). In most software, only a small part of the software is differentiating and adds extra value to the product, compared to the competitors' products. The remaining parts of the software are either infrastructure, or common to the domain. These parts of the software could be acquired elsewhere to reduce the challenges related to the increasing size and complexity. Another recent trend is that open source software (OSS) development and the use of OSS software have increased during the last couple of years. For example, the OSS project Apache has 66% of the market share for web servers (Raja and Barry 2005; Karels 2003). Creating a survey on industrial involvement in open source will shed light on this new and interesting area, which can bring many benefits to companies, and participants in OSS.

1.1 Rationale and Background

The ITEA COSI[1] project (Co-development using inner & Open source in Software Intensive products) has observed a shift in software development, to more cooperation, new coalitions, and more collaboration with open source communities. These forms of network-enabled collaborations create new challenges for software developers. COSI acknowledges the success of several open source products, but notices at the same time that the advantages and drawbacks of integrating OSS into software are not well understood (Lacotte 2004).

The overall goal of the COSI project is to understand how industry can benefit and learn from open source software and development practices. NTNU is part of the Norwegian COSI project, providing experience and knowledge, primarily related to empirical research in software engineering. NTNU has through its participation in the COSI project been given the responsibility, of leading some of the work, describing current industrial involvement in open source, and open source related processes. These baseline descriptions can be gathered from relevant literature and from the industrial partners' current processes.

Another observation made by the COSI project, is how companies can achieve more efficient software development. This can be achieved by using commodity OSS or by developing this commodity software in cooperation with others. Parts of solutions which do not give any competitive advantage can be replaced by OSS or by partner-developed software. This situation is presented in Figure 1.1 where cooperative development is shown in the middle area. The figure will be explained in detail, later in this thesis.

[1]Project web site: `http://itea-cosi.org/`

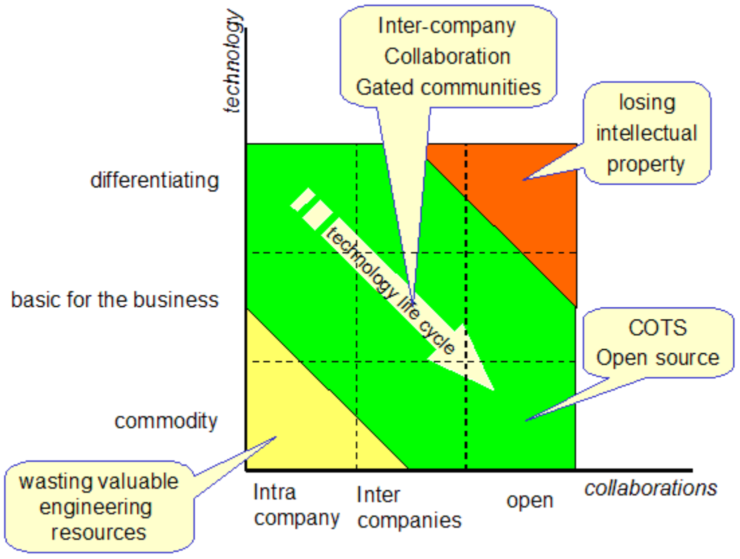


Figure 1.1: Efficient and effective development(van der Linden 2005)

To better understand the aspects of open source, described in the goals of the COSI project, it is necessary to perform a literature review, and create a research design. The research design should aid the work of describing the industrial partners' open source related processes. The distribution of the COSI partners over several countries, pose an extra challenge to the researchers. It is therefore important to benefit from NTNU's long traditions in doing empirical software engineering research.

The industrial partners participating in the COSI project have through their involvement in the project, agreed to participate in empirical studies to reach the goals of the COSI project. In particular, the Norwegian partners will be closely involved with the studies performed by NTNU. These partners include the companies Keymind, Linpro, and eZ systems. Keymind is a six person consultancy company, with the main business unit in Lillehammer. They are focused on developing, maintaining, and supporting IT health care solutions. eZ systems is a company with business units in several countries, with the main office in Skien. The company owns and manages their Open Source Content Management System, eZ publish. Linpro is probably the largest open source based consulting company in Norway, and is situated in Oslo. These three companies are participating in the COSI project because of their OSS related role. Further, there are a larger number of international partners participating in the COSI project. These are companies and academic institutions from a wide range of countries. These companies and the Norwegian partners will be described in more detail in Chapter 13.

1.2 The Purpose of this Thesis

The purpose of the research performed in this project, is described in terms of a project description, an overall research goal, and five research questions. This study explores use of open source software and practices in the industry, as described in the original project description.

> *The students should perform a literature study related to open source, and industrial involvement in open source. Based on this literature study, the students should create a survey in the context of the COSI project. This survey should aid the work of describing current OSS involvement, and OSS related processes.*

The project description contains two clear but relatively open ended tasks: doing a literature study, and preparing a survey. Since the COSI project is a project with industrial partners, we chose to focus on aspects of the literature related to industrial involvement in open source, and industrial utilization of open source. We further chose to base our research goal on our findings in the literature, as we will describe later.

While reviewing the open source literature, we found few descriptions of true empirical research. If the industry is going to learn from open source, more empirical research will be necessary. Because of the shortage in the literature, we chose to do an explorative study of industrial open source involvement. Based on our original project description and our findings in literature, we formed the overall research goal. We elaborated five research questions based on this goal. The elaboration of the research goal and the research questions is fully explained in Chapter 11, but we will also provide the research goal and the research questions here.

Research Goal:
Explore the following four roles: open source owner, open source participant, inner source participant, and user of open source components. Explore the relationships these roles have to the open source community, their development processes, their communication processes, and their motivations behind assuming these roles.

Research Questions:
Research Question 1: Why do companies use OSS components, and how do they find and evaluate these components?
Research Question 2: Why do companies choose to make their products Open Source, and how can this be done successfully?
Research Question 3: Why and how do companies use development methodologies from Open Source, in ISS development?
Research Question 4: How does the use of OSS influence the development processes of companies?
Research Question 5: How and why do companies participate in the development of OSS projects controlled by a community outside the company?

1.3 Our Contributions

The contributions of this work can be divided into four parts: the literature study, new knowledge, a basis for further research, and a reusable research design.

The first contribution is the *literature study*. This study constitutes the second part of this thesis. The results of this study will be used as an input to deliverables in the ITEA COSI project.

The second and most important contribution is *new knowledge*. This knowledge is first of all, in form of both qualitative and quantitative answers to our five research questions. We have also proposed twelve hypotheses based on our results. These hypotheses and our results are described in Chapter 16. The knowledge is applied in form of guidelines, for management of open source products, inner source projects, and selection and evaluation of open source components. These guidelines are enclosed as Appendix D.1, D.2, and D.3.

The third contribution is also important. While our work answers some questions, new and interesting questions can be formed based on this work. Our work base a *platform for future work* both for us, and the COSI project. Ideas for future work are provided in Section 18.2.

The fourth contribution will aid us and the COSI project in future work. We have developed a *reusable research design*, primarily in form of a questionnaire. This design can be developed further and reused. This will save us for a lot of work when we continue research in the COSI project. Our research design is described in Chapter 12, our interview guide is attached as Appendix C, and our questionnaire is available in Appendix B. The questionnaire is also implemented in a Web-based tool.

1.4 Content and Structure

This thesis consists of five parts distributed in a total of 18 chapters.

Part one, *Introduction*, is this part of the document and it contains one chapter.

Part two is the *Prestudy*. It contains the results from our literature review. Chapter 2 gives an overview and an introduction to the phenomenon, open source. This overview is followed by a short summary of the historical background of open source in Chapter 3. Chapter 4 discusses some characteristics of open source, and open source communities. These characteristics are followed by a presentation in Chapter 5 of some of the development practices used in open source software development. In Chapter 6, we present commercial use of open source, before we continue with software development using open source components, in Chapter 7. The second part is concluded by Chapter 8 and industrial involvement in open source where companies initiate, manage, or support open source projects.

Part three presents the *Research* of this thesis. We start the third part by presenting our research process in Chapter 9. This research process is followed by a description of our literature review, in Chapter 10. Based on the literature review and the original project description, we arrive at the research questions described in Chapter 11. In Chapter 12, we present our research design. This chapter includes both the rationale behind the chosen research design, and the final design. In Chapter 13, we present the context of our work, and the companies we visited. The two last chapters contain a description of how we collected data, Chapter 14, and how we analyzed these data, Chapter 15.

Part four presents the *Results and conclusions*. Chapter 16 contains results for each of the research questions. Each research question is given a dedicated section in that chapter. After the results have been presented, the results are summarized and discussed in Chapter 17. This discussion contains the most important results, a discussion of the validity of the results, as well as a critical evaluation of the goal achievements. Finally, Chapter 18 concludes this work with conclusions and suggestions for future work.

Part five contains appendices like the interview guide, the final version of the questionnaire, the open source definition, and our guidelines.

An overview of the content in this thesis can be found in Table 1.1.

Part	**Chapter**	**Content**
Introduction	1	Introduction
Prestudy	2	Overview of Open Source
	3	Historical Background
	4	Open Source Practices
	5	Software Development in OSS Communities
	6	Commercial Use of OSS
	7	Software Development with OSS
	8	Industrial Involvement in OSS Projects
Research	9	Research Process
	10	Literature Review
	11	Research Questions
	12	Research Design
	13	Research Objects and Context
	14	Data Collection
	15	Analysis
Results	16	Results
	17	Discussion
	18	Conclusion and Further Work
Appendices	A	The Open Source Definition
	B	Questionnaire
	C	Interview Guide
	D	Guidelines

Table 1.1: The content of this thesis

Part II

Prestudy

Chapter 2

An Overview of Open Source

The following chapters will present an overview of what *open source* is, its history and how the open source communities work, all according to the available literature. Writing such an overview is not a trivial task because of several issues. It is difficult because of the wide diversity in the open source community. SourceForge.net, `http://sourceforge.net/`, is probably the biggest open source repository online with close to 120 000 registered projects and more than 1.3 million registered users[1]. The projects vary from big successful projects with several hundred contributors to small one-man projects that are left to die. Secondly, open source is a fairly new concept. The term 'open source' is less than ten years old, thus open source an arena for research is still in its youth. Thirdly, most of the research has focused on relatively few, big and successful projects (Monteiro, Østerlie, Rolland, and Rørvik 2004).

Because of the big variety and the so far fairly limited, and perhaps un-nuanced, empirical research in this field, this presentation will only cover what is presented and commonly accepted in the literature. There have been written vast amounts of papers about open source, but fairly few researchers have done empirical work related to open source. Many publications are built on other researchers', limited, empirical work and treat idealistic, economic or other non software engineering issues. Because of the big scope of open source and the amount of papers written about open source, we have of course not been able to review all available literature. For a description of how we found our sources please see Chapter 9.

We believe that the world of open source is both bigger and more varied that what literature describes, but we will anyhow present our findings by starting with a definition of open source. Open source is in our eyes three things, it is the software, licensed with an open source license; a set of development practices used to develop this software; and it is the community around this software. When we use the term open source we mean the open source software, unless otherwise is specified.

This part of the thesis includes a definition of what open source software is followed by a short overview of the history of open source in Chapter 3, before we continue with other topics related to the community, open source licenses, and to the development practices in Chapter 4 and Chapter 5, industrial involvement in open source is discussed in Chapter 6, 7 and 8.

2.1 What Is Source Code?

Computers can only calculate with binary numbers, 0 and 1. In the childhood of computing all programs were written as 0s and 1s. This was a tedious work with many sources of errors. The programming languages have evolved, incorporated higher abstractions, and become more

[1]May 2006

similar to natural language, but the computer still uses binary numbers for all its calculations and operations. To be able to execute a program, normally written by a programmer in a high level programming language, the source code, has to be compiled by a compiler to an executable binary file. Source code is the text files, written by a programmer normally in a high level programming language, making the basis for a compiled binary file the computer can execute.

2.2 What Is Open Source Software?

There are several views of what Free Software (FS) , Open Source (OS), Open Source Software (OSS), Free Open Source Software (FOSS) or Free/Libre[2] Open Source Software (FLOSS) is.

The Free Software Foundation (FSF) and it's pioneer Richard Stallman have their Free Software Definition (Free Software Foundation 2006) which gives the requirements to a piece of software to be considered free software. Stallman and FSF had an idealistic mind when these requirements were written and it is important to underline that it is free as in freedom, not as in gratis or free beer (Free Software Foundation 2006). The definition refers to four kinds of freedom (Free Software Foundation 2006).

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

The Open Source Initiative (OSI) has on the other hand its Open Source Definition (opensource.org 2006b). The Open Source Definition was written as a pragmatic reaction to the misinterpretation of free software as "gratis software". It has its roots back to 1998 when it was originally drafted by Bruce Perens(Perens 1998). The Open Source Definition is currently available in version 1.9, see appendix A.

The definition contains the following 10 requirements to products which want to call themselves open source. The list below is partly adopted from Understanding Open Source Software by Mark H. Webbink (Webbink 2003).

- No royalty or other fee imposed upon redistribution.
- Availability of the source code.
- Right to create modifications and derivative works.
- May require modified versions to be distributed as the original version plus patches.
- No discrimination against persons or groups.
- No discrimination against fields of endeavour.
- All rights granted must flow through to/with redistributed versions.
- The license must not be specific to a product.

[2]Libre is Spanish for free, as in freedom

- The license must not restrict other software.
- The license must be technology-neutral.

Most of the popular open source licenses like, General Public License (GPL) and "Lesser" General Public License (LGPL), Mozilla Public License (MPL), MIT license, Apache Software License and New BSD license, all comply to these requirements. A full list of in all 58[3] licenses, can be found at `http://www.opensource.org/licenses/`.

Free Open Source Software (FOSS) and Free/Libre Open Source Software (FLOSS), try to unite the two terms Free Software and Open Software. Richard Stallman said "I recommend the term Free/Libre Open Source Software as a way they can mention both movements and give equal weight to both", in an interview with Indymedia at the School of Informatics, Edinburgh University, on 27th May 2004 (Indymedia 2004). The term FLOSS originates from the European Commission project, Free/Libre and Open Source Software Study, initiated by Rishab Aiyer Ghosh. The project can be found at `http://flossproject.org/`.

We do not want to address the disagreement between the pragmatic view on Open Source and the idealistic view on Free Software in this thesis. As a consequence we adapt the term FLOSS to cover the understanding of both the open source and the free software community. Throughout this thesis the terms Free Software, Open Source, and Open Source Software (OSS) could be used interchangeably and all of them should be understood as FLOSS, unless specific distinction between the terms is explicitly made.

[3]February 2006

Chapter 3

Historical Background of Open Source

This chapter will give a brief overview of the history of open source, some of its most important moments and products, and provide some references for further reading.

3.1 Hacker Culture

In the early computer age most of the software development was performed by scientists and engineers who found it normal to share and exchange research results as well as software (von Krogh and von Hippel 2003). It was hard to get a program to work and to be effective thus sharing was natural. Sharing allowed other scientists and engineers to improve, change, and play with code written by others. In 1953, the Project for the Advancement of Coding Techniques (PACT) was formed between Lockheed, Douglas and North American Aviation. It was probably the first code sharing initiative across company borders (Leonard 2000). The openness in this and other similar communities was an important part the hacker culture, making the foundation of the open source community we know today. A hacker is not a person who breaks security measures, but "someone who loves to program and enjoys being clever about it." (Stallman 1999, 53)

3.2 The Internet

One of the motivations behind what we today know as the Internet was to be able to share knowledge and programs between the participants (Tuomi 2003). In 1968, Pentagons Defence Advanced Research Project Agency (DARPA) started the Advanced Research Project Agency Network (ARPANET). In 1969, the first Request for Comment (RFC) was published, allowing interested and skilled people outside the core group to provide feedback in form of corrections or wishes. This was the start of a trend of openness in the history of the Internet.

During the seventies many of the ARPANETs major research centres wanted to get connected. Many of these organizations had different and incompatible hardware platforms, and this incompatibility lead to the development of the platform independent TCP/IP. TCP/IP later replaced the old communication protocols in ARPANET in 1983.

The Internet, built on TCP/IP, provided scientist and engineers a communication platform where they could cooperate over distances. The Internet also allowed them to share both research results and software, and it has served both researchers and open source enthusiasts as a necessary infrastructure. This infrastructure gives them a place where they can work and share their results. The open source movement would not have been what we know today without the Internet.

3.3 UNIX

During four weeks of the summer of 1969, Ken Thompson wrote the first version of an operating system called UNICS (Weber 2004), later renamed to UNIX. It is said that Thompson wrote UNIX because he had a personal itch, he wanted to play his favorite computer game, Space Travel, on a PDP-7 (Feller and Fitzgerald 2002).

Keeping UNIX clean and simple was seen as very important and it was constructed with the 'Keep it simple, stupid.' philosophy in mind (Raymond 2001). When they made the first manual for the operating system another important trend was started. A person was assigned to each subprogram as an owner, or responsible (Weber 2004). Having different owners in a growing system, lead to another important feature, modularity. It was important to build simple and well functioning systems that worked together.

Users in the community around UNIX started to create and provide software and tools that could run on the UNIX platform. One example is the Berkeley Software Distribution (BSD) which was made in 1977. The BSD distribution incorporated many Internet utilities like BIND and Sendmail (Feller and Fitzgerald 2002). These utilities are today part of the backbone of the Internet.

The BSD distribution of UNIX contained large parts of code written by AT&T. All the releases of BSD came with the code and because some of it was written by AT&T you needed an AT&T license to have legal access to it. When AT&T shifted towards a more commercial focus in 1984 the openness of BSD became a problem (Weber 2004). AT&T wanted to benefit from their rights to the distribution by charging a license fee.

In parallel to the development of UNIX, the development of the Internet and the TCP/IP protocol took place. A UNIX implementation of TCP/IP was developed at Berkeley and became part of the BSD distribution. As a result of AT&Ts commercial actions, Berkeley, re-released the TCP/IP stack in a release called Networking Release 1(Weber 2004). This was a true open source product with a liberal license, later called BSD-style license.

The popularity of this release led to the idea of rewriting all of the BSD-code resulting in an almost finished operating system, released as Networking Release 2. The system was released with the hope that someone else would finish it (Weber 2004).

The break-up between AT&T and Berkeley lead to a long legal battle which was not resolved until the early nineties. This again, made a lot of developers getting more ideological ideas around code sharing and free software. Many of these developers started working on other projects like for instance Linux.

3.4 GNU and the Free Software Foundation

In the early eighties, MIT licensed code created by employed hackers to a commercial company. This restricted access to the source code of that software, even to the ones who made it (von Hippel and von Krogh 2003).

Richard Stallman, a programmer working at the MIT Artificial Intelligence Laboratory, was stressed by this and in 1983 he announced the GNU project (Barahona, Quiros, and Bollinger 1999). He later wrote the GNU Manifesto explaining the goals of the GNU project and in 1985 he formed the Free Software Foundation. The Free Software Foundation was (and still is) an idealistic organisation supporting the free software movement, and especially the GNU project. Stallman wanted to use existing copyright law to guarantee some basic rights to all future users of software. Free, as in freedom not as in gratis, was one of the most important issues. With this in mind he formed the widely used General Public License (GPL). Under this license everyone should have permission to run, copy, modify and redistribute software and the code of software, but you cannot add any restrictions or limits to this freedom.

The Free Software Foundation has created several widely used programs like GNU Emacs, the GCC compiler, the GDB debugger and many others.

3.5 Linux

In 1991, just a short while after the release of the Networking Release 2, Linus Thorvalds started to work on the Linux operating system. Inspired by the Minix operating system, created at the Vrije University of Amsterdam by Andrew Tanenbaum, he wanted to create a UNIX-like operating system for PCs. Later the same year, he released the source code for the Linux kernel onto an Internet use-group (Weber 2004). Together with the code he left a note asking for help, comments or modules that could be added to the kernel. The kernel attracted high numbers of interested people who gave comments, fixed bugs, and in other ways participated in the community leading to a release of the 1.0 version in 1994.

The most important Linux feature was sociological. The new development model was unique. Linux was developed by huge numbers of volunteers coordinated through the Internet (Raymond 2001).

Linux is today one of the most successful open source projects ever, with a big share in the desktop, server and embedded market. An IDC forecast in 2004 reported a 26 % annual growth in the Linux market, including servers, PCs and packaged software over the next five years (Abramson 2004). Analytical companies and other, less serious sources, have also estimated the market share of Linux. It is probably impossible to find the exact answer, but the popularity of Linux is without doubt increasing.

3.6 Apache

The Apache HTTP Server is another of the most important developments in the open source history with a reported market share of close to 70% (Netcraft 2006). The development of the Apache server started after Rob McCool left the development team of the National Centre for Supercomputing Applications (NCSA) http server in 1995. The NCSA server was at the time the most popular http server, but when McCool left the development stopped.

Many webmasters had made their own patches and contributed to the server. Some of these webmasters started to organize via email to coordinate the distribution of the patches to the server and later formed the Apache Group (Fielding 1999). All modifications to the server are voted over by this group and the server's further development is thereby controlled by this group (Østerlie and Rolland 2003).

The Apache Group later became the Apache Software Foundation (ASF) `http://www.apache.org/`. The ASF is a non-profit corporation which oversee several important open source projects like the HTTP Server , ANT, Tomcat and several others.

3.7 Mozilla and Open Source

In January 1998, Netscape released the source code of their browser and named the project Mozilla (Feller and Fitzgerald 2002). Netscape was quickly loosing market share to Microsoft and Internet Explorer and they had to do something. Inspired by Eric Raymond's paper The Cathedral and the Bazaar(Raymond 2000), Netscape released the code open to everyone.

Within hours after the release, people around the world were submitting patches(Feller and Fitzgerald 2002).

The Free Software Foundation's free software was not immediately an industrial success and the term open source was seen as a replacement. The term open source came from a brainstorming session at Palo Alto, California related to Netscape's kick-off of the Mozilla project in 1998 (opensource.org 2006a). The definition of open source was later formed by Bruce Perens (Perens 1998), currently available in version 1.9 at http://opensource.org/ (opensource.org 2006b).

Open Source movement acknowledge the same licenses as the Free Software Foundation. The Open Source movement differentiates itself from the Free Software Foundation primarily in philosophical matters, emphasizing more on the practical benefits of such licenses instead of the moral issues (von Krogh and von Hippel 2003).

3.8 Summary

We have seen that the open source movement has its roots back to sixties when scientists and engineers shared results and software. The arrival of the Internet made sharing and coordination over distances easier. By using the Internet in their distributed 'Linux development model', the Linux community grew to produce one of the most successful open source products ever.

UNIX was a starting point for many of the later events in the history of open source. Stallman's idealistic Free Software Foundation wanted to make a free clone of UNIX. The term 'open source' came as a pragmatic reaction to 'free software' when Netscape as a commercial actor started to release source code through the Mozilla project.

Some of the literature paint a very idealistic and almost heroic picture of the open source communities, are these pictures really true? Are, other or maybe more nuanced stories told elsewhere in literature? With the entry of more and more companies and a growing focus in media it is likely that the open source movement and the communities are changing, but into what?

Chapter 4

Open Source Practices

There are many different open source projects and it is hard to state characteristics that are valid for all for them. We will anyhow present and discuss some general characteristics of open source communities and open source development in this chapter.

Open source development and is said to be many things. We get the following list of properties by combining qualities and characteristics from (Mockus, Fielding, and Herbsleb 2002, 310), (Feller and Fitzgerald 2002, 84) and (Weber 2004).

- OSS reduces the number of times the wheel is reinvented
- OSS development takes advantage of large numbers of volunteers by making it interesting to join and participate. Note that more and more companies are also participating in OSS projects.
- OSS includes involvement of both advanced users and highly talented and motivated developers.
- The OSS communities provide prompt feedback to user and developer contribution.
- OSS solves the developers' problems or, what Raymond called, personal itches.
- OSS utilizes truly independent peer review.
- OSS makes use of extremely rapid release schedules and solves problems in parallel whenever possible.
- In OSS development, work is not assigned; people undertake the work they choose to undertake.
- There is neither explicit system-level design, nor detailed design in OSS projects.

These statements are probably not true for all open source projects. (Monteiro, Østerlie, Rolland, and Rørvik 2004) state that there is a tendency to exaggerate the uniqueness of open source software development compared to traditional software development. This view is also supported by (Fuggetta 2003). The amount of quality research, done by different researchers on differing projects, we discovered through our literature study was unfortunately quite small. This supports the views of Monteiro et al. and Fugetta.

The first four statements, related to reuse, motivation, participants, and communication will be discussed in this chapter. In addition to discussing these characteristics, we will also present some of the most common open source licenses, some license related issues, and briefly discuss open standards. The last five statements, related to development processes, will be discussed in the next chapter.

4.1 OSS Avoids Reinventing the Wheel

It is quite obvious that open source reduces the need for reinventing the wheel. Open source allows developers and users to reuse solutions to problems in form of software and code.

Many open source products have been said to be highly innovative. It is easy to mention successful products like Linux, Apache, MySQL and other, but are they innovative? Open source has been accused for never inventing the wheel in the first place. The open source movement has been accused for primarily re-implement ideas created by others. Linux could for instance use UNIX as a template when it was first developed, and Apache was not the first web-server available on the Internet. Klincewicz surveyed 500 of the most active projects registered at Sourceforge.net and found that the technical newness or innovation in these projects was low (Klincewicz 2005). Questions have also been asked about innovation since it is normally a small number of people who write most of the code (Mockus, Fielding, and Herbsleb 2002). As a consequence of this, many OSS projects are the result of one, or a few highly motivated and inspired visionaries who lead the project (Divitini, Jaccheri, Monteiro, and Trœtteberg 2003).

4.2 Making It Interesting for People to Join

How and why people join open source projects will be discussed in this section. Motivations behind participation in open source communities are one of the few things where some empirical work has been done.

4.2.1 How Do People Join Open Source Projects?

Joining an open source community as a developer is like joining any other community. You have to seek it out, start being active in it and learn its rules to be accepted into it. To get to know a community, you could start browsing the web pages of the community, participate in forums and mailing lists, and read relevant documents. von Krogh et al. describe *joining scripts* which people follows to be accepted into a community (von Krogh, Spaeth, and Lakhani 2003). Following these joining scripts means to understand what the community expects of one's behavior as a newcomer and to participate in the technical discussions in forums and mailing lists. Newcomers can also be accepted on the background of code contributions, or what the authors call *feature gifts*, to the community. When accepted, a newcomer take on tasks primarily related to their prior knowledge or prior work.

Berquist and Ljungberg say that newcomers have to be socialized into the community's gift culture (Bergquist and Ljungberg 2001). Giving code as a gift to the community is something that has to be learned. In return for the gifts, the individuals will get a reputation and thus more respect and status. The reputation based structure, or meritocracy, of open source projects will be discussed further in chapter 5.

4.2.2 Why Do Individuals Participate in Open Source Projects?

Most open source communities consist of individuals but we will also briefly investigate company participation in open source communities in Chapter 6 and 8.

Developers in open source projects use their own resources, knowledge, and experience to create public value in what is called the "private-collective" innovation model (von Hippel and von Krogh 2003). The risk and costs are put on the individual while the benefit is given to the public. What makes people spend their spare-time working for others?

There are several explanations to this phenomenon, arguing that the participants get private benefits from their contribution. (von Hippel and von Krogh 2003) argue that the private benefit has to exceed the effort a developer puts into the work they do. (Lakhani and von Hippel 2002) show that the community members of the Apache community see that the learning benefits of providing online user-support to other users out-weights the work of doing it. Other benefits from working in open source could for instance be money, status, enjoyment, or affiliation with the community.

Several studies have been performed on this topic, for instance (Lakhani and Wolf 2005) on selected projects from SourceForge, (Hars and Ou 2002) on selected individuals from different open source forum, (Hertel, Niedner, and Herrmann 2003) on developers around the Linux kernel, and the Floss study led by Rishab Aiyer Ghosh for the European Commission (Ghosh 2002). These studies are performed within a small part of the open source community and the results may not be valid for the whole community, but they still give us valuable indications of the motives driving individuals who participate in open source projects.

The motivational factors shown in these studies are often divided into extrinsic and intrinsic motivations. *Extrinsic motivation* is when a person is rewarded or encouraged by something outside the person. This could for instance be getting a job through participation in an open source project, getting paid, solve a software problem or a personal itch, improve programming skills, or to get respect and status. Extrinsic motivation can be divided into immediate pay off and future rewards. *Intrinsic motivation* on the other hand is to do something for your own satisfaction or enjoyment. This could be things like being creative, acting according to community norms, or increasing the welfare of others.

(Hertel, Niedner, and Herrmann 2003) reported the desire to be identified with the Linux community, improve their own software and career, get status, support the Linux community, pure enjoyment, and the wish to improve Linux tools because of earlier time losses as the most important motivational factors. (Lakhani and Wolf 2005) found that enjoyment, being able to be creative together with the need for intellectual stimulation, and improving programming skills were the most important reasons for participation in open source. (Hars and Ou 2002) found reasons like enjoyment, altruism or idealism, identification with community, direct compensation or anticipated return, learning, showing their skills and personal need for product to be the most important reasons for participation. The FLOSS reported learning and knowledge sharing as the most important motivations (Ghosh 2002). A summary of these motivational factors is presented in Table 4.1.

Intrinsic Motivation	
	Need to feel competent
	Altruism: increase the welfare of others
	Community identification
	Enjoyment-based
Extrinsic Motivation	
Future rewards	Revenues from related products and services
	Increasing human capital learning and improving skills
	Self-Marketing or career advancement
	Peer recognition
Immediate payoffs	Salary or payment
	User-need for particular software

Table 4.1: Individual motivations for open source participation

The studies mentioned above, also report that about 20 % to 40 % of the responding developers get paid on a regular or irregular basis for their work within open source projects. Some get paid directly for their efforts while others use some of their day-job time working with OSS projects.

There are as presented above, several different factors behind participation in an open source project. It is not possible to say that one reason is more important than the others because the researchers highlighted different motivations as the most important ones. It is most likely that most participants are motivated by more than one factor as well.

4.3 Who Are Involved in Open Source Projects?

The OSS community is no longer consisting of hackers and ideologist only. The participation of companies is growing. (Dahlander and Magnusson 2005) describe how Nordic software companies take part in open source projects as part of their business strategy. There are growing numbers of paid developers participating in OSS projects, as seen in the previous section.

The studies by (Hertel, Niedner, and Herrmann 2003), (Lakhani and Wolf 2005), and (Hars and Ou 2002) may as mentioned not be representative for the whole open source community, but they have some interesting points. Almost all the respondents are men and most of them are between 20 and 40 years old with an average age around 30 years. A majority is reported to have a college degree or higher. Around 50 percent is working within the IT-industry. Other interesting groups were students and people who had programming as a hobby. The respondents were from several countries with a majority from North-America and Europe. Similar characteristics are described in the European Commissions FLOSS report from 2002 (Ghosh 2002).

By looking closer at the people involved in open source projects, we see that they are motivated, skilled, and many of them are also users of the application. This gives them good understanding of the problem domain. Many have an outside job (Mockus, Fielding, and Herbsleb 2002; Ghosh 2002), many are members of more than one project (Hars and Ou 2002), and some are paid to take part in the projects.

4.4 Communication in Open Source Communities

Informal hallway conversations, as well as organized and scheduled meetings, generally take place online in a publicly visible manner (Scacchi 2002). The open source communities are massive users of tools like chat, email, web-forums and mailing lists.

(Gutwin, Penner, and Schneider 2004) describe how simple text based communication tools, like mail lists and chat, are used to maintain group awareness. Awareness involves knowing who you work with and what they do. The authors emphasized that it was important to the members of the community to keep information public. Keeping the information open makes it easier to join a discussion since the history of the discussion is available online. Gutwin et. al. further reported that email was heavily used by the community members. Developer can use email and mailing lists to find out what is happening with a component, find the expert in an area, to get answers to their questions, to inform others of what they were doing, to get feedback on code (Jørgensen 2001), and to keep track of what the others are doing. The authors further emphasized the significance of overhearing discussions on mailing lists as a awareness tool.

The community members also used the mailing lists to get answers to questions. Posting a question was done assuming the right person would answer (Gutwin, Penner, and Schneider 2004). This allows the one who knows the answer, and who has time to respond, to answer it, and it prevents the top experts from being flooded with questions. In the Apache community user-to-user support is free and done by the users themselves. (Lakhani and von Hippel 2002) reported that one of the reasons why people responded so quickly in this community was because most of the time they already knew the answer to the question and answering it did not take much effort. This construct is of course dependent of a certain mass of people, and it could be fragile if nobody knew the answer to the question, or nobody bothered to answer it. In a study of the FreeBSD community, (Jørgensen 2001), reported that it could be hard to start a discussion and to get feedback on code-commits if you were not part of the inner circle of developers.

Email is used quite a lot, but on the other hand, (von Hippel and von Krogh 2003) noted that much of the communication in a community happens beyond public email. (Monteiro, Østerlie, Rolland, and Rørvik 2004) show that IRC is used to a greater extent than email in the Gentoo project. (Divitini, Jaccheri, Monteiro, and Trœtteberg 2003) take it further and argue that a lot of the political play actually happens privately.

Since community members mostly use informal communication tools it could be necessary to divide the community into groups. In the Apache web server, the core group is fairly small, and they are able to keep clean interfaces towards the other groups, thus making the groups independent. The small size of the groups, and the independence between them, allow informal communication and process to be used (Mockus, Fielding, and Herbsleb 2002). On the other hand, the Mozilla project has a bigger core team and higher dependency between the groups. Because of the group's size and the dependency they have to use more formal means to coordinate their work (Mockus, Fielding, and Herbsleb 2002).

Acknowledging that not all OSS projects are equal, is important, also when it comes to communication. (Crowston and Howison 2005) show that the, centralization of communications vary widely in open source development teams. Some projects were completely centralized, meaning that all communication, from everyone, was directed to just to one person. Other projects had a fairly high decentralization, meaning that the communication happened between several or many people.

4.5 Open Source Licenses

The normal purpose of copyright and licenses in software development is to protect creative work from others by making it illegal to copy or change a program. In open source, the purpose of licensing is "to deny anybody the right to *exclusively* exploit a work" (Laurent 2004, 4), or to preserve the user's freedom to do what she wants with the software. Open source licenses give the user freedom to use, change and redistribute the program. At the same time many of these licenses force the user to release any modifications under the same license.

There are a lot of different licenses in the open source community and some of the most popular ones will be presented below. All licenses presented can be found at OpenSource.org's home page at `http://www.opensource.org/licenses/index.php`.

4.5.1 Open Source License Types

GNU General Public License (GPL)

GPL is perhaps the most popular OSS license with more than 50 000[1] registered projects at SourceForge.net (SourceForge.net 2006) and more than 28 000 registered projects at freshmeat.net (freshmeat.net 2006). It was created by Richard Stallman and The Free Software

[1]All numbers are from February 2006

Foundation to be used with many of the FSF's projects like GNU Emacs, GCC and BASH. Even though it is a FSF license, it is certified by the Open Source Definition.

The GPL grants the user the four freedoms described in section 2.2 including the freedom to change and redistribute GPL software in any way she would like, as long as the modified versions and the source code are distributed under GPL as well. This viral behaviour is also called *copyleft*. The GNU project defines copyleft as '...a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well.' (GNU Project 2005)

GNU Lesser General Public License (LGPL)

Another popular OSS license, with more than 8 000 registered projects (SourceForge.net 2006), is the LGPL. LGPL was also created by FSF and Stallman, and it is a license meant to be used with libraries. It is not as restrictive as the GPL. LGPL software can be used with other free or proprietary software as a linked library. But if the original software is modified or extended, the derived versions have to use LGPL as well.

Stallman recommends using GPL to give free software developers an advantage over proprietary developers. By using GPL, every derivative product must be distributed freely as well (Feller and Fitzgerald 2002, 20).

Berkeley Software Distribution (BSD) and the MIT License

There are more than, 5000 projects registered with the BSD license, and more than 1300 registered projects with the MIT license at SourceForge.net (SourceForge.net 2006). Both licenses are more liberal than the GLP license. The BSD license has three requirements to modified versions of software (University of California 1999). It is equivalent to the MIT license (Massachusetts Institute of Technology 2006) except from the last no-endorsement clause.

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the original authors nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The Artistic License

The Artistic License was designed to be used specifically with Perl, and have an emphasis on the rights of the copyright holder. It is because of this perhaps not the most suitable license for free software development (Laurent 2004). It is still used by more than 1300 projects at SourceForge (SourceForge.net 2006).

Mozilla Public License (MPL)

MPL was first created when Netscape was released as a free and open product in 1998. MPL allows code with MPL to be combined with code licensed under another license. This is prohibited by GPL which requires all code to be GLP code. BSD on the other hand permits BSD code to be combined with code under other licenses. Code copied or changed under the MPL is required to stay under the MPL.

MPL has also worked as a basis of other licenses like the Apple Public License, the Nokia Open Source License, and the Sun Public License.

Apache Software License

The Apache license is similar to both the MIT and the BSD license allowing distribution and modification with few restrictions. It allows redistribution under other licenses as long as the, restrictions in the Apache license is still complied.

Public Domain

Public domain is not actually a license. It refers to artistic works without an owner or with no legal restrictions to its use. Or as Rosen said, "It is simply “free”, as in the terms “free beer” and “free speech”." (Rosen 2002) It is quite popular at SourceForge with close to 2000 projects registered with this license or more correctly *lack* of license (SourceForge.net 2006).

4.5.2 Challenges with Licensing in Open Source

There are several challenges related to open source licenses. First of all, there are many of them. The number of licenses makes it hard to have a complete overview of all of them, when they can be used, what they permit, what they forbid, and how they can be combined. Some licenses require that the same license is used with all new or extended versions of the software. All this may lead to fear, uncertainty and doubt among the users of open source software. Companies in particular need to be careful not to make any mistakes using code or components licensed with an open source license in their proprietary products, since most open source licenses forbids this.

This leads to another challenge, enforcing the licenses. Since most of the software is written by people without their own legal department, it could be hard for them to enforce their rights to their own intellectual property. There have been examples of law suits where owners of open source products have sued, and won over software vendors misusing their software (Gonzalez 2005).

A third challenge is patents and patenting. Since licenses only cover copyright aspects and not patents, open source software could be vulnerable to patenting. If someone outside the open source community patents software constructs used in the open source software, the community could be forced to remove these parts of their software. This again would make the license worthless (Gonzalez 2005). It is therefore important to keep software and the ways of constructing it open and available to the open source community, and perhaps even patent ideas to be able to share them.

4.6 Open Standards

Licenses, intellectual property and patents are very important issues related to open source. Open standards have high relevance and importance to open source as well. Open standards are also important to preserve the user’s freedom and independence.

Sutor defines a standard as ’a blueprint or a set of plans that can be implemented’ (Sutor 2006). He further states that high transparency, high community involvement, democratic the creation processes, low costs for software developers and customers related to the standard, and high degree of freedom are characteristics of a true open standard. The more restrictive a standard is the less open it is.

(Perens 2006) says that the following characteristics must be met for an open standard: It must be available and it must be free, free as in freedom. The standard cannot discriminate anyone or anything. It must be allowed to extend the standard. It is at the same time, necessary with licenses that protect the standard from sub-versioning with purpose of making everyone dependent of the new sub-versions.

Having an open standard can allow anyone to implement them, thus increase compatibility between different providers and platforms giving both the customer and the developer greater freedom.

4.7 Summary

In this chapter we have looked at some particularly interesting aspects of the open source community, who, how and why people do participate in OSS communities, how they communicate in OSS communities, and some open source licensing practices. Individuals who participate in open source projects are motivated by both intrinsic and extrinsic motivational factors, and many developers also get paid by their employer to contribute to the open source project.

These individuals, participating in open source development, use simple informal text-base communication tools like email, web-forums, and chat. These communication tools are used together with other tools, like CVS and Bugzilla, to support and co-ordinate the development process.

4.8 Challenges

When a company participates in open source, what is it motivated by? We will look a bit at company motivations in chapter 6. Do for instance companies value the possible learning effects from open source participation? The developers employed in a company what do they get from participating in an open source project through their day time job? How many of these employees participating in an open source project, is actually getting paid to do so? How do companies get involved in OSS communities and which companies participate? The questions related to company participation in OSS are many.

When companies participate in open source projects do they use the same tools as used in the communities, and do they participate actively in other community activities, like answering to forum postings and emails? How companies communicate with open source communities is an interesting issue.

Another interesting issue is open source licenses. These are made to secure the user's freedoms. When companies use open source components are they aware of, are they able to differentiate, and do they respect, the licenses used in open source projects?

Chapter 5

Software Development in Open Source Communities

The software development practices used in OSS projects have some similarities and some differences with commercial software development. In this section, the characteristics of software development typically used in open source projects will be described. Since open source communities are so diverse, it is impossible to describe a single common practice which applies to all OSS projects. Therefore, the most important aspects which are common to typical successful OSS projects will be reported. Further, the development process will be described in the perspective as seen though the COSI-project: how the software development practices in open source communities are relevant to the Norwegian software industry. The use of typical OSS development practices inside companies will be described in the coming section as inner source development.

5.1 Organization of Open Source Projects

Most open source projects are organized by either, individuals, groups, organizations or companies. Most of the large open source projects like The Apache Web Server, BIND, FreeBSD, Sendmail and Eclipse are organized by organizations like Apache Software Foundation, Free Software Foundation, Eclipse Foundation Inc, Python Software Foundation and others (Feller and Fitzgerald 2002). Many of the smaller projects are organized by individuals using, for instance, SourceForge or other free hosting services as a home for their projects. Last but not least more and more projects are organized by companies using open source as part of their business strategy.

A study of two open source projects, Mozilla and FreeBSD, by (Holck and Jorgensen 2005), shows that most work in those projects are performed as one-man projects, where a single developer works on implementing a particular feature. With closed source it is only the developers in the company who have access to the code and it is only they who can read it, understand it, find errors in it and correct it. They will most likely get complaints from customers when something is wrong, but the developers do not get any further help from the customers finding the defect. With open source, everyone who is skilled enough can read and understand the code, find errors and eventually fix them. As Eric Raymond famously wrote, "Given enough eyeballs, all bugs are shallow" (Raymond 2000), meaning that when many developers read and try to understand the code they will discover and fix all bugs.

5.2 Software Development Process

The following is an overview of some of the most important characteristics of the development process which will be described subsequently:

- Contributions are made from many volunteer developers which implement new functionality according to their skills and interests.
- OSS development has a unique contribution process. Typically developers have control over particular modules in the system, and have responsibility for reviewing changes to those modules.
- Many phases of the process are performed less formally, and are instead implied activities.
- Developers are often globally distributed, which means that development must be coordinated well.
- The OSS development process has similarities to agile development.
- Development occurs asynchronously between developers, often though the use of a collaboration infrastructure which includes mailing lists, source code repositories and bug-reporting systems. Further, development is parallel and concurrent, with less clearly defined phases than a traditional iterative or waterfall model.
- Quality assurance is handled in the development process though peer review, a specialized release process and early testing by users.

A study by (Potsar and Chang 2004) describes the open source development model as an evolutionary model, which changes over time according to the interests of the involved people. Development is performed by many developers concurrently. This is quite unlike commercial software development, which often is structured more in phases. Coordination of developers is therefore essential in order to develop software while working on the same source code while not being present physically at the same place.

Agile methods	**Open Source development**	**Plan-driven methods**
Agile, knowledgeable, collocated and collaborative	Agile, globally distributed, knowledgeable and collaborative	Plan-oriented, adequate skills, access to external knowledge
Small teams and products	Large teams, small products	Large teams and products
Inexpensive refactoring	Inexpensive refactoring	Expensive refactoring
Designed for current requirements	Open, designed for current requirements	Designed for current and new requirements

Table 5.1: OSS development method compared to other methods, according to (Warsta and Abrahamsson 2003).

Warsta et al. has compared open source development with other development methods, and found that it was similar in many ways to agile development. Agile software development has the four following characteristics (Warsta and Abrahamsson 2003): Primarily the development method is incremental, with small releases which are available frequently. Next, it is cooperative, where developers and customers cooperate closely during development. Further, the development method is straightforward, meaning it is lightweight and easy to use. Finally, it is able to adapt to changes easily. According to (Warsta and Abrahamsson 2003), OSS shares these characteristics to a large extent. Table 5.1 compares the open source development method with agile and traditional plan-driven methods.

5.2.1 Roles in Open Source Communities

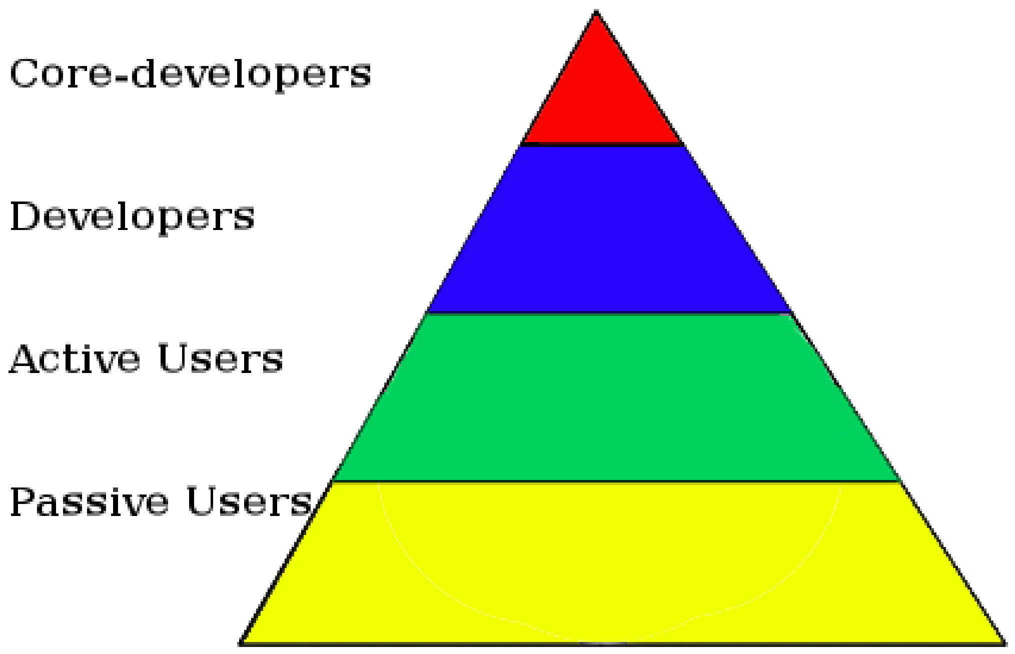


Figure 5.1: The layered structure of open source projects

For open source projects to be successful it is important to have a base of developers, testers, users and interest around the project. Projects which do not manage to create such a critical mass, will get in to problems and most likely close.

In a big community, do everyone contribute equally? Open source communities are often described to have hierarchical or onion like structure with several layers (Crowston, Annabi, Howison, and Masango 2004; Mockus, Fielding, and Herbsleb 2002; Scacchi 2004). The teams consists of a core with relatively few developers, up to 10-15, who contribute close to 80 % of the code (Mockus, Fielding, and Herbsleb 2002), co-developers, bug-fixers, bug-reporters and users or free riders. This leaves the OSS projects with a pyramid like staffing, where the group below is larger by an order of magnitude than the one above. On the top we have one or few project owners, together with a small group of core developers, then a bigger group of developers and bug-fixers, yet a bigger group of active users who participate in the community reporting bugs, reading forums, news and so on, and at the bottom we have free riders, or normal users as in figure 5.1.

Many open source projects are organized as meritocracies, for example the Apache project. This means that the developers who have contributed the most valuable and high-quality code also are the ones who make the most important decisions. As in the example of the Apache project, new members of the project are added formally when they are nominated by an existing member and receives a unanimous vote (Fielding 1999).

The OSS development process is made possible though the cooperation of a number of people. Figure 5.2 shows the most important roles which are involved in this process, based on a study of two open source projects by (Holck and Jorgensen 2005). OSS is typically developed

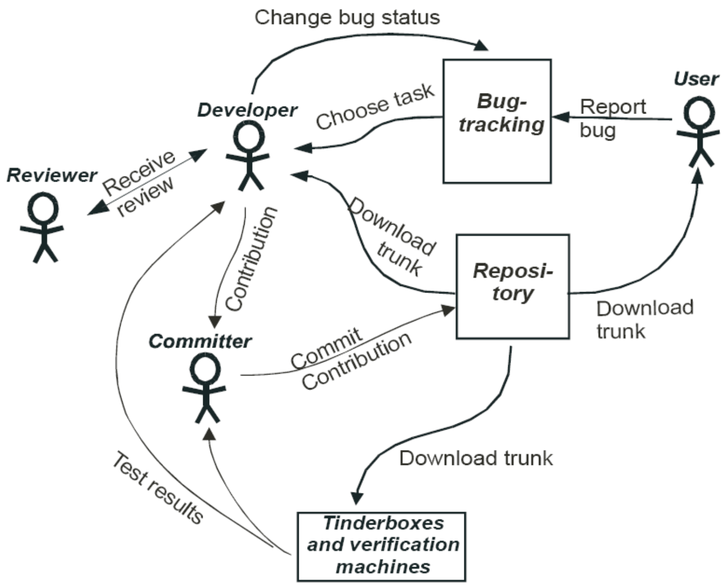


Figure 5.2: Overview of the open source development process(Holck and Jorgensen 2005).

though an interaction between developers, users, reviewers and committers. These people rely on a technological infrastructure to cooperate, manage the source code, reported defects and testing.

Developers, is anyone who contributes code to the project, and can be volunteers or employed by a company managing the project. Volunteers can typically choose which task they want to work on, based on skills and interests. The interaction between a developer and a user is important in this model. There is a cycle where developers contribute code to the software, which then is committed to the source code repository, and tested by users (Holck and Jorgensen 2005). If the user finds any faults, this is reported back to the bug-tracking system.

It is common for open source projects to practice module ownership. This means that a developer has the responsibility for accepting new code to a particular module. The roles reviewer and committer are central to the quality assurance in OSS projects. The reviewer must check the code from developers, and decide if it is of good enough quality. This process can take several iterations before the code is accepted by the reviewer. The committer is the person responsible for adding or changing the source code repository. By having a different person reviewing and committing code written by another developer, this is hoped to increase the quality of the code. However, it also results in additional work, so it is not practiced in all OSS projects. (Holck and Jorgensen 2005)

5.2.2 Project Start-up

An open source project is normally formed by an individual or a small group having an idea or a problem they want to solve (von Krogh and von Hippel 2003), or as put by Raymond "Every good work of software starts by scratching a developer's personal itch" (Raymond 2000). Stallman on the other hand, says that new software is planned; it is not impulsive creations (Stallman 1999, 64).

An early version of the product is made freely available on the internet together with the source code and mailing lists (von Hippel and von Krogh 2003). People can download the code, play with it, make modifications to it and post ideas and code-contributions on the mailing list and the project website. The contributions can be reviewed and used by anyone interested. If the value of the contribution is high it can be included in the project code.

The project code is normally guarded by trusted developers who function as gatekeepers (von Krogh, Spaeth, and Lakhani 2003). For instance, the Apache server uses a voting mechanism to decide which code is to be integrated into the main release (Mockus, Fielding, and Herbsleb 2002), others may have a person responsible for a module or a piece of code.

5.2.3 Requirements Specification

Creating a requirement specification is typically an implied activity in a OSS project. Many OSS projects are also designing software where the requirements are already clear, because there is often software which provides the same functionality, according to a study by (Scacchi 2004). The study showed that the requirements where created as a by-product of discussions between developers. These discussions took place on mailing lists and other communications forms, but were never formally described in documents. In fact, (Potsar and Chang 2004) states that the requirements are already known by open source developers before they start an OSS project. This is because many OSS projects are started because individual developers discover areas where software can improve a situation, and develop a solution themselves. This is quite unlike most commercial software projects, which have to interview users and customers in order to discover the requirements.

5.2.4 Software Design and Implementation

The design phase of OSS is often performed in parallel with the implementation. This means that there is often no formal software and architecture design documentation, which can lead to a system which is difficult to maintain because the implementation diverges from the design. (Potsar and Chang 2004)

OSS is often implemented trough a process called continuous integration (Holck and Jorgensen 2005), which means that developers integrate new functionality into the system at the same time. The free access to the source code has an important impact on the development process, and is one of the critical factors which distinguish it from traditional commercial software development. Software for version control is therefore critical in order to manage and coordinate the development when multiple users and developers have the opportunity to modify the source code. The tools CVS and Subversion are commonly used for this purpose by OSS projects. According to (Scacchi 2004) the version control system coordinates development, and manages which additions to the software that should be made. It also prevents conflicting modifications to an extent.

One can question how OSS projects are able to successfully create software without a clear design phase. The following three reasons are proposed by (Narduzzo and Rossi 2004):

- The design of many OSS systems are based on reusing the architecture of existing modular software. This enables fast development and the possibility to reuse other code, in addition to avoiding unanticipated design-problems to occur because of using a new design.

- When OSS projects develop new software where there is no design to reuse, the architecture evolves from a draft into a useful modular architecture over time.

- Since the source code is available and code is not owned by any entity, developers can systematically improve multiple modules in a system in order to improve the architecture.

5.2.5 OSS Release Process

A central part of the open source development process, and how it ensures quality, is the release process. Frequent releases are made in order for users to get access to new functionality as early as possible, and defects are reported by the users back to the developers (Hang, Hohensohn, Mayr, and Wieland 2004). Figure 5.3 shows how a release process in the open source project Mozilla is organized. There are two branches of development. The topmost branch is the stable branch for version 1.0, where no new functionality is added and only defects are removed. The development branch at the bottom of the figure is where new functionality is added. Before a major release is made, there is a period of feature freeze, when no new functionality can be added. This is to ensure that the software receives sufficient testing before it is released.

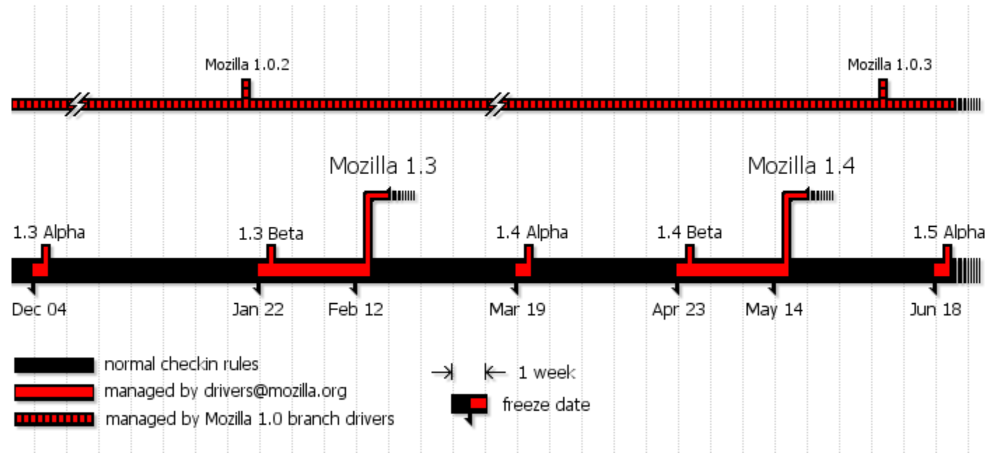


Figure 5.3: The Mozilla development roadmap showing mayor release dates for two branches.(Mozilla OSS project 2002)

5.2.6 Tool Support

(Gutwin, Penner, and Schneider 2004) describe that many communication tools are used by developers in OSS projects. This is a necessity because, people are often distributed globally, and they are often working in parallel. Asynchronous communication tools such as email, mailing lists, forums and web pages are used, as well as synchronous tools such as chat.

Further, there is a wide range of software engineering tools used by OSS projects. This includes software modeling tools for creating UML diagrams, such as Dia. Defect-tracking systems such as Bugzilla are also used, which is a online system to coordinate information about defects, suggestions and support-requests from users.

For software configuration management, (Scacchi 2004) states that tools such as CVS and Subversion, are used in OSS projects. Such tools are used to coordinate development between several people, and are also used to mediate control over which code changes that are accepted.

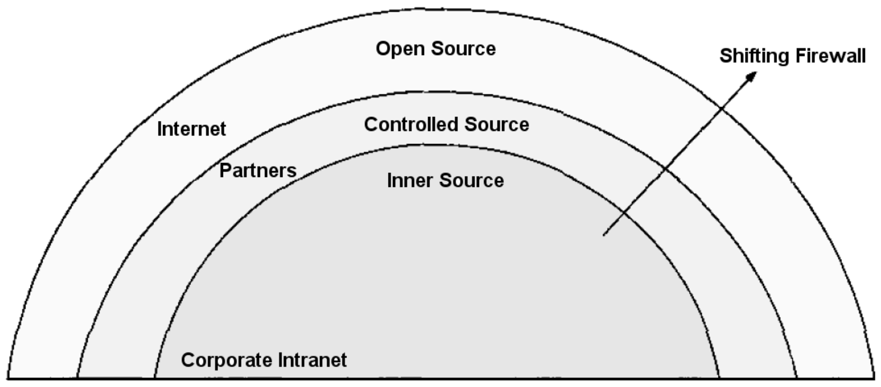


Figure 5.4: Relationship between Open and Inner Source(Dinkelacker, Garg, Miller, and Nelson 2002)

.

5.3 Inner Source Development

Inner source development (ISD) uses aspects of open source software engineering inside a business unit, company, or consortium. The software developed is only open and available to a closed consortium of partners. This development methodology has been described by the COSI project, which characterizes the development as an "open, concurrent model of collaboration" where the source code is made available to the participants (van der Linden 2005). It can be thought of as a special case of an open source community, where the participants are involved based on their role in the company or business unit.

ISD has for instance been used in a project at Hewlett-Packard, where this form of development was used to develop and maintain approximately two dozen software systems (Dinkelacker, Garg, Miller, and Nelson 2002). Another case study of using open source development methodologies inside a company has been performed by (Gurbani, Garvert, and Herbsleb 2006). This case study has shown that companies can benefit from using open source development practices, and suggested specific success criteria for inner source development.

The experiences from these studies will make the foundation for describing the advantages and challenges of ISD. However, there are few other experiences with using inner source development in real projects. This is therefore an interesting topic for the survey which will be performed as part of this thesis.

Figure 5.4 shows the relationship between open and inner source. The figure suggests that inner source is shared on the corporate intranet, controlled source is shared with partners, and open source is shared with anyone on the Internet.

5.3.1 Characteristics of Inner Source Development

Sharing of the program source code is an important characteristic of ISD, which reduce the amount of code which is reinvented by several different people in the same company. A common repository of source code can be reused in several projects, which improves code reliability and quality (Dinkelacker, Garg, Miller, and Nelson 2002).

Distribution of development teams in different organizations and geographic locations can be made easier by using ISD (van der Linden 2005). Further, by using inner source it is easier for a company to start, or change collaborations with other development teams at different departments or companies. This is because developers, who already have participated in ISD inside as a partner, will already be familiar with the source code repository, development tools and coding standards (Dinkelacker, Garg, Miller, and Nelson 2002).

Shared, community debugging is another characteristic of ISD. It has been reported to increase code quality, because of the large number of testers and because the developers reputation is at stake (Dinkelacker, Garg, Miller, and Nelson 2002).

The following success criteria for inner source projects have been suggested by (Gurbani, Garvert, and Herbsleb 2006):

- Technologies are shared between several product groups.
- Software requirements are not well understood.
- Different product groups customize software to their needs.
- The software has a modular architecture, so it will be possible to merge changes from several contributors.

5.3.2 Inner Source Development Practices

The following development practices are common OSS development practices used inside a company using ISD (Dinkelacker, Garg, Miller, and Nelson 2002; van der Linden 2005):

- Developers can choose the task that they are working on in the source code shared as inner source. This can be a challenge to organize, because of the responsibility this puts on the developers.
- Developers own and control modules in the software. The owners of different modules can be located at different locations.
- Early and frequent releases of the software, with source code included. This allows developers to get fast feedback on new releases.

5.4 Summary

Chapter 5 has been about several aspects of software development in open source communities. First the organization of typical open source projects was discussed. It was shown that open source projects are organized either by individuals, groups, organizations or companies, and that most projects are one-man projects.

Secondly the development process used in open source projects has been explained in detail, covering all phases and relevant roles in the process. The development process was claimed by literature to be globally distributed, performed in parallel, and with many similarities with agile development. However, it should be noted that open source projects are very diverse, so the open source development process described here is not used in all projects.

Finally inner source development was presented as a way for companies to use aspects of open source software engineering inside a company.

It has been shown that software development in open source communities is quite unlike traditional commercial software development. Based on the research described in this chapter, there are several topics which can be investigated further and become the basis for a research question. How can companies use and benefit from the development processes described in this chapter? Very little information has been published about inner source development. This is therefore an attractive area to be studied further, if the appropriate study objects are available. "Architecture, development model, license models, business models, tool support, and infrastructure are strongly related." (Deng, Seifert, and Vogel 2003) How does the use of open source development processes, tools, software, and source code have an impact on companies using such methodologies?

Chapter 6

Commercial Use of OSS

Several studies shows that commercial companies are increasingly using OSS as an important part of their software and interacting with the open source community as part of this process(Karels 2003; Rossi and Bonaccorsi 2005; van der Linden 2005). There are many challenges related to using OSS for commercial purposes, so choosing the correct strategy for using such software is important. The payoffs can be considerable if used properly. This chapter will focus on the commercial aspects of using OSS in a software company which develops and uses software.

6.1 Open Source Business Models

To make profit on OSS is inherently difficult because the software itself is available at no cost. There are some business models which have been used successfully, most of which can be grouped in the two categories Distributors and Retailers, and OSS-related services, as illustrated in Figure 6.1, which is based on a survey by (Ghosh 2002).

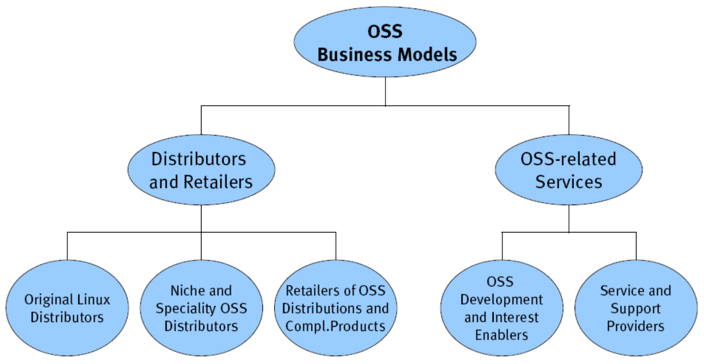

OSS Business Models
Distributors and Retailers
OSS-related Services
Original Linux Distributors
Niche and Speciality OSS Distributors
Retailers of OSS Distributions and Compl.Products
OSS Development and Interest Enablers
Service and Support Providers

Figure 6.1: Open Source business models (Ghosh 2002).

The business model used by most Linux distributors is based on software integration. These companies operate by interacting with the many open source communities, collecting software and package the software as a product (Karels 2003). This means that several OSS software packages, libraries and components are integrated as a complete and usable product. This has the advantage that the company does not have to develop a complete operating system.

OSS is often only available for download on the Internet, so these companies can earn money by distributing the software on CD-ROM. However, since the software is available freely, convincing customers to purchase the software is difficult, and therefore the margins can be low. Brand building and advertising is therefore important. There are many companies based on this model which sells the operating system Linux and FreeBSD. Red Hat Software is an example of a successful company using this business model, developing and distributing the Linux operating system (Ghosh 2002).

The business model called "Niche and speciality OSS distributors" in figure 6.1 have a business model based on developing specialized software in cooperation with an open source community. According to the study, the most typical product from companies using this business model is specialized software such as applications, development and administrative tools. The cooperation is often in a form of symbiotic relationship as described in Section 6.3. This business model is used by for example ActiveState, which cooperates with the communities surrounding the programming languages Perl and Python, selling additional services.(Ghosh 2002)

The business model labeled "Retailers of OSS distributions and complementary products", is based on selling the software from OSS distributions, in addition to documentation and other related products to the mass market. Linuxbutikken is an example of a business using this model in Norway (Ghosh 2002).

Providing support for customers and users of OSS is also a common business model. While many open source communities provide a form of support through mailing-lists, companies often depend on more reliable support provided as a service. Therefore the companies involved in the development of OSS also have a great deal of knowledge about their products, and can use this knowledge to provide customers with support (Karels 2003).

6.2 Commercial Utilization of OSS

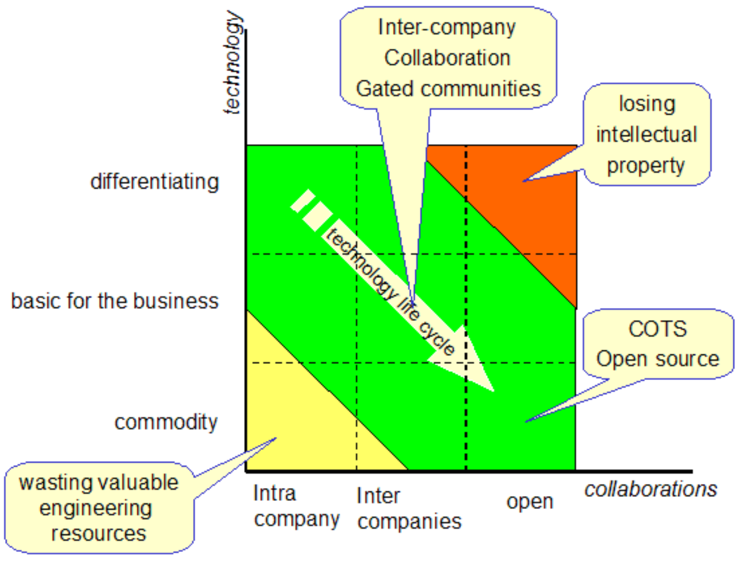


Figure 6.2: Developing software internally versus acquired from OSS, according to the COSI project (van der Linden 2005).

Companies manufacturing software develop and use software from a variety of sources. The software which is developed internally in a company is an important part of the total software package. This software is important for the competitive advantage that a company has over its competitors. The *differentiating* software developed internally at the company is what differentiates it from the products of other companies. However, a company can also use software which is developed by other sources, such as commercial-off-the-shelf components and open source components. Such components are often used by many companies, and are considered to be commodity, accessible by several companies. If the software is acquired from other sources and is used by companies in the same business area it is referred to as *basic for business*, while if the software is commonly used by all businesses then it is referred to as *commodity*. The COSI project depicts this trade-off between using differentiating and commodity software as shown in Figure 6.2. If a company is developing software internally which it could have acquired through other sources, it is inefficient because the software could have been acquired at a lower cost than developing it inside the company. On the other hand, if a company publicly releases differentiating software which is developed internally, then the company would loose money because other companies also get access to this software. The figure suggests that for a company to efficiently develop software, it must develop differentiating software internally, and acquire commodity software from other sources, such as OSS or COTS (van der Linden 2005).

(Behlendorf 1999) states that software can be classified according to a spectrum where infrastructure is at one end of the spectrum, and end-user applications at the other. According to the spectrum, most OSS is likely to be classified as infrastructure, while commercial software is classified towards the end-user end of the spectrum. This is in accordance with the figure created by COSI shown in Figure 6.2, which recommends that commodity software should be open source, while differentiating software should be developed internally in the company. Acquiring software components from other sources than those developed "in-house" often puts

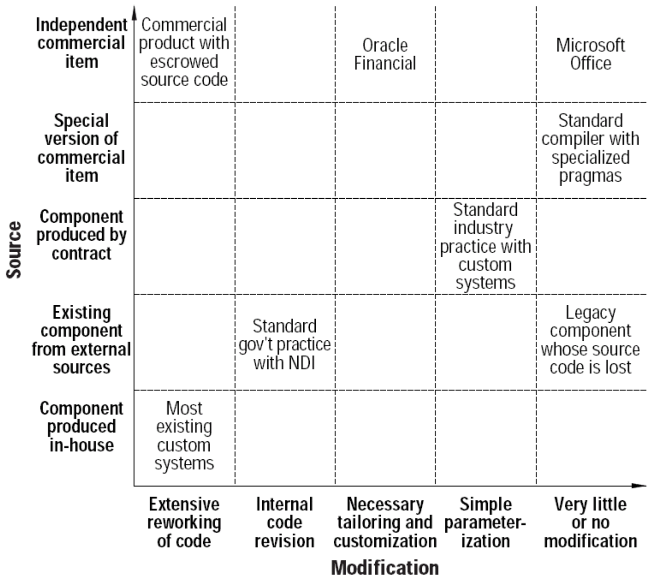

Independent commercial item
Special version of commercial item
Component produced by contract
Existing component from external sources
Component produced in-house
Source
Commercial product with escrowed source code
Oracle Financial
Microsoft Office
Standard compiler with specialized pragmas
Standard industry practice with custom systems
Standard gov't practice with NDI
Legacy component whose source code is lost
Most existing custom systems
Extensive reworking of code
Internal code revision
Necessary tailoring and customization
Simple parameter-ization
Very little or no modification
Modification

Figure 6.3: Classification of software according to source and modifiability(Carney and Long 2000).

restrictions on the amount of changes a company can make to those components. (Carney and Long 2000) illustrate the extent to which a software component can be modified as shown in Figure 6.3. Software developed in-house can be modified freely, compared to software developed externally by contract or by another company where there are fewer possibilities for making modifications. In this context, OSS is developed by a source other than the company, or in some form of cooperation with the company. As a result, the company has access to modify the software freely, depending on which open source license is used.

6.2.1 Motivations of Companies Using OSS

There are many reasons why a commercial company would be interested in using OSS, in addition to the fact that the software is available at almost no cost. Rossi et al. has performed a survey (Rossi and Bonaccorsi 2005) on the motives for Italian companies using OSS. The result of the survey was that two categories of motivation were observed within the companies. One type of motivation is extrinsic, which is an external factor which motivates the activity, such as profit. The other motivation is intrinsic, meaning that the pleasure of performing the activity itself is the cause of the motivation, such as performing creative and mentally challenging activities such as software development. The survey showed that extrinsic motives did have high importance for the companies. The extrinsic motives will be discussed first (Rossi and Bonaccorsi 2005).

Reduced time to market for a product is an important commercial motive for a company to use OSS. By re-using existing software, this means that the company will not have to pay the price of developing this software internally. This means that the company will get access to new technology and features more quickly (Goldman and Gabriel 2005). Further, OSS contributes to innovation and new technology even for small companies (Rossi and Bonaccorsi 2005).

The open source community can contribute to maintenance, improvements and support for the software. However this is not necessarily true, since it can be difficult to plan what the community contribution will be. The community involvement depends upon many factors, such as the complexity of the software, because volunteer developers can have problems understanding a very large software system (Goldman and Gabriel 2005).

OSS can have higher quality than the alternatives, because of certain aspects of the development process. In particular with frequent releases, defects are likely to be found quickly and removed. Further, since the source code is available, defects can be found more easily and repaired by members of the open source community (Goldman and Gabriel 2005).

Intrinsic motives for companies are that the company has similar values as the open source movement, and that source code and knowledge in the company should be shared with the open source community (Rossi and Bonaccorsi 2005).

6.2.2 Dual Licensing

Copyright holders can choose which license they want to use for their products. If a software vendor has developed a product all by herself she can use whichever license she wants. Some choose to use both a proprietary license and an open license, this is called dual licensing.

A dual license model gives the freedom of choice by allowing you to choose between a proprietary license and an open license (Meeker 2005). If you choose the open source approach the software can be licensed under for instance GPL. This license requires you to share you derived products and code with the community but you can acquire the software for free.

If you want to protect your modified software you can choose a proprietary license. By choosing a proprietary license you will most likely have to pay for the software and the code, but you are free to do whatever you want with it without having to share it with the community.

Several companies have chosen this approach, Trolltech license the Qt framework under GPL and under their own proprietary license. eZ, MySQL and others do the same for their products. They release the product to the community with an open license, for instance GPL, and they also release it with a proprietary license to customers, who do not want to share their changes with others.

6.3 Relationships between Companies and OSS Communities

There are several approaches to organizing the relationship between a software company and an open source community. A case study performed by Dahlander (Dahlander and Magnusson 2005), shows that the relationship can take one of three forms: a symbiotic, commensalistic or parasitic relationship. These are stereotypical relationships, so real companies may have a role which is somewhere in between the three forms.

In the symbiotic relationship, both the software company, and open source community gains benefits from the relationship. This means that both parties take part in decision making, and sharing of resources and knowledge. The case study showed that companies using the symbiotic approach try to return benefits to the OSS community, through the release of internally developed source code and providing infrastructure to the OSS community. This was the case with MySQL, which is a database system which is developed cooperatively by an OSS community and a software company.

The commensalistic relationship is one where either the software company or the OSS community gains benefits, while the other does not benefit or take harm from the relationship. This can be achieved if the software company uses the software produced by the OSS community, while not being involved in the actual software development itself.

The parasitic relationship is one where either the software company or the OSS community is positively affected by the relationship, while the other can be negatively affected by the relationship. This relationship should be avoided, and has not been seen in practice in the case study by Dahlander.

6.4 Summary

This chapter has looked at the commercial use of OSS in companies. It has been shown that software can be built as a combination of acquiring OSS as commodity software and developing differentiating closed-source software in Section 6.2. There are many pitfalls and tradeoffs to be made when deciding which parts of the software to base on OSS, so further studies on this topic would be interesting.

Further a way to classify the relationship between a company and an OSS community has been described as, either, parasitic, symbiotic or commensalistic. Future studies could be done to find out how these different relationships give a company influence in an OSS community, and how these can be used to create a successful relationship for a company interested in participating in OSS development.

Moreover several commercial motives were described in this chapter. Which of these motives are most important for companies, and what types of motives do these include?

Finally business models used by companies participating in OSS, has been introduced. Further studies on how the business models influences the choice of software licenses and interaction with the OSS community can be performed.

Chapter 7

Developing Software with OSS Components

This chapter will present usage of OSS components, and how companies typically use and select such components. Some selection processes for OSS components will be described, as well as how OSS components are used by companies. This will be shown through a recent study performed on Norwegian companies using OSS. Open source and commercial software development have been described as complimentary activities, where each can extend and benefit from the other (O'Reilly 1999).

7.1 Survey of OSS and COTS Usage in Norwegian Companies

A survey on Norwegian software companies using OSS and COTS has been performed in 2006. The survey was performed by the Research Council of Norway, ICT Norway, Innovation Norway and NTNU. The survey was answered by 138 companies, where 55 companies answered a voluntary section about the usage of OSS and COTS. As part of this thesis, access to relevant results was provided for analysis.

Of the 55 companies which responded, 41.8% answered that the company was using component based development with COTS, while 50.9% answered that they were using component based development with OSS. Further, 90.9% of the companies answered that they were systematically reusing internally developed components or other products. Figure 7.1 shows the motivation these companies had for choosing to use OSS or COTS based development. The motivation "Lower costs" was agreed upon by almost 80% of the respondents, indicating that the vast majority of respondents agree that using OSS or COTS contribute to reduced costs. This is a expected since OSS is available at no or low cost. Based on the data, it is not specified why not 100% of the respondents answered that using OSS or COTS was a motivation for reducing costs. It can be hypothesized that there are other costs than the cost of acquiring the software is a factor which increase the costs of using OSS and COTS, such as the cost of selecting the OSS or COTS components from a variety of sources. "Shorter development time" and "More standardized products" was also answered by the majority of respondents as motivations for using OSS and COTS. This is also as expected, because reusing finished software components means less of the code has to be implemented by the developers themselves. Finally "Improved quality" was chosen by fewer respondents, indicating that this motivation was not as important as the other motivations.

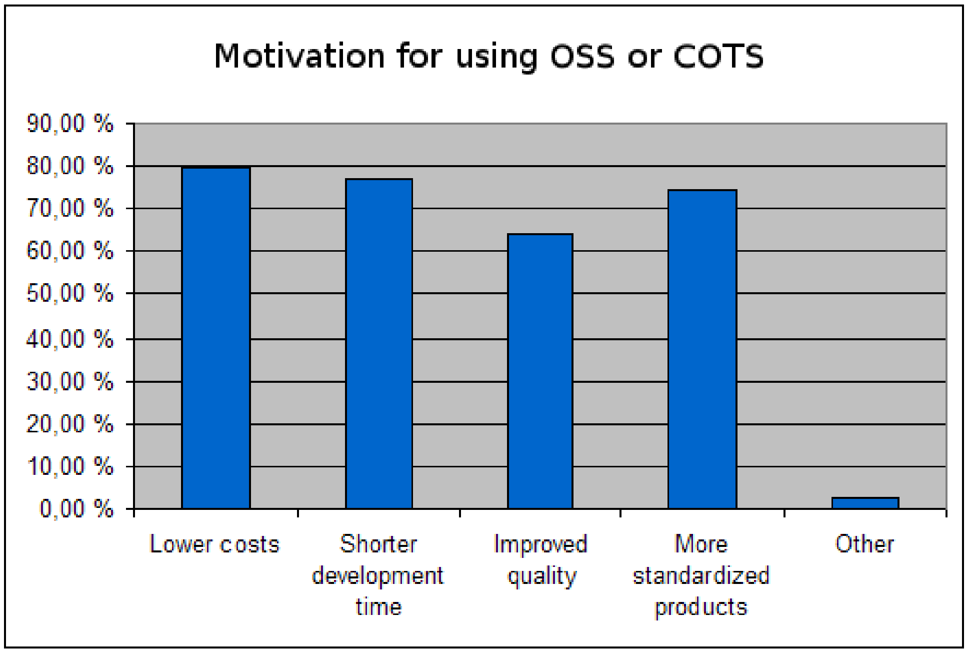


Figure 7.1: Motivation for using OSS or COTS in Norwegian companies

7.2 Development of Software Using OSS

There are several motivations for companies to acquire OSS and use it as an important part of the software development as described in the previous section. In this section, the focus will be on how OSS is used in software development by companies.

One approach to using open source software is to use it as Off-The-Shelf components. Off-The-Shelf (OTS) has been stated to be increasingly used by software companies according to a study by (Conradi and Li 2005). OTS can be either commercial, called COTS, or open source. COTS, is short for *Commercial-Off-The-Shelf*. In this report a software product with the following characteristics will be referred to as COTS (Torchiano and Morisio 2004):

- COTS, is not developed at the same place which it is used.
- COTS can be both open source and closed source. In this report, only the aspects of COTS related to open source will be relevant.
- COTS, is not commodity software which is distributed with the operating system.
- COTS, is integrated into the software being developed.
- COTS, is not controlled directly by the developers which use it.

A survey by (Conradi and Li 2005) has investigated certain aspects of how off-the-shelf components are used in industrial projects. The survey showed that all OSS was delivered with the source code, while one third of COTS components were provided with source code. However, the survey found that the components were rarely modified, even if the source code was available.

Another survey by (Li, Bjornson, Conradi, and Kampenes 2004) performed on 16 Norwegian software projects, shows that the companies were increasingly using COTS in the hope that it would result in faster time-to-market and increased productivity. The survey indicates that COTS has to be taken into account in the development process. (Li, Conradi, Slyngstad, Torchiano, Morisio, and Bunse 2005) therefore proposed a revised COTS-based software development process. The revised development process contains four proposed activities related to using software components, which will be described in more detail later:

- Deciding whether to acquire an OSS component, or develop software with the same functionality internally in the company, referred to as a build vs. buy decision.
- Selecting the best suited OSS components from a variety of competing alternatives.
- Learning to develop software based on the selected OSS components.
- Integrating the selected OSS components into the system which will be developed.

Build vs Buy Decision

Based on empirical results, (Reifer, Basili, Boehm, and Clark 2003) state that there are many aspects of using COTS which need to be considered before it can be used. The study shows that the maintenance cost of software based on COTS is more than or equal to the cost of developing customized software. Using COTS is therefore not always the best solution, especially in systems which have a long expected life-time. Before deciding to use COTS, it is therefore common to decide whether to build the component from scratch or acquire it from other sources, such as OSS. This decision should be made early in the development process (Li, Bjornson, Conradi, and Kampenes 2004).

7.3 Selection of OSS Components

It is recommended to have a selection process to decide which OSS components to use. Three selection processes will therefore be described. (Li, Bjørnson, Conradi, and Kampenes 2004) has performed a survey of selection processes in Norwegian IT companies, which provided relevant results about how companies select their components. An important result from this survey was that companies did not use any formal process for selecting COTS components. Further, the selection processes used in a project depended upon the context of the project. One reason companies did not use a formal selection process was that developers already had experience with some COTS components, and this experience was important in the decision of which components to choose. The results of the survey showed that out of 30 respondents, 16 were using an experience based selection process, while 11 were using a hands-on trial-based selection process.

Experience based selection

The first method by (Li, Bjornson, Conradi, and Kampenes 2004), is a selection-process based on familiarity with existing COTS components. If a project member is already familiar with a COTS component from a previous project, then this was a key factor for choosing that particular component again.

Hands-on trial-based selection

The second method by (Li, Bjornson, Conradi, and Kampenes 2004) is a trial-based selection process. This method was used because hands-on trial was stated to be the only way developers could trust the quality of the components. Further, this trial enabled developers to learn using the components. The Internet is used to find candidate components, and then a small number of these are downloaded and evaluated. This method is hands-on trial-based, and is described with the following steps (Li, Bjørnson, Conradi, and Kampenes 2004):

1. The Internet was first used to search for components, using keywords of the required functionality.

2. If many candidates were found, then 2-3 were selected based on brief reviews of them. Important selection criteria in this step are cost and marketplace trends.

3. Finally, a trial-version of each candidate is downloaded and tested locally.

Formal selection process

A third and more formal method for selection of OSS is proposed by (Madanmohan and De' 2004), which is found in Figure 7.2. The process is based on three activities, which can be performed several times: Collection, Incubation and Critical Revision.

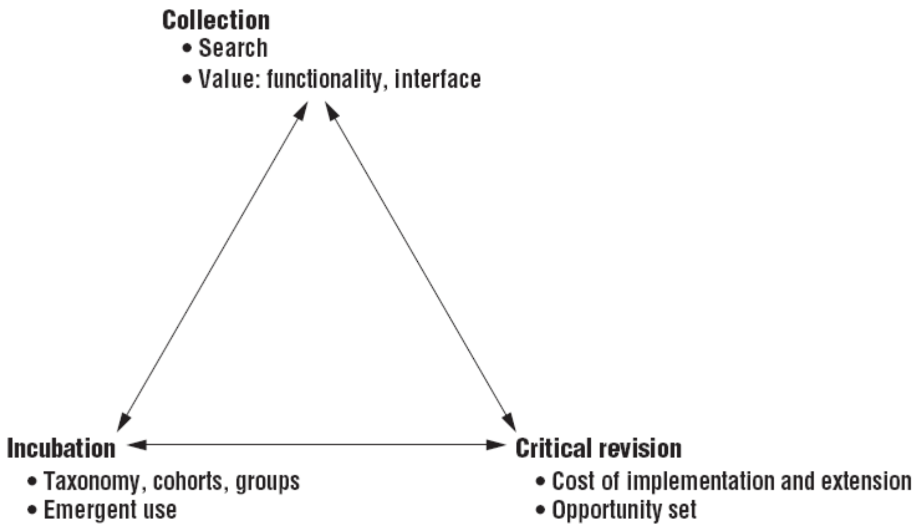

Collection
• Search
• Value: functionality, interface
Incubation
• Taxonomy, cohorts, groups
• Emergent use
Critical revision
• Cost of implementation and extension
• Opportunity set

Figure 7.2: A selection model for OSS (Madanmohan and De' 2004).

The collection activity involves searching for new components from several sources. The incubation activity is when the developers themselves evaluate the components, while critical revision involves rethinking the usage of the components in a critical way to decide if the components need to be modified to fulfil new requirements as the system evolves.

Understand how to use the OSS components

According to (Li, Bjornson, Conradi, and Kampenes 2004), it is necessary to learn and understand new software components before they can be used. This is easier to than in commercial components, because the source code of the component is available.

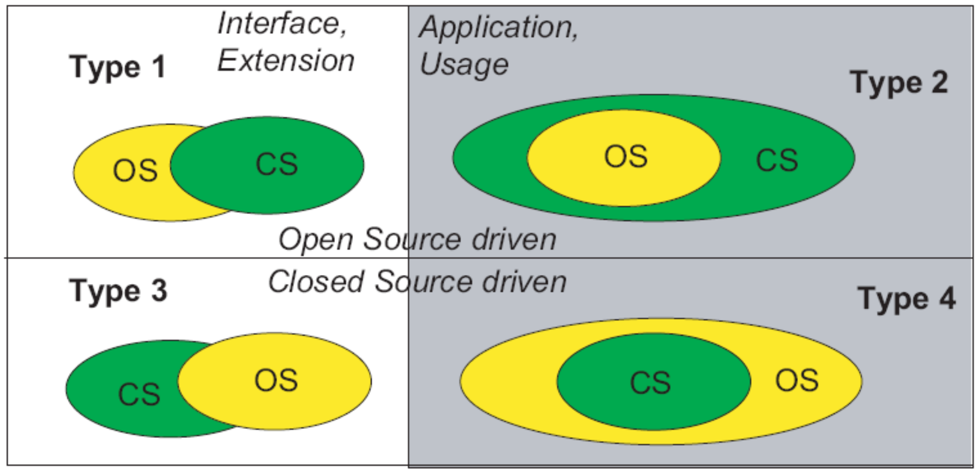


Figure 7.3: Possible ways to integrate OSS and commercial software, by (Deng, Seifert, and Vogel 2003).

Integration of Open Source with Commercial software

There a different ways to develop software which is based on open source and commercial software in some combination. Figure 7.3 shows possible approaches for achieving this. The development can be driven by either open source or commercial software as the core. There is also a distinction between tightly coupled OSS and non-OSS components. Depending upon the license associated with the component, it can either be tightly coupled and heavily modified, or lightly coupled, only used as a "black box" (Deng, Seifert, and Vogel 2003). In order to integrate the OTS components, it is common to create additional code related to the integration, which is called glue-ware (Li, Bjornson, Conradi, and Kampenes 2004).

7.4 Summary

There are many areas related to using OSS in component-based software development which could have been studied further. In the future work section of (Li, Bjornson, Conradi, and Kampenes 2004), studying "the actual COTS-based development processes" has been suggested. In addition, several challenges related to developing software based on OSS components have been described (Reifer, Basili, Boehm, and Clark 2003).

The study performed by (Li, Bjornson, Conradi, and Kampenes 2004) about usage of OSS components, selection of OSS components and some of the risk factors of using OSS have been described. Further, statistics from the survey performed by ICT Norway described in this chapter has provided information about Norwegian companies using OSS and COTS, and the motivations behind these decisions.

By repeating some relevant topics of these two surveys, it is hoped to verify the results from those surveys, as well as provide more qualitative information about current usage and challenges.

Chapter 8

Initiating, Managing and Supporting Open Source Projects

Industrial involvement in open source can take several forms. We are only interested in the involvement related to the companies which develop software, either by managing open source projects of by participating with development efforts in a project run by others. This is unfortunately not well described in the literature. Some information can be found at home pages like MySQL, Trolltech, Sleepycat, Jboss, Ez, and others, but this information is not always objective or complete. It seems to be a clear need for more research (and scientific publications) related to how companies initiate, develop, and control open source products. We will present some findings from the literature and information from the product providers in the next sections.

8.1 Commercial Open Source Project Start-up

Open source projects initiated by individuals normally start with something small as presented in Chapter 4. Open source projects initiated by companies can be started in at least three ways, as an original idea, either as an idealistic idea or a business idea, as the last possibility of a failing project, or as a project without economical potential.

Releasing a product as open source when you do not intend to make money of it makes sense. This has been done with several tools and applications which do not have any particular value to the company owning it. The product may benefit their customers and releasing the product may give the company some credit. If it does not, the company is at least not losing any important intellectual property.

The start of the Mozilla project was quite the opposite. As we saw in chapter 3, the Mozilla project was started in 1998 when Netscape saw that they were loosing the browser war against Microsoft and Internet Explorer. Netscape announced that they were giving away their browser for free, with the source code (Hamerly, Paquin, and Walton 1999). People at Netscape then established Mozilla.org as a separate organization. Mozilla.org was meant to act as a coordinator for everyone working with the Mozilla software. An interesting challenge related to releasing the software as open source, was removing proprietary components which were included into the browser. When the browser was made open source, all the components still under proprietary licenses had to be replaced.

The third option is having an idea and releasing the product with clear idealistic or business goals in mind. Sleepycat Software was started to guarantee the quality of their open source database, Berkeley DB. According to Sleepycat's web page, Margo Seltzer and Keith Bostic formed Sleepycat Software in 1996 because the users of their database wanted commercial support and new features (Software 2006). Cygnus Solutions is another company started by entrepreneurs who saw the potential in open source (Tiemann 1999). In the late 80's John Gilmore, Michael Tiemann, and David Henkel-Wallace created Cyguns Solutions providing commercial support, maintenance, and ports to new platforms for several GNU products. They did not primarily have their own products but they provided services to other open source products at the same time as they developed these products further. Having your own product as an open source product could give you the same advantages. Open source products can have the advantage of having an active community around the product and you can focus on delivering services to the product instead of selling licenses.

8.2 Managing and Supporting Open Source Projects

Management of the Mozilla project was done by people from both inside and outside of Netscape. The browser was organized in modules and every module had its "owner" (Hamerly, Paquin, and Walton 1999). These owners were found both inside and outside of Netscape. Controlling the modules allowed the owners to decide whether to accept or reject code-proposals to a module. These decisions were based on the quality of the code and whether the licenses were compatible with the Netscape Public License (NPL) or not.

Many open source projects supported by companies can also be managed by organizations outside the company itself. Many companies support open source projects through foundations like The Apache Software Foundation [1], Eclipse[2], Mozilla [3] or the Free Software Foundation [4]. These foundations work more or less independently from the industry supporting and managing the development of their respective open source products.

On the other hand, when industry supports open source projects more directly, they can contribute with for instance development effort, infrastructure, or funding. As an example, IBM claims that they have supported more than 120 open source projects including Linux with more than $1 billion (IBM 2006).

8.3 Summary

We have seen that industrial open source projects can be initiated in at least three different ways as an original idea, as the last possibility of a failing project, or as a project without economical potential. We have also shown some examples of how industrial open source projects are managed.

8.4 Challenges

An important observation is that it is surprising to see how few research publications we have been able to find about company initiation of, control of, and participation in, open source communities. Most of the information available is written by the companies themselves and it is most of the time not any quality research.

Another, interesting thing to note is that several of these relatively small open source companies have been bought by bigger companies. Redhat bought Cygnus back in 1999 and JBoss announced on their web page, April 10th 2006 that Redhat were going to aquire JBoss.

[1]`http://apache.org/`
[2]`http://apache.org/`
[3]`http://www.mozilla.org/`
[4]`http://www.fsf.org/`

Oracle announced on their web page the February 14th that they were buying Sleepycat. Is this a constant trend? Are small open source companies with good ideas, swallowed by bigger companies?

Open source projects were said to be started because a problem were 'scratching a developers itch'. Is this true for projects started by companies as well? Firstly, are companies driven by the same motives or 'itches' as individuals or do they have other motives, and in case which? Secondly, how are projects started in a company setting? How is the decision to make a product open source taken? Are products started as open source projects or are they made open source at a later stage?

Part III

Research

Chapter 9

Overview of the Research Process

This part of the thesis contains a description of our research. We start by presenting an overview of the research process. In the following chapters we describe important steps in this process. We discuss issues related to the research question, the research design, the context of our research, and how we collected and analyzed our data. For a complete overview of the content of this part, please see Section 9.3.

9.1 The Research Process

The original project description was presented in the introduction of this thesis. We selected our research process with this description in mind. (Remenyi, Williams, Money, and Swartz 1998) describe an eight phase research process for research in business and management. We adopted this process with some minor modifications inspired by the process described by Jacobsen (Jacobsen 2005). The eight phases in (Remenyi, Williams, Money, and Swartz 1998, 64-65) are:

1. Reviewing the literature
2. Formalizing a research question
3. Establishing a methodology
4. Collecting evidence
5. Analyzing the evidence
6. Developing conclusions
7. Understanding the limitations of the research
8. Producing management guidelines and recommendations

These phases should be self explaining and the reader is advised to consult the book for further descriptions of the process. It is important to notice that several of these phases will go in parallel and that the work is iterative.

Activity	Date
Project start-up	15th of January
End of literature review	25th of February
Preliminary research question	1st of March
First version of the questionnaire	9th of March
Company visits and interviews	13th-15th of March
Final version of the questionnaire	31st of March
Distribution of the questionnaire	7th of April
Questionnaire deadline	15th of May
Project deadline	21st of June

Table 9.1: Timeline of the research

9.2 Project Timeline

This section contains an overview of the major activities in this project. These activities are explained further in the remaining of this, and the next part of the thesis.

In the beginning of the project, we sketched a plan similar to the timeline show in Table 9.1. We put time-slots instead of exact dates in this plan because a lot of the work is highly related. The work was also done in parallel and in iterations. For instance, the literature study which said to be finished the 25th of February was improved and change after this date as well. The changes occurred because of new knowledge or feedback from colleagues or supervisors. The work was performed using short term goals which were agreed on, on a daily or weekly basis.

9.3 Presentation of the Research

Our process and the outcome of our process are presented as follows. A description of our literature study can be found in Chapter 10, and the result of this literature study can be found in the second part of this thesis. The formalization of the research question and the final research question can be found in Chapter 11. The research design and a theoretical foundation for this design can be found in Chapter 12. The context of the study, and the study objects are presented in Chapter 13. We conclude this part of the thesis with collection and analysis of the evidence in Chapter 14 and 15.

Process step	**Relevant parts**
Reviewing the literature	Chapter 10 and the prestudy
Formalizing a research question	Chapter 11
Establishing a methodology	Chapter 12
Collecting evidence	Chapter 14
Analyzing the evidence	Chapter 15
Developing conclusions	Chapter 17
Understanding the limitations of the research	Chapter 17
Producing management guidelines and recommendations	This thesis

Table 9.2: Research process and relevant chapters

The sixth and the seventh step of the process, development of conclusions, and understanding the limitations of the research, can be found in Chapter 17. Both steps are included in the discussion of our findings, and the chapter is in the next part of the thesis.

The last step of the process is packaging and presentation of the research. The presentation of our work is this thesis, and how we wrote this thesis will not be explained in any further detail. A summary of the steps in the research process and the relevant parts of this thesis can be found in Table 9.2.

Chapter 10

Reviewing the Literature

Conducting a literature survey is part of the original problem description, presented in the introduction. By carrying out such a study, we will fulfil parts of our requirements to this thesis, it will help us to broaden our knowledge about open source, and give us a good theoretical basis for building further research. The literature study will further provide input to the research question and the choice of a research design. How we performed our literature study will be presented in this chapter. The result of this literature study is presented in the pre-study part of this thesis.

10.1 Sources

Material for the state of the art survey was found in many ways, through several different sources. These sources and how we searched them will be described in this section.

As a starting point we used articles from the curriculum of both *IT8003 Advanced Topics Regarding IT* and *Organizations and TDT10 Software Technology COTS and Open source software.* Broadening this search was done by searching several journal databases, as presented in Table 10.1, with different search words. The terms we used while searching were both wide and more specific. The term "open source" is one extreme, and terms like "open source community joining" would be a more specific search term.

Journal Database	**URL**
The ACM Digital Library	`http://portal.acm.org/portal.cfm`
EBSCOhost Electronic Journals Service	`http://ejournals.ebsco.com/Home.asp`
ScienceDirect	`http://www.sciencedirect.com/`
JSTOR - The Scholarly Journal Archive	`http://www.jstor.org/`
Computer and Information Science Papers CiteSeer Publications Research Index	`http://citeseer.ist.psu.edu/`
IEEE Xplore: Dynamic Home Page	`http://ieeexplore.ieee.org/Xplore/dynhome.jsp`

Table 10.1: Journal databases

In addition to these journal databases we reviewed papers from several workshops and conferences as shown in Table 10.2, and from internet resources and Web journals as shown in Table 10.3.

Workshop name	URL
1st Workshop on Open Source Software Engineering	`http://opensource.ucc.ie/icse2001/`
2nd Workshop on Open Source Software Engineering	`http://opensource.ucc.ie/icse2002/`
3rd Workshop on Open Source Software Engineering	`http://opensource.ucc.ie/icse2003/`
4th Workshop on Open Source Software Engineering	`http://opensource.ucc.ie/icse2004/`
5th Workshop on Open Source Software Engineering	`http://opensource.ucc.ie/icse2005/`
C & T 2003 Workshop on Open SourceSoftware Movements and Communities	`http://opensource.ucc.ie/ct2003/`

Table 10.2: Open Source Workshops

Site name	URL
The Free Software Foundation	`http://www.fsf.org/`
Open Source Initiative (OSI)	`http://opensource.org/`
opensource.ucc.ie: resources for researchers	`http://opensource.ucc.ie/`
Free / Open Source Research Community	`http://opensource.mit.edu/`
First Monday - Peer reviewed journal on the Internet	`http://www.firstmonday.org/`

Table 10.3: Web Resources

Articles were also found after recommendations from fellow students, researchers and supervisors. Last and perhaps most importantly, references in the articles we evaluated were searched for other interesting references.

In addition to articles and Web resources, we also searched the library at the university, using the *BIBSYS* tool at `http://ask.bibsys.no`. We searched the database for particular books, with open searches, and with more specific search words.

Lastly, we also used Google [1] and Wikipedia [2] to some extent to find pointers to literature and more information about specific topics. This information was always cross-checked with other sources, preferably with the source of the information.

10.2 Selection and Evaluation

Based on our project description, we tried to select papers relevant to industrial involvement in open source. This manual selection would of course influence the result of the literature study. Because the amount of papers published where open source is mentioned is quite large, it is nearly impossible to read everything. There have been published open source related papers, in many different fields, but we are primarily interested in software engineering and preferably empirical work. We have therefore only selected papers which we found relevant to this focus. The selection was performed by evaluating the title of the paper, or more often also the abstract, introduction and conclusion. The number of citations was also taken into account when using CiteSeer, and sometimes also which other papers, the paper cited.

[1] `http://www.google.com`
[2] `http://en.wikipedia.org/wiki/Main_Page`

Chapter 11

Research Questions

We will in this chapter first present our final research questions, followed by a short overview of how they were defined.

11.1 Research Goal and Research Questions

A high-level research goal is defined to determine the overall purpose of the research. Based on this goal, several research questions are defined to fulfil the research goal. The research goal and questions are presented in Table 11.1.

Research Goal:
Explore the following four roles: open source owner, open source participant, inner source participant, and user of open source components. Explore the relationships these roles have to the open source community, their development processes, their communication processes, and their motivations behind assuming these roles.
Research Questions:
RQ1: Why do companies use OSS components, and how do they find and evaluate these components?
RQ2: Why do companies choose to make their products Open Source, and how can this be done successfully?
RQ3: Why and how do companies use development methodologies from Open Source, in ISS development?
RQ4: How does the use of OSS influence the development processes of companies?
RQ5: How and why do companies participate in the development of OSS projects controlled by a community outside the company?

Table 11.1: Research Goal and Research Questions

11.2 Elaboration of the Research Goal and Research Questions

Based on the literature study, the goals of the COSI project (Chapter 13) and the objectives of this thesis (Chapter 1), we started to form a research goal, and possible research questions. These proposals were evaluated in collaboration with our supervisors and several of the proposals were rejected, primarily because of time constraints, but also because it is important to limit the scope of this thesis.

On one hand we have the results from the literature survey. We found that industrial involvement in open source is poorly described in the literature. The literature survey also showed that literature paints a somewhat unbalanced picture of open source. On the other hand we have the overall goal of the COSI project, which is to better understand how open source, and open source practices can be utilized in a commercial setting. Finally, we have the responsibilities of NTNU in the COSI project, which is to establish a baseline of current open source practices in industry. This includes both describing the industrial partners' processes and reviewing work described in research literature.

We saw that we could best contribute to NTNU's work, to the COSI project, and to improve the knowledge about open source by conducting a literature survey, and by doing more empirical work related to industrial involvement in open source. To better understand industrial involvement in open source it is necessary to empirically explore their relationships to open source.

We have identified four open source related roles that a company can have in the COSI setting. A company can either, own and manage an open source product, it can participate in inner source development, it can participate in open source development, or it can use open source components provided by a community without actively participating in the community. An overview of these roles can be found in Table 11.2. We defined these roles based on the COSI documentation, our literature review, information about the Norwegian partners in the COSI project, and conversations between one of the authors of this thesis and employees from the partners in the COSI project. These conversations took place at the COSI kick-off meeting in Netherlands between January 17th and 19th.

The literature did not provide any unexplained phenomenon we could try to explain or any clear hypotheses we could test. Based on the observations from the literature, we saw that we needed to *explore* these four roles further to be able to observe new phenomenon, propose new hypotheses, and build new theory. The literature study showed that many aspects of open source are closely related, and it did not discover any clear focus areas. The COSI project tries to cover several aspects of open source. The focus of our study will be software development, but we cannot leave out all other aspects. We believe that including some aspects beside software development, and doing a broader study will provide us and the COSI project the best results. It is very important not to loose focus, but since we do not exactly know what we are looking for, we fear that if we focus too much we might overlook important issues.

We wanted to explore the four roles described in Table 11.2, the motivations behind assuming these roles, and the processes a company can be involved in when having these roles. We have thus formed the *explorative* research goal in Table 11.1. Based on this high-level research goal, research questions can be created to fulfil that goal. Exploring every aspect of these roles is impossible, so some areas have to be prioritized.

Reuse of components, and component selection, is something that is normal to traditional software development. Industry has started to use more and more OSS components as well. We wanted to explore why they use OSS components, how they find them, and how these components are evaluated. This resulted in RQ1.

Giving away your software for free as open source does not sound smart, or is it? More and

Role	Description
Open Source Manager	An open source manager is a company which controls the development of a product, licensed with an open source license. The company controls or manages access to the code and it decides the further development of the product.
Open Source Participant	An open source participant is a company which participate in the development of an open source product controlled by others. Participation can take several forms, from core development effort, to requesting assistance from the community around the product.
Inner Source Participant	An inner source participant is a company which uses open source development practices in a closed consortium of partners, for instance several companies working together, several departments in a company working together.
Open Source Component User	An open source component user is a company which develop software using open source components without participating actively in the community around the community. The component is thus used as any other components acquired from any other component provider.

Table 11.2: Company roles in open source involvement

more companies, like Trolltech, MySQL, Sleepycat, and others, have started to use this business model. We want to understand the advantages and disadvantages of this model. In addition, we want to understand the challenges of making a product open source. To understand this better we formulated RQ2.

Inner source software (ISS) development is a fairly new term, and it is not described in detail, in literature. We wanted to explore the characteristics of ISS. This made us formulate RQ3.

Inner source software development is related to the use of OSS development practices in a smaller setting. How is software developed by companies using OSS components, and how is software developed by companies which participate in OSS development? What are the characteristics of this kind of software development? We wanted to explore OSS related software development, and RQ4 was formulated because of this.

Communicating in the open source communities is primarily done by simple text based tools like email and chat. How do companies who are involved in open source related software development communicate? We wanted to explore this issue, and we formulated RQ5.

By doing explorative research on these issues related to industrial involvement in open source, we intend to provide guidelines for open source involvement, share our experiences with the industry, and create a platform for further research.

Chapter 12

Research Design

The research design is the blueprint for the collection, measurement, and analysis of data, and will be used to answer the research questions (Cooper and Schindler 2001).

This chapter will present the research design of this thesis. The rationale behind choosing this research design will be described first. Next, the relevant research methods involved in this research design will be explained. The main research methods which will be described, includes interviews and a questionnaire. The research design of these survey methods will be described subsequently.

12.1 Choice of Research Strategy

The research design is based on the research questions of this thesis, found in Chapter 11. Based on these research questions, relevant research methods and procedures will be chosen. The research goal was stated to be explorative, where the purpose of the research goal is to produce new knowledge about a topic which is relatively unknown.

There are three empirical strategies which are commonly used when performing research in software engineering: experiment, case study and survey. An *experiment* is a formal and controlled investigation within a laboratory environment, which provides better control over the results than the two other methods. Experiments are often used to confirm existing theories. A *case study* is often used to monitor and observe projects. Case studies can be used in projects to compare the effects of a new method, compared to a baseline before that method was introduced. A *survey* is an investigation usually performed in retrospect, and the most common survey methods used to collect information is through interviews or questionnaires (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). Which of these strategies would be most suited to answer the research questions, given the constraints of interacting with the companies participating in the COSI project? Since it is the least costly strategy for the participating companies, and can be performed within the time-constraints of this thesis, it was decided to perform a survey.

A survey can be either descriptive, explanatory or explorative (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). In a *descriptive* survey, the research questions are already known, and the survey will describe a situation or some characteristics of a population. *Explanatory* surveys try to find reasons and causal relationships between observations. In *explorative* surveys the research questions are not clearly defined, and such surveys are performed when the topic is not clearly understood. Explorative surveys are further useful when it is not known which problems that can occur during the study (Cooper and Schindler 2001).

Explorative research questions can be solved by performing exploratory studies. The study in this thesis will be performed as an explorative survey, because there were no appropriate research questions based on the literature study, and because we want to discover new knowledge about open source development, as outlined in the research questions. Although the literature study has shown that previous studies have been performed on some aspects of open source, this study will concern areas where there is very little published research; such as the focus on open source projects managed by companies and inner source software development. The purpose of this study is therefore to answer the research questions by performing an explorative study to find new knowledge about open source.

When designing a research study, (Cooper and Schindler 2001) recommend using a two stage research design for explorative studies. The first stage is the most explorative one, where the purpose is to define research questions and develop the research design. The second stage is more descriptive and is performed on a larger sample of the population. Based on the explorative research goal and questions, it was decided that first a qualitative study should be performed on a small number of respondents, followed by a qualitative and quantitative study on more respondents, based on the information learned during the first part. These methods will be described in the following section.

12.1.1 Qualitative Studies

Qualitative and quantitative studies are two research approaches which will be used to collect information for this study. The difference between the two approaches is that *qualitative* methods collect information as text, while *quantitative* methods collect information as numbers. Both methods are used for performing empirical studies, and each of them are used in different situations depending on the desired result (Jacobsen 2005).

The output from a qualitative survey is typically a textual description of a particular situation or object being studied. The following research methods are often used in qualitative studies(Cooper and Schindler 2001; Jacobsen 2005; Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000):

Interview

In-depth interview is the most commonly used qualitative method. An interview is performed when a respondent and a researcher has a dialog where information about the respondent is collected. The interview can have various degrees of being structured, ranging from a conversation without strict structure or order, to a fully structured interview with predefined questions and alternatives for the answers. Expert interviews can also be performed, where people with specific knowledge about a topic is interviewed. Another common form of interview is a group interview.

Observation

Observation of the study objects, for a first hand observation of the participants in the study. For example, it is possible to observe how developers cooperate and perform their work using a particular development process.

Case studies

A case study, which is an investigation of a single object or phenomenon, is often performed within a limited time period. It is common to perform case studies on specific software projects, and collect data at specific times to measure the effect of new techniques or methods. Case studies are typically longitudinal, which will be described in section 12.2.

Other Qualitative Methods

One qualitative method is document analysis from secondary information sources, for example analyzing requirement specifications. Another method is to perform an ethnography, which is a study of how a cultural group describes itself.

12.1.2 Quantitative Studies

Quantitative research on the other hand, has the goal of measuring, counting or comparing objects. Research objects are observed in their natural setting, searching for definitions and meaning (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). The output of a quantitative study is typically the measured numbers of the situation or object being studied, or statistics describing a sample taken from a larger population.

The following research methods are commonly used in quantitative studies (Jacobsen 2005):

Questionnaire

Questionnaire is the most common quantitative method, and can be both quantitative and qualitative. Information can easily be collected from a large number of respondents, and the results can be compared. Information can be standardized by creating questions which are answered by the respondent by selecting the answer from a list of alternatives created by the researcher. Important concepts have to be *operationalized*, which means to create concrete questions about more abstract concepts. The answers from the respondents can then for example be analyzed statistically.

Other Quantitative methods

Other quantitative methods can include studies of secondary data, such as processing and analyzing a database of defects reported from users.

12.1.3 Choice of Research Methods

Based on the preceding research methods and explorative research goal, and questions, it was decided to first perform a structured interview, followed by a questionnaire.

The interview was chosen as a qualitative method because it was the best trade-off between usage of time, and its ability to provide detailed textual description for the research questions. First, the conditions for the study was that three Norwegian companies presented in chapter 13, participate in the COSI project, and therefore have agreed to participate with some employees in a limited time period. An interview would therefore provide sufficiently detailed information in a relatively short period of time. Further, given that very little was known about the companies before this study began, an interview would allow the respondents to answer, elaborate, and clarify their situation and role during the interview. Finally, given that interviews are the most common qualitative method, and therefore most likely to be familiar to the respondents, it was decided to perform an interview. The interview should be structured as much as possible, to get comparable results from the respondents, and because the research design of the interview then can be used as a basis for creating and improving the research design for the questionnaire. However, the interview must also have open-ended questions because of the explorative research question. See section 12.3 for the design of the interview.

The questionnaire was chosen as a qualitative and quantitative method, because it is the most cost-efficient way to get information from the potentially large number of companies participating in the COSI-project. Since the companies are distributed in several different countries, they can respond to an electronic questionnaire over the Internet. Analyzing and comparing the quantitative results from the questionnaire will then be possible. Questionnaires, as a survey method, can contain both quantitative and qualitative questions depending on what information is required. See section 12.4 for the design of the questionnaire.

12.2 Approaches to Creating a Survey

Surveys can have an inductive or deductive strategy, and be either longitudinal or cross sectional, descriptive or causal, and have other characteristics which are decided during the design of a survey. This section will clarify some of these design decisions. These are philosophical issues related to the way people can observe the reality, and how a survey best can describe the reality. (Jacobsen 2005) describes two perceptions of reality: the positivistic view which implies that reality can be studied objectively, in opposition to the hermeneutic view, based on understanding which implies that the reality is made and interpreted by people. Since this survey has the goal of describing reality both qualitatively and quantitatively, it is necessary to be aware of the limitations and choose the most suited research methods.

12.2.1 Inductive or Deductive Reasoning

Deduction is a form of reasoning where the conclusion follows from the reasons given, opposed to induction, where the conclusions are drawn from facts or evidence (Cooper and Schindler 2001).

Deductive research is based on existing experiences and theory, while inductive studies create theories from their own observations. Qualitative studies are often based on deductive reasoning, for example in the case of a questionnaire, because information must be ordered before the survey is performed, for example the alternatives for answers must be prepared beforehand. Therefore, deductive reasoning can only create a limited amount of new knowledge within the scope decided beforehand by the researchers (Jacobsen 2005).

Inductive research on the other hand are more open to new knowledge than deductive studies, and are useful for exploring topics which are unknown in advance (Jacobsen 2005).

The study performed as part of this thesis will use aspects of both inductive and deductive research. As a consequence of the explorative research goal, inductive reasoning will be used to a large extent to create conclusions based on evidence from the interviews. Further, both inductive and deductive reasoning will be used to create conclusions based on the questionnaire. The questionnaire is deductive, in that it will be designed based on a literature study of existing research studies, but also inductive since there will be open ended qualitative questions.

12.2.2 Longitudinal or Cross-sectional Study

The study will be *cross sectional*, rather than *longitudinal*. This means that the study will represent the situation reported by the responding companies at a specific point in time, rather than comparing changes over time (Cooper and Schindler 2001). This decision is already given, because of the limited time-period available for this thesis. Future studies could be performed to compare the results from this study, to perform a longitudinal study which would reveal long-term trends and changes.

12.2.3 Descriptive or Causal Study

Studies can be *descriptive*, which describes the, who, what, where and when of a particular research area. Studies can also be *causal*, which search for causal relationships between variables, why certain decisions are made (Cooper and Schindler 2001). The study in this thesis will aim to be both descriptive and causal.

12.3 Interview Design

The interview is designed to provide qualitative answers to the research questions. The design of the interview is explorative, where the main objective is to discover new aspects of open source related to the research questions. The lessons learned from the interview will be used to create a questionnaire, which is described in the next section. The process of creating the research design for the interview has therefore been closely related and mixed with the design of the questionnaire.

12.3.1 Respondents

The interviews were conducted at the three Norwegian software companies Keymind, Linpro and eZ Systems, where developers and managers were interviewed. These respondents will be described further in Chapter 13.

The companies were selected using *convenience sampling*, which means that the companies were selected because they participate in the COSI project in Norway, and therefore are available for interview. However, selection of respondents is related closely to how much the results of the interview can be generalized to a wider range of companies (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). All companies have reported that they have a role in OSS, so the respondents are taken from the correct population of companies. However, three companies will not be representative for all companies with a role in OSS, and this has a big impact on how the results can be generalized. The validity of the results will be discussed and elaborated upon in the next part of this thesis.

When performing an interview with three companies, it will be possible to focus on the special and unique aspects of these companies, rather than searching for results that can be widely generalized (Jacobsen 2005).

12.3.2 Relation to Research Questions

The interview is supposed to further improve and fulfil the research goal from Chapter 11, as well as further improve the research design for the questionnaire. Therefore, the research questions were not fully formalized before the interviews began, since the interviews were supposed to help define them. The research goal was more formalized before the interviews began.

Based on initial descriptions given from the companies, it was already hypothesized that these companies had the roles open source manager, open source participant, and user of open source components. This would be verified during the interview.

Each part of the interview would *explore the relationships these roles have to the open source community, their development processes, their communication processes, and their motivations behind assuming these roles*, since this was the initial purpose of the research goal.

12.3.3 Interview Design Application

Next the high-level interview design had to be applied to create an interview which could be performed with respondents.

An *interview guide* was created to give an overview of the topics that should be covered by the interview. This guide can be found in Appendix C. The interview would be structured to a large extent, with predefined questions, containing both multiple-choice questions with predefined answers and open ended questions. During the interview it will still be possible to ask follow-up questions to further elaborate on any areas where more information is needed. A structured interview is easier to analyze, and will ensure that the respondents are answering questions about the topics which were planned (Jacobsen 2005). The interview guide was written in Norwegian to be used during interviews with native Norwegian respondents.

The interview would provide information which should be used to create the questionnaire. Therefore, the interview guide contained questions from an early version of the questionnaire, which would be refined through the interviews to create the final questionnaire.

12.3.4 Interview Limitations

There are many problems which can occur when designing and performing an interview, and these effect the validity of the results. Some of the problems are related to the sample of respondents answering the interview, while other limitations can be categorized as non-response and response errors.

Performing interviews is time-consuming, so it is difficult to interview a representative number of responding companies. This means that the validity of the results will have to be discussed when analyzing the results.

Non-response errors occur when the respondent for some reason will not answer the interview. This reduced the sample size, and has an impact on the validity of the results. *Response errors* occur when the reported data does not match the actual true data. These errors can occur if the respondent does not answer the questions accurately or completely. People often over or underestimate measures. Response errors can also occur if the interviewer makes mistakes, such as not following the interview guide (Cooper and Schindler 2001).

12.4 Questionnaire Design

A questionnaire is a very common way to collect both qualitative and quantitative data from a large number of respondents. The information can be standardized, which makes it possible to compare results from different respondents. This section will describe the design of the questionnaire used in this study. The final questionnaire can be found in Appendix B, in paper format. The questionnaire will also be available in an online format, containing the same questions. The final questionnaire in Appendix B also contains answers from respondents, to limit the number of pages used for attachments.

12.4.1 Respondents

The respondents to the questionnaire will be all companies participating in the COSI project. These companies are located in different countries, so business units at different locations may submit separate responses. The number of companies participating in the COSI project is approximately 16. There is also a small number of research institutions participating in the COSI projects, although most of these will not have a role in OSS. These respondents will be described further in Chapter 13.

The companies were selected using *convenience sampling*, with participation in the COSI project as the only selection criteria. This selection is not representative for any larger population of companies, so the results will mainly be valid for the participating companies only. However, the sample size for the questionnaire is larger than the sample size for the interview, which will have a positive effect on the validity of the potential results.

12.4.2 Relation to Research Questions

The questionnaire should reflect all aspects described in the research goal and research questions. Based on the research questions and the results from the interview, the four roles open source owner, open source participant, inner source participant, or user of open source components should be further explored. These are very diverse roles, so it became apparent early during the design of the questionnaire that these were separate, partially overlapping roles, and that this had to be reflected in the research design. It was also assumed that companies could have several roles, based on the initial description given from the companies about themselves.

After having performed the interview, the four roles described in the research goal had become more clearly understood, and these four roles became the basis for the organization of the questionnaire. The four parts of the questionnaire will be described next, and how they are related the structure of the questionnaire to the research questions and literature study.

Initiating and Managing OSS

This part of the questionnaire is related in the role 'Open Source Manager' described in the research goal, and its sole purpose is to answer research question 2. The part contains questions based on Chapter 8 from the literature study. The literature study found many unanswered questions related to companies initiating and managing OSS projects, which will be elaborated upon in this part of the questionnaire.

Participating in Inner Source Development

Inner source development has been discussed in section 5.3 in the literature study, and is also one of the roles companies take related to OSS according to the research goal. This part of the questionnaire will answer research question 3.

The part about inner source in the questionnaire is highly explorative because very little information is written about the topic in published articles, and none of the companies participating in the interview-part of this study have stated that they participate or use inner source.

Interacting with or Participating in OSS Communities

This part will describe the aspects of companies interacting with or participating in OSS communities, related to communication processes and relationships to the open source community of the research goal. This part of the questionnaire is related to research question 5. The questions will be based on the literature study in Chapter 5. These questions will validate the existing research in company interaction with OSS communities, and possibly discover new facts about the topic in explorative, open-ended questions. The motivational factors for such involvement will be important.

Developing Software with OSS Tools and Components

This part of the questionnaire is related to the role OSS component user in the research goal. Questions will be mostly quantitative, with some open ended questions in addition. This part of the questionnaire is related to research question 1 and 4.

A large part of these questions will be related to exploring development processes which are used for selecting OSS components by companies. Some questions will be based on the literature study in Chapter 7, while other questions will be reused from a survey titled 'Survey on Risk Management in OTS-based Development in IT Industries in Several Countries' (Li, Conradi, Slyngstad, Torchiano, Morisio, and Bunse 2005) at NTNU. The authors of that survey have given permission to reuse and compare the questions and results of relevant parts. The reused questions can be found in the final questionnaire as part of questions 42 and 50. This is a deductive form of reasoning, where the existing results from the previous study are compared with new observations from the COSI companies.

12.4.3 Questionnaire Design Application

This section will describe how the questionnaire was created by applying the design with methods commonly used in questionnaires.

Filter questions

The questionnaire has several parts including a part for each role described in the research question, as well as a part to collect information about the respondent, product and company. Therefore *filter questions* are used to allow respondents to answer only the parts which are relevant for the respondent. It is important that respondents reply only to questions which are appropriate for the validity of the results. Further, the filter questions should not be too many and too complex to confuse the respondents.

Types of Questions and Scales

The questionnaire will contain both qualitative and quantitative questions. According to (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000) qualitative studies perform measurements of questions on the nominal and ordinal scales, while quantitative studies perform measurements of questions on interval and ratio scales. These scales will be explained in this section.

The qualitative question types are open-ended questions where the respondent can answer freely. These types of questions are used when it was difficult to create predefined answers, and when unique answers from each respondent were of interest.

Quantitative questions are used to classify the answer of the respondent into a predefined and standardized format. These are the types questions mainly used in the questionnaire:

The first is questions where the answer is chosen from alternatives on a *nominal scale*, which means that the answer only represents a form of mapping into categories, such as a classification, label or choice of alternatives. For example, the answers 'Yes', 'No', and 'Don't know' are three answers on a nominal scale. Since the various alternative answers have to be predefined before the questionnaire is answered by respondents, it can sometimes be difficult to create all the alternatives that apply for a particular respondent. Therefore, most questions have the possibility to answer an 'Other (please specify):' alternative which the respondent can use when the predefined alternatives are not sufficient. The ability to specify 'Don't know', is also available in most questions, when the respondent is not able to answer the question (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000).

Questions of *ordinal scale* ranks the answers according to some ordering scale (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). In the questionnaire, *interval scale* is used to classify the respondents opinion on questions, on a scale where the distance between the different ratings have a meaning, for example a scale from 'Small' to 'Large'. For many questions of interval scale, a Likert scale is used. Many questions are formulated as 'To what extent do you agree with the following statements?'. Then the following scale for answers is used: 'Don't agree at all', 'Hardly agree', 'Agree somewhat', 'Mostly agree', Strongly agree' and 'Don't know'. When a question is formulated as a question about size or amount, then the ordinal scale 'Not at all', 'Small', 'Some', 'Large', 'Very large' and 'Dont' know' is used. These scales were chosen because they are used in other studies, and when comparing the results from other studies, it is important that the scales are compatible.

Finally, some questions of *ratio scale* were used to answer questions about numerical measures such as age.

12.4.4 Questionnaire Limitations

One major limitation of questionnaires is that questions can only be asked about topics which we subjectively think is important. As a consequence, it can be difficult to define the correct questions, and results can be interpreted to answer the expectations that the researcher initially had, rather than what the actual results are (Jacobsen 2005).

12.5 Summary of Research Design

Table 12.1 summarizes the research design of the study in this thesis, which has been described in this chapter.

Research Goal	Explore the following four roles: open source owner, open source participant, inner source participant, and user of open source components. Explore the relationships these roles have to the open source community, their development processes, their communication processes, and their motivations behind assuming these roles.
Research Methods	Structured Interview and Questionnaire. Qualitative and quantitative study. Cross-sectional time perspective. Inductive and deductive reasoning.
Interview Design Respondents Sampling method	Structured interview. 3 companies: Keymind, eZ Systems, Linpro. Convenience sampling.
Questionnaire Design Respondents Sampling method	Questionnaire based on research question and interview results. 16 international companies participating in the COSI project. Convenience sampling.

Table 12.1: Summary of research design

Chapter 13

Research Context and Study Objects

In this chapter we will describe the context of our research, the COSI project, the Norwegian COSI project, and the partners in the Norwegian COSI project.

13.1 The COSI Project

The COSI project (Co-development using inner & Open source in Software Intensive products) is an European ITEA project. It was started late 2005/early 2006 and it will be running until the end of 2008. The project consists of both industrial and academic partners from Sweden, Finland, Spain, Netherlands, and Norway. The industrial partners range from big multi-national companies like Nokia, Philips, Telefonica, and Telvent; to smaller companies like Meritie and Keymind with six employees each. These partners develop everything from hardware and embedded software to more traditional software. A list of the partners can be found in Table 13.1.

COSI is based on the following observations related to shifts in software development, and the fact that for most software only a small part is differentiating. The project proposal suggests that only 5 to 10% of most software is differentiating (van der Linden 2005).

- Shift from in-house development towards subcontracting and integration
- Shift towards coalitions (e.g. open platforms)
- Shift towards collaboration with open source development communities

The consequence of these shifts is illustrated in figure 13.1. The project predicts that more and more software is produced in, or with open or inner source development. More about these shifts and the commercial use of open source can be found in Chapter 6.

The project intends, first of all, to increase the awareness of the industrial usage of distributed collaborative software and open source. Secondly, to understand the development model used in open source communities, and to learn how industry can benefit from this manner of developing software. Thirdly, to deploy key aspects of open source software engineering inside a limited scope, in what we call inner source development. It is particularly interesting to understand how to achieve trust and understanding in such collaboration. Fourthly, to understand how to take advantage of the software shift from differentiating to commodity software. This means keeping a company from losing intellectual property to other participants in a community or collaboration, and avoiding wasting valuable engineering resources developing

Company	**Role**	**Country**
Philips	Industry	Netherlands
Finalist	Industry	Finland
Nokia	Industry	Finland
Meritie	Industry	Finland
Gamelion	Industry	Finland
VTT Technical Research Centre of Finland	Research	Finland
Helsinki School of Economics	Research	Finland
ICT-Norway(partner)	Industry	Norway
eZ Systems(subcontractor)	Industry	Norway
Linpro(subcontractor)	Industry	Norway
Keymind(subcontractor)	Industry	Norway
NTNU(subcontractor)	Research	Norway
Telvent	Industry	Spain
Telefonica R&D	Industry	Spanin
European Software Institute (ESI)	Research	Spain
Universidad Politecnica de Madrid	Research	Spain
Combitech Systems	Industry	Sweden
Högskolan i Skövde	Research	Sweden

Table 13.1: The COSI partners

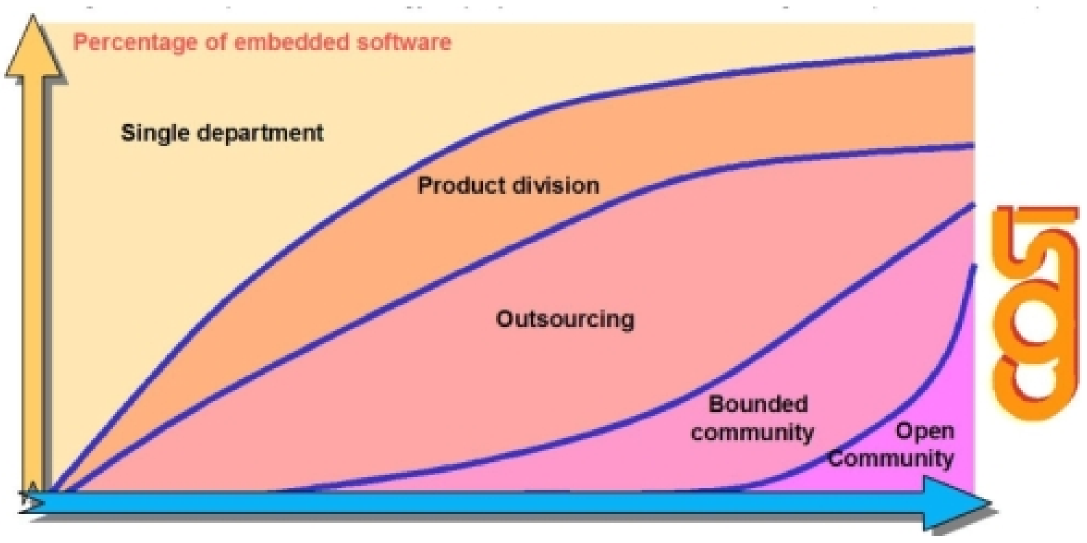


Figure 13.1: Prediction of the percentage of parts in embedded software(van der Linden 2005)

infrastructure or working alone with something that could be developed as open source or in collaboration with others. In other words finding, and staying in the middle area of Figure 13.2. The lower left corner should be avoided because this software could be acquired cheaper elsewhere, and the upper right corner should be avoided because it would mean giving away the competetive edge to competitors. This figure has also been explained in section 6.2 in the prestudy.

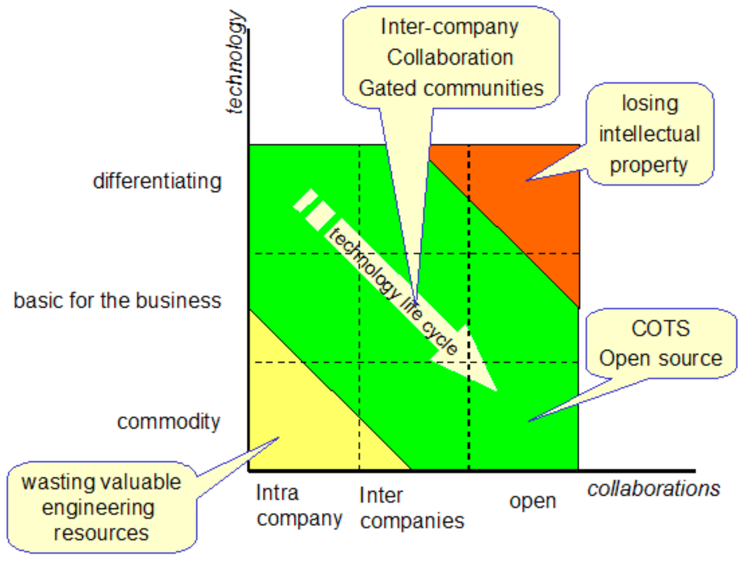


Figure 13.2: Efficient and effective development(van der Linden 2005)

For more information about the COSI project, please see `http://itea-cosi.org` and the full project proposal (van der Linden 2005).

13.2 Norwegian COSI

The Norwegian sub-project is sponsored by the Norwegian Research Council and led by ICT-Norway. ICT-Norway has more than 320 members, with a combined annually turnover of approximately 12 billion Euro (IKT-Norway 2006). In addition to ICT-Norway, there are also three industrial partners, which will be described below, and one academic partner, NTNU.

The focus of the Norwegian project is process improvement of development processes related to open and inner source software development. The Norwegian project is further responsible for deliverables to the European project related to development processes. Each industrial partner is responsible for reporting their processes related to open source, and NTNU is responsible for gathering these experiences and presenting them to the European project.

In the next subsections, the industrial partners in the Norwegian project will be briefly presented. The information is gathered from their respective websites, public documents available about the companies, from the research proposals (van der Linden 2005; Dahle and Conradi 2005), from the interviews, and from conversations with employees in the different companies.

13.2.1 Keymind

Keymind Computing AS, formed in 1998, is a small consulting company focused on developing, maintaining and supporting IT solutions. They have six employees, located in three different offices.

Their main project is digital screening, or mammography, within the Norwegian health care. Keymind develops, maintains, and supports the integration of digital equipment with existing IT-solutions in the mammography program. The mammography solutions, is a distributed solution located on several sites. The solution uses replicated databases to store the patient information and the test results. The patient information is among other things used to call in the patients automatically to regular controls.

Keymind is a user of open source components and open source tools. These tools and components are used in the development of their solutions. Keymind believes using open source can give them even further economical benefits then they have today. The cost of maintenance is proportional with the amount of code, and Keymind believe they can cut maintenance costs by using code maintained by a community. Keymind are participating in the COSI project to be able to analyze and further develop their use of OSS. For further information about Keymind Computing, see their website at `http://www.keymind.no/`.

13.2.2 Linpro

Linpro has since their beginning in 1995 focused on services around the Linux platform and use of open source software. Linpro is growing and with more than 50 employees, they are probably the largest open source based consulting company in Norway.

Linpro have competence within several areas like the Linux desktop, security, project management, network, databases, several programming languages, and operation of IT systems. They provide development and integration services, support, some products developed by themselves, training, and courses. They also provide operation of IT systems for several big Norwegian customers like Elkjøp, A-pressen, Netcom and Telenor. In addition to these services, they have their own webshop which is used to sell Linux related products and some server hardware.

Linpro provides services on an open source platform, they develop software using open source tools and components, and the whole company has it's roots to the open source community. All this makes Linpro a very interesting partner for the COSI project. Linpro wishes to improve their cost/risk-evaluation, delegate more of their software development to open source communities, and improve their relationship in the open source community. More information about Linpro is available at `http://www.linpro.no/`.

13.2.3 eZ systems

eZ systems[1], established 1999, has since their start-up had success with their Enterprise Open Source Content Management System, eZ publish. eZ has their main office in Skien, Norway, but they are also located in Oslo (Norway), Ukraine, France, Canada and Germany. They have at the moment more than 70 employees from close to 20 countries, but are still growing. Through participation in the COSI project, eZ wishes to gain insight into how they can grow to become a larger global actor than they are today.

Their most important product is eZ publish. It is an open and freely available content management solution developed in php. eZ systems provides services, product responsibility, and other related software to this solution. eZ Publisher is licensed under the GPL license and their own proprietary licenses. This allows the user to select which kind of licensing scheme he or she wants. Around their product they have a community consisting of customers from all over the world, and more than 20 000 registered users. More information both about the community and about eZ can be found at their website `http://ez.no/`.

[1]eZ systems is pronounced "easy systems". It is spelled with lowercase e and s, and uppercase Z

Chapter 14

Collecting Evidence

We will in this chapter describe how we collected data and information from our interviews and from the respondents to our questionnaire. We start by describing how we conducted the interviews before we describe how we improved our questionnaire, and the tool we used to gather responses to the questionnaire.

14.1 Interviews

Through the COSI project we were able to visit each of the three Norwegian companies in the period March 13th, and 15th. During these visits we conducted interviews with one, two, or three employees (in groups), see Table 14.1. The interviews were conducted in Norwegian, as semi-structured interviews using an interview guide and an early draft of the questionnaire to structure the conversation (Appendix C). To allow the interview objects to be prepared for the interview, we sent them the draft of the questionnaire. The draft helped us structuring the conversations, at the same time we got valuable feedback on the questionnaires structure and content. We focused the interviews using only the part(s) of the questionnaire relevant to each company, and the role they have related to open source.

Company	**Interview objects**	**Company role**
Keymind	2 developers	User of OSS components
Linpro	1 developer	OSS owner, OSS participant, and user of OSS components
eZ systems	Primarily 1 CEO and 1 developer, but another developer joined us for parts of the interview	OSS owner

Table 14.1: Interview objects

Our supervisor, Carl-Fredrik Sørensen, followed the interviews together with the researchers, as an observer. Both the researchers and our supervisor were already introduced to the interview objects before the interviews, and the objects were familiar with the purpose and the context of the interviews. This and the fact that the interviews were conducted in the offices of the companies allowed the employees to feel safe. The only unfamiliar objects in the surroundings were our audio recording equipment.

One of the researchers took notes during the interviews, using pen and paper, while the other researcher led the interview. To avoid loosing details and to reduce the problem of having a biased researcher taking notes, all interviews were recorded using a digital recorder.

The interview objects were most of the time allowed to talk freely about what they found relevant to the questions in the questionnaire, but what they said could be followed up with further questions. These were primarily asked by the researcher leading the interview, but at times also by the researcher taking notes and our supervisor.

The interviews lasted in total between three and four hours, only interrupted by short breaks. Even though the interviews were quite long, all the interview objects seemed to answer openly and patiently to the questions asked during the whole interview. This could indicate that they felt comfortable with the situation.

14.2 Questionnaire

This section will describe how we improved our questionnaire using feedback from project participants and from colleagues at NTNU. We will present how the questionnaire was distributed, and how we collected data.

14.2.1 Improving the Questionnaire

While working with the questionnaire we sought advice from fellow students, PhD students, and our supervisors Professor Reidar Conradi and Dr. Carl-Fredrik Sørensen. We did this to remove any confusion and to improve our questions. The reviewers read through the questionnaire and provided valuable feedback both to our questions, with additional questions, layout and wording. The part of the questionnaire about inner source participation came as a direct result of one of these revisions.

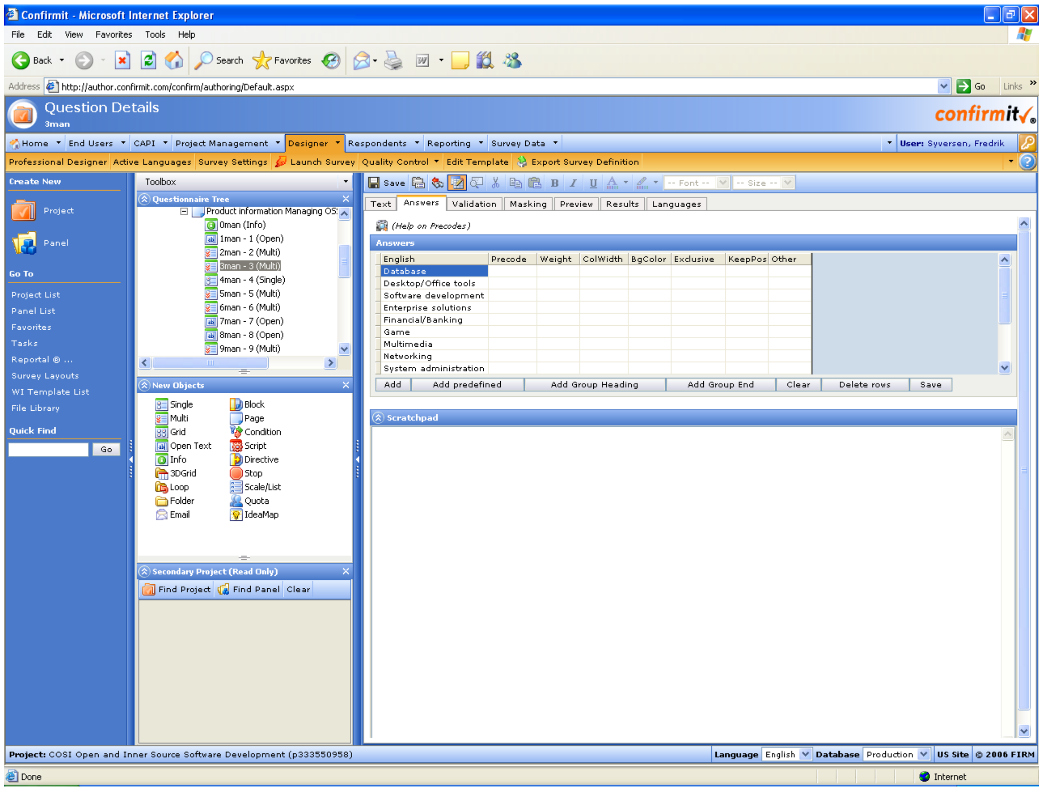
Drafts of the questionnaire were also sent out to the partners in the international COSI project. We got feedback from some of them and we were able to resolve some misunderstandings. One example of what we corrected after feedback from the project was the use of the terms "open source" and "inner source". These were reported to be used in a confusing way, and these confusions were resolved. Another example was confusion made by mixing the terms "open source project" and the term "open source product". And a last, and pretty important example, is the scope of the response. Originally, we wanted the respondents to answer on the behalf of their company. This should be unproblematic for SMEs, but it would lead to difficulties for big, global companies like Nokia and Philips. We therefore changed the scope of the response to concern the respondent's business unit.

The questionnaire was also sent to the Norwegian companies before the interviews. We used, as mentioned, these drafts to both structure the interviews and to get feedback on the questionnaire. The feedback from the interviews was used to improve the questionnaire further. Many new and interesting questions were found through the interviews, some questions were also found to be uninteresting and they were therefore removed. The writing of several questions, were also improved to avoid misunderstandings.

One example could for instance be one of the questions related to use of open source components. The original statement "Requirements to the product were changed a lot" was changed to "Requirements to the product your company developed were changed a lot", because some of the respondents thought it was the requirements to the component which were changed, not the requirements to the final product which embody the component that were changed.

14.2.2 Implementation and Distribution of the Questionnaire

The final implementation of the questionnaire was done in a tool provided by ICT-Norway, called confirmit[1], `http://www.confirmit.no/`. ICT-Norway normally use conformit for their questionnaires, and they wanted us to use it too since the questionnaire is part of the COSI project. They kindly provided us with a license to the tool, and we were able to benefit from having a ready-to-use tool, and an infrastructure for data collection. conformit is an online tool for implementation and distribution of questionnaires. It also contains simple programming constructs, like if and loops, and some reporting and exporting features. It is possible to both do simple reporting in the web-interface, and to export data to e.g. comma-separated files, Microsoft Excel, or SPSS. The implementation of our questionnaire was done through a web-interface as shown in Figure 14.1.



Address: http://author.confirmit.com/confirm/authoring/Default.aspx

User: Syversen, Fredrik

Product information Managing OS:
- 0man (Info)
- 1man - 1 (Open)
- 2man - 2 (Multi)
- 3man - 3 (Multi)
- 4man - 4 (Single)
- 5man - 5 (Multi)
- 6man - 6 (Multi)
- 7man - 7 (Open)
- 8man - 8 (Open)
- 9man - 9 (Multi)

Answers

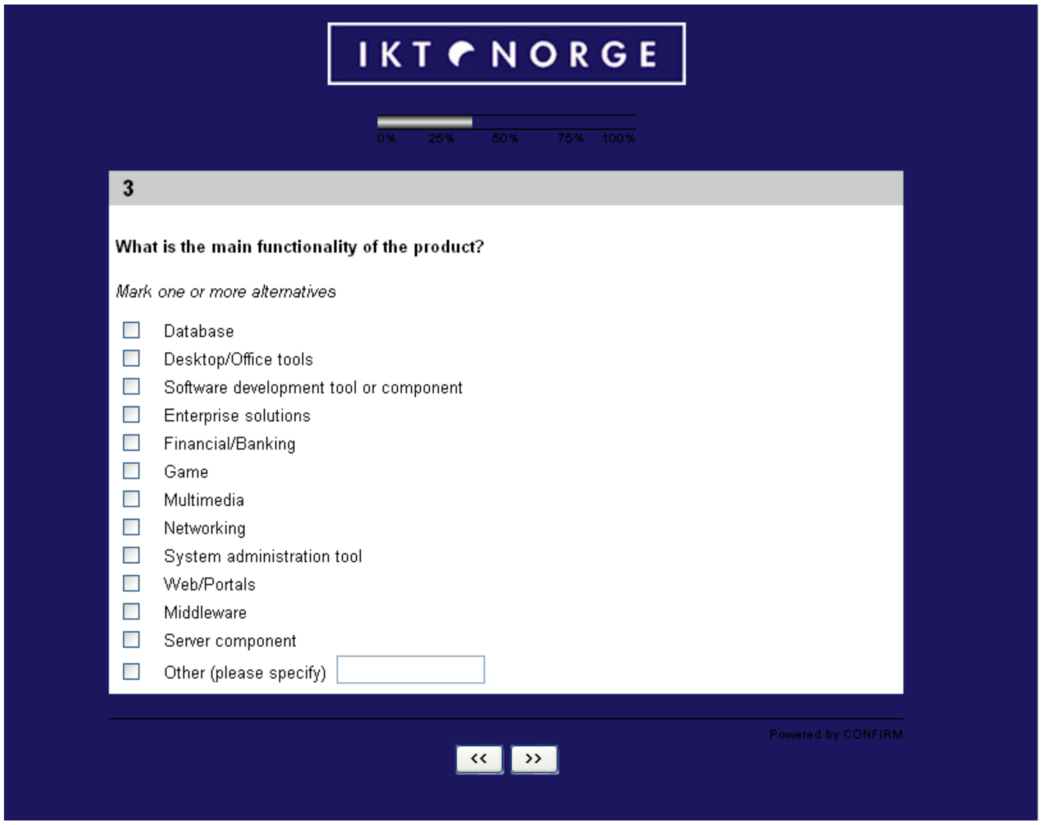
English	Precode	Weight	ColWidth	BgColor	Exclusive	KeepPos	Other
Database							
Desktop/Office tools							
Software development							
Enterprise solutions							
Financial/Banking							
Game							
Multimedia							
Networking							
System administration							

Project: COSI Open and Inner Source Software Development (p333550958)   Language: English   Database: Production   US Site   © 2006 FIRM

Figure 14.1: Conformit web-interface when designing a questionnaire

Having such a tool allowed us to focus on the development of questionnaire, and leave everything else to the tool. The development of the questionnaire was started before conformit was available, and it was done using a word processor. The implementation in conformit was done quite fast, and the tool provided us with predefined templates which gave our questionnaire the same look as other ICT-Norway questionnaires, as shown in Figure 14.2.

[1]confirmit is actually written with a lowercase c



3

What is the main functionality of the product?

Mark one or more alternatives

- ☐ Database
- ☐ Desktop/Office tools
- ☐ Software development tool or component
- ☐ Enterprise solutions
- ☐ Financial/Banking
- ☐ Game
- ☐ Multimedia
- ☐ Networking
- ☐ System administration tool
- ☐ Web/Portals
- ☐ Middleware
- ☐ Server component
- ☐ Other (please specify)

Figure 14.2: Conformit web-interface as seen be respondents

14.3 Problems while Collecting Data

All the industrial partners in the COSI project committed to respond to the questionnaire, and the questionnaire was sent out to these partners. Our supervisor, Carl-Fredrik Sørensen, who is NTNU's contact person in the COSI project, was responsible for the distribution of the questionnaire. He helped us by contacting the COSI partners and asked them to complete the survey. Getting responses was unfortunately more difficult than expected, and contacting the partners had unfortunately to be done several times, both through email and by phone.

Time is always a problem and some of our interview objects did not have time to prepare as much as wanted for the interviews. It did not turn out to be a problem but we could have received more feedback on the questionnaire, if they could have used a bit more time to prepare for the interviews.

14.4 Feedback from the Respondents

Since the respondents are participating in the COSI project they should be interested in the results and motivated to respond to the questionnaire. We had, as mentioned, some problems getting responses, but we hope the results will be interesting for the respondents. One of the respondents commented that "I am very interested in the outcome".

We knew that the scope of the questionnaire, were quite large, but since we divided the questionnaire in four parts we hope that this would not be a problem. Because the research questions are explorative we included some open questions in the questionnaire. Some of the feedback we got on the questionnaire was related to these open questions, and the time used to complete the survey. One respondent commented "Asking to describe a process in a survey seems too much to ask", another said "To describe the development process takes a lot of time. Please try to make that part easier to fill inn (checkboxes or similar mechanism)."

Other respondents were uncertain about which part to respond to. Some of the bigger companies reported that it was hard to get people to respond because they were not working with open source, but rather global development. Others were uncertainty about which section to respond to, especially related to inner source. One respondent requested clearer distinction between personal participation in open source and participation initiated through his employer, saying "I have participated [in] OSS projects in two ways: - as a hobby - at work[.] I would have appreciated more clear distinctions between the two".[2] Other respondents were a bit uncertain about when you are considered as a participant. Is it enough to read send an email asking for help every now and then, or do you have to develop code and new functionality to the product to be considered as a participant?

The respondents reported to use between 10 and 180 minutes to complete the survey, with and average of about 45 minutes. This is perhaps a bit too much, and the scope of the questionnaire could perhaps be reduced and more focused, to get more responses, and to make it easier for the respondents to respond.

On the other hand, one responded suggested to include even more into the questionnaire, saying "The questionnaire was very good for OS/Inner source projects dissemination [and] for analyzing them thoroughly. There could also be a selection for OS platforms as they are quite usable today. eg. JBoss, Ez etc."

[2]Some spelling mistakes were corrected, and some words were inserted to make this and other comments more readable

Chapter 15

Analyzing the Evidence

In this chapter we will give a short presentation of how we analyzed the data recorded through the interviews and our questionnaire. The results are presented in Chapter 16.

15.1 Interview

After the interviews were completed they were transcribed in Norwegian, using a word-processor. This was a time consuming process, but it was fruitful work. By transcribing the interviews we had to listen through the interviews. This made us reflect over the conversations, and the things which were said during the interviews. This reflection made us able to find important topics from the interviews. These topics were discussed, and we used the transcriptions, and the recordings of the interviews to find more information about these topics. The selection of interesting topics is of course a totally subjective process depending on the individuals doing it.

We used the transcriptions to find quotes and topics from the interviews, but we also used the recordings once more to get the wording of the statements exact, and to fill in some gaps in our transcriptions. To back-up our findings, we translated some statements and included them in the thesis as quotes. Translating a quote directly from Norwegian to English is difficult, but we tried to preserve the original wording.

15.2 Questionnaire

The results from the questionnaire were gathered using the web-tool described in the previous chapter. We exported these results to a spreadsheet, and included them into the original questionnaire, see Appendix B. This document contains the answers from respondents to the parts of the questionnaire related to the research questions, while parts 1 and 6 are removed to ensure the confidentiality of the respondents. Answers to qualitative questions have been included without modification, while answers to quantitative questions have been reported as the number of answers to each alternative answer.

The respondents were taken from the COSI project. The number of respondents is quite low as a consequence of the small population. We had in total 24 responses from 11 different companies on the questionnaire. These responses were distributed on the four parts as shown in Table 15.1. One of the 24 respondents did not answer any of the four sections.

We used the word document with the answers to see if we could find any trends, or any qualitative answers that could give us insight into the respondent's relations to open source.

Part	**Number of respondents**
1 Initiating and Managing OSS	3
2 Participating in Inner Source Development	6
3 Interacting with or Participating in OSS Communities	5
4 Developing Software with OSS Tools and Components	9

Table 15.1: Distribution of respondents

The number of respondents was, as mentioned above, quite low, and the results are not statistically valid. Validity is further discussed in Section 17.3. The answers from the questionnaire were related to the issues discussed in the interviews and presented thematically in the next part of the document.

Part IV

Results and Conclusions

Chapter 16

Results

In this chapter we present our results from the interviews and the questionnaire. The process of discovering these results is described in the previous part of the thesis. The results will answer the research questions described in Chapter 11.

16.1 Overview of Results

The results of the interviews and the questionnaire are presented in the subsequent sections, organized according to the research question answered. First, Research Question 1 will be answered in Section 16.2 where the motivations for use of OSS components and the process of evaluating such components will be described. Next, Research Question 2 will be answered in Section 16.3 where we will describe why and how companies make their products open source. Then, Research Question 3 will be answered in Section 16.4 where the usage and motivations for inner source development will be presented. In Section 16.5, Research Question 4 will be answered by describing the influence usage of OSS has on the development processes in companies. Finally, the answers to Research Question 5 will be presented in Section 16.6 where company participation in OSS projects will be described.

As described in the research design in Chapter 12, this study is based on an explorative research goal. As a consequence of this, several hypotheses will be created based on the results to the research questions. A hypothesis is defined in Webster's Comprehensive Dictionary of the English Language as:

> *"A hypothesis is a statement of what is deemed possibly true, assumed and reasoned upon as if certainly true, with a view of reaching truth not yet surely known; especially in the sciences, a hypothesis is a comprehensive tentative explanation of certain phenomena, which is meant to include all other facts of the same class, and which is assumed as true till there has been opportunity to bring all related facts into comparison; if the hypothesis explains all the facts, it is regarded as verified; till then it is regarded as a working hypothesis, that is, one that may answer for present practical purposes."* (Read 2003, 623)

Our understanding of this definition is that a hypothesis is a tentative explanation of certain phenomena that can be tested and either falsified or supported. We will in the next sections present several observations formulated as hypotheses. These hypotheses could be evaluated by other qualitative or quantitative findings. Some of them are already supported in literature, while others are new understandings.

The first hypothesis is based on an overall evaluation of the results from the interviews and questionnaire:

H 1 Companies participate in Open Source with the roles open source owner, open source participant, inner source participant, and user of open source components

This hypothesis is based on the 24 responses from 11 different companies to the questionnaire and the interviews in three companies. It is clear that companies have different roles related to open source, because companies answered one or more of the parts of the questionnaire related to their role in OSS. Our findings support the initial descriptions of the four roles described in Chapter 11.

16.2 RQ1: Using and Evaluating OSS Components

We will in this section present our results related to the first research question.

RQ 1 Why do companies use OSS components, and how do they find and evaluate these components?

16.2.1 Using Commodity OSS

When we asked our interviewees about the most important open source components they used in their solutions, the respondents mentioned software like Hibernate, MySQL, Apache, Linux, and other infrastructure software. The respondents to our questionnaire also gave the impression that the OSS components they used were primarily commodity software. Eight of nine respondents answered that between 80 and 100% of the open source software they used were either infrastructure or common to everyone in the market (Question 48). Only one of the respondents said that more than 20% of the OSS they used gave their software advantage over their competitors. One of the developers we interviewed stated that:

"We believe that, at least 60 percent of our code is open source..., it may be more, but I do certainly not believe we have less...*and 100 percent of it is commodity*" [Interviewee A].

Based on these observations we formed the following hypothesis.

H 2 Open source components are primarily regarded as commodity software.

This hypothesis supports the view of OSS presented in the COSI project. OSS components are primarily used as commodity software, providing only basic functionality to the solutions it is integrated into.

16.2.2 Motivations behind Using OSS Components

The reasons to use open source components could be many. We will here present some of our findings related to the motivations for using OSS components.

No Cost

The first and perhaps most obvious reason to choose open source is the price. Open source components can most often be acquired for free. Both the interviews and Question 42d in the questionnaire indicated that price is one of the main motivations for selecting open source.

For a software developer, getting a component that solves the problem is normally the most important issue. The actual cost of acquiring proprietary software licenses is usually not that high compared to the cost of the man-hours necessary to adapt the component. It is therefore better to have a component that requires little work. The fact that the open source components are free is perhaps not the most important motivation for using them.

Availability of the Component

The fact that most open source components can be acquired for free is closely related to another, and even more important motivation; availability of the component.

"The most important is actually easy access. It is easy to find it [the OSS components], it is easy to test it, insert it into a test environment, and see that it works then we can include it." [Interviewee A]

Fully functional open source software is normally available on the web site of the creators or at sites like SourceForge. The component can therefore be easily downloaded and tested free of charge, without any further problems. Proprietary software is mostly available only for a short test period or with reduced functionally. Eight of the nine respondents to the questionnaire also agreed that availability of the components is an important motivation for using OSS components (Question 42h).

Availability of the Source Code

Availability of the source code and the possibility to change it are other important motivations for using OSS components. Being able to read and change the code makes it easier to extend the component, and it allows you to find, fix, or avoid any defects in the component. Interviewee C said it was very important to be able to correct defects in the code because guaranteeing the quality of the component was part of their business model. Reading the code can also be performed to evaluate the quality of the code and thereby the component itself.

Question 42e indicates the same. Having the source code enables you to modify it. It is perhaps not always easy to change the code because of the complexity of the OSS component. Our interviewees preferred not to change the source code because it required them to understand the component, and it increases the number of dependencies to the component. However, they said it was assuring to know that they could change it if they had to.

Availability of Information, Openness and Honesty

The availability of information, openness, and honesty are motivations which were mentioned by our interviewees. Open source communities are usually open and honest about the status of their software. The history of the software and the number of open and closed bug-reports, are normally available on the product's web site.

"We think that there is more information around open source components in general than around commercial components and a lot more openness.... If there is something that does not work, then the community is not afraid to tell, which is very good." [Interviewee A]

This openness allows software developers to evaluate the real status of the component. Interviewee A said that openness was an important reason to choose open source components.

"What you see is what you get. You know what you get." [Interviewee C]

The distribution of the answers to our questionnaire, question 42i, was more evenly distributed, but we still believe that openness and honesty are advantages of open source software. Interviewee D said, as an open source provider, that openness was very important to them. They meant that openness and treating the community fair were important to keep the community members from turning their back on them.

Based on the findings related to the availability of components, source code, and information we have formed the following hypothesis.

H 3 High availability of the components, of the source code, and of the information related to the components is one of the most important motivations for using OSS components.

Software developers use OSS components because the components, the source code, and the information about the component are highly available. Providers of proprietary components have to react to this. If they want to succeed they have to increase the availability of their components, information about them, and perhaps also the source code. Making their components open source is perhaps one possible way of succeeding. How to make a product open source will be discussed in Section 16.3.

Standard Compliance

Open source and open standards go hand in hand. Open standards are important because they allow software developers to similar interfaces when they create software. Using similar interfaces gives interchangeability, which again enables genuine competition.

"More important than a market consisting of open source, is a market which consists of open standards where you have genuine competition between products." [Interviewee C]

Seven of the nine respondents to the questionnaire said that standard compliance was important to their company (Question 46a). Six of the nine respondents said that they mostly agreed that standard compliance was a reason to use OSS components (Question 42k). One respondent also mentioned open interfaces as one of the criteria when they evaluated OSS components. This indicates that standard compliance is quite important, but perhaps not crucial. Interviewee A explained this by saying that finding a component that solves the problem is more important than standard compliance.

Unimportant Motivations

Some motivations were also found to be of low or of no importance to the respondents. Questions 42f and 42g indicate that *company policies* and the *customer* seldom influence the decision to use open source or not. However, Interviewee C, who work in an open source company which base their business model on open source, said that company policies were one of the main reasons for choosing open source components. The other respondents to the questionnaire are not typical open source companies. It is therefore reasonable to assume that the type of company will influence whether company policies will affect the decision to use open source components.

16.2.3 Selection Process

The interviewees who use OSS components described informal search and evaluation processes similar to the familiarity, and trial-based selection process described by (Li, Bjornson, Conradi, and Kampenes 2004). See Section 7.3 for a description of these selection processes. The answers to Question 44 and 45 of the questionnaire also give more or less the same picture of OSS component selection. We will now describe an example of an hypothetical selection process based on these findings. We will also discuss how to find, and how to evaluate OSS components a bit closer.

A process for selection of OSS components can be presented as shown in Figure 16.1. The components are initially found through *informal Internet searches* or through *assessment of previously known components*. The developer may know of the component or may ask colleagues or a community for recommendations. The components found through these two sources form a long-list. These components and their communities are *evaluated* against the most important functional and non-functional requirements. Many of the components are rejected, and the developer is able to form a short-list of possible components. The components on this short-list are further evaluated by downloading and *testing* them. The short list is reduced to one candidate, which is *implemented* in the final product. In some cases the process may end up without any candidate component. Then, the developer must either search for other components or implement the functionality.

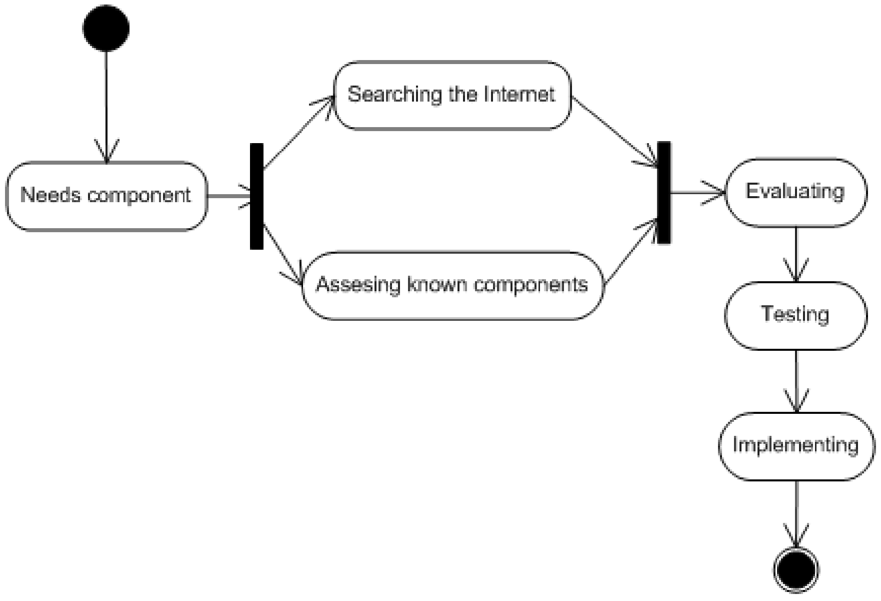


Figure 16.1: Selection process of OSS components

A problem with this process is that they do not know what they miss. Interviewee A said that they felt rather confident about their selection process and the outcome. They would normally describe the problem fairly well before searching for a component, and because the problems are well defined, they would normally find the right solution to the problem.

The selection process descriptions seemed to be quite informal at all the companies we interviewed.

"It is an undocumented and informal process..." [Interviewee A]

Even though the process is informal, interviewee A, B, and C had noticed that they used more or less the same procedure every time they searched for OSS components. They had thought about formalizing, and describing the process, but they had not yet done it. The processes seemed to be more formal in some of the companies which responded to our questionnaire, but their answers were quite brief so this has to be investigated further.

Based on the interviews and the results from the questionnaire, we believe that many software companies, which use OSS components, are following a process similar to the one described above. We also believe that quite often, no process is used because the choice is already made based on previous experience and knowledge. When asked about why they choose to use Perl one developer answered:

"The ones who started [the company] were Perl enthusiasts and they continued solving their task, in Perl" [Interviewee C]

We believe that this kind of decision making is quite common, you use or continue to use something because you know it. It is perhaps not the perfect solution, but since you know it is easy to continue using it.

16.2.4 Finding OSS Components

Finding OSS components are primarily done in two ways. Firstly, you can have knowledge about some OSS component yourself or you can ask someone who have knowledge about components. Asking someone else can for instance be done by talking to people at your company, asking expert networks, ask in mailing lists, or ask in web-forums, or by calling former colleagues.

To learn about more components, it would help to visit community sites like SourceForge, and Slashdot, subscribe to mailing lists, and to read discussions on web-forums who are concerned with OSS and OSS development.

The second way of finding OSS components is through Internet searches. These searches can be done through search engines like Google or Yahoo, or through some of the many sites mentioned by the respondents, see Table 16.1.

Site	**Url**
SourceForge.net	`http://sourceforge.net/`
Freshmeat.net	`http://freshmeat.net/`
Tigris.org	`http://www.tigris.org/`
Java-Source.net	`http://java-source.net/`
Javalobby	`http://www.javalobby.org/`
TheServerSide.com	`http://www.theserverside.com/`
IBM's resource for developers	`http://www-128.ibm.com/developerworks/`
Slashdot	`http://slashdot.org/`
Kuro5hin	`http://www.kuro5hin.org/`
koders	`http://koders.com/`

Table 16.1: OSS Sources on the Internet

The sources in Table 16.1, and search engines can either be used to find components or comparisons of several components. Interviewee B said he preferred to look for lists of evaluated components, because this was time-saving.

16.2.5 Evaluating OSS Components

A problem with Google and other search tools is the amount of information they give you. To be able to evaluate this information, and the components you find, you need some sort of evaluation procedure. Our findings related to evaluation of OSS components are presented next.

The evaluation, as presented above, is a two step process. It is unlikely that the process is totally linear in real life, but we saw a division between early or brief evaluation, and late and more careful evaluation or testing.

In the brief evaluation, a component could either be rejected pretty fast, or kept for closer evaluation. The evaluation criteria used for this early evaluation depend on many factors and we have not been able to see a clear division between the ones used early, and the ones used later. Evaluation criteria like checking the main functionality or just a first impression of the component may make the developer reject a component. The selection processes are quite informal; because of this most evaluation criteria depend, on the individual, the project, the requirements to the project, or possibly several other factors. The outcome of the process is because of these dependencies, not deterministic. We will present some of the evaluation criteria we discovered later.

The late evaluation is a closer evaluation, often testing. The component could be downloaded and tested in a small prototype. The developer may review the documentation, and perhaps also the code.

Interviewee C said that reading the code was a basic part of the quality assurance work they did for every open source product they used. This part of the selection process was a bit more formal. They used checklists to check the maturity of the functionality, the project, the documentation, the fulfilment of the customer's requirements, and so on.

The developer may also pay attention to the mailing list and web forum for some days, or maybe a week or two to check the activity of the community.

The companies we interviewed did not feel that this evaluation process was particularly time-consuming. They were also satisfied with the outcome of the process, most of the time. However, some problems were mentioned. They could at times be unsuccessful finding any useful components. When this happened, they had to implement the functionality themselves. Other times they had underestimated the work necessary to implement the selected component, but we got the impression that this happened very seldom.

16.2.6 Information Sources

We have already mentioned some sources which can be used to evaluate OSS components. Information about many open source projects can be found at the web sites listed in Table 16.1. Our interviewees and our respondents also mentioned recommendations from peers or colleagues, community mailing lists and web forums, books and magazines concerning open source, project web sitez, the source code, the project documentation, and project statistics online, as other important information sources. The documentation seemed to be particularly important. This includes code documentation, user documentation, and project documents like roadmaps, change logs, and so on.

16.2.7 Evaluation Criteria

"Both the code, the project documentation, the wiki, and the mailing lists have to create trust" [Interviewee C]

Creating trust is important and we will now present some of the criteria developers evaluate OSS components against.

Functionality

The first and most obvious criteria, is functionality. A component without any of the required functionality will be discarded immediately. Working with a badly written component can be very time-consuming, thus functionality is not more important than everything else. Interviewee C said, it is better to have a high quality component which lack some features, than a fully featured component with low quality.

License Types

Another quite obvious criteria, is the type of license(s). All our interviewees were very concerned with licensing. The license has to be compatible with their software and the customer's needs. It was normally not a problem since most OSS components are licensed with the BSD, MIT, LGPL, or the GPL license. Several of the respondents to the questionnaire also mentioned licenses when they listed evaluation criteria.

"We are always very concerned with tidy licensing." [Interviewee A]

Maturity and Condition of the Component

Maturity and the condition of the product are other criteria which were mentioned by several respondents (Question 45, 49g, and 49h). By maturity and condition, we think of the stability of the component. Is it available in a stable release, or is it just an alpha release? Our interviewees seemed to be a bit conservative when it came to new functionality. They preferred to use stable releases with a track record and a reputation instead of using components with leading, or bleeding edge functionality. Immature and unstable products would be rejected more easily, because of the fear of bugs or defects.

Other respondents did not find stable releases and a good reputation as important. We do unfortunately not know why, but we are still quite convinced that the maturity or stability, and the reputation of both the product and the community are used as criteria when evaluating an OSS component.

Development Process

The development processes used in the community are related to the maturity of the community. Interviewee C said that he looked at the development methods in the community when he evaluated an OSS component. If the community had unit-testing, frameworks for testing, and procedures for bug reporting, this would be sign of quality.

Future Versions, Backwards Compatibility and Maintenance

Interviewee A said that because they built and delivered solutions to a customer, they thought only about the component(s) available today. They did not consider future versions and future functionality. However, one respondent to the questionnaire said that prospect for future development was one of their criteria when they used OSS components.

Interviewees A, B, and C said that updating to a new version of the component could be challenging. Especially when updating to a totally new version of the component. Updating to a new version may require a lot of work. You have to learn the new version and you have to do the work of integrating the new version with your code. Why would you do this as long as the old version works fine? If the old version works, there may not be any reasons to replace it. Keeping an old version could lead to maintenance problems. If the community stops to support the old versions of the software, you will have to do all the support yourself.

Especially if a component is not backwards compatible, it could be difficult to upgrade to the new version. Interviewee A and B said that they had not implemented the new version of Hibernate, because it was not backwards compatible. Upgrading to the new version would just be too much work. In contrast, interviewee C meant that some open source communities are very concerned with this, and that the focus on backwards compatibility has increased a lot the last years.

Documentation and Information

It is not only the maturity of the component that is important. Also the maturity of documentation (Question 49b). If the documentation has low quality or low availability, this could be an indication of an immature product. Information about OSS components is, normally available but lack of information will make it harder to use the product, and it is therefore a reason for rejecting a component. Interesting information could be any kind of information about the product itself or about the community.

Reputation

Information about the reputation the track record of an OSS component, and its community seemed to be quite important. If no reputation or track record were available, this was seen as a bad sign because it was unlikely that anyone used the component.

"Lack of reputation is a bad sign" [Interviewee B]

Checking the reputation and references to the product and its community seems to be an important part of the evaluation process.

"We always check references related to others who have used it [the component], and the reputation of the component." [Interviewee A]

One of the respondents to the questionnaire also said that the reputation of the provider was important. If the component was provided by someone like Apache or if it was supported by an industrial partner like IBM, this was seen as a proof of quality.

Activity in the Community

Closely related to the reputation of the community and the reputation of the product, is the existence of a community and the activity in the community. A product with a good reputation is more likely to have an active community than a product with a bad reputation. An active community will most likely make it easier to get help if you have a problem, new features are more likely to be implemented, and bugs are more likely to be reported and fixed faster. Question 40e indicates that this is important to the respondents.

One possible disadvantage of having a big or strong community could be that it is harder to influence the community and thus the requirements to the product. Interviewee D said that because they use PHP, they are able to influence the development of PHP. If they chose Java this would have been impossible because Java is controlled in a lot stricter way than PHP.

The activity can be measured by looking at the number of *downloads*, number of *users* and *developers*, number of *messages* on the mailing list or on web forums, the number of *updates* of both the software and the documentation, the number of open and closed *bug reports*, and so on. In addition, many of the web sites like SourceForge provide *statistics* for the available products.

Support

The chance of getting support is closely related to the activity in the community. Getting support can be done by reading project documentation, subscribing to mailing lists, or by posting inquiries on a mailing list or a web forum. In a community with no or low activity, you are not likely to get any answers.

The lack of support is one of the challenges with OSS components. Some of our respondents answered that lack of support is a reason to discard an open source component (Question 49a). Others, for instance Interviewee C, said that providing support to and guaranteeing the quality of open source components are parts of their business idea. This could perhaps explain the answers to Question 49c where some answered that it is important that someone guarantees the quality of the OSS component, while others mean it is not important at all.

Implementation

The quality of the implementation is of course important, Question 49f. A high quality component is easier to use and modify. Interviewee C said he would check the quality of the implementation by inspecting the code. This inspection would be performed by starting at the main class of the program and look at the most important modules, and some random modules. He would check if it was programmed i a confidence-inspiring way, if classes, method, and variable names are reasonable, if the code was documented, if the code is understandable, if the structure is clear, if it is easy to extend the program, and so on. Exactly how the implementation creates trust is difficult to say, but as a programmer you use your gut feeling. You will trust code written in a way, you as a developer understand.

"You get a sense of these things right away, when you look at the code." [Interviewee C]

Interviewee D said, that one of the reasons why their open source solution gained popularity, was because it had clean and tidy code which developers could understand and use.

Another important issue related to implementation, are dependencies, Question 49i. If the component has many dependencies to other software it will more often be rejected.

"The worst components are those you are going to use only a small part of, and they create a lot of dependencies to other components. That is something you try to avoid." [Interviewee A]

Having dependencies is at the same time a cost benefit question too. Interviewee C said that removing all the dependencies is demanding, and as a consequence, the software would be very expensive. Most customers would not pay for perfect software. The developer did not see big disadvantages either, because it was likely that the customer would contact them later to upgrade their, not-perfect software.

Other Non-functional Qualities

As with every other component that is being evaluated some non-functional qualities of the component may be important. We believe this is depending more on the solution the developer is making, than the fact that it is an open source component that is being evaluated. Some of the qualities which were mentioned were performance, scalability, and open interfaces.

16.2.8 Summary and Challenges Related to Using OSS Component

Based on our findings, and supported by the findings of (Li, Bjornson, Conradi, and Kampenes 2004), described in Section 7.3, we propose another hypothesis.

H 4 Selection and evaluation of OSS components, is primarily done as a process based on previous knowledge, informal searches, and subjective evaluations of the components.

This hypothesis is explained in the previous sections. It indicates that software developers do not use any formal process when they search for and evaluate OSS components. Developers do however use similar procedures every time, but they may vary from developer to developer.

We have answered the first research question by presenting motivational factors for using OSS components, and we have presented a hypothetical selection process. Some of the lessons learned are summarized in a preliminary guideline we have made, see Appendix D.1. This guideline is meant as an input to the work in the COSI project. We hope this guideline can function as a basis for future work. We have also recognized some challenges related to use of OSS components. These challenges will be discussed in the remaining part of this section, and they could be a basis for future research.

The selection processes we described here are informal and it is not deterministic. Having a formal and deterministic process is not a goal in itself. Most of the time good enough is good enough, and finding a component that solves the problem is sufficient. However, some of our interviewees said they used more or less the same procedure every time they evaluated OSS components. Describing these procedures, and increasing the awareness of them, can be something that might help the companies using OSS components.

Most companies do not have extra resources to evaluate tools and components if they do not need them. The amount of OSS tools and components is enormous, how can a company deal with this problem? Interviewee A said that it was very difficult for them to assess new open source tools, simply because they were so small. Is it possible to provide some sort of method or infrastructure that could help companies to assess such tools and components?

Another challenge we discovered during our interviews is related to maintenance. It is a twofold challenge. First of all, when should a company upgrade to a new version of an open source component. Upgrading could, as we have commented earlier, require a lot of resources. Avoiding upgrading could leave you with a lot of maintenance work.

The second challenge related to maintenance is, when should you exchange code written by yourself with OSS? This is recognized by COSI as one of the software shifts. Closely related to this is the question of, when should you start to make your software open source, or when should you start to participate in OSS development, instead of developing this software, yourself?

16.3 RQ2: Companies Initiating and Managing OSS

Our second research question is related to companies which initiate and control their own open source products.

RQ 2 Why do companies choose to make their products Open Source, and how can this be done successfully?

Our findings related to this research question will be presented in this section. We will start by showing some of the motivations behind the decision to make a product open source. Then we will discuss some of the advantages and disadvantages of offering an open source product. We report some important issues related to going open source with a product, before we present some challenges related to offering an open source product.

Many companies will probably face challenges related to the software shifts described in the COSI project. If you know that some technology do not give you any competitive benefits, it could as well be acquired or developed as open source. Developing these components alone will only give you extra development costs and maintenance expenses.

"We are at the moment doing something related to the Web client, Ajax, which most likely will be commodity in relatively short time." [Interviewee A]

If you know that some technology will be available as open source soon, the question is why do not you make it open source yourself, or together with others? All the companies we interviewed, face or have faced this challenge too. How and when to make a product open source?

16.3.1 Motivation for Making an Open Source Product

There may be other motivations for making a product open source than saving development and maintenance cost. One respondent to Question 15 in the questionnaire said that they wanted to make an open source product because making the software open would help spreading their software all over the world. Many users would result in many potential customers, and some of those potential customers would be likely to pay for the services provided by the company.

The part of the questionnaire which treated open source management, did not have more than three respondents and it is hard to say anything certain, because of this. *Increased attention*, *saved effort by shared development*, *increased product quality*, *reduced time-to-market* because bugs are found faster by a community, were motivations that all three respondents agreed to in Question 10. *Idealism*, and *community contribution* were other motivations mentioned during the interviews.

"We thought it was the right way to do it [releasing the product as open source]...then we had to find out how to make money on it later" [Interviewee D]

The motivations of making a product open source are thus closely related to the advantages of owning an open source product.

16.3.2 Possible Advantages and Disadvantages Owning an Open Source Product

There are both possible advantages and disadvantages related to owning an open source product. By just owning an open source product, you may benefit from increased attention and publicity, and you may get some credit for giving something back to the open source community. The real benefits will first come when you manage to attract a community around your product. Some possible advantages and disadvantages related to having an open source product will be presented next.

Availability and Openness

In Section 16.2, we reported that availability was one of the most important reasons to use OSS components. An open source provider needs to be available to attract customers and new community members. Being available means both making people know about your product, and keeping them interested in it. In other words, you need to make your product famous and you need to be open to your community members and treat them in a way that makes them stay.

Openness is important in open source communities. It is easier and more motivating to contribute if you are informed. Openness also works as a trust-making guarantee of quality. Being open may also give you a lot of valuable feedback, for instance through open mailing lists or open discussion forums on your web site. This feedback, or perhaps critics, enable you to improve, but the openness makes you vulnerable too.

"We cannot hide." [Interviewee D]

When you are open to everyone, you cannot hide anything. If for instance, an open source provider releases information about all the bugs in the system, he is forced fix them. If he does not fix them, then everyone can see the amount of bugs in the system. The openness might hurt you instead if you unable to fix the bugs. If customers see all the open bug reports they may loose faith in your product.

H 5 Opening up the development of an industrial open source product may create trust and generate feedback, but it requires excellence from the open source provider.

Market Related Issues

Most open source software can be downloaded for free. If you only live of selling your software, you should perhaps not give it away. When people can download your software for free, you have to focus on the customers, who actually wants to pay for your licenses or your services. Interviewee D described some of the properties of what he saw as the differences between a business model for a traditional software vendor and a company which develops open source. These difference between are exemplified in Figure 16.2. According to him, open source companies focus more on maintenance and total quality guarantee, and on services related to their products, than traditional industry.

Many open source providers may sell licenses to their software as well, but to be able to sell licenses they need to have intellectual property rights to 100 percent of the code. Accomplishing this may be very hard if you do not think about it from the start.

People who buy licenses to an open source product, are customers who want to base their own product on the open source software and sell it to others. According to Interviewee D,

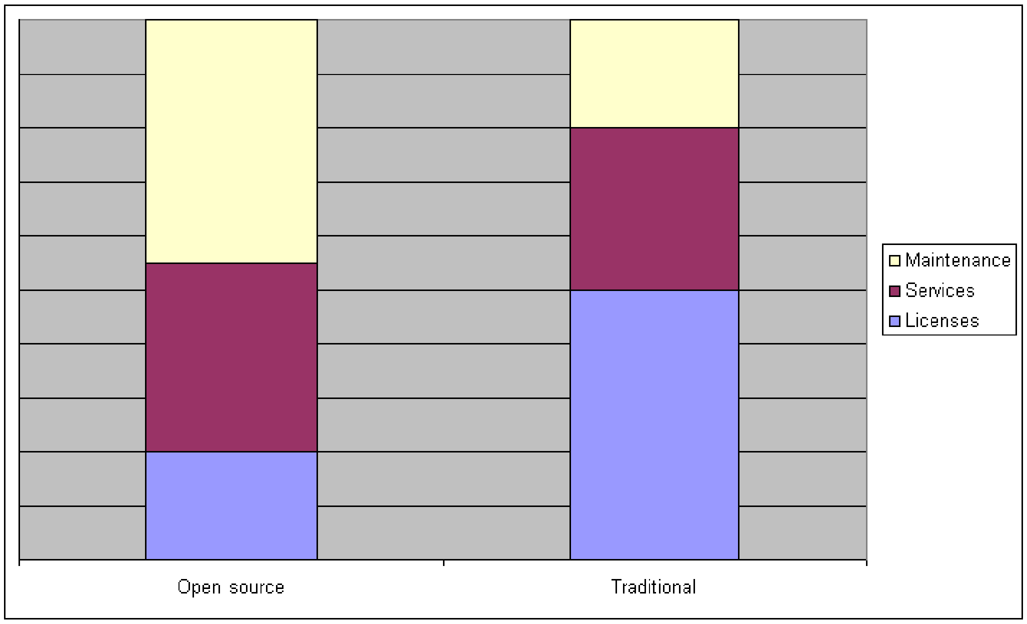


Figure 16.2: Share of income in a traditional and an OSS business model

it may be companies which do not want to distribute their software based on the open source software, or it may be big companies which do not have clear guidelines for how to deal legally with open source license. These customers buy proprietary licenses to the same software you can download for free with an open source license.

Publishing the product as open source means that it is available to anyone. This availability may give you the benefit of bottom-up sales. If someone, for instance a technician, download your product for free and start using it, his company might want to pay for your services later.

The availability may also make it harder for you to focus your marketing strategies because anyone may turn up as your customer.

"One of the challenges with OSS is that you get a horizontal marketing strategy." [Interviewee D]

If you have accomplished to attract a community to your product, you may benefit from the advantages of having a community. The community members are likely to talk about the product with colleagues and friends, and if they are satisfied with it, they will possibly save a lot of marketing expenses.

"We have a marketing department now, but for a long time we did not have it." [Interviewee E]

"It [the product] sold itself." [Interviewee D]

As the interest in the product and community grows, it is likely that someone starts to create add-ons, extensions, or complementary software to your product. These add-ons may create more and more momentum around your product, and thus increase the market. By increasing the market, new users and customers will be attracted to your product. You are perhaps not able to keep the same market share as in a smaller market, but as illustrated in Figure 16.3, it is better to have a small share of a big market that having a big share of a small market.

"What makes the biggest pie, not how can we get the biggest piece of the pie." [Interviewee

D]

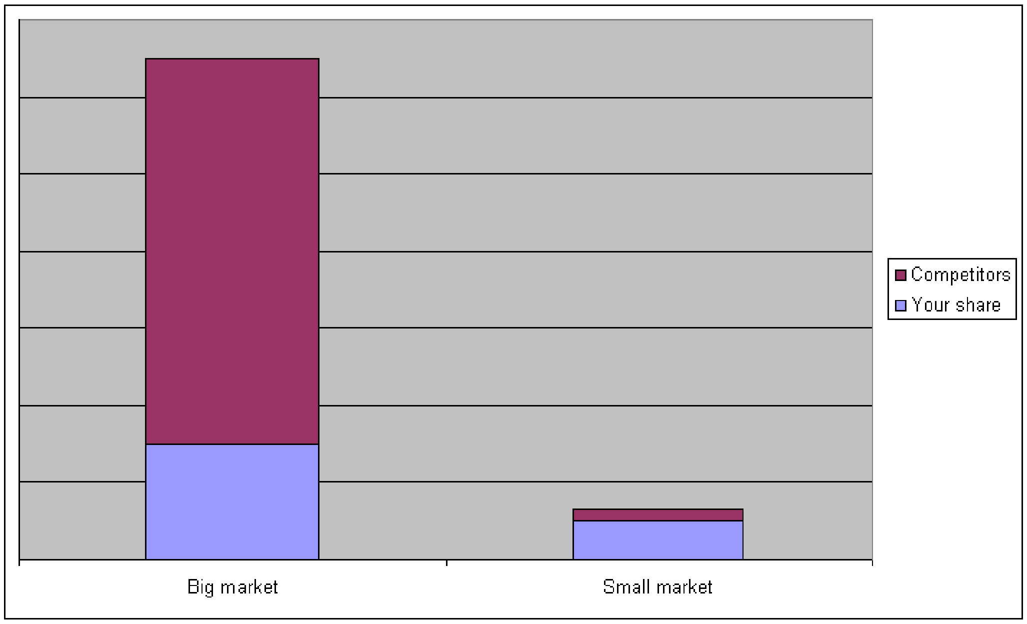


Figure 16.3: A big market with a small share and a small market with a big share

Development

Development of an open source product will only benefit from being open source as soon as there is an active community around the product. Many users means that your software is used and tested by many people, and it is more likely that bugs or defects are found. If more bugs are found and reported, you will be able to increase the quality of your software by correcting these defects. All the respondents to the questionnaire agreed that the product quality is increased because the community find more bugs, (Question 12l). Being able to benefit from the resources outside the company to find more bugs faster is good news for you. It means that you can cut costs related to testing, that your software matures faster, and it may also help you to reduce the time-to-market.

"We can probably fix more bugs than our competitors who are not open." [Interviewee D]

There is no need to wait until the product is released to involve the community. The community can give feedback in earlier stages as the requirements phase, planning phase, specification phase and so on.

"We have open specifications...and we get feedback already in the specification stage." [Interviewee D]

This is beneficial for both parts. You as a software developer, are able to reduce the risk by developing the software the market wants and the community is able to influence the product in the direction they want.

Contribution may come in different forms. It may come as feedback to specifications, new requirements, as a bug-report, or it may come as code. Some community members may provide solutions together with their bug-reports and thus solve the problem. They may also provide new functionality they want included in the product.

Getting contributions and suggestions is positive, but many contributions and contributions with low quality generate a lot of extra work. Someone will have to review the contributions, answer the contributors, and perhaps include code contributions or requirements into the product.

"We have seen that we have to rewrite many of the contributions we get and spend as much time as if we had done it ourselves." [Interviewee D]

To reduce the amount of overhead work, you need to find a balance between allowing community members to participate, and controlling the development yourself. Increasing the quality of the product and reducing the number of bugs before the software is released to the community, is one way to reduce the traffic from the community.

Clear routines for contributions and informing the community members of these routines is another way to increase the quality of the contributions. It is necessary to inform, perhaps also train, and educate the community members. This means keeping all your information up to date and being open to the community.

"We have clear code-standards, it [the code contributions] should not only work, but it should be done like this. ... It is our job to inform the community." [Interviewee D]

Another way of reducing the work related to contributions from the community, is allowing trusted members of the community to check in their own code. Benefiting from the contributions from many developers is good, but allowing many people to change the same code at the same time, may lead to coordination problems and it may break the build. It is therefore important to find a balance between having strict control over the product and its code, and allowing everyone to contribute. Interviewee D said that they tried to keep strict control over the core of the product and encourage innovation around it.

To benefit from many of the advantages related to having your product as open source, you need to reach a critical mass of community members, and you need to get a certain momentum in the community. The open source model will not scale if the community is small and inactive. If the market is small, it is perhaps not right to make your product open source.

H 6 The number of possible users in the market and the size of the community around your product severely influence the further development of the product and your benefits of having it as an open source product.

The hypothesis indicates that to be successful with an open source product it is very important to have a community around it. This is because most of the benefits of having the product as an open source product are achieved only when such a community exists.

Support

Another possible advantage of a community is the fact that they could support each other. Users help other users, as shown by (Lakhani and von Hippel 2002), and if you provide the necessary infrastructure, your community might also support itself, to some extent. A community may also lead to extra work load for your developers so it is important to keep a balance.

"If everyone in the community would contact them[the developers in the company], then they would never be able to do anything." [Interviewee E]

16.3.3 Making an Open Source Product

Making an open source product is easy, succeeding is a lot harder. Just licensing a product with an open source license will not give you any other benefits than perhaps increased attention and goodwill from users and potential customers. To be able to really benefit from an open source product, it is it is necessary to attract and keep a living community around the product. So what does it take to achieve this?

Interviewee C told about one project they did as an open source project a few years ago. This product was originally sponsored by an external sponsor. The sponsor did not have any technical competence themselves and they were depending on the company to do all the development. The product was only used by the sponsor and not internally in the company. When the cash flow stops, the motivation for maintaining the product also disappeared. Since the sponsor did posses any technical competence to continue the project, it ceased.

We observed some issues we believe influenced the success of this open source project. First of all, the company did not make any money continuing the development of the product. Secondly, the company did not have any internal use of the product. Thirdly, the developers were not users of the final product. Fourthly, there was no technically competent community around the project that could have developed the product further. This project was in our eyes, doomed to fail.

Keeping an open source product alive and in development, seems to be a task that requires some effort.

A company need to be motivated to set aside personnel to the development of an open source product. They may be motivated to do it if they can make money offering it as an open source product or if they use the product themselves. If the company does not have these resources, it is necessary to have a community with technical competence to maintain the development of the product. Based on these observations, we formed the following hypothesis.

H 7 Critical success factors for succeeding with an industrial open source products are: Your company can make money offering it as an open source product. The product is needed in the company. The developers are users of the product. An active community with technically competent people exists around the product.

The hypothesis indicates that there are many factors that influence the success of an industrial open source product. It is therefore not strange that we observed some insecurity related to the effort necessary to make a product open source. Interviewee B said he thought it would be easier to maintain a product without sharing it with others. He said that you do not need to think about what you can or cannot share with others when you keep it closed. He continued saying that you do not need to think about other stakeholders than yourself and your customer. Another developer commented:

"It is quite demanding to maintain it. You need to use a lot of time to maintain the webpages, documentation, mailing lists, and creating a community". [Interviewee C]

We have seen that it requires some resources to be successful with an open source product. If you do not have anything to loose by just licensing a product with an open source license and make it available to the world, you may do that and hope that someone will start working on it. To be successful and to be able to attract a community, it is important to do more than just making the product available. Interviewee C said that they presented their product on both national and international conferences for the targeted user group. Interviewee D told that they posted their product on several web sites like freshmeat.net, wrote articles about it, published papers, were active in other open source communities, in web forums, and in other open source related settings, to create publicity around their product. Based on these observations, we have formed the following hypothesis.

H 8 You must as an owner of an open source product work to attract a critical mass of community members. When you have succeeded, you need to work hard to keep them.

The hypothesis indicates that community members do not just turn up. As an owner of an open source product you need to work hard to attract and to maintain your community.

16.3.4 Communication in OSS Projects Managed by Companies

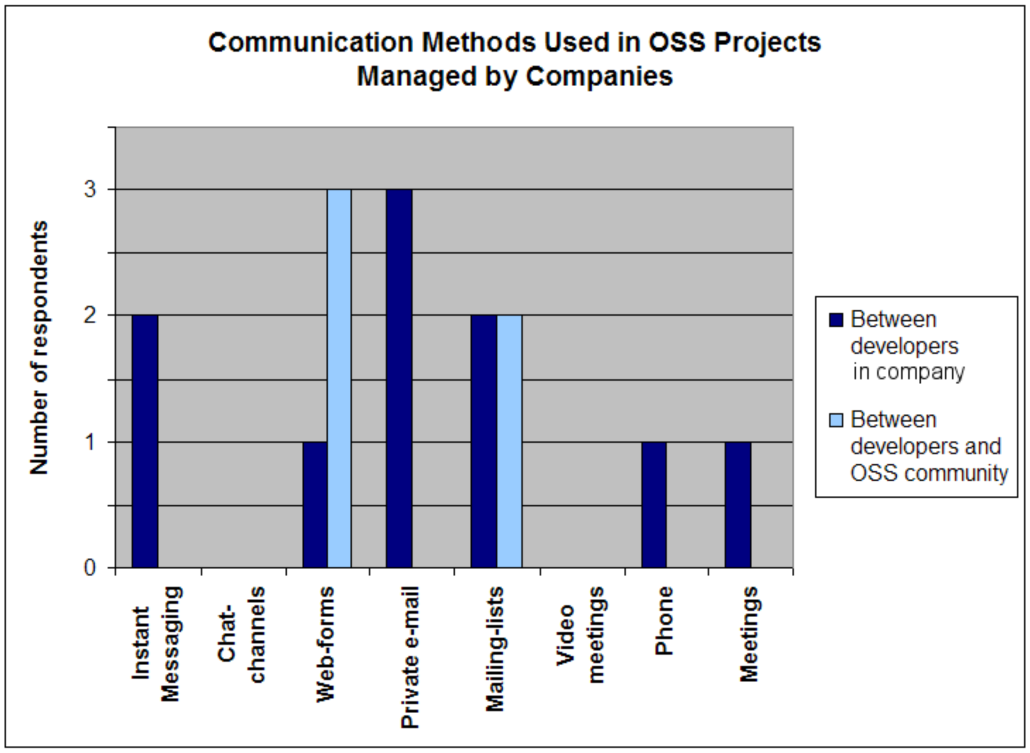


Figure 16.4: Communication methods used in OSS projects managed by companies.

OSS projects consist of people with the roles core-developer, developer, active users, and passive users, according to the prestudy in Section 5.2.1. Which communication methods do these roles use when they participate in OSS projects managed by companies? Figure 16.4 shows the results of communication methods used between *developers inside* the company, and between *developers in a company and community members.* The results are based on 3 respondents to Question 22, and 23 in the questionnaire. Participants using a particular communication method to the extents "Large" or "Very large", were summed. The results are shown in the Figure. Respondents using communication methods to a lesser extent were ignored because it was necessary to focus on the communication methods most important to the respondents.

The results in Figure 16.4 indicate that developers on the inside of the company communicate primarily using instant messaging, private email, and mailing lists, while the communication between developers in the company, and community members primarily are through web-forums and mailing lists. These results suggest that developers inside the company communicate using other methods than when they communicate with the members of the OSS community. This suggests that there is a communication-gap between the company and participants in the OSS community surrounding OSS projects managed by companies. This can be perhaps be explained with the fact that it is necessary to shield the developers from some of the communication in the community.

16.3.5 Discussion of OSS Management and Related Challenges

This section has been used to respond to the first research question. This has been done by presenting issues related to motivation for creating an open source product, possible advantages and disadvantages related to this creation, and how to do this. This section is concluded by a discussion of what we see as the two biggest challenges related to offering an open source product, *how to attract a community*, and *how to benefit from having one.*

To be able to attract a large community and to be able to benefit from their contributions, it is important to have a good, well-documented product that is easy to use. If the product has few bugs, it is easy to use, and the documentation is good, then the amount of unnecessary feedback and contributions will be reduced. Many OSS products are used as components in other software, because of this it is important to have good user documentation, clean and well-documented application programming interfaces (API), and well documented code.

"This is a very big job to be honest." [Interviewee D]

To include more code from the community and their partners, it is necessary to increase the quality of the contributions. This could be done by standardizing the routines for contribution and informing the community and the partners.

Informing and involving the community and your partners is also very important, but you need to find a balance between full democracy and dictatorship. Time is never something you have too much of when you develop software, it is thus important that a community is not turned into something bad, by spending too much time on democratic processes.

"Democracy takes time compared to dictatorship" [Interviewee D]

Interviewee D said that they thought making a hierarchic structure for decisions and reporting and a flat structure for information was the right thing for them. It is important to inform the community and your partners. He continued, saying that it is important to stay true to the open idea because you build your name as a brand. If you turn your back on the community, they will turn their back on you.

To summarize the knowledge learned related to management of open source projects, we have made some guidelines. These guidelines can be found as Appendix D.2 and are meant as an input to the COSI project. We hope they could be used as a basis for further development and be used as a resource for the industry.

16.4 RQ3: Inner Source Development

This section will present our results related to the third research question.

RQ 3 Why and how do companies use development methodologies from Open Source, in ISS development?

Inner source development (ISS) has been described in section 5.3 in the prestudy, and was the topic of one part of the questionnaire. The definition of inner source development is the following: *Inner source development uses aspects of open source software engineering inside a business unit, company, or consortium. The software developed is only open and available to a closed consortium of partners.*

It was clear from the prestudy that very few published results could been found about the subject, which is why the research question approached this topic in an explorative manner. The results presented in this section are primarily from the questionnaire, but also to some extent from the interviews. The respondents to this part of the questionnaire consists of 6 inner source projects, from 4 companies participating in the COSI project.

Based on the results from the questionnaire, it was clear that companies had different ideas about the definition of inner source. One respondent answered the part of the questionnaire about inner source, when the company was using a distributed from of development called global software development. All responses to the questionnaire indicating that the company was using development methodologies from open source on the inside of the company was therefore accepted.

16.4.1 Characteristics of Inner Source Development

The following result, are considered important aspects of inner source development. These results are based on the answers to Question 28 and 30, of the questionnaire.

- Inner Source uses OSS software engineering practices within a limited scope such as a business unit, company, or consortium of partners.
- Development is often distributed at several physical locations and possible acros several time zones.
- Source code is often shared between the partners in the inner source project.
- Email and mailing lists are often used by participants in inner source projects.
- Code ownership is often practiced.

Note that there was some variety in the characteristics of inner source projects, so the list just presented contains aspects which were considered important to many of the respondents. It is by no means complete, nor can it be considered a final conclusion.

16.4.2 Structure of ISS Projects

Role	**Business Unit in Company**	**Company**	**Collaborating Companies**
Core Developers	21-50	11-20	11-20
Developers	101-500	51-100	51-100
Active Users	1001-5000	1001-5000	1001-5000
Passive Users	5001-10000	5001-10000	5001-10000

Table 16.2: Number of participants in an ISS project.

Table 16.2 shows the number of participants in an inner source project at company A[1]. This specific inner source project, was selected as an example of the organization of an inner source project.

The organization of the six inner source projects in the questionnaire were reported to be organized in different ways. They had different distributions of people: at a particular business unit, in the company, and in other collaborating companies. It is therefore not possible to draw any conclusions about all inner source projects based on these answers.

In section 5.2.1, OSS projects were described to have a layered structure with few core-developers, more developers, and even more active and passive users. The respondents from inner source projects are to a large extent also organized in a similar layered structure. If the participants who have the same role in different companies or business units are added

[1]The companies which responded to the questionnaire were given a letter-code to conceal their true identity.

together, we see that the overall project has a hierarchical structure. The organization of table 16.2 is consistent with the organization of typical OSS projects. It has a small number of core-developers and a large number of passive users. This observation can also be made on most of the other respondents to this part of the questionnaire.

16.4.3 Inner Source Development Process

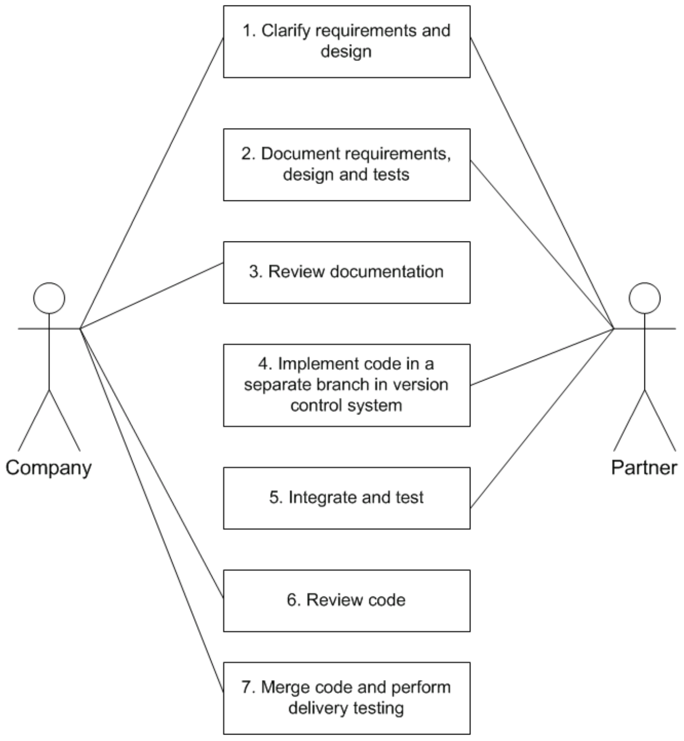

1. Clarify requirements and design
2. Document requirements, design and tests
3. Review documentation
4. Implement code in a separate branch in version control system
5. Integrate and test
6. Review code
7. Merge code and perform delivery testing
Company
Partner

Figure 16.5: Example of Inner source development process

The development processes used in inner source projects were reported in Question 29 of the questionnaire. One of the companies participating in the interview also used some aspects of inner source development. The qualitative answers to the questionnaire suggest that each of the inner source projects have a specially adapted development process. To illustrate the differences in these projects, some examples will be shown.

Company E reported that they use the Rational Unified Process (RUP) and managed the project according to a project plan. This development process is typically used in many traditional software development companies.

Company I reported that the project had a development process which was similar to the process used in many OSS projects. In that company, there was a core developer reviewing contributions and new requirements. These contributions were sent as patches to a mailing list. The source code was licensed with the GPL license.

Company A has a development process as shown in Figure 16.5. The development process is based on cooperation between developers at the company and developers at a partner company. The process is iterative, where steps 2 to 7 are repeated several times during the development.

Interviewee C stated that their development process was "emulating open source methodology on the inside" of the company. The company did not specifically call the development process 'inner source', but based on the description of the development process, it can be deduced that the development process shares many of the characteristics of open source development.

The following is how one of the interviewees describes the development process:

"We have been emulating open source methodology on the inside, an informal leader, and participants who participate as good as possible. The structure is very flat and the participants have the same rank and responsibility in the project solution. A work group of 2 to 5 persons is created without an appointed project leader and they create a solution-specification. We have traditionally tried to create requirements-specifications, solution-specifications etc. using the waterfall method, but during the implementation phase we have a very flat project structure where the participants have been almost equal, and together said that 'we are going to make this', without a clear leadership and delegated tasks. People have taken odd jobs as they become available from the todo-list." [Interviewee C]

Based on this description from the interviewee, it is deduced that the company is actually using many aspects of inner source development implicitly.

16.4.4 Communication in Inner Source Development Projects

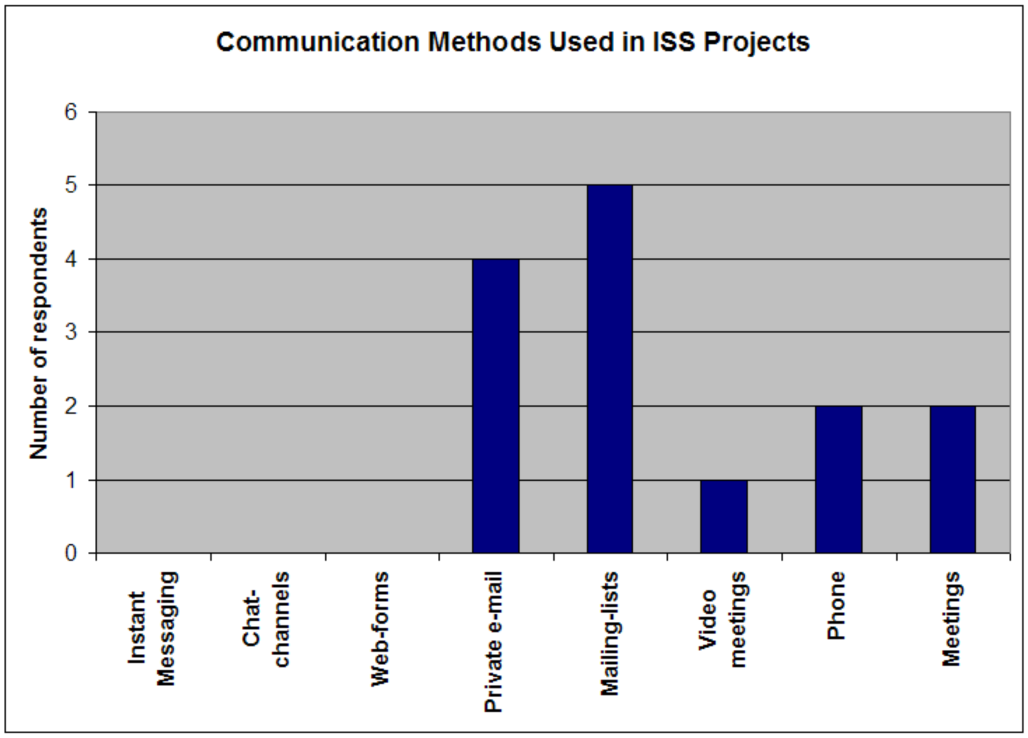


Figure 16.6: Communication methods used in ISS projects

In section 16.4.2, Inner source projects were described as being distributed at different locations, such as at different business units or collaborating companies. How does the participants in such projects communicate? Figure 16.6 shows the results of Question 28. This question is related to the communication methods used by participants in inner source projects. The results are based on 6 answers to this question. Participants using a particular communication method to "Large" or "Very large" extent were summarized and the results shown in the figure. Respondents using a communication method to a lesser extent or answering "Don't know" have not been included. The figure is only an indication of a *possible trend* of the communication methods which were most important to the respondents. The figure shows that e-mail and mailing lists are used by most participants, while phone and meetings are used to a lesser extent.

16.4.5 Motivations for Using Inner Source Development

Question 26 in the questionnaire dealt with experiences gained from companies using inner source development. The question was supposed to discover the motivations behind why companies participate in inner source development. The answers to this question seems to indicate that companies gain advantages by using inner source development such as allowing distribution of development and cooperation with partner companies. However, too few respondents answered this section to draw any conclusions.

16.4.6 Discussion of Inner Source

Based on the feedback from respondents about their understanding of the term 'Inner Source', it was clear that this is still not a very established term in the industry. A respondent from company A stated that their development model is classical and heavy-weighted, but it is distributed at different sites. Other respondents also answered that they were using formal development methods such as RUP. It can be hypothesized that there is some overlap between distributed development and inner source development.

Based on the results provided in this section, the following hypothesis about inner source has been formulated:

H 9 Companies use open source software engineering practices to perform distributed software development.

The software engineering practices mentioned in the hypothesis are based on the results presented in this chapter. Important practices which are used include the following:

- Sharing source code with partners
- Code ownership
- Using OSS development tools such as CVS and Subversion
- Communicating using email and mailing lists in a distributed setting
- Having a community of users and developers
- Using Agile development methods such as RUP and XP, in combination with inner source development.

The results related to the communication methods used in inner source projects indicate that participants in ISS projects communicate in ways which are similar to OSS projects. In section 4.4 of the prestudy, email and mailing lists were two important communication methods used in OSS projects. The results from the questionnaire suggest that ISS projects also communicate using similar methods.

Furthermore, the definition of inner source used so far has been described as the use of certain open source software engineering practices in companies. The results of the questionnaire give an ambiguous list of software engineering practices, commonly used by the respondents, who state that they are using inner source development. There is quite a lot of variation, and each company using inner source has adapted it to the company's situation. So which software engineering practices *must* be used for a company to be formally using inner source? This is an open question, which can be answered by further studies.

To summarize the knowledge learned about inner source projects, we have made proposals to some guidelines. These guidelines can be found as Appendix D.3 and are meant as an input to the COSI project.

16.5 RQ4: Development Processes when Using OSS

This section will present our results related to the fourth research question.

RQ 4 How does the use of OSS influence the development processes of companies?

To answer this research question, results from the interviews and the questionnaire have been collected and analyzed. The results from the interviews are loosely based on the open-ended Question 26, 45, and 48 from the interview guide. These results will be presented in Section 16.5.1. The results from the questionnaire are based on Question 31, 33, 40, 43, 46, 50, and 51 from the questionnaire, and will be presented in Section 16.5.2.

16.5.1 Interview: OSS Related Development Processes

The results of the fourth research question will be presented, based on the structured interview with three companies participating in the COSI project. The results will be presented as three separate examples of the development processes, used in the three companies. Each of the companies has a different OSS related role and we will present the development processes as the interviewees themselves describe their processes.

Case 1: Development Processes of Company Using OSS

Interviewee A and B answer on behalf of a company using OSS components. Interviewee A states that the company has their own description of the development methodology. This methodology is based on Feature Driven Development. The development process is an iterative and incremental software development process and it has aspects of Agile software development. Feature driven development focuses on development with small teams, light methodologies, and heavy customer involvement (Radinger and Goeschka 2003). Interviewee A states that the company has modified FDD to handle architecture since FDD does not take this into account. Further, FDD had been modified to handle model driven development. The domain is modelled using the Unified Modelling Language. Database bindings, and user interfaces are generated with the UML models as a starting point.

When asked whether the documentation of the development process was a "living" document, the interviewee stated that the document was updated once in a while, when it was necessary. Parts of the development method are used frequently while other parts are not, dependent on the project.

Further, when asked about whether the development process was agile, the interviewee stated that the development process was agile, that prototypes were made, and that iterations with the customer were carried out frequently.

Finally, the interviewee was asked to which extent OSS components were modified or if something was developed to encapsulate the component. The respondent replied that it was preferable to build a generator around the components, rather than modifying the components.

Case 2: Company Using Development Practices from OSS

Interviewee C stated that they were using several development practices from OSS. They were emulating an open source methodology inside of the company. They had a flat organizational structure where participants were of the same rank and had the same responsibilities in the project. Requirements specifications and solution specifications were specified using the waterfall method, while implementation was performed with a flat structure.

The interviewee describes his experience with a project where many aspects of Extreme Programming (XP) were used. The following is how the interviewee described experiences with using XP:

"We tried to do XP I had an ambition to do it as a light-weight process with two week iterations. The reasons for this are many: I have a strong belief in the model, I like principles such as pair-programming, short iteration-cycles, and keeping the customer involved." [Interviewee C]

There was a strong need for an iterative process in this project because the customer did not fully understand the possibilities and limitations of the software. The iterations lasted from a couple of weeks to a couple of months. Interviewee C also described some difficulties related to interaction with the customers because the customer had low understanding of software development. Modelling tools like Use case diagrams were used to solve the difficulties and to involve the customer.

Test-driven development was not initially performed in the project, but during the project there was a demand for test-driven development. For the last six months test-driven development was implemented.

The interviewee also expressed a wish to standardize the processes in the company. Currently, developers often use the methods and tools they prefer, and solved problems the way they wanted. They were also using mailing lists, instant messaging, and other informal ways of communication. Standardizing on processes would be preferable according to the interviewee. He expressed a need to further improve this development method because it currently had aspects which the interviewee did not like. According to the interviewee, more formal development methods should be put in place to allow the developers to work more efficiently.

Case 3: Development Processes of Company Managing OSS

Interviewees D, E, and F responded for a company which manage and control an OSS project. They were therefore interviewed about the development process they used when developing the open source product.

The development process used in the company was described as having aspects from both Extreme programming and Rational Unified Process (RUP), both of which are iterative processes. They had reduced the formal parts of the processes compared to RUP. Initially, Interviewee F stated that they tried to be very formal and document the solution. At later stages, they use a more evolutionary process where several iterations are performed.

Interviewee E stated that the following phases are used in the development process: Analysis, Design, Implementation, Test, and Review.

Interviewee D stated that since the company manages an open source project, the community comes with feedback to new requirements. Specifications for new features are released for review, on mailing lists where the community can come with feedback. Many decisions are taken in a democratic way rather than as a dictatorship.

Finally, interviewee D stated that the company was interested in changing the development process to be more structured, and less like "open source". The motivation for this was to improve the quality of the product. Keeping the quality of the product was a challenge, especially when integrating code from several sources, such as code from the community, or from different partners.

16.5.2 Questionnaire: Development Processes in Companies Related to OSS

The questionnaire also covered Research Question 4 and the results from this will be presented next. Five respondents with the role "interacting or participating in OSS communities" and nine respondents with the role "developing software with OSS tools and components" have described several aspects of their development. These results of how use of OSS influences the development processes, will be presented next.

Development Processes when Participating in OSS

The respondents were in Question 33d asked whether their company had changed its development processes because of participation in OSS projects. Unfortunately, too few respondents (five respondents) answered this question to give a conclusive answer. Some of the respondents answered that they had changed their development processes, while others did not. One extreme was the response to Question 40 by company A, which stated that they adapted their development process to that of the OSS project they were participating in. Another respondent from company G stated that an iterative and incremental process was used, but did not state that the development process had been adapted in any way towards the OSS project.

The results therefore do not give a definitive answer to whether companies adapt their development processes when using OSS, although some of the respondents stated that they did.

Development Processes when Using OSS

The respondents from companies assuming the role of users of OSS components reported their development process in Question 43. There was no clear trend in the answers. Out of the nine respondents, two were using a traditional waterfall based method, two were using Rational Unified Process in some ways, one was using Feature Driven Development, while the rest used ad-hoc versions adapted to their particular needs.

The following interesting results were found when the respondents were asked about their development process in Question 46:

- Frequent releases (more than every other week) were not used at all or to a small extent by respondents.
- The majority of respondents were using code repository tools like Subversion or CVS.
- There was no conclusive result about whether the respondents read parts of the source code of the components that they used. Some did, while others did not.
- Few of the respondents modified the OSS components that they used in their projects.
- Code ownership was practiced by some of the respondents, while not by others.

In Question 50, the respondents were asked about their experiences related to using OSS components in the development of their product. The following are statements which the majority of respondents **disagreed** with:

- It was difficult to identify whether defects were inside or outside the OSS components.
- OSS components could not be sufficiently adapted to changing requirements.
- OSS components negatively affected system reliability.
- OSS components negatively affected system security.
- OSS components negatively affected system performance.

This means that the respondents did not think that using OSS components reduced system reliability, security, or performance. This result also means that the respondents find it easy to determine the location of a defect inside or outside of an OSS component, and that it was easy for the respondents to modify OSS components to changes in requirements. This result shows that the respondents have great confidence in using OSS components based on their experiences. Whether the answers are biased for ideological reasons by the respondents has not been investigated, but it is possible.

Software Development Challenges

Question 51 asked the respondents about what they thought were their main software development challenges related to using OSS. This is a summary of the challenges the respondents have with using OSS:

- Finding high quality OSS components.
- Influence the roadmap of components.
- Maintenance of OSS, for example when the OSS component does not have backward compatibility.
- Unacceptable license requirements for some OSS. One company reported they were not able to use GPL licensed software.
- Definition of processes such as selection of components, integration, maintenance, and evolution management.
- Synchronization of changes.

16.5.3 Discussion of Development Processes

The survey results related to Research Question 4 has just been described in this section.

The interviews provided tree descriptive results about the development processes used in those companies. How do these results contribute towards answering this research question? The following trends have been observed in the companies interviewed:

- The companies interviewed use light-weight, **agile development processes**, based on one or more of the following methods: Feature Driven Development, Rational Unified Process, or Extreme Programming.
- The companies want **more standardized and formal development processes**.

Note that the use of OSS has not been shown to be the reason why the companies use agile development methods. It is only an observation that there is a correlation between the use of OSS and agile development methods in the three interviewed companies.

The results from the questionnaire were less conclusive than the results from the interviews. There was a lot of variation in the answers, and too few respondents to use any statistical methods. For example, the development processes used by the different companies had no clear trend unlike the answers from the interviews where all the development methods were Agile.

Some challenges related to using OSS were described in this section. Software developers face these challenges every day, when using OSS. It is important to solve several of these challenges to simply the process of using OSS.

One of these challenges is how the use of OSS affects the development processes. One company reported that they had changed their development processes to that of the OSS community when interacting with the community. When using OSS components, it is necessary to include a selection process to be able to select the right components. When receiving contributions from a community, you have to adapt your development processes to benefit from these contributions.

Further, the selection and evaluation of OSS components is an important process which has an impact on the development processes in companies using OSS. Aspects related to selection and evaluation of OSS components, have been described in the first research question.

Based on these observations, the following hypothesis is presented:

H 10 Companies have to adapt their development processes when using OSS.

Some respondents said that they did not have to change their development processes to be able to benefit from use of OSS. We believe that some adaptation is necessary, but further studies need to be performed to see if this is true. Future research may also elaborate on how the development processes are optimally adapted to OSS.

16.6 RQ5: Open Source Participation by Companies

The fifth research question is concerned with industrial participation in open source projects, controlled by communities outside the company. We wanted to know why and how companies are involved in these projects and communities. We will present the results to the fifth research question in this section.

RQ 5 How and why do companies participate in the development of OSS projects controlled by a community outside the company?

There was some uncertainty about what it takes to be participating in an open source project. Do you have to actively develop code often to be participating, or is it enough to just browse the news or mailing lists from time to time. We were interested in the interaction companies have with the open source communities, and we said that participation is defined by undertaking any of the roles defined in Section 5.2.1. However, we got only five responses the part of the questionnaire which contained open source participation. We will report some of the answers from the questionnaire together with some observations from the interviews related to participation in open source projects.

16.6.1 Activity and Contributions

The respondents to the questionnaire did not seem to contribute to any large extent (Question 31). To the question of why they did not participate that much in open source projects, one developer said:

"We see it more as a product."[Interviewee B]

Some respondents to the questionnaire reported that they did some testing, bug-reporting, and bug fixing, but it does not seem to be important activities. Bug-fixing and bug-reporting were also mentioned during our interviews. Interviewee D said that one of their developers spent about half a year fixing a bug in PHP. Interviewee C said that they got a patch included in the Linux core. The developers seem to perform defect related activities, but maybe not to a large extent.

"I do bug fixes which I need myself and then contribute them to the community. Occasionally, I just take bugs from the tracker and fix them just to contribute." [Respondent to the questionnaire]

The respondents also reported that they did not contribute in any particularly degree with the following: new requirements to the project, architecture or design, documentation, add-on components, integration towards other products (glueware), free user support, distribution of software, infrastructure (e.g. servers or storage space), and money.

Several of our interviewees and some of the respondents to the questionnaire said that they subscribed to mailing lists, asked for user support via email or forums, and read news through related channels. Some of the respondents to the questionnaire reported the same, but the answers were distributed. The response to Question 36 and the interviews indicate that the companies we asked are primarily users, not developers in these open source communities.

Based on the observations above we formulize the following hypothesis:

H 11 Most industrial software developers do not participate in the development of open source software. They are primarily involved as active or passive users, with occasional bug-fixes.

This hypothesis indicates that most industrial software developers do not participate in the development of open source software. They are rather users of the software, but they may contribute with some bug-fixes or other smaller contributions. The hypothesis supports the definition of the layered organization of open source projects described in Section 5.2.1.

Interviewee C said that many developers participated in open source communities through their spare time, and because of different reasons this was perhaps not possible for everyone anymore. It was therefore important to acknowledge participation in these communities, make it more visible internally in the company, and make it part of the developers' jobs.

16.6.2 How Satisfied Is the Industry with OSS?

According to the responses to Question 32, the companies seem to be satisfied with the OSS products, and perhaps surprisingly, the product support they get. They also say that they are satisfied with the relationship between the companies and the communities, and with the ability they have to influence the OSS product. However, Interviewee B said that he found it hard to influence the development of OSS projects. He would have liked to influence them more, but he meant that you had to be part of the development team to influence it.

16.6.3 Motivation

The answers to Question 38 and 39 about company and personal motivations were quite scattered. Benefiting from testing performed by the community seems to be one company motivation, while required use of OSS by the company does not seem to be important.

For the individual, idealism seemed to be one motivating factor, while obligation to the community, and reputation among peers seemed to be unimportant. Learning, were also mentioned as one motivational factor by some respondents.

16.6.4 Communication between Companies and the OSS Projects

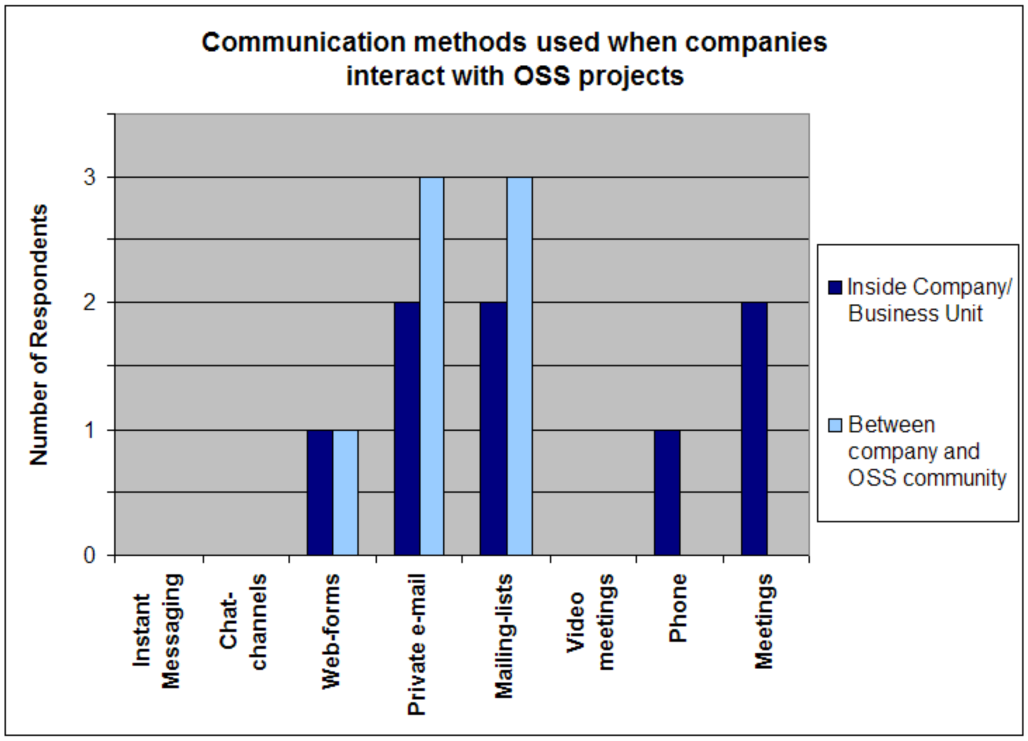


Figure 16.7: Communication methods used when companies participate in OSS projects

How do the developers from a company and members of an OSS community communicate with another when the developers participate in OSS projects controlled by communities outside the company? Figure 16.7 shows the results from Questions 34 and 35. Participants using a particular communication method, answering "Large" or "Very large" were summarized in the figure. The results from the five respondents are shown in Figure 16.7.

The results indicate that on the inside of the company, private e-mail, mailing lists, and formal and informal face to face meetings were used the most. Participants in the company and community members outside the company, communicate primarily using private e-mail and mailing lists.

16.6.5 Licenses

Licensing was not part of our questionnaire, but we wondered how important this was to the companies. We asked our interviewees about licensing, because we wanted to know if they cared, and if they were aware of the possible challenges related to licensing. We were relieved by the fact that this seemed to be extremely important to all the companies we interviewed. Licensing was not seen as a big problem, because most OSS use well-known licenses like GPL, LGPL, BSD, or the MIT license.

Interviewee D as an open source provide, spoke quite a lot about offering the right license to the right customer. To give the user the freedom to choose, they licensed their software with up to four different licenses. Based on these observations we have formed the following hypothesis.

H 12 Industrial actors are aware of and they respect open source licenses.

The consequence of this hypothesis is that the open source community does not need to worry too much about industrial actors exploiting them and stealing their software. Nevertheless, we believe it is important to maintain control over the software developed in the community, and to use clear and well defined licenses.

Chapter 17

Discussion

This chapter will present a discussion of the results from the survey. First, the main contributions in this thesis will be presented. Then, the goal achievement of each research question will be evaluated. Moreover, the validity of the results will be discussed. Finally, improvements to the survey will be suggested.

17.1 Main Contributions

The contributions of this thesis can be divided into four parts, the literature study, new knowledge, a basis for further research, and a reusable research design. Our contributions will be summarized, before we discuss the significance of the answers to the research questions.

17.1.1 Literature Study

The first contribution of this thesis is the literature study. The literature study is a presentation of relevant research, which has been published about the topics described in the research goal. As presented in Chapter 10, a large number of articles and books have been used to describe the state-of-the-art research. The process of creating this literature study has been documented and described. The results constitute the second part of this thesis, the Prestudy. The results of this study will be used as input to deliverables in the COSI project. Further, the results of the literature study base a foundation for the research design. This means that knowledge gained from the literature study has been used to rule out research questions which have already been answered. This has enabled us to create relevant and realistic research questions.

17.1.2 New Knowledge

The second and most important contribution from this thesis is the new knowledge, which has been discovered through the survey. This knowledge can be divided into: answers to the research questions, our hypotheses, and to the guidelines.

Answers to Research Questions

The answers to the research questions are provided in Chapter 16 in form of, both qualitative and quantitative descriptions. The goal achievement of each research question is discussed in Section 17.2 and will therefore not be discussed here.

Hypotheses

We have proposed twelve hypotheses based on the results we found through our interviews and questionnaire. These hypotheses are believed to constitute a valid understanding of the reality based on observations made through the survey. Hypotheses are proposed explanations of observations, they can not be considered as definitive proof. Further work is needed to test the hypotheses, and possibly falsify some of them. The hypotheses are provided here, but they are discussed together with the results in Chapter 16. The hypotheses can be used to create new research questions, and they are therefore an important part of the platform for future research. Our hypotheses can be found in Table 17.1.

H1	Companies participate in Open Source with the roles open source owner, open source participant, inner source participant, and user of open source components.
H2	Open source components are primarily regarded as commodity software.
H3	High availability of the components, of the source code, and of the information related to the components is one of the most important motivations for using OSS components.
H4	Selection and evaluation of OSS components, is primarily done as a process based on previous knowledge, informal searches, and subjective evaluations of the components.
H5	Opening up the development of an industrial open source product may create trust and generate feedback, but it requires excellence from the open source provider.
H6	The number of possible users in the market and the size of the community around your product severely influence the further development of the product and your benefits of having it as an open source product.
H7	Critical success factors for succeeding with an industrial open source products are: Your company can make money offering it as an open source product. The product is needed in the company. The developers are users of the product. An active community with technically competent people exists around the product.
H8	You must as an owner of an open source product work to attract a critical mass of community members. When you have succeeded, you need to work hard to keep them.
H9	Companies use open source software engineering practices to perform distributed software development.
H10	Companies have to adapt their development processes when using OSS.
H11	Most industrial software developers do not participate in the development of open source software. They are primarily involved as active or passive users, with occasional bug-fixes.
H12	Industrial actors are aware of and they respect open source licenses.

Table 17.1: Our proposed hypotheses

Guidelines

Preliminary guidelines have been created based on the results of the survey. These guidelines are created based on Research Question 1, 2, and 3. The answers to theee research questions provided immediate results, which could be useful for companies participating in the COSI project. Results to Research Questions 4 and 5 are not of a type where guidelines can be created directly. Guidelines for the selection and evaluation of open source components can be found in Appendix D.1. Guidelines for management of open source products can be found in Appendix D.2. Finally, guidelines for companies participating in inner source projects can be found in Appendix D.3. Feedback from the industry directly on the guidelines would be useful to improve them further.

17.1.3 Platform for Future Work

Our work answers many questions, but new ones can be based on this work. We have gained new understanding of several topics and our work will function as a platform for further work in the COSI project. Our work may also work as a platform for others to build upon as well. Some ideas for future work are presented in Section 18.2.

17.1.4 Reusable Research Design

The research design we have developed can be extended and reused. We have provided descriptions of our design and our work, and we have made the interview guide and the questionnaire available. The questionnaire is also implemented using a web tool. This tool is available and it would be small job to improve, extend, and perform this survey repeatly. Our research design is described in Chapter 12, our interview guide and questionnaire, are available in Appendix B, and C.

17.1.5 Significance of Results

What is the significance of the results found in the survey, and how are these results compared to those found in literature?

Research Question 1

From the answers to the first research question, we have described a selection-process for selection of OSS components. This selection process, has many similarities to the selection processes for COTS components described in Section 7.3, in the prestudy. Our results to Research Question 1 therefore suggest that the selection process described in the literature study is quite commonly used.

Research Question 2

The results to the second research question described experiences from companies which manage their own OSS product. As described in the prestudy in Chapter 8, few publications have been made about companies which initiate their own OSS product. Therefore, the results to this research question describe interesting and new knowledge about companies managing their own OSS product.

Research Question 3

Answers to the third research question described inner source development. As mentioned in the prestudy in Section 5.3, inner source development has been discussed in very few publications. Therefore, answers to this research question present results which describe new experiences from inner source development projects. This knowledge, is something companies interested in participating in inner source projects, can learn from.

Research Question 4

Based on the answers to the fourth research question, we proposed a hypothesis that companies using OSS have to change their development processes to some extent when using OSS. The most significant result to this research question is a hypothesis which can be the foundation for future research.

Research Question 5

The fifth research question has been answered by describing how companies participate in OSS development. The results of this work include hypotheses which can be the foundation for future research. The results to this research question were not definitive, and we therefore have not compared these results to other studies.

17.2 Goal Achievement

The original goal of this work is described in the introduction of this thesis. The two main objectives were, to do a literature study, and to create a survey design. The results of these should aid the work in the COSI project.

The first part of this goal has been fully completed. The results of the literature study can be found in the second part of this thesis, the Prestudy. The literature study will function as input to one of the deliverables in the COSI project.

The second part of the main goal was decomposed into a research goal, and five research questions. Both the research goal and the research questions are described in Chapter 11.

We described four roles which a company can undertake in the COSI setting, open source owner, open source participant, inner source participant, and user of open source components. The focus of the research goal was to explore the relationships these four roles have to open source and open source practices.

17.2.1 Evaluation of Answers to the Research Questions

The research goal was as mentioned decomposed into five research questions, RQ1-RQ5. We will now evaluate the goal achievement of each of these five research questions.

Research Question 1

The first research question, **RQ1 Why do companies use OSS components, and how do they find and evaluate these components?** is related to the role *user of open source components*. We have for this role answered all parts of the research question by providing an overview of important motivations for use of OSS components, a description of a search and evaluation process, description of several evaluation criteria, and some guidelines for OSS component selection and evaluation. The results are described in Section 16.2 and the guidelines are available in Appendix D.1. The guidelines are provided as an input to the COSI project and future work should be focused on improving these preliminary guidelines.

Research Question 2

We feel confident with the goal achievement of the second research goal, as well. **RQ2: Why do companies choose to make their products Open Source, and how can this be done successfully?** is related to the role as an *open source owner*. To answer this research question, we have described motivations behind making a product open source, showed advantages and disadvantages of having a product as open source, provided tips, and requirements to those who are going to make their product open source, and summarized our findings in form of preliminary guidelines. The results are available in Section 16.3 and the preliminary guidelines for companies managing OSS projects are available as Appendix D.2.

Research Question 3

The third research question, **RQ3: Why and how do companies use development methodologies from Open Source, in ISS development?**, is answered by providing an overall description of inner source development. This description includes a description of the open source development practices, which can be used inside a company. Further, we describe the development practices, and the development processes used in inner source projects. Finally, we present a a description of how inner source projects are organized.

As stated in the research goal, the survey covered inner source in an explorative way, because little was known about the topic in advance. The results of this research question have therefore contributed towards a better understanding of inner source development. It is clear that companies use open source practices in their development, but further research in needed to explore this phenomenon.

Inner source development is a relatively new concept, and some of the respondents reported that they were uncertain about the definition of "inner source". Nevertheless, the goal achievement of this research question is good. Inner source development has been explored, and described as good as possible, based on the responses to the questionnaire. These descriptions are presented in Section 16.4 and the preliminary guidelines for inner source development are available as Appendix D.3.

Research Question 4

The answers to the fourth research question, **RQ4: How does the use of OSS influence the development processes of companies?** are presented in Section 16.5. These answers are related to the role *inner source participant*.

This research question was answered based on the interviews of three companies, participating in the COSI project and the responses to the questionnaire. The answers to this research question were provided as three examples of the development processes used in the companies which were interviewed. For each of these companies, the development processes were described, and influences from the use of OSS were sought. Further, the development processes, and the development practices reported by companies responding to the questionnaire, were described.

Based on the results from the interviews and questionnaire, a discussion was presented related to whether the use of OSS has influenced the development processes of those companies.

It was hypothesized that companies, had to change their development processes to some extent when using OSS. However, further studies are necessary to determine this. The number of respondents to the questionnaire was relatively low, and the companies themselves were not unified about this research question. To sum up, we have succeeded at answering this research question, by provided new insight into development processes used in companies with a role related to OSS.

Research Question 5

The fifth research question, **RQ5: How and why do companies participate in the development of OSS projects controlled by a community outside the company?**, is answered by describing how companies participate in OSS development. In addition to these descriptions, we have described some motivational factors and some issues related to communication. The results can be found in Section 16.6.

There was unfortunately some uncertainty among the respondents about what participation is, or what it takes to be a participant. Additionally, it could be hard to differentiate between personal, and company initiated participation. We are interested in all the interaction companies have with open source communities. We have provided some results, and we think it would be interesting to work with this issue further. This research question has been answered to some extent, but as described above, the responses from companies are that they are unclear about their participation in OSS projects.

17.2.2 Summary

All the research questions have been answered based on the results from the survey. Detailed qualitative and quantitative answers have been provided to all the five research questions. Based on these results, preliminary guidelines have been created to provide input to the COSI project. Further, hypotheses have been created as a foundation for future research.

However, the number of responses to the questionnaire was low, the use of convenience sampling, and the fact that some of the respondents did not agree on the definitions of some of the terms in the questionnaire, threaten the validity of our results. These and other threats to validity are discussed in the next section.

Our results are, because of these threats to validity primarily presented as hypotheses, rather than conclusions. The results of each research question consist of descriptions of the observations made through the interviews and questionnaire, and our analysis of these answers.

To summarize the discussion of our goal achievement, we would say that we are very satisfied with the results we achieved, but we see that there are room for improvement. We have completed the tasks from the original project description, and we have answered all of our research questions. Some the possible improvements of our work and a discussion of the validity of our work will be described further in the next sections.

17.3 Validity

Validity is related to how much we can trust the results we have found. It is always important to take a critical look at your work, and assess the quality of both this work, and its results. (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000) state that adequate validity refers to the fact that the results must be valid for the population of interest. The authors divide validity into four types: internal, external, construct, and conclusion.

The *"truth"* can be described in different ways, as described in Section 12.2. It is because of this, better to talk about intersubjectivity rather than truth. Intersubjectivity is a description of something which many people agree upon (Jacobsen 2005, 214). We cannot say that our descriptions of our results are true, but we believe they are understandings that can be shared by others. We will now discuss possible threats to our descriptions, and our understandings of the results.

17.3.1 Internal Validity

Internal validity is related to whether the results are valid for the population the sample is taken from (Jacobsen 2005, 214).

Review and Reporting

(Jacobsen 2005) describes two measures to ensure the internal validity of a qualitative study. These are, validation through review by others, and validation through critical self-review. These two measures can also be used to ensure the validity of quantitative studies.

The self-review should review both your sources, and the information from these sources. We have of course reviewed our information, and discussed our results several times. In addition to our self-review, we have shared both the data and the results with our supervisors, primarily Carl-Fredrik Sørensen. Some issues have also been presented to fellow students and researchers. Both supervisors and colleagues provided us valuable feedback, and functioned as sparring partners.

We are going to report our findings to the respondents of the questionnaire and to our interview objects, through the COSI project. This will unfortunately not happen until after our project deadline. We are therefore at the moment not able to get any feedback on our results from our respondents.

Selection

(Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000) mention selection as one threat to internal validity. We have been forced to use convenience sampling both for the selection of the interview objects and for the questionnaire.

Using convenience sampling, like we were forced to do in our research, may reduce the internal validity of the results. Our supervisor sent the survey to the companies participating in the COSI project and the partners selected the respondents themselves. This leaves us without control over response, and dropout rates of the questionnaire, and we do not know whether the respondents are representative for the companies they represent. We are therefore unable to do any analysis of the ones who chose, not to answer the questionnaire.

Another drawback of convenience sampling is that people who really want to answer, or as in the COSI setting, who are obligated to answer, may answer differently than other respondents (Kitchenham and Pfleeger 2002b). It would have been preferable to use some kind of randomization to select our sample, but this was not possible.

Control of Human Factors

There are several human factors which influence the responses to a questionnaire, and it is hard to control these factors. (Robson 2003) mentions human memory, knowledge, experience, motivation, personality, and that people may want to be seen in a good light as possible problems with questionnaires. We have tried to develop a questionnaire that allows the respondents to answer honestly by improving it through several reviews, and from feedback from the interviewees and from the COSI project.

Further, the questionnaire contained a meta-question where respondents could clarify assumptions that they made when answering the questionnaire. This gave us feedback on the questionnaire and allowed us to only use results which came from respondents who understood the questions that they were answering.

Finally, the interviews were recorded on tape. This allowed us to listen and transcribe all the interviews. This contributed towards increasing the internal validity because then we could be surer that the reported results are accurate.

Maturation

Further, (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000) describe maturation as a threat to internal validity. Maturation is the unwanted effect that respondents can answer differently over time. One cause of maturation is that respondents think that answering questionnaires is uninteresting and possibly answer differently because of this. Very long questionnaires with a poor interface may contribute towards maturation. This was an issue we were aware of during the design of the questionnaire. The questionnaire created in this study was quite long with 69 questions in total. However, respondents only had to answer the parts of the questionnaire which was relevant to their OSS related role. The time to fill in the questionnaire was therefore decreased and the risk of maturation was reduced.

Mortality

Mortality is another risk to internal validity as described by (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). Mortality means that respondents drop out of the study. A cause for this could be the same as described for maturation, which is that some respondents could have had problems answering the questionnaire or finding it uninteresting to answer. We observed only one respondent to the survey who did not answer any of its parts. Other than this instance, there was no observed mortality.

17.3.2 External Validity

External validity is related to whether the results are valid for other populations. The main threat to external validity is the sampling of respondents to the survey. The respondents were few, 24 respondents from 11 companies. Further, the respondents are possibly not representative for larger population.

Generalization

The results have adequate external validity if they are valid for the population we wish to generalize (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000). However, (Lee and Baskerville 2003) say that as a consequence of Hume's truism, a theory have to be tested in the setting before it can be said to be generalizable.

The purpose of qualitative studies is normally not to generalize, but to explain and understand a phenomenon (Jacobsen 2005, 222). Our goal was to create an understanding of the phenomenon, *use of open source in industry*, and create a theory based on this understanding.

(Lee and Baskerville 2003) present a framework of four types of generalizing and generalizability. Generalization from empirical work to theory is one of the four types of generalization. (Yin 2003) refers to this type of generalization as analytical generalization. Lee and Baskerville refer to several examples and different theories, and conclude that generalization from empirical work to theory is possible. We have based on our observations, presented several hypotheses. These hypotheses need to be tested in other settings to be generalizable to that setting according to (Lee and Baskerville 2003). Testing these hypotheses could be an input to future work.

Population

The population of our research is the COSI project. This project consists of industrial partners who want to use open source, and who are interested in improving their use of open source, and open source practices. Our population is therefore not typical for all software companies. The fact that we work with, one special group of industrial partners, reduces the external validity of our research.

Sampling

The sampling method, convenience sampling, could also be a threat to the external validity of the results. It is unlikely that the few respondents we had are typical for the whole population. This is also related to the statistical generalization. The statistical power of our survey is very low, with only 24 respondents, and as few as three respondents to one part of the survey.

Yet another thing that may affect the external validity is the fact that our interview object also answered to the questionnaire. This means that their answers may have been given extra weight because we have used their answers from the interviews, and we have used their answers from the questionnaire.

17.3.3 Construct Validity

(Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000) states that construct validity is related to generalizing the results to more a general underlying theory. To meet the test of construct validity, you have to select what you want to study and demonstrate that the selected measures reflect what you wanted to study (Yin 2003, 35). In other words, did our interviews and our questionnaire measure or collect the data we wanted?

(Jacobsen 2005) mentions measuring the right thing as one problem, especially with questionnaires. Questions and statements in both our questionnaire and our interviews may have been misunderstood, or misinterpreted because they were improperly phrased. If the interview objects, or the respondents misunderstood something, they may have answered something else than what we wanted to know.

We had some ambiguity related to some terms used in the questionnaire, but besides that, we did not get any indications that neither the questionnaire nor the interviews were misinterpreted. Some of these misunderstandings and our corrections are described in Chapter 14.

(Robson 2003) mentions testing and review as the best ways of ensuring construct validity. Our supervisors, fellow students and researchers, and partners in the COSI project have reviewed our questionnaire, and provided us with feedback. We have also tested the questionnaire by using it as an interview guide. If it was possible, it would have been preferable to be able to pre-test the questionnaire with a small sample of respondents.

17.3.4 Conclusion Validity

Conclusion validity is related to finding relationships. There are two possible threats to conclusion validity. The first threat is that relationships are found where no relationships exist. The other threat is that existing relationships are ignored. Further, it is important that the relationships are causal and that the relationships are in the correct direction of causal influence (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000).

When using qualitative research methods, validity can be increased by triangulation. Triangulation is to control the results and data by combining different research methods. We have used both interviews and a questionnaire. Other methods, like observations, could also have been used if the time span of this work had been larger.

One possible threat to the conclusion validity is the low statistical power. The statistical power of this work is as mentioned before, very low. This is an obvious problem. Other threats to the conclusion validity could be fishing, poor question wording, and bad questions (Wohlin, Runeson, Höst, Ohlsson, Regnell, and Wesslén 2000).

The researchers lack of experience conducting interviews, and creating questionnaires is potentially another problem. This lack of experience may lead to bad questions, and perhaps leading questions. We have received good support and feedback from our supervisors, so we do not think this is a major problem, except when it comes to describing processes. We got feedback from some of the respondents to the questionnaire that open questions were not suitable to describe processes. It was difficult, and it was too time consuming. Both the feedback from the respondents and the feedback from our supervisors are described further in Chapter 14.

17.3.5 Reliability

Reliability is related to stability and consistency. If the same procedures are followed in later investigations, they should find the same results and arrive at the same conclusions (Yin 2003, 38). The goal of reliability is to reduce the errors and the biases in a study.

We have tried to document our research in Chapter 12, 13, 14, and 15. This documentation is provided to increase the reliability of our research. We have also made both the interview guide and the questionnaire available as Appendix C, and B. This enables other researchers to replicate our work.

(Yin 2003) emphasizes that reliability is about *doing the same research*, not about replicating the same results in another setting. It should be possible to do very similar research to what we have described here, and get very similar results. However, it is important to be aware that the same respondents we used may change their mind, or learn, and therefore answer differently the next time the same research design is executed.

17.4 Improvements

In this section, we present some of the thoughts about possible improvements to our work.

The two most obvious threats to the validity of our work are the limited sample size, and the sampling method used. By redistributing the questionnaire to a bigger sample and by using a random-sample technique to select the respondents, we would reduce this problem. Having a defined sample would also allow us to do analysis of the ones who did not answer the questionnaire.

Some of the respondents reported that the questionnaire was too time consuming and that it was difficult to describe development processes in a small box. Another improvement would be to review the questionnaire and perhaps remove or rephrase some questions.

(Kitchenham and Pfleeger 2002a) describe how to increase the motivation of the respondents, by showing how the results of the study can be useful to them. Increased motivation may lead to higher response rate, and we believe that we should have focused more on showing the usefulness of our results to the respondents. The article by Kitchenham and Pfleeger is part of an article-series which discusses how to increase the reliability and the validity of surveys. We were unable to follow these advices because we discovered the series too late. Nevertheless, learning from the advices given by Kitchenham and Pfleeger would also improve our questionnaire.

Validation of our work can also be achieved through comparison with the literature and against other studies, or by presenting it to people with deep knowledge and insight in the field. This and the other improvements of our work could be done as an extension of our present work.

Chapter 18

Conclusion and Future Work

18.1 Conclusion

Through this thesis, the results of a survey of industrial involvement in open source have been presented. Open source is an area which is becoming increasingly interesting for companies, as part of a business model, but also as a set of development practices.

The overall goal of the COSI project is, as stated in the introduction, to understand how industry can benefit from open source software and open source development practices. The COSI project has hypothesized that the use of commodity OSS is one way to achieve more efficient software development.

The initial goal of this thesis was to create a survey of open source within the context of the COSI project. We defined four roles in the COSI setting which a company can undertake: open source owner, open source participant, inner source participant, and user of open source components. The goal of this work is to understand these roles better. Our work will, together with other work in the COSI project, form a baseline description of the companies' software development processes, communication processes, and relationships to the open source communities. These baseline descriptions will help the participating companies to better understand, how they use and benefit from open source software and development practices.

The survey was performed as a combination of structured interviews and a questionnaire The interviews were performed with the Norwegian partners of the COSI project. Feedback from these partners, were used to develop and improve the questionnaire. The questionnaire was answered by the companies participating in the ITEA COSI project.

The research goal has been achieved and our work has four main contributions: Firstly, a literature study describing the state-of-the-art of open source software development. This literature survey is used in deliverables in the COSI project. It allowed us to build our work, on the work of others, and it allowed us to compare our results with the results with other researchers. Secondly, new knowledge has been discovered in the form of answers to the research questions, twelve hypotheses, and guidelines for companies involved in open source software development. Thirdly, the new knowledge found can be used as a platform for future work. Fourthly, we have developed a research design which is reusable and extendable. We provide descriptions of this design and it can be reused by us, or other researchers.

Each of the five research questions, have been answered and we have provided extensive, descriptive answers. The results of these research questions provide interesting and unique answers to questions which have not been studied in detail in the literature. The answers to Research Question 1, provide new insight into how developers select and evaluate OSS components. Based on answers to the survey, we have described a hypothetical OSS component selection process. Answers to Research Question 2 describe open source projects managed by companies. There were very few publications about companies managing their own OSS projects. The unique access to such respondents through the COSI project, allows us to harvest and share experiences from such projects. Research Question 3 covers inner source development. The answers to this research question can provide new insight for companies interested in using open source development practices and to improve development in a distributed setting. Research Question 4 covers how the use of OSS influences the development processes in companies. The answers provide relevant experiences for all companies using OSS. Finally, results from Research Question 5 describe companies participating in OSS projects. The results to this question can aid companies interested in such participation.

Many interesting results have been found through this survey, but new questions are posed. Our work should be extended and validated by more research within the same research area. We strongly believe that our work is a step in the right direction and we encourage other researchers to continue work in this area. More work in this area will help industry and the open source communities to benefit as much as possible from industrial involvement in open source.

18.2 Future Work

This section describes areas to be explored in future studies which are related to open source and possibly performed within the context of the COSI project.

When we did our literature survey, we found limited empirical research about industrial involvement in open source. To increase the understanding of this topic and to increase the benefits the open source communities have from industrial involvement and the benefits industry has from involvement in open source, it is important to do more empirical research. The work described in this thesis can be extended in several ways.

It is first of all possible to redistribute the survey to a bigger sample, using more appropriate sample techniques. By building on our research design it should be plausible to get good, valid results.

A second interesting topic is selection and evaluation of OSS. We have provided descriptions of interesting characteristics of such processes, but it would be very interesting to explore these processes further. This could be done by both interviews, and questionnaires.

How to create and maintain an OSS community is another important topic. It is important to understand how this can be done successfully to really benefit from the open source development model. To really succeed, it is important to attract a community to your product. How to do this and how to benefit from the community are two important questions. It is as we have shown, necessary to spend some resources to be able to succeed with an open source project. How much and how to spend these resources most efficiently are two questions that should be explored.

The three topics above have been found interesting by NTNU, and they have made new project descriptions around these three topics available for new computer science students. We hope that some of these students will find interest in our work and continue it.

Related to these topics are our guidelines. They are only preliminary and thus they need to be improved to be of greater use. We hope that these guidelines can be developed further and be used in the COSI project, as useful input in their work.

Further, a longitudinal survey could be performed to investigate how company involvement in open source change over time. The COSI project lasts until 2008, so future surveys could be performed comparing the baseline descriptions found in this thesis with future changes.

Some respondents expressed some dissatisfaction with the open questions in the questionnaire related to process descriptions. It should therefore be possible to find other and more suited models to describe software development processes. It is interesting to describe both processes used in open source related development and in software development which use open source practices.

We have in this thesis proposed twelve hypotheses. These hypotheses are our interpretations of the results we found. We believe they form a good starting point for future work. Testing these hypotheses could be done related to some of the tasks described above.

Index

Glossary

Altruism	altruism is the practice of placing others before oneself, 21
ANT	a tool for automatic building of software, 17
API	The programming interface of a piece of software. The API describes how to access the functions of the software., 103
BASH	Bourne Again Shell (a shell environment), 24
BIND	Implementation of the Domain Name Server (DNS) which maps domain names to IP-addresses, 16
CBD	COTS-based development, 42
Compiler	a program translating source code in a language like C, 12
Consortium	association companies engaging in a joint venture, 32
Copyleft	a word-play with the more known copyright Copyleft is a general method for making a program or other work free and requiring all modified and extended versions of the program to be free as well, 24
COSI	Co-development using inner & Open source in Software Intensive products, 1
COTS	Commercial off the shelf, 42
Cross sectional study	A study performed once, 66
CVS	Concurrent Versions System, 31
Extrinsic motivation	When a person is rewarded or encouraged by something outside the person, 21
FLOSS	Free Libre Open Source Software where Libre is Spanish for free as in freedom, 12
GCC	GNU Compiler Collection, 24

Hacker	a person who is very skilled at computer programming and spends a lot of time programming , 15
HTTP server	Earlier known as the Apache web server. It is the most used web server on the Internet today, 17
Idealism	Idealism is the behavior and beliefs of someone who has strong ideals and tries to base their behavior on these ideals, 21
IDI	Department of Computer and Information Science, i
Intersubjectivity	Intersubjectivity referes to shared meanings constructed by people in their interactions with each other, 121
Intrinsic motivation	When a person is involved in some activity without some obvious external incentive present. This could be the feeling of creativity, 21
IRC	Internet Relay Chat is a chatting environment where the users can log on to a server, 23
ITEA	Information Technologi for European Advancement, 1
Longitudinal study	A study repeated at several points in time, 66
NTNU	Norwegian University of Science and Technology, i
Qualitative	Concerned with information as text, 64
Quantitative	Concerned with information as numbers, 64
Reliability	The reliability of a survey is concerned with how well we can reproduce the survey data., 125
Sendmail	Eric Allman's mail messaging routing program which is currently running most of the e-mail traffic today, 16
Tomcat	a Java Server Pages and Java Servlet extension of the HTTP server, 17
Triangulation	Triangulation is to control the results and data, 124

Validity	The validity of a survey concerns how well the instrument measures what it is supposed to measure., 125

References

Abramson, R. (2004). Linux Looms Larger Than Thought. Online: `http://www.thestreet.com/_yahoo/tech/ronnaabramson/10199089.html`, accessed 2006-04-24.

Barahona, J. G., P. H. Quiros, and T. Bollinger (1999). A Brief History of Free Software and Open Source. *IEEE Software 16*(1), 32–34.

Behlendorf, B. (1999). *Open Sources: Voices from the Open Source Revolution*, Chapter Open Source as a Business Strategy, pp. 149–170. O'Reilly Media, Inc.

Bergquist, M. and J. Ljungberg (2001). The Power of Gifts: Organizing Social Relationships in Open Source Communities. *Information Systems Journal 11*, 305–320.

Carney, D. and F. Long (2000). What Do You Mean by COTS? Finally, a Useful Answer. *IEEE Software 17*(2), 83–86.

Conradi, R. and J. Li (2005). Observations on Versioning of Off-the-Shelf Components in Industrial Projects (short paper). In *SCM '05: Proceedings of the 12th International Workshop on Software Configuration Management*, Lisbon, Portugal, pp. 33–42. ACM Press, New York, NY, USA. ISBN: 1-59593-310-7.

Cooper, D. R. and P. S. Schindler (2001). *Business Research Methods*. McGraw-Hill. ISBN: 0-07-231451-6.

Crowston, K., H. Annabi, J. Howison, and C. Masango (2004). Effective Work Practices for Software Engineering: Free/libre Open Source Software Development. In *WISER '04: Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research*, Newport Beach, CA, USA, pp. 18–26. ACM Press, New York, NY, USA. ISBN: 1-58113-988-8.

Crowston, K. and J. Howison (2005). The Social Structure of Free and Open Source Software Development. *First Monday 10*(2). Online: `http://www.firstmonday.org/issues/issue10_2/crowston/index.html`, accessed 2006-05-01.

Dahlander, L. and M. G. Magnusson (2005). Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms. *Research Policy 34*(4), 481–493. available at http://ideas.repec.org/a/eee/respol/v34y2005i4p481-493.html.

Dahle, H. P. and R. Conradi (2005). Norsk Cosi. Proposal for the Norwegian COSI.

Deng, J., T. Seifert, and S. Vogel (2003, May). Towards a Product Model of Open Source Software in a Commercial Environment. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, pp. 31–37. International Conference on Software Engineering.

Dinkelacker, J., P. K. Garg, R. Miller, and D. Nelson (2002). Progressive Open Source. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, Orlando, Florida, pp. 177–184. ACM Press, New York, NY, USA. ISBN: 1-58113-472-X.

Divitini, M., L. Jaccheri, E. Monteiro, and H. Trœtteberg (2003). Open Source Processes no Place For Politics. In J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (Eds.), *Taking Stock of the Bazaar Proceedings of the 3rd Workshop on Open Source Software Engineering*, pp. 39–44.

Feller, J. and B. Fitzgerald (2002). *Understanding Open Source Software Development.* Addison Wesley. ISBN :0-201-73496-6.

Fielding, R. T. (1999). Shared leadership in the Apache project. *Communications of the ACM 42*(4), 42–43.

Free Software Foundation (2006). The Free Software Definition. Online: `http://www.fsf.org/licensing/essays/free-sw.html`, accessed 2006-02-10.

freshmeat.net (2006). freshmeat.net Statistics. Online: `http://freshmeat.net/stats/`, accessed 2006-02-17.

Fuggetta, A. (2003). Open Cource Coftware an Evaluation. *The Journal of Systems and Software 66*, 77–90.

Ghosh, R. A. (2002). Free Libre and Open Source Software: Survey and Study. International Institute of Infonomics, University of Maastricht, The Netherland; Berlecon Research GmbH, Berlin, Germany, http://www.infonomics.nl/FLOSS/report.

Glass, R. L. (2002). Sorting out Software Complexity. *Communications of the ACM 45*(11), 19–21.

GNU Project (2005). What is Copyleft? Online: `http://www.gnu.org/copyleft/`, accessed 2006-02-16.

Goldman, R. and R. Gabriel (2005). *Innovation Happens Elsewhere: Open Source as Business Strategy.* Morgan Kaufman Publishers. ISBN: 1-55860-889-3.

Gonzalez, A. G. (2005). Legal Challenges to Open Source Licences. *SCRIPT-ed A Journal of Law and Technology, School of Law at the University of Edinburgh 2*(2), 301–308. Online: `http://www.law.ed.ac.uk/ahrb/script-ed/vol2-2/challenges.asp`, accessed 2006-04-26.

Gurbani, V. K., A. Garvert, and J. . Herbsleb (2006). A Case Study of a Corporate Open Source Development Model. In *ICSE '06: 28th International Conference on Software Engineering*, Shanghai, China, pp. 472–481. ACM Press, New York, NY, USA. ISBN: 1-59593-375-1.

Gutwin, C., R. Penner, and K. Schneider (2004). Group Awareness in Distributed Software Development. In *CSCW '04: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, Chicago, Illinois, USA, pp. 72–81. ACM Press, New York, NY, USA. ISBN: 1-58113-810-5.

Hamerly, J., T. Paquin, and S. Walton (1999). *Open Sources: Voices from the Open Source Revolution*, Chapter Freeing the Source - The Story of Mozilla, pp. 197–206. O'Reilly Media, Inc.

Hang, J., H. Hohensohn, K. Mayr, and T. Wieland (2004). Benefits and Pitfalls of Open Source in Commercial Contexts. In S. Koch (Ed.), *Free/Open Source Software Development*, pp. 222–241. Hershey, PA: Idea Group Publishing.

Hars, A. and S. Ou (2002). Working for Free? Motivations for Participationg in Open-Source Projects. *International Journal of Electronic Commerce 6*(3), 25–39.

Hertel, G., S. Niedner, and S. Herrmann (2003). Motivation of Software Developers in Open Source Projects: in Internet-based Survey of Contributors to the Linux Kernel. *Research Policy 32*, 1159–1177.

Holck, J. and N. Jorgensen (2005). Do not Check in on Red: Control Meets Anarchy in Two Open Source Projects. In S. Koch (Ed.), *Free/Open Source Software Development*, pp. 1–26. Hershey, PA: Idea Group Publishing.

IBM (2006). New to Open Source. Online: `http://www-128.ibm.com/developerworks/opensource/newto/`, accesssed 2006-05-01.

IKT-Norway (2006). IKT-Norway Web site. Online: `http://www.ikt-norge.no/templates/Page.aspx?id=474`, accessed 2006-06-07.

Indymedia (2004). Free Software - Free Society! Interview with Richard Stallman. Online: `http://www.indymedia.org.uk/en/2004/05/292609.html`, accessed 2006-02-10.

Jacobsen, D. I. (2005). *Hvordan Gjennomføre Undersøkelser? Innføring i Samfunnsvitenskaplig Metode* (2nd. ed.). Hyøskoleforlaget. ISBN: 82-7634-663-4.

Jørgensen, N. (2001). Putting it all in the Trunk: Incremental Software Development in the FreeBSD Open Source Project. *Information Systems Journal 11*, 321–336.

Karels, M. J. (2003). Commercializing Open Source Software. *Queue 1*(5), 46–55.

Kitchenham, B. and S. L. Pfleeger (2002a). Principles of Survey Research Part 4: Questionnaire Evaluation. *SIGSOFT Softw. Eng. Notes 27*(3), 20–23.

Kitchenham, B. and S. L. Pfleeger (2002b). Principles of Survey Research: Part 5: Populations and Samples. *SIGSOFT Softw. Eng. Notes 27*(5), 17–20.

Klincewicz, K. (2005). Innovativeness of Open Source Software Projects. Online: `http://opensource.mit.edu/papers/klincewicz.pdf`, accessed 2006-04-16.

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

Lakhani, K. R. and E. von Hippel (2002). How Open Source Software Works: "Free" to User-to-User Assistance. *Reserach Policy 32*, 923–943.

Lakhani, K. R. and R. G. Wolf (2005). *Perspectives on Free and Open Source Software*, Chapter Why Hackers Do What They Do: Understanding Motivations and Effort in Free/Open Source Software Projects, pp. 3–23. MIT Press.

Laurent, A. M. S. (2004). *Understanding Open Source & Free Software Licensing.* O'Reilly. ISBN: 0-596-00581-4.

Lee, A. S. and R. L. Baskerville (2003). Generalizin Generalizability in Information Systems Research. *Information Systems Research 14*(3), 221–243.

Leonard, A. (2000). The Free Software Project. Online: `http://www.salon.com/tech/fsp/contents/index.html`, accessed 2006-02-07.

Li, J., F. O. Bjornson, R. Conradi, and V. B. Kampenes (2004). An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry. In *METRICS '04: Proceedings of the Software Metrics, 10th International Symposium on (METRICS'04)*, Washington, DC, USA, pp. 72–83. IEEE Computer Society. ISBN: 0-7695-2129-0.

Li, J., F. O. Bjørnson, R. Conradi, and V. B. Kampenes (2004). An Empirical Study on COTS Component Selection Process in Norwegian IT Companies. In *International Workshop on Models and Processes for the Evaluation of COTS Component (MPEC'04)*, Edinburgh, pp. 27–30. IEE Press.

Li, J., R. Conradi, O. P. N. Slyngstad, M. Torchiano, M. Morisio, and C. Bunse (2005). A State-of-Practice Survey on Risk Management in Off-the-Shelf Component-Based Development. In *In Proc. of the 4th Int. Conf. on COTS-based Software System (ICCBSS'05)*, Bibao, Spain, pp. 278–288. Springer-Verlag.

Madanmohan, T. R. and R. De' (2004). Open Source Reuse in Commercial Firms. *IEEE Software 21*(6), 62–69.

Massachusetts Institute of Technology (2006). The MIT License. Online: `http://www.opensource.org/licenses/mit-license.php`, accessed 2006-02-11.

Meeker, H. (2005). Dual-Licensing Open Source Business Models. Online: `http://linux.sys-con.com/read/49061.htm`, accessed 2006-02-15.

Mockus, A., R. T. Fielding, and J. D. Herbsleb (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol. 11*(3), 309–346.

Monteiro, E., T. Østerlie, K. H. Rolland, and E. Rørvik (2004). Keeping it Going: The Everyday Practices of Open Source. Draft, available online: `http://idi.ntnu.no/~thomasos/paper/keeping_it_going.pdf`, accessed 2006-02-10.

Mozilla OSS project (2002). Mozilla Development Roadmap. Online: `http://www.mozilla.org/roadmap/roadmap-05-Jun-2002.html`.

Narduzzo, A. and A. Rossi (2004). The Role of Modularity in Free/Open Source Software Development. In S. Koch (Ed.), *Free/Open Source Software Development*, pp. 84–102. Hershey, PA: Idea Group Publishing.

Netcraft (2006). Netcraft February 2006 Web Server Survey. Online: `http://news.netcraft.com/archives/web_server_survey.html`, accessed 2006-02-07.

opensource.org (2006a). History of the OSI. Online: `http://www.opensource.org/docs/history.php`, acessed 2006-01-31.

opensource.org (2006b). The Open Source Definition. Online: `http://opensource.org/docs/definition.php`, acessed 2006-01-31.

O'Reilly, T. (1999). Lessons from Open-Source Software Development. *Communications of the ACM 42*(4), 32–37.

Østerlie, T. and K. H. Rolland (2003). Unveiling Distributed Organizing in Open Source Development: Practices of Using, Aligning, and Wedging. In *Proceedings of the Workshop on Open Source Software Movements and Communities*, pp. 26–33. Pre-Workshop Draft Version Online: `http://opensource.ucc.ie/ct2003/OSSMovementsCommunities-2003.draft.pdf`, acessed 2006-02-07.

Perens, B. (1998). The Open Source Definition. Online: `http://www.perens.com/Articles/OSD.html`, accessed 2006-01-31.

Perens, B. (2006). Open Standards Principles and Practice. Online: `http://perens.com/OpenStandards/Definition.html`, accessed 2006-02-15.

Potsar, V. and E. Chang (2004). Open Source and Closed Source Software Development Methodologies. In *Collaboration, Conflict and Control – Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburgh, Scotland, pp. 105–109. IEEE Computer Society.

Radinger, W. and K. M. Goeschka (2003). Agile Software Development for Component Based Software Engineering. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Anaheim, CA, USA, pp. 300–301. ACM Press, New York, NY, USA. ISBN: 1-58113-751-6.

Raja, U. and E. Barry (2005). Investigating Quality in Large-scale Open Source Software. In *5-WOSSE: Proceedings of the fifth Workshop on Open Source Software Engineering*, St. Louis, Missouri, pp. 1–4. ACM Press, New York, NY, USA. ISBN: 1-59593-127-9.

Raymond, E. (2001). *The Cathedral & The Bazar - Musings On Linux And Open Source By An Accidental Revolutionary* (Revised edition ed.). O'Reilly. ISBN: 0-596-00108-8.

Raymond, E. S. (2000). The Cathedral & the Bazaar. Online: `http://www.catb.org/~esr/writings/cathedral-bazaar/`, accessed 2006-02-07.

Read, A. W. (Ed.) (2003). *The New International Webster's Comprehensive Dictionary of The English Language*. Trident Press International. ISBN: 1-58279-558-4.

Reifer, D. J., V. R. Basili, B. W. Boehm, and B. Clark (2003). Eight Lessons Learned during COTS-Based Systems Maintenance. *IEEE Software 20*(5), 94–96.

Remenyi, D., B. Williams, A. Money, and E. Swartz (1998). *Doing Research In Business And Management*. SAGE Publications. ISBN: 0-7619-5949-1.

Robson, C. (2003). *Real World Research*. Blackwell Publishing. ISBN: 0-631-21305-8.

Rosen, L. (2002). Geek Law: Why the Public Domain Isn't a License. *Linux J. 2002*(102), 12. Online: `http://delivery.acm.org/10.1145/580000/571797/6225.html?key1=571797&key%2=2517000411&coll=Portal&dl=ACM&CFID=68573562&CFTOKEN=5521419`, accessed 2006-02-15.

Rossi, C. and A. Bonaccorsi (2005). Why Profit-Oriented Companies Enter the OS Field?: Intrinsic vs. Extrinsic Incentives. In *5-WOSSE: Proceedings of the fifth Workshop on Open Source Software Engineering*, St. Louis, Missouri, pp. 1–5. ACM Press, New York, NY, USA. ISBN: 1-59593-127-9.

Scacchi, W. (2002). Understanding the Requirements for Developing Open Source Software Systems. *IEEE Proceedings–Software 149*(1), 24–39.

Scacchi, W. (2004). Free and Open Source Development Practices in the Game Community. *IEEE Software 21*(1), 59–66.

Software, S. (2006). About Us. Online: `http://sleepycat.com/company/aboutus.html`, accessed 2006-04-28.

SourceForge.net (2006). SourceForge.net: Software Map. Online: `http://sourceforge.net/softwaremap/trove_list.php?form_cat=14`, accessed 2006-02-11.

Stallman, R. (1999). *The Gnu Operating System and the Free Software Movement, Open Sources*, Chapter Open Source as a Business Strategy, pp. 53–70. O'Reilly Media, Inc.

Sutor, B. (2006). Open Standards vs. Open Source. Online: `http://www-128.ibm.com/developerworks/blogs/dw_blog_comments.jspa?blog=%384&entry=105886`, accessed 2006-02-15.

Tiemann, M. (1999). *Open Sources: Voices from the Open Source Revolution*, Chapter Future of Cyguns Solutions - An Entrepreneur's Account, pp. 71–90. O'Reilly Media, Inc.

Torchiano, M. and M. Morisio (2004). Overlooked Aspects of COTS-Based Development. *IEEE Software 21*(2), 88–93.

Tuomi, I. (2003). Internet, Innovation and Open Source: Actors in the Network. *First Monday 6*(1), 1–1. Online journal: `http://www.firstmonday.org/`.

University of California (1999). The BSD License.

van der Linden, F. (2005). COSI Full Project Proposal. Version 1.3, Confidential.

von Hippel, E. and G. von Krogh (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science 14*(2), 209–223.

von Krogh, G., S. Spaeth, and K. R. Lakhani (2003). Community, Joining, and Specialization in Open Source Software Innovation: a Case Study. *Research Policy 32*, 1217–1241.

von Krogh, G. and E. von Hippel (2003). Editorial Special Issue on Open Source Software Development. *Research Policy 32*(7), 1149–1157.

Warsta, J. and P. Abrahamsson (2003). Is Open Source Software Development Essentially an Agile Method? In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, Portland, Oregon, pp. 143–147.

Webbink, M. H. (2003). Understanding Open Source Software. *The New South Wales Society for Computers and the Law Journal 51*, 1–1. Online: `http://www.nswscl.org.au/journal/51/Mark_H_Webbink.html`, accessed 2006-02-10.

Weber, S. (2004). *The Success of Open Source*. Harvard University Press (Camebridge). ISBN: 0-674-01292-5.

Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén (2000). *Experimentation in Software Engineering - An Introduction*. Boston / Dorrecht / London: Kluwer Academic Publishers.

Yin, R. K. (2003). *Case Study Research Design and Methods* (3rd. ed.). Sage Publications.

Part V

Appendices

Appendix A

The Open Source Definition

The Open Source Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

The Open Source Definition

Appendix B

Questionnaire

Survey on Open Source Software Development

Dear colleague,

This survey concerns the current state of practice of Open Source Software (OSS) development. The main focus areas of this research are motivation for managing and participating in OSS development, relationships with OSS communities and processes for development of, and development with, OSS tools and components.

The survey has been developed as part of the European ITEA project, COSI (Co-development using inner & Open source in Software Intensive products), where ICT Norway and NTNU (Norwegian University of Science and Technology) are part of the Norwegian subproject.

This questionnaire is organised in six parts where part 1 and 6, and only **one** of the parts 2 through 5 should be completed per response. Which part you should fill out depends on your relationship to OSS related development. If you are related to OSS through several products or projects we would be grateful if you completed more than one copy of this questionnaire.

The second part, *Initiating and Managing OSS* should be completed by respondents who work in a company or local business unit which controls or has controlled the development of a product licensed with an OSS license. **The third part,** *Participating in Inner Source Development*, should be completed by respondents who have participated in software development projects using OSS software engineering practices within a limited scope, such as in a closed consortium of trusted partners. **The fourth part,** *Interacting with or Participating in OSS Communities* should be completed by respondents who interact with or participate in an OSS project controlled by someone outside their company or local business unit. **The fifth part,** *Developing Software with OSS Tools and Components***,** should be completed by respondents who participate or have participated in software development projects using OSS tools or components.

The first part, *Demographic Information***,** contains questions related to the OSS product which the respondent is related to, and should be completed by all respondents. **The sixth part,** *Demographic Information***,** contains questions related to the respondent and the respondent's company, and should be completed by all respondents.

The information reported in this questionnaire will be treated confidentially.

Thank you for participating in this survey.

Contact information:
Dr. Carl-Fredrik Sørensen,
IDI, NTNU, NO-7491 Trondheim
Email: Carl-Fredrik.Sorensen@idi.ntnu.no
Phone: +47 73 59 07 31 Fax: +47 7359 4466

Survey Overview

Survey Objective	The purpose of this questionnaire is to survey the current state of practice of OSS related Software Engineering, involving ICT companies in several countries. The *units of study* are OSS products owned by companies, software development projects using OSS practices, OSS projects where companies participate, or software system which contains OSS components.
Confidentiality	All responses to this questionnaire will be handled confidentially, including information on whether the company has participated. The results will describe and analyze the current state-of-practice of OSS related development in the ICT industry. The final report will not contain detailed information about the company or the respondent. Neither will it be possible to draw any conclusions about a given company.
Benefits	A final, technical report of the survey will be electronically available to all respondents by email or through a web-link. Diffusion will also take place through ICT Norway and the ITEA project, COSI (Co-development using inner and Open source in Software Intensive products).
Some general instructions	Use an "X" when you mark an answer to a question, e.g., in a table. Filling-in of the questionnaire should not take more than half an hour.

Table of Content

Part 1. Product Information

In this part information about the product developed by your company or the community will be elaborated.

1) What is the name of the product?
Name: ________________________________

2) In what business area are most of the product's customers?
[Mark one or more alternatives]

☐ Consulting companies
☐ Software companies
☐ Manufacturing Industry
☐ Telecommunications
☐ Public/Health
☐ Educational institutions
☐ Banking
☐ Non-profit organizations
☐ Media
☐ Private/domestic
☐ Other:________________________________

3) What is the main functionality of the product?
[Mark one or more alternatives]

☐ Database
☐ Desktop/Office tools
☐ Software development tool or component
☐ Enterprise solutions
☐ Financial/Banking
☐ Games
☐ Multimedia
☐ Networking
☐ System administration tool
☐ Web/Portals
☐ Operating system
☐ Middleware
☐ Server component
☐ Other:________________________________

4) What kind of product is this?
[Mark one alternative]

☐ It is commodity software or infrastructure and has basic functionality common to software in several market segments

☐ It has functionality which is basic for a certain business and has similar functionality to software in the same market segment

☐ It has differentiating functionality which is unique for this product in the market segment

5) What is the product primary used as?
[Mark one or more alternatives]

☐ Component in other software or infrastructure

☐ Standalone user product

☐ Infrastructure or middleware

6) Which programming language(s) are used in the implementation of this product?
[Mark one or more alternatives]

☐ C

☐ C++

☐ C#

☐ Java

☐ Perl

☐ PHP

☐ Python

☐ Visual Basic

☐ JavaScript

☐ Other: ____

7) Please list the five most important OSS-components used in this product, if any.
This could for example be MySQL, Hibernate or Apache HTTP Server.

Number	**Component**
C1	
C2	
C3	
C4	
C5	

8) Please list the five most important OSS development tools used to develop this product, if any.
This could for example be Bugzilla, Emacs, GNU Compiler Collection or Subversion.

Number	Tool/technology
T1	
T2	
T3	
T4	
T5	

9) Which license(s) are used for the product?
[Mark one or more alternatives]

☐ Don’t know

☐ The Artistic License

☐ (New) Berkeley Software Distribution (BSD)

☐ General Public License (GPL)

☐ Lesser General Public License (LGPL)

☐ The MIT license

☐ Own proprietary license(s) (please specify):________________________________
Other: ____

Part 2. Initiating and Managing OSS

This part is only relevant for respondents working in companies, or business units, owning products licensed with OSS licenses. If your company or local business unit controls more than one product licensed as an OSS product, we would be grateful if you completed this section several times. Please start with the one you are most familiar with and answer the following questions.

2.1. Business Motivations

10) To what extent do you agree that the following motivations *were* important when turning this product into an OSS product?

We are interested to know the motivations behind turning this product into an OSS product, or in other words, what did you want to achieve by choosing an OSS licensing scheme for this product.
[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. Idealism	**1**				**2**	
b. Increased attention and publicity					**3**	
c. Improved relationship to OSS community by returning value to the community	**1**		**1**	**1**		
d. Saved effort by shared development. Less development effort is put on the company because of community contributions				**3**		
e. Decreased maintenance effort by getting help from the community			**1**	**2**		
f. Increased product quality. Having a big community reduce the number of bugs (defects), because more bugs are found				**1**	**2**	
g. Increased innovation by having a big community (increase the number of new requirements and ideas)			**1**	**2**		
h. Reduced time-to-market because a big community can test and review the product and thus find and correct bugs (defects) faster				**2**	**1**	
i. Reduced time-to-market and cost by making integration of other OSS products or components possible	**1**			**2**		
j. Other (please specify)						

11) If there were commercial motivations behind turning this product into an OSS product, what was business idea behind doing it?

[Mark one or more alternatives]

☐ **1 / 3 -** No commercial motives

☐ **2 / 3 -** Sell proprietary licenses (Dual licensing)

☐ **2 / 3 -** Integration and deployment services

☐ **2 / 3 -** Development and sales of proprietary add-on or related software

☐ **2 / 3 -** Provide and sell support

☐ **2 / 3 -** Provide and sell total product responsibility

☐ **1 / 3 -** Packing and distribution of software

☐ **1 / 3 -** User training

☐ Other: ____

12) To what degree do you agree with the following statements related to the advantages and disadvantages by having this product as an OSS product?

We are interested in the advantages and disadvantages related to having this product as an OSS product today. If there is other important advantages or disadvantages please specify these in the last row.

[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. The company has increased attention and publicity because this product is an OSS product			**1**		**2**	
b. It is hard to focus on one group of customers since the product is available to everyone			**1**			**2**
c. A big community increases the number of potential paying customers				**1**	**1**	**1**
d. The company's relationship to the OSS community has improved because this product is open source			**1**	**2**		
e. Costs are reduced because of shared development. The community provide functionality and less development effort is put on the company			**3**			
f. Maintenance costs has decreased because the community helps with maintenance		**1**	**2**			
g. Innovation has increased because a big community increase the number of new requirements and ideas			**1**	**1**	**1**	
h. It is more difficult to plan releases because of constant input from the community		**2**			**1**	
i. Input and requests from the community leads to extra work	**1**		**2**			

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
j. Code provided by the community has too low quality to be included directly into the product		**2**	**1**			
k. Many contributions from several different community members increase the effort to keep a sound architecture, design and implementation		**1**	**1**	**1**		
l. Increased product quality. Having a big community reduce the number of bugs (defects), because more bugs are found				**2**	**1**	
m. It is more difficult to protect intellectual property because the product is open source		**1**			**2**	
n. It is more difficult to protect business secrets because the product is open source					**1**	**2**
o. Security is reduced because the openness (of the code) makes it easier to find security holes	**1**	**1**				**1**
p. The time-to-market has improved because the product is tested and reviewed by many and bugs are found faster			**1**	**2**		
q. The time-to-market and the cost have been reduced because other OSS products can be integrated into the product			**1**	**2**		
r. Having this product as open source enable us to sell related products or services			**1**	**2**		
s. Selling licenses to the product is difficult because the product is freely available			**2**			**1**
t. Other (please specify)						

2.2. Creating an Open Source Product

13) When was the product initiated?
Please specify the date in the format YYYY-MM-DD, for example 2006-07-26.

Date: Company A: 1996-01-01, Company B: 2000-11-1, Company F: 2004-01-01

14) When was the product first released, with an OSS license?
Please specify the date in the format YYYY-MM-DD, for example 2006-07-26.

Date: Company A: 2005-06-22, Company B: 2000-11-1, Company F: 2005-06-01

15) How was the development of this product, licensed with an OSS license, initiated? Please describe a timeline including the main events which took place, the most important decisions and how much time and effort were spent initiating this project?

Company A:
The product is already being developed for about 10 years.

Each party contrubutes to the project bsed on there own businesscases.

Milestone ID	Milestone date	Contents / Deliverables
1	27/5/2005	Create proposal for open source licence.
2	15/6/2005	Discuss Lincence with legal department
3	15/6//2005	Investigate CVS mechanism
4	22/6/2005	Create project(s) in Sourceforge
5	22/6/2005	Install CVS within Philips and AGFA
6	1/7/2005	Prepare Code
7	1/7/2005	Finalize design Documentation
9	15/7/2005	Place code under Sourceforge
10	15/7/2005	Set up Sourceforge enviroment(s) (Defect tracking/ forum)
11	1/8/2005	Create Sourceforge homepage(s)
12	1/8/2005	Promote Sourceforge project

Total cost for putting the project into th open source is approximately 4 to 6 week effort (160 to 240 hours)"

Company B:
In 1999 eZ publish was only one of many things we were working on. As a small company (4-5 persons) at that time we were working with that our limited number of customers wanted.

We were actually focusing on some software targeted for traders and financial institutions. Along the way we also had some local customers we were developing development websites for.
This web development resulted in product named eZ publish.

All of us in the company at that time was big Linux and OSS enthusiasts so it was actually not a big issue. ""Of course we should release it as open source"". But in the beginning this was just something we would to on the side. Our main target was the trader software. And this software was not going to be OS.

However, eZ publish got so much Company that we actually realized that this was an opportunity we couldn't let go..
Our thoughts were -
- By licensing this by OS this will spread around all the world
- Many users will result on many potential customers
- Even though the software itself is free, many would need other service which we can get payed for
- Duallicensing it was an idea we had from the beginning. People should also have the oppertunity to buy a commercial licence if the opensource licence is not suitable to them

Company F:
No information.

2.3. *Activity in the Community*

16) How many times has the product been downloaded by the community?
Number: Company A: 6000, Company B: 1600000, Company F: No information

17) How many person-months were spent by the employees in your business unit during the last year related to this product?
Number: Company A: 40, Company B: 170, Company F: 2

18) How was the company able to attract participants and interest to this product from outside the company?
[Mark one or more alternatives]

☐ **1 / 3 -** No community exists

☐ **0 / 3 -** A community already existed for the product before it was made open source

☐ **0 / 3 -** Commercials and advertisements for the product

☐ **0 / 3 -** Scientific publications or publications in popular press about the product

☐ **0 / 3 -** The company participates in other OSS communities

☐ **2 / 3 -** The product was made available on websites (your own and/or sites like sourceforge.net)

☐ **0 / 3 -** Conferences, etc. for developers and users

☐ **1 / 3 -** Forums, mailing-lists and other infrastructure was made available to the community

☐ Other:____

19) Approximately how many people participate in the following roles in the OSS community around the product?

Company A	**Outside the company**	**Inside the company**
a. Core-developers	<5	<5
b. Developers	<5	<5
c. Active users	51-100	21-50
d. Passive users	501-1000	51-100

Company B	**Outside the company**	**Inside the company**
e. Core-developers	5-10	5-10
f. Developers	5-10	11 - 20
g. Active users	1001-5000	21-50
h. Passive users	> 10000	11 - 20

Company F	**Outside the company**	**Inside the company**
i. Core-developers	<5	<5
j. Developers	<5	<5
k. Active users	<5	5 - 10
l. Passive users	<5	5 - 10

20) To what extent do *internal users or developers* in your business unit perform the following activities related to this OSS product?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Test the product			**1**	**2**		
b. Report bugs (faults)			**1**	**2**		
c. Fix bugs (defects)		**1**		**1**	**1**	
d. Provide new requirements			**1**	**2**		
e. Implement new functionality			**1**	**1**	**1**	
f. Develop add-on components	**1**	**1**	**1**			
g. Provide locale adaptations of the product	**2**					**1**
h. Create documentation	**1**	**1**		**1**		
i. Provide free user support		**2**		**1**		
j. Develop architecture or design			**1**		**2**	
k. Other (please specify)						

21) To what extent do *users or developers* from the community perform the following activities related to this OSS product?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Test the product			**1**	**2**		
b. Report bugs (faults)			**1**	**2**		
c. Fix bugs (defects)	**1**		**2**			
d. Provide new requirements			**2**	**1**		
e. Implement new functionality		**1**	**2**			
f. Develop add-on components	**1**		**1**	**1**		
g. Provide locale adaptations of the product	**1**		**1**			**1**
h. Create documentation		**1**	**2**			
i. Provide free user support		**1**	**1**	**1**		
j. Develop architecture or design		**2**	**1**			
k. Other (please specify)						

22) To what extent are the following communication methods used between *developers inside* the business unit related to this product?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Instant messaging	**1**			**2**		
b. Chat-channels	**3**					
c. Web-forums	**1**	**1**		**1**		
d. Private e-mail				**2**	**1**	
e. Mailing-lists		**1**		**2**		
f. Video meetings	**3**					
g. Phone			**2**	**1**		
h. Formal and informal face to face meetings			**2**	**1**		

23) To what extent are the following communication methods used between *internal developers in your business unit and, users and developers from the community* related to this product?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Instant messaging	1		2			
b. Chat-channels	3					
c. Web-forums				2	1	
d. Private e-mail		1	2			
e. Mailing-lists			1	2		
f. Video meetings	3					
g. Phone	1	1	1			
h. Formal and informal face to face meetings	1	1	1			

2.4. OSS Development process

24) Which development process is used in the development of products licensed with OSS licenses in your company or business unit?

Please describe how you treat contributions and new requirements from the community, the phases, key activities and roles of the development process and any process tools used.

Company A:
"Each contributor implements the functionality that is requested for the internal organisation.
Every 3 months all contributor met do agree on 6 months roadmap.
Each week progress and new activities (such as Pr's) are discussed in a development meeting.
Contributions for other are reviewed an integrated by the core developers."

Company B:
- Contributions are received by mailinglist or bug system. The contribution must exists of the following:
  - the code itself
  - Description of what it does
  - Documentation on how it works
- We are looking on the functionality, is this something we want in the product
- code quality is review. Is it implemented in the right way?
- Code can be rejected. An explenation is then given (what is not correct according to the guidelines)
- Verifying that we have a contract for transfer of copyright from the contributor
- Code is approved"

Company F:
Not information

25) To what extent do you agree with the following statements related to the development process used to develop this product?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large		Don't know
a. Compliance to open standards is important to your company				**2**	**1**		
b. You have modified the code written by other open source projects		**1**	**1**		**1**		
c. You have shared modified code with other open source projects			**2**	**1**			
d. Code ownership is practiced, one or few developers function as gatekeepers for each their part of the code, controlling the code that is checked into the code-repository			**1**	**2**			
e. Code repository tools like Subversion or CVS are always used in inner source software development					**3**		
f. More than one branch of code is kept active at any time (both stable and unstable)				**1**	**2**		
g. The product had early releases, with pre-releases before the final version		**1**			**2**		
h. The product had frequent releases, with a release more frequent than every other week	**1**	**1**	**1**				
i. Test-driven development is used and tests are written prior to implementation		**1**	**1**	**1**			
j. The same development environment and development tools are used by all participating developers.			**1**	**1**	**1**		
k. Developers can choose their work assignments themselves	**1**	**1**	**1**				
l. The developers of the product are also users of the product or domain experts		**1**		**1**			**1**
m. The end users are involved in the development process every day		**3**					
n. The status and plans of the product are open to the community (number of active bugs, roadmaps etc.)				**2**	**1**		
o. The developers take time to respond to requests and proposals from the community every day			**3**				

	Not at all	Small	Some	Large	Very large	Don't know
p. Everyone in the community is free to participate and contribute to the development of the product (requirements, patches, proposals etc.)				**2**	**1**	
q. Communication and documentation is kept informal		**1**	**1**	**1**		
r. Development of the software is done on several distributed locations		**1**		**1**	**1**	

Part 3. Participating in Inner Source Development

This part concerns inner source development, where OSS software engineering practices and principles, are used in a limited scope. These practices could for instance be sharing source code among partners, or performing distributed development using collaboration methods. The scope of an inner source development project could be a business-unit, company, or a consortium of trusted partners. Please select one project using inner source development where your company participates. For a clarification of the terms related to inner source see page 43.

3.1. Experiences from Inner Source Development

26) To what extent do you agree with the following statements about your experiences from the inner source development?

We are interested to know the experiences you have had with using OSS software engineering practices in this project.
[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. Eased distribution of development teams across organisational and geographical boundaries		**1**	**1**	**1**	**2**	**1**
b. Increased flexibility in starting, stopping, and changing of collaborations		**1**	**1**	**1**	**2**	**1**
c. Saved effort by shared development between the partners. Less development effort is put on the company because of contributions from partners			**1**	**2**	**1**	**2**
d. Decreased maintenance effort by getting help from the partners				**3**	**1**	**2**
e. Increased product quality. Having a big community reduce the number of bugs (defects), because more bugs are found			**2**	**1**	**2**	**1**
f. Reduced time-to-market because partners can test and review the product and thus find and correct bugs (defects) faster			**2**		**2**	**2**
g. Increased innovation by having a big community (increase the number of new requirements and ideas)			**2**	**1**	**1**	**2**
h. Other (please specify) 'Not at this moment						

3.2. Participants in Inner Source Development

27) Approximately how many people participate in the following roles in this inner source project?

Company A	**Business Unit in Company**	**Company**	**Collaborating companies**
a. Core-developers	21-50	5 - 10	< 5
b. Developers	101-500	21-50	< 5
c. Active users	21-50	51-100	< 5
d. Passive users	11 - 20	501-1000	< 5

Company I	**Business Unit in Company**	**Company**	**Collaborating companies**
e. Core-developers	< 5	< 5	< 5
f. Developers	< 5	< 5	< 5
g. Active users	< 5	< 5	< 5
h. Passive users	< 5	< 5	5 - 10

Company E	**Business Unit in Company**	**Company**	**Collaborating companies**
i. Core-developers	< 5	< 5	< 5
j. Developers	< 5	< 5	5 - 10
k. Active users	5 - 10	5 - 10	11 - 20
l. Passive users	5 - 10	< 5	5 - 10

Company C	**Business Unit in Company**	**Company**	**Collaborating companies**
m. Core-developers	< 5	< 5	< 5
n. Developers	< 5	< 5	< 5
o. Active users	< 5	< 5	11 - 20
p. Passive users	< 5	< 5	501-1000

Company A	**Business Unit in Company**	**Company**	**Collaborating companies**
q. Core-developers	21-50	11 - 20	11 - 20
r. Developers	101-500	51-100	51-100
s. Active users	1001-5000	1001-5000	1001-5000
t. Passive users	5001-10000	5001-10000	5001-10000

Company A	Business Unit in Company	Company	Collaborating companies
u. Core-developers	11 - 20	< 5	< 5
v. Developers	21-50	51-100	< 5
w. Active users	< 5	51-100	< 5
x. Passive users	< 5	51-100	< 5

28) To what extent are the following communication methods used by *participants* in this inner source project?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Instant messaging	**2**	**1**	**2**			**1**
b. Chat-channels	**4**		**1**			**1**
c. Web-forums	**4**	**1**	**1**			
d. Private e-mail	**1**		**1**	**1**	**3**	
e. Mailing-lists			**1**	**3**	**2**	
f. Video meetings	**2**	**2**	**1**	**1**		
g. Phone		**1**	**3**	**1**	**1**	
h. Formal and informal face to face meetings		**1**	**3**	**2**		

3.3. Inner Source Development processes

29) Which development process was used in this inner source development project?
Please describe how you treat contributions and new requirements from partners, the phases, key activities and roles of the development process and any process tools used, and which of the partners, users or customers in the project who have access to the source code.

Company A:
We use a process called InnerSource, which is based on the Open Source principles. The process defines three collaboration styles 'use as is', 'contribute', and 'virtual team'.

Company I:
There exist a core developer that decides if the contributions and new requirements are included in the middleware. The source code is available to all partners accordingly with the GPL license of the component. Patches or new functionality is send by e-mail to the core developer who decides to include or not in the component the modifications. Requirements are also requested by e-mail. There exist SCM of the component in order to control the different releases.

Company E:
Project management according to projectplan. RUP is used for early stage prototyping and as a proof of concept. Source code is handled via CVS trees.

Company C:
"1.- Contributions and new requirements from partners: We gather and study new requirements and we include them in the new version requirements of the product.

2.- The phases:
Project Management.
Tool Specification
Software Testing
Software Packaging & Commercialization
Following the life cycle (""Software Development life cycle"") specified in the process ""Product & Service Development"" of the internal quality system.

3.- Key activities and roles of the development process:
Key Activities:
Project planning, project tracking: project leader
detailed tool specifications: project leader
requirements implementation: developers
integration & system testing: developers and project leader
preparation of the customer package: developers and project leader

4.- Process tools used:
Project Management: MS Project
Project Development: MS Visual Studio

5.- users or customers in the project who have access to the source code:
Nobody at this moment"

Company A:
"Typical development process with a partner
1. Partner and Company:Initial face to face discussions to clarify global requirements and design.
2. Partner: Development partner works out requirements, design and tests in documentation

3. Company: Review of this documentation
4. Partner: Implement code on a fixed Clear Case version
5. Partner: Integrate code and test the implemented code
6. Company: Review code
7. Company: Merge the code into the main VOB and perform delivery testing

Steps 2-7 are repeated several times during development. The order code be somewhat different in occasional situations."

Company A:
"- contributions and requirements from partners are input to CCB
- after the conception phase of a project the set of requirements is fixed
- core developers implement the requirements
- a number of developers work together with customers in separate projects
- the way in which contributions are fed back still needs to be detailed"

30) To what extent do you agree with the following statements related to the development process used to develop this product?
[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large		Don't know
a. Compliance to open standards is important to your company		**1**	**3**		**1**		**1**
b. You have read parts of the inner source software's code developed by other partners	**1**		**2**	**3**			
c. You have modified the code written by another partner	**1**	**1**	**1**	**3**			
d. You have shared modified code with partners	**1**		**2**	**3**			
e. Code ownership is practiced, one or few developers function as gatekeepers for each their part of the code, controlling the code that is checked into the code-repository	**1**	**1**		**4**			
f. Code repository tools like Subversion or CVS are always used in inner source software development	**1**			**3**	**2**		
g. More than one branch of code is kept active at any time (both stable and unstable)		**1**	**2**	**3**			
h. The product had early releases, with pre-releases before the final version			**1**	**4**	**1**		
i. The product had frequent releases, with a release more frequent than every other week	**1**	**2**	**3**				
j. Test-driven development is used and tests are written prior to implementation	**3**		**2**	**1**			
k. The same development environment and development tools are used by all participating developers.		**1**	**2**	**3**			

	Not at all	Small	Some	Large	Very large	Don’t know
l. Developers can choose their work assignments themselves	**2**		**2**	**1**		**1**
m. The developers of the product are also users of the product or domain experts	**1**		**3**	**2**		
n. The end users are involved in the development process every day	**1**	**4**	**1**			
o. The status and plans of the product are open to the community (number of active bugs, roadmaps etc.)	**2**	**1**	**1**	**2**		
p. The developers take time to respond to requests and proposals from the community every day		**2**	**2**	**2**		
q. Everyone in the community is free to participate and contribute to the development of the product (requirements, patches, proposals etc.)			**4**	**2**		
r. Communication and documentation is kept informal	**1**	**1**	**4**			
s. Development of the software is done on several distributed locations		**1**		**4**	**1**	

Part 4. Interacting with or Participating in OSS Communities

This part is concerned with the relationship between your company and OSS communities. Please select one work-related OSS project *you know*, where you, your company or other employees in your company are involved. If you are familiar with several OSS projects we would be grateful if you completed the questionnaire several times. Please start with the project you are most familiar with. We are interested in involvement from reading mailing lists and providing new requirements to major implementations of new functionality.

4.1. Relationship to the OSS Community

31) To what extent does your company/business unit perform the following activities, or contribute the artefacts or resources below to the community?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. New requirements to the project	**1**	**3**		**1**		
b. New functionality with code	**1**	**2**	**1**	**1**		
c. Architecture or design	**2**	**2**		**1**		
d. Testing	**1**	**1**	**3**			
e. Bug-reports (faults)		**2**	**3**			
f. Bug-fixes (defects)	**1**	**1**	**2**	**1**		
g. Documentation	**4**		**1**			
h. Provide local adaptations of the product	**1**	**2**	**1**	**1**		
i. Add-on components	**3**	**1**	**1**			
j. Integration towards other products (glueware)	**4**	**1**				
k. Free user support	**4**	**1**				
l. Distribution of software	**3**	**1**	**1**			
m. Infrastructure (e.g. servers or storage space)	**5**					
n. Money	**4**		**1**			
o. Subscribe to mailing-lists from the community		**1**	**2**		**2**	
p. Read OSS related news channels and forums like sourceforge.net, slashdot.org, freshmeat.net etc.		**1**	**1**	**1**	**2**	
q. Other (please specify) **Company A:** 'Sponsor company that gives commercial support on OSS, see http://www.theaceorb.nl/en/acetao.html				**1**		

32) To what extent are you satisfied with the following elements in the OSS community in this project?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Documentation (availability and quality)		**1**	**1**	**1**	**2**	
b. The functionality of the product				**3**	**2**	
c. The non-functional quality attributes of the product			**1**	**3**	**1**	
d. The code quality			**2**	**1**	**1**	**1**
e. Product support			**1**	**3**	**1**	
f. Response to bug reports			**1**	**2**	**1**	**1**
g. Response to new requirements			**2**		**1**	**2**
h. Response time		**1**		**2**	**1**	**1**
i. Other (Please specify)						

33) To what degree do you agree with the following statements?

[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. The company/business unit's relationship with the OSS community is successful				**3**	**1**	**1**
b. Your company/business unit's ability to change or influence the OSS product and its requirements would have been greater if the company contributed more to the community			**1**	**4**		
c. Participation in OSS communities is rooted in your company/business unit's business model		**2**	**1**	**1**		**1**
d. Your company/business unit has changed its development processes because of participation in OSS projects	**1**	**1**	**1**	**1**		**1**

34) To what extent are the following communication methods *used inside the company/business unit* related to this OSS project?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Instant messaging	**3**	**1**	**1**			
b. Chat-channels	**5**					
c. Web-forums	**3**			**1**		**1**
d. Private e-mail		**1**	**2**	**1**	**1**	
e. Mailing-lists		**1**	**2**	**2**		
f. Video meetings	**5**					
g. Phone	**1**	**1**	**2**	**1**		
h. Formal and informal face to face meetings		**1**	**2**	**2**		

35) Which of the following communication methods are used in communication with *community members* outside the company/business unit related to this OSS project?

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Instant messaging	**2**	**1**	**1**			**1**
b. Chat-channels	**1**	**2**	**1**			**1**
c. Web-forums	**1**	**1**			**1**	**2**
d. E-mail	**1**		**1**	**2**	**1**	
e. Mailing-lists			**2**	**2**	**1**	
f. Video meetings	**4**					**1**
g. Phone	**4**					**1**
h. Formal and informal face to face meetings	**3**	**1**				**1**

36) How many, from your company or business unit, have the following roles in the community around this OSS project?

Company B	**Number:**
a. Core-developers	< 5
b. Developers	< 5
c. Active users	< 5
d. Passive users	11 - 20

Company G	**Number:**
e. Core-developers	< 5
f. Developers	< 5
g. Active users	< 5
h. Passive users	< 5

Company D	Number:
i. Core-developers	< 5
j. Developers	< 5
k. Active users	< 5
l. Passive users	5 - 10

Company A	Number:
m. Core-developers	< 5
n. Developers	< 5
o. Active users	< 5
p. Passive users	< 5

Company C	Number:
q. Core-developers	< 5
r. Developers	< 5
s. Active users	5 - 10
t. Passive users	< 5

37) How does the company or business unit organize participation in this OSS project?

[Mark one or more alternatives]

☐ **3 / 5 -** As an internal project

☐ **0 / 5 -** Through the line organization

☐ **2 / 5 -** At an individual level

☐ **1 / 5 -** Participation is not organized by the company, but voluntarily by the individuals

4.2. Motivation behind Interaction with and Participation in OSS Projects

38) What is the company's or business unit's motivation for participating in the OSS project?

☐ The company/business unit has never participated as a company in any OSS projects. Participation has been in form of individual contributions.

[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Influence the community	**1**	**2**	**1**	**1**		
b. Learn from the community		**2**		**3**		
c. Reduce time to market	**1**	**2**		**1**	**1**	
d. Increase product quality (fewer defects)		**1**	**2**	**2**		
e. Benefit from development and testing performed by community users				**3**	**2**	
f. Improve relationship to the OSS community			**3**	**2**		
g. Use of the OSS product is required in the company	**4**				**1**	
h. Other (Please specify) Research					**1**	

39) What is your personal motivation for participating in OSS projects?

[Mark one alternative per row]

☐ I have never personally participated in any OSS projects

	Not at all	Small	Some	Large	Very large	Don't know
a. I get paid by my employer	**2**	**1**	**1**	**1**		
b. Learning			**2**		**3**	
c. Curiosity	**1**		**3**		**1**	
d. Idealism	**1**			**2**	**2**	
e. Desire to identify with the community	**1**	**2**	**1**		**1**	
f. Feel obligated to contribute to the community	**2**	**2**	**1**			
g. Future work possibilities	**2**		**2**	**1**		
h. Personal interests of the product	**1**	**1**		**3**		
i. Altruism (Feel satisfaction by doing something for other people)	**1**		**1**	**2**	**1**	
j. Reputation among peers	**2**	**2**	**1**			
k. Other (Please specify) my thesis					**1**	

4.3. Processes Used when Interacting with OSS Projects

40) Which development process is used in your company/business unit when your company participates in an OSS project?

Please describe the key activities, tools and roles in the development process.

Company B:
The OSS project is fundamental to our product. It is important for us to be involved and know where this project is heading. For instance if there is a problem/defect that is crucial for our own product it is very important that we can get this fixed. Without contribution and commitment to this community that is more difficult.

Company G:
Iterative and incremental. We combine "bits and peaces" from other OSS product to create our own.

Company D:
"Team leading - Project leader. Word, Excel, MS Project.
Requirements, design - System Architect, Technical project leader and developers. UML.
Implementation and module test - Developers. GCC.
Integration - Technical project leader/integrator and developers. Subversion, Cruise Control."

Company A:
Adapt to process of OSS project

Company C:
Not specified

41) Which development process do you use when you personally participate in OSS projects?

Please describe the key activities, tools and other involved roles.

Company B:
CVS is used for version controll
Most of our time is used on bug fixing

Company G:
I do bug fixes which I need myself and then contribute them to the community. Occasionally, I just take bugs from the tracker and fix them, just to contribute.

Company D:
As above depending on my role.

Company A:
Adapt to process of OSS project.

Company C:
Not specified.

Part 5. Developing Software with OSS Tools and Components

If you have participated in more than one development project in your company where you have used OSS components, please select the project that you are most familiar with and answer the questions in the next sections. If you do not use any OSS components in your development, please skip forward to Part 6.

5.1. Motivations Related to Using OSS Components

42) To what degree do you agree that the following elements were the motivation behind using OSS components in this project?

If other motivations were important, please specify in the last row.

[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. Reduced risk of component provider going out of business	**2**	**3**	**2**	**1**	**1**	
b. Reduced risk of component provider changing market strategies	**1**	**3**	**3**	**1**	**1**	
c. Reduced risk of selected component evolving into an unwanted direction	**1**	**3**	**3**	**1**		**1**
d. Component could be acquired for free		**2**	**1**	**3**	**3**	
e. Source code was available and could easily be changed		**2**	**1**	**3**	**3**	
f. To use OSS components was decided by the customer	**6**		**3**			
g. Political reasons (company policy, licensing conditions)	**3**	**3**	**1**		**1**	**1**
h. The components were available for test and use	**1**			**4**	**4**	
i. Honesty and openness from the provider about the status of the component	**1**	**2**	**3**		**3**	
j. Reduced risk by ability to remove defects yourself	**1**	**1**	**2**	**3**	**2**	
k. Compliance to standards		**1**	**2**	**6**		
l. Idealism	**3**	**2**	**2**	**1**	**1**	
m. Other (please specify)						

5.2. Development Processes Using OSS Components

43) Which development process was used in this project?
Please describe key activities and roles in the component based development process.

Company H
Traditional waterfall approach, release based process.
Milestone based project management process.

Requirements are received from users: e-mail requests to contact persons, in surveys, meetings.
Tool feedback database in use.

Concept group that covers representatives from user organizations takes care of roadmapping.
Steering group decides of the release content.
Feasibility studies by the development organization.
Design and implementation subcontracted.

Deployment to customers planned for each release.
Training needs covered and training planned for each release.
Testing by users.

Company D
Requirements and main design by System Architect.
Detailed design, implemenation and test by developers.
Cross-team reviews of requirements, design and implementation.

Company A
Agile version of RUP

Company I
Requirements gathering was performed by two organizations. Design and implementation was done by one organization. Bug reports and patches by several.

Company E
"Modified RUP was used for testing and choosind a suitable OS product and platform.
Key activities and roles:
Meritie: Inner source project lead / high level design / constraints
Subcontractors: Component additional design / implementation, documenting, OS integration
OSC: OS project lead / design / development and continuous improvement. Provides interfaces for os/inner source"

Company I
Ad-hoc process, not a well defined process it is employed.

Company J
Process based on Feature Driven Development (FDD) modified with additions of how to manage architecture and MDD.
Phase 1: Planning & high level design and architecture
Phase 2: Development: Number of feature iterations involving analysis and design for feature, implementation and verification of feature

Phase 3: Test & evaluation of release

Company F
previously comented

Company A
Normal waterfall

44) Where do you find OSS components?

Please name the places you search to find OSS components.

Company H
From our customers (that are users from the Nokia business units) that have used the OSS based tools and wants to have them under the global service provided by us.
Also, components are found from Internet, community web pages.

Company D
Google.
Sourceforge.

Company A
Internet

Company I
N/A

Company E
tigris.org
google.com
java-source.net
jboss.org

Company I
Google, Sourceforge, freshmeat, javalobby, TheServerSide

Company J
SourceForge, IBMs developerworks, and other OS sites.
Google in general

Company F
websites as sourceforge and simillar ones

Company A
Internet

45) Which process do you use to select OSS components?

Please describe the key selection criteria, activities and roles in the selection process.

Company H
No specific approach from literature.

Road mapping process. Concept group selects components and proposes them (release contents) to the steering group. Customer feedback process: proposals.
Feasibility studies: selection and implementation details.

Company D
Key selection criteria; Maturity, robustness, user base, code/documentation quality.
Activitities; Selection if possible through peer/college recommendation otherwise info through mailing lists, books, magazines.
Roles; Selection usually made by system architect or lead developers.

Company A
- Fit in our product's architecture
- Provides required functionality and performance
- Evaluate alternatives
- Public license used
- Get authorisation from our legal department

Company I
ad-hoc process, mainly taking into account provided functionality

Company E
The selection strategy had following constraints: (in order of magnitude)

1. Quality
2. Robustness
3. Scalability
4. Open interfaces
5. Steady prospects for future development
6. Licencing policy"

Company I
Basically the process consist in:
-- Identify several components/products with similar functionality
-- Check the code stability in some degree
-- Select the most suitable one

Company J
Depends on the size, complexity and role of the component.
This process is NOT described in our development methodology.

Smaller components are selected by developers based on need.
When a developer /development team faces a problem in a featureiteration:

* are there existing solutions ""out there""?
  - look for component meeting the functional requirements
    (preferably without too much more)
* shortlist based on
  - license
  - reputation
  - bug track
  - activity in the community
  - weight of the community itself (like Apache) or of the industrial partners in the community (like IBM)

* download and test the component
* integrate into development

Larger components are evaluated in testbed and is far more complex to evaluate and select.

Company F
previously comented

Company A
There is one person responsible for the PROCESS of introducing OSS components. He gets requests from engineers, talks with project managers, and ensures that
- effort is spent to check the component
- component is in line with long-term objective
- determines risks and mitigation based on impact of introduction of that OSS component"

46) To what extent do you agree with the following statements related to the development process used to develop this product?
[Mark one alternative per row]

	Not at all	Small	Some	Large	Very large	Don't know
a. Compliance to open standards is important to your company			**2**	**4**	**3**	
b. You have read parts of the OSS components' code used in this project		**2**	**3**	**3**		**1**
c. You have modified the OSS components' code used in this project	**2**	**5**		**2**		
d. You have shared modified code with the OSS community	**3**	**2**	**3**	**1**		
e. Code ownership is practiced, one or few developers function as gatekeepers for each their part of the code, controlling the code that is checked into the code-repository	**1**	**2**	**4**	**2**		
f. Code repository tools like Subversion or CVS are always used in inner source software development			**1**	**3**	**5**	
g. More than one branch of code is kept	**1**	**2**	**3**	**1**	**2**	

	Not at all	Small	Some	Large	Very large	Don't know
active at any time (both stable and unstable)						
h. The product had early releases, with pre-releases before the final version	**1**	**1**	**3**	**2**	**2**	
i. The product had frequent releases, with a release more frequent than every other week	**2**	**7**				
j. Test-driven development is used and tests are written prior to implementation	**2**	**3**	**3**	**1**		
k. The same development environment and development tools are used by all participating developers.		**1**	**1**	**5**	**2**	
l. Developers can choose their work assignments themselves	**4**	**2**	**2**	**1**		
m. The developers of the product are also users of the product or domain experts		**2**	**4**	**2**	**1**	
n. The end users are involved in the development process every day	**1**	**5**		**3**		
o. The status and plans of the product are open to the community (number of active bugs, roadmaps etc.)	**3**	**2**	**1**	**2**		**1**
p. The developers take time to respond to requests and proposals from the community every day	**1**	**4**	**2**	**1**		**1**
q. Everyone in the community is free to participate and contribute to the development of the product (requirements, patches, proposals etc.)	**4**	**1**	**1**	**1**	**1**	**1**
r. Communication and documentation is kept informal		**4**	**2**	**2**	**1**	
s. Development of the software is done on several distributed locations	**3**	**1**	**3**	**1**	**1**	

5.3. Development with OSS Components

47) How much of the product application functionality in your company's finished product is provided by OSS components?

Company H:	**81-100 %**
Company D:	**0-20 %**
Company A:	**0-20 %**
Company I:	**81-100 %**
Company E:	**0-20 %**
Company I:	**0-20 %**
Company J:	**41-60 %**
Company F:	**21-40 %**
Company A:	**0-20 %**

48) In your company's product, how much of the OSS components belong in the following categories?
[Please fill in percentages which should sum up to 100%]

Company H

81-100 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
0-20 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment

Company D

81-100 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
0-20 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment

Company A

81-100 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
0-20 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment

Company I	
61-80 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
21-40 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment
Company E	
41-60 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
21-40 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment
Company I	
61-80 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
41-60 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment
Company J	
81-100 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
0-20 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment
Company F	
21-40 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
21-40 %	Basic for business or similar to functionality as software in the same market segment
21-40 %	Unique for this component in the market segment
Company A	
41-60 %	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
21-40 %	Basic for business or similar to functionality as software in the same market segment
0-20 %	Unique for this component in the market segment

49) To what degree do you agree that the following issues make you discard an OSS component?
[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. No or little support from the community or provider	**1**		**3**	**2**	**3**	
b. Documentation with low availability and quality		**1**	**2**	**4**	**2**	
c. No one guarantees product quality of the OSS component	**1**	**3**		**4**	**1**	
d. The product lifetime and roadmap of the OSS component is unknown	**1**	**2**	**1**	**4**	**1**	
e. No or little activity in the community		**1**	**3**	**3**	**2**	
f. Low quality code. It is messy, in other (natural/written) languages etc.	**1**		**3**		**5**	
g. The component has no stable releases		**3**	**1**	**4**	**1**	
h. The component has no track record or reputation		**3**		**4**	**2**	
i. The component has too many dependencies (platform, other components, standards etc.)		**2**	**2**	**4**	**1**	
j. Other (please specify):						

50) To what extent do you agree with the following statements related to the use of OSS components in the development of your product?
[Mark one alternative per row]

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. The project was delivered long after schedule or had severe delays	**3**	**3**	**1**	**2**		
b. Effort to select OSS components was not satisfactorily estimated		**6**	**2**	**1**		
c. Effort to integrate OSS components was not satisfactorily estimated		**2**	**4**	**3**		
d. OSS components negatively affected system reliability	**2**	**7**				
e. OSS components negatively affected system security	**5**	**4**				
f. OSS components negatively affected system performance	**3**	**5**	**1**			
g. Requirements to the product your	**1**	**2**	**6**			

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
company developed were changed a lot						
h. OSS components could not be sufficiently adapted to changing requirements	**3**	**5**	**1**			
i. It was difficult to identify whether defects were inside or outside the OSS components	**1**	**8**				
j. It was difficult to plan system maintenance, e.g., because different OSS components had asynchronous release cycles	**1**	**3**	**4**			**1**
k. It was difficult to update the system with the latest OSS component version	**1**	**3**	**4**	**1**		
l. OSS components were not satisfactorily compatible with the production environment when the system was deployed		**5**	**4**			
m. Information about the reputation and technical support ability of community providing the components were inadequate or not available	**2**	**4**	**2**	**1**		
n. Community of provider did not provide enough technical support/ training	**1**	**5**		**1**	**1**	**1**
o. If you have a regular selection process, to what degree did this process reduce the risks of the project?		**1**	**2**	**1**		**5**
p. OSS licenses restrict us from using the components the way we wanted		**2**	**2**	**3**	**1**	**1**
q. Adaptation of OSS components resulted in high maintenance cost	**1**	**3**	**4**	**1**		
r. It is difficult to influence the evolution of OSS components		**1**	**4**	**2**	**2**	

51) **What is currently the main software development challenges related to using OSS components in software projects?**

Please describe what you see as your company's biggest challenges related to software development using OSS components and tools.

Company H:
No answer.

Company D:
Synchronization of changes.

Company A:
-

Company I:
Definition of several processes: selection fof components, integration, mainteinance and evolution management

Company E:
Finding high quality OS components for some projects. Integrating inner source to OS. To have monotonous quality in the product after integration.

Company I:
-- Lack of documentation related to API and component architecture

-- Bad scripts used for configuring and compiling the whole integrated system

Company J:
We use OS in a conservative way, selecting well established components and not relying on future releases. Our main challenges are still:
* Maintenance (example of forks resulting in lack of backward compatibility)
* Influence on the roadmap of components

Company F:
not information

Company A:
Several OSS components use license agreements (like GPL) that are not acceptable for our company.

Part 6. Demographic Information

6.1. About the company

52) What is the name of your company or business unit?
Company name: ______________________

53) Where is your company/business unit located?
Please fill in the country and town where you are currently working
Company location: ______________________

54) What is the type of company?
[Mark one alternative]

☐ Stand-alone: The highest reporting entity with no parent organization above it.

☐ Subsidiary: An independent entity with majority interest held by a parent.

55) What is the ownership of your company?
[Mark one alternative]

☐ Publicly traded on a stock exchange

☐ Privately held company

☐ Government, education, or non-profit organization

56) What is the staff size of your mother company in your own country (full- & part-time persons) ____?

57) What is the staff size of your local business unit (full- & part-time persons) ______?
[This staff size may be the same for small and medium sized companies]

58) What is the main business area of your company?
[Mark one alternative]

☐ IT/ Telecom industry (Ericsson, Nokia, NERA, Nordic VLSI etc.)

☐ Telecom service provider (Telenor, Telefonica, Orange etc.)

☐ Software House / Software Vendor (Opera, Microsoft etc.)

☐ Software / IT consulting company (Accenture, CapGemini E&Y, TietoEnator etc.)

☐ Retail product with large degree of embedded software (Philips, Sony etc.)

☐ Other software-intensive company (please specify): ______________________________

6.2. About the respondent

59) What is your gender?

- ☐ Male
- ☐ Female

60) How old are you?

- ☐ <25
- ☐ 26-30
- ☐ 31-25
- ☐ 36-40
- ☐ 41-45
- ☐ 46-50
- ☐ 51-55
- ☐ 56-60
- ☐ 60-65
- ☐ 65<

61) What is your current position in your business unit?
[Mark one or more alternatives]

- ☐ IT manager
- ☐ Project manager
- ☐ Software architect
- ☐ Software developer
- ☐ Web designer
- ☐ Tester/Quality Assurance
- ☐ Other: ___________________________

62) How long have you been working in the current business unit?
Years: ___________________________

63) How long have you been working with software development?
Years: ___________________________

64) How long have you been working with OSS?
Years: ___________________________

65) How many projects using OSS component have you been involved in?
Projects: _________________________

66) How many OSS projects have you been involved in?
Projects: _________________________

67) What is your highest completed education?
[Mark one alternative]

☐ Bachelor (BSc)

☐ Master (MSc)

☐ Ph.D.

☐ Other education: ____________________

68) Is your degree software-related for instance informatics, computer science or telecommunications?
[Mark one alternative]
□ No
□ Yes

6.3. *Feedback on the questionnaire*

69) How much time in minutes was used to complete the survey?
Minutes: __________________________

Thank you for taking the time to answer this questionnaire. If you have any feedback on this questionnaire then please give it in the field below. If you had to make any assumptions on any on the questions these can also be provided here.

Definitions of terms used in this survey

Roles *Core-developer:* A developer responsible for the main functionality of the system, adding new functionality and making the most important design and architecture decisions. *Developers:* A developer which makes code contributions to the software, fixing defects and creating add-on functionality. *Active user:* A user which is involved in several activities, including suggesting new requirements, reporting fault descriptions etc. *Passive user:* A user which is not contributing to the project, instead the user only passively uses the product.
Differentiating software: Software that is unique and gives a business edge compared competing products with similar functionality.
Commodity software: Undifferentiated software, like infrastructure, common and available to everyone in the market.
Component: Software components are *program units* of independent production, acquisition, and deployment that can be composed into a functioning system. We focus on components that are neither shipped with the operating system, provided by the development environment, nor included in any pre-existing platform. Typical examples of OSS components are Hibernate, Xerces (XML parser) and OpenEJB.
Open Source Software (OSS): Software where the source code is available with a license allowing modifications and redistribution of modified versions of the software and code.
Inner Source: Inner source development uses aspects of open source software engineering inside a business unit, company, or consortium (association companies engaging in a joint venture). The software developed is ***only*** open and available to a closed consortium of partners.
Open Source community: The environment where OSS is created, used and discussed.
Licensing *Open Source License:* Licenses which are compliant to the Open Source definition, for instance GPL, LGPL and New BSD. *Dual licensing:* A licensing model where a piece of software is licensed under two different licenses with the purpose of providing freedom to the user, by allowing them to choose between an OSS licenses or a proprietary license.

Appendix C

Interview Guide

Formål: Formålet med intervjuet er å finne ut mer om små og mellomstore bedrifters forhold til bruk av og deltagelse i utvikling av åpen kildekode.

Presentere oss: Andreas siv.ing student ved NTNU som gjennomfører diplomoppgaven sin, Carl-Fredrik post.doc ved NTNU og leder for to arbeidspakker i COSI-prosjektet og Øyvind PhD-stipendiat ved NTNU.

Gjennomføring: Undersøkelsen gjennomføres som del av norskCOSI (Co-development using inner & Open source in Software Intensive products) hvor IKT-Norge, eZ Systems, Linpro, Keymind og NTNU er med som partnere.

Informasjon: Informasjonen som kommer frem i disse intervjuene vil bli bruk i arbeidet med å utforme en forbedret spørreundersøkelse, arbeidet relatert til arbeidspakke 2 i COSI-prosjektet samt studentenes oppgaver. OBS! Hvis nødvendig må oppgavene våre kanskje "hemmeligstemples". **Konfidensialitet!/?**

Opptak: Vi ønsker å ta opp samtalen på bånd slik at det er lettere å analysere det som ble sagt i ettertid. Hvis du føler ubehag med dette må du si i fra.

Før vi begynner, er det noe du lurer på?

Hvis det er ting du ikke vil svare på må du si i fra om det.

Opplysninger om respondenten:
Navn

Kjønn

Alder

Kontaktinformasjon (e-post, telefon, adresse?)

Underveis:
Vi må lytte.
Vi må ta notater (avklare rolle).
Se om vi kan avdekke om det er flere termer vi bør forklare.

Start: Vi vil begynne med å gå gjennom spørreskjemaet slik det foreligger for dere i den samme rekkefølgen.

Part 1. Prosjekter kontrollert av selskapet

1) Kontrollerer firmaet du jobber i ett eller flere åpen kildekodeprosjekter?

1.1. Informasjon om prosjektet

2) Hva heter prosjektet
Name: ______________________________

3) Hvilket forretningsområde (kunder) henvender produktet seg til?

a. Passer kategoriene som er oppgitt

4) Hva er den viktigste funksjonaliteten til produktet?

a. Passer kategoriene som er oppgitt

5) Hvordan vil du klassifisere produktet?
- ☐ Commodity software
- ☐ Felles for alle produkter i det samme markedssegmentet
- ☐ Det har unik fuknsjonalitet, ulik alle andre produkter

6) Hvordan er produktet først og fremst brukt?
- ☐ Komponent i annen programvare
- ☐ Sluttbrukerprodukt
- ☐ Komponent i infrastruktur

7) Hvilke programmerinsspråk er brukt i produktet?

8) Hvilke teknologier og eller kompoenter basserer produktet seg på? Hvilke liseneser er disse komponentene/teknologiene lisensiert under?

1.2. Business related

9) Hvilke lisenser bruker dere for deres produkt?

10) Hva var motivasjonen bak beslutningen om å gjøre produktet open source?

a. Dekker de følgende kategoriene deres motivasjon?

Motivations	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree		Don't know
b. Idealistic							
c. Increase attention and publicity							
d. Improve relationship to open source community							
e. Saved costs by shared development (less development effort on the company)							
f. Decrease maintenance costs							
g. A big community gives increased product quality (reduce the number of bugs)							
h. A big community gives increased innovation (increase the number of new requirements and ideas)							
i. Improve time-to-market							

11) Hvordan ønsker dere å tjene på å gjøre produktet åpent?

□ Salg av lisenser
□ Levere tjenester (angi hvilke?)
□ Annet: ____

12) Hva er fordelene med å ha produktet som åpen kildekode?
[Mark one alternative per row]

Advantages	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree		Don't know
a. Increased attention and publicity							
b. Improved relationship to open source community							
c. Saved costs by shared development (less development effort on the company)							
d. Decreased maintenance costs							
e. Improved time-to-market							
f. A big community gives increased product quality (reduce the number of bugs)							
g. A big community gives increased innovation (increase the number of new requirements and ideas)							
h. Enables the use and integration of other OSS products							

13) Hva er ulempene ved å ha produktet som åpen kildekode?
[Mark one or more alternatives]
- ☐ Selling the product is difficult licensing restrictions
- ☐ Difficulty of protecting intellectual property
- ☐ Difficulty of protecting business secrets
- ☐ Security is reduced because openness (of the code) makes it easier to find security holes
- ☐ More difficult to plan releases
- ☐ Other: ______________________________

1.3. Start-up

14) Når ble prosjektet påbegynt?

15) Når ble prosjektet åpen kildekode?

16) Hvor mye arbeid ble brukt på prosjektet det siste året (i måneder)?

17) Kan du beskrive prosessen for å starte prosjektet/for å få det startet som et åpent kildekodeprosjekt?

1.4. Community

18) Eksisterer det et community rundt produktet?
- ☐ Yes, established by the company
- ☐ Yes, established by the community itself
- ☐ No

19) Hvordan har firmaet klart å bygge dette communityet?

- ☐ Community already existed for the product before it was made open source
- ☐ Commercials and advertisements
- ☐ The company participates in other open source communities
- ☐ The product was made available on a website
- ☐ Conferences for developers and users
- ☐ Forums and other community supporting infrastructure was made available to the community
- ☐ The community was created by founders of the company
- ☐ Other:____

20) Hvor mange deltagere innehar disse rollene i communitiet?

Role	**Outside the company**	**Inside the company**
a. Core-developers		
b. Developers		
c. Active users		
d. Passive users		

21) Til hvilken grad utfører ansatte i selskapet de følgende aktivitetene? Er det andre aktiviteter som også utføres av de ansatte?

Activity	Not at all	Small	Some	Large	Very large	Don't know
a. Bug-reporting						
b. Bug-fixing						
c. New requirements						
d. New functionality						
e. Add-on components						
f. Localization						
g. Other(please specify)						

22) Til hvilken grad utføres de følgende oppgavene av folk utenfor selskapet? Er det også andre oppgaver de utfører?

Activity	Not at all	Small	Some	Large	Very large	Don't know
a. Bug-reporting						
b. Bug-fixing						
c. New requirements						
d. New functionality						
e. Add-on components						
f. Localization						
g. Other(please specify)						

23) Hvor mange ganger har produktet blitt lastet ned?

24) I hvilken utstrekning brukes følgende kommunikasjonsarenaer innad i bedriften i dette prosjektet?

[Mark one alternative per row]

Arena	Not at all	Small	Some	Large	Very large		Don't know
a. Instant messaging							
b. Chat-channels							
c. Web-forums							
d. E-mail							
e. Mailing-lists							
f. Video meetings							
g. Phone							
h. Formal and informal face to face meetings							

25) I hvilken utstrekning brukes følgende kommunikasjonsarenaer utad av bedriften mot community i dette prosjektet?

[Mark one alternative per row]

Arena	Not at all	Small	Some	Large	Very large		Don't know
a. Instant messaging							
b. Chat-channels							
c. Web-forums							
d. E-mail							
e. Mailing-lists							
f. Video meetings							
g. Phone							
h. Formal and informal face to face meetings							

26) Hvilke utviklingsprosesser er hovedsakelig brukt i dette prosjektet? Kan du beskrive de viktigste aktivitetene og rollene?

Part 2. Participating in an open source project

27) Deltar ditt firma aktivt i åpne kildekode communities?

28) Hvorfor deltar dere i åpen kildekodeprosjekter?

☐ Ability to influence the community
☐ Learn from the community
☐ Reduced time to market
☐ Higher product quality
☐ Benefit from development and testing from community users
☐ Improved relationship to the OSS community
☐ Requires usage of the OSS product
☐ Other: ____

29) Deltar du personlig i noen åpen kildekodeprosjekter?

30) Hvorfor deltar du i åpen kildekodeprosjekter?

☐ You get paid
☐ Learning
☐ Idealism
☐ Desire to identify with the community
☐ Feel obligated to tribute to the community
☐ Future work possibilities
☐ Personal interests
☐ Altruism (Feel satisfaction by doing something for other people)
☐ Reputation among peers
☐ Other: ____

2.1. About the project

31) Hva heter prosjektet?
Name: ________________________

32) Hvilke brukere retter produktet seg mot?

- ☐ Consulting Companies
- ☐ Software companies
- ☐ Industry
- ☐ Telecommunications
- ☐ Public/Health
- ☐ Educational institutions
- ☐ Banking
- ☐ Other:______________________________

33) Hvilke funksjonalitet er den viktigste I produktet?

- ☐ Database
- ☐ Desktop/Office tools
- ☐ Development
- ☐ Enterprise solutions
- ☐ Financial/Banking
- ☐ Games
- ☐ Multimedia
- ☐ Networking
- ☐ System administrator
- ☐ Web/Portals
- ☐ Operating System
- ☐ Server Component
- ☐ Other:______________________________

34) Hvordan vil du klassifisere produktet?

- ☐ Commodity software
- ☐ Felles for alle produkter i det samme markedssegmentet
- ☐ Det har unik fuknsjonalitet, ulik alle andre produkter

35) Hvordan er produktet først og fremst brukt?

- ☐ Komponent i annen programvare
- ☐ Sluttbrukerprodukt
- ☐ Komponent i infrastruktur

36) Hvilke programmeringsspråk er brukt i prosjektet?

2.2. Organization

37) Hvordan er bedriftens deltagelse i prosjektet organisert?

- ☐ Project
- ☐ Individual level
- ☐ Line organization

38) Hvor mange fra ditt firma deltar i følgende roller?

Role	**Number:**
a. Core-developers	
b. Developers	
c. Active users	
d. Passive users	

39) Til hvilken grad ufører dere de følgende arbeidsoppgavene?

Activity	Not at all	Small	Some	Large	Very large	Don't know
a. Bug-reporting						
b. Bug-fixing						
c. New requirements						
d. New functionality						
e. Add-on components						
f. Localization/translate						
g. Other:						

2.3. Relationship with community

40) I hvilken grad er du tilfreds med følgende bidra fra community?

Contribution	Not at all	Small	Some	Large	Very large	Don't know
a. Infrastructure						
b. Product with code						
c. Requirements						
d. Bug-fixes						
e. Money						
f. Product support						
g. Other:						

41) Hva tilbyr dere til community?

Contribution	Not at all	Small	Some	Large	Very large	Don't know
a. Infrastructure						
b. New functionality with code						
c. Requirements						
d. Bug-fixes						
e. Money						
f. User support						

g. Distribution						
h. Other(please specify)						

42) I hvilken grad er du enig i følgende utsag?

Statement	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. The relationship with the open source community is successful						
b. Your company's ability to change or influence the open source product and its requirements would have been greater if the company contributed more to the community						
c. Participation in open source communities is rooted in your company's business model						

43) I hvilken grad er følgende kommunikasjonskanaler brukt innad i bedriften i forhold til kommunikasjon i prosjektet?

Arena	No	Very small	Small	Some	Great	Don't know
a. Instant messaging						
b. Chat-channels						
c. Web-forums						
d. E-mail						
e. Mailing-lists						
f. Video meetings						
g. Phone						
h. Formal and informal face to face meetings						

44) I hvilken grad er følgende kommunikasjonskanaler brukt i kommunikasjon med community i forhold til kommunikasjon i prosjektet?

Arena	No	Very small	Small	Some	Great	Don't know
a. Instant messaging						
b. Chat-channels						
c. Web-forums						
d. E-mail						
e. Mailing-lists						

f. Video meetings						
g. Phone						
h. Formal and informal face to face meetings						

45) Hvilke utviklingsprosesser blir brukt for å delta i prosjektet.

Part 3. Use of open source tools and components

3.1. Information about usage of open source

46) Hva er de viktigste åpen kildekode vertkøyene dere bruker samt deres lisenser?

47) Hva er de viktigste åpen kildekode komponentene dere bruker og deres lisenser?

48) Hva er den vanligste metoden for utvikling med OSS komponenter?

3.2. Motivations behind and disadvantages of usage of open source components

49) I hvilken grad er du enig i følgende utsag rundt motivasjon for bruk av oss komponenter?

Motivations	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a) Reduce risk of component provider going out of business						
b) Reduce risk of component provider changing market strategies						
c) Reduce risk of selected component evolving into an unwanted direction						
d) Component can be acquired for free						
e) Source code is available and can easily be changed						
f) It was decided by the customer						
g) Political reasons (company policy, licensing conditions)						
h) Other (please specify):						

3.3. Development with open source components

50) I hvilken grad er du enig i følgende utsagn relatert til utvikling med OSS komponenter:

Role	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
a. Compliance to open standards is important to your company						
b. Your company has changed its development processes because of participation in open source projects						
c. Your company changed its development processes because of using open source components						
d. You have read parts of the open source components' code which are used in this project						
e. You have modified the code of the open source components which are used in this project						

51) Hvor mye av produktet er OSS komponenter? KODELINJER??BYTE??

52) Hvor mye av de komponentene som er OSS er:

__________%	Commodity software or infrastructure which has very basic functionality common to most software in many market segments
__________%	Basic for business or similar to functionality in, all software in the same market segment
__________%	Unique for this product in the market segment

53) Beskriv prosessene deres for å velge OSS komponenter.

3.4. *Development with OSS components*

4.1 Hva er din mening om de følgende punktene relatert til utvikling med åpen kildekode komponenter?

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree		Don't know
a) The project was delivered long after schedule or has had severe delays							
b) Effort to select OSS components was not satisfactorily estimated							
c) Effort to integrate OSS components was not satisfactorily estimated							
d) OSS components negatively affected system reliability							
e) OSS components negatively affected system security							
f) OSS components negatively affected system performance							
g) Requirements to the product were changed a lot							
h) OSS components could not be sufficiently adapted to changing requirements							

	Don't agree at all	Hardly agree	Agree some what	Mostly agree	Strongly agree	Don't know
j) It was difficult to identify whether defects were inside or outside the OSS components						
k) It was difficult to plan system maintenance, e.g. because different OSS components had asynchronous release cycles						
l) It was difficult to update the system with the last OSS component version						
m) OSS components were not satisfactorily compatible with the production environment when the system was deployed						
n) Information on the reputation and technical support ability of community or provider were inadequate						
o) Community of provider did not provide enough technical support/ training						
p) If you did use a formal selection process, to what degree did this process reduce the risks of the project? q) Licenses put restrictions on how to use the components						
r) Adaptation of OSS components resulted in high maintenance cost						
s) It is difficult to influence the evolution of OSS components						

54) Hva ser du på som ditt selskaps største utfordringer relatert til bruk av OSS?

Part 4. About the company

55) What is the type of company?
[Mark one alternative]

□ Stand-alone: The highest reporting entity with no parent organization above it.
□ Subsidiary: An independent entity with majority interest held by a parent.

56) What is the ownership of your company?
[Mark one alternative]

□ Publicly traded on a stock exchange
□ Privately held company
□ Government, education, or non-profit organization

57) What is the staff size of your embedding company (full- & part-time persons) ____?

58) What is the staff size of your local business unit (full- & part-time persons) ______?
[This staff size may be the same for small and medium sized companies]

59) What is the main business area of your company?
[Mark one alternative]

□ IT/ Telecom industry (Ericsson, Nokia, NERA, Nordic VLSI etc.)
□ Telecom service provider (Telenor, NetCom etc.)
□ Software House / Software Vendor (Opera, Microsoft etc.)
□ Software / IT consulting company (Accenture, CapGemini E&Y, TietoEnator etc.)
□ Other software-intensive company (please specify): ______________________________

Part 5. About the respondent

60) What is your current position in your business unit?
[Mark one or more alternatives]
- □ IT manager
- □ Project manager
- □ Software architect
- □ Software developer
- □ Web designer
- □ Tester/Quality Assurance
- □ Other: ___________________________

61) How long have you been working in the current business unit?
Years: ___________________________

62) How long have you been working with software development?
Years: ___________________________

63) How long have you been working with OSS?
Years: ___________________________

64) How many projects using OSS component have you been involved in?
Projects: _________________________

65) How many open source projects have you been involved in?
Projects: _________________________

66) What is your highest completed education?
[Mark one alternative]
- □ Bachelor (BSc)
- □ Master (MSc)
- □ Ph.D.
- □ Other education: ____

67) Is your degree software-related (informatics/computer science/telecommunications)?
[Mark one alternative]
□ No
□ Yes

Appendix D

Guidelines

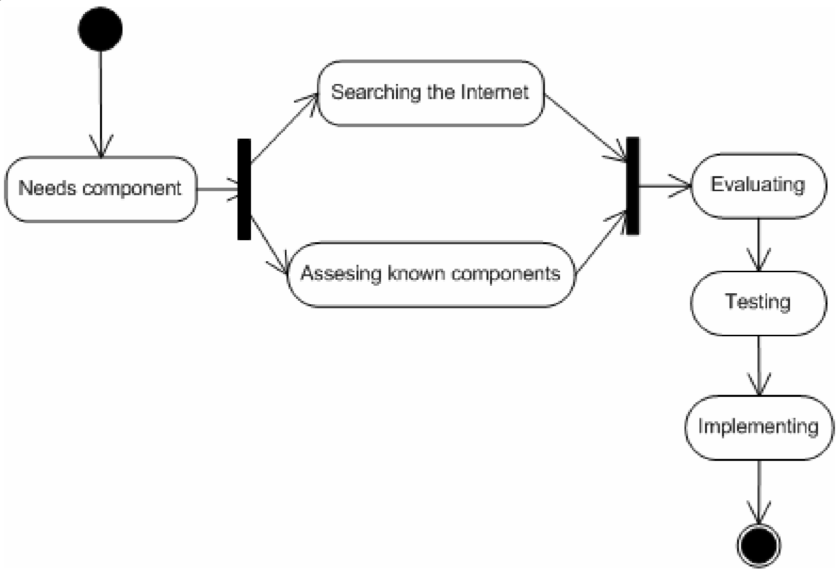
D.1 Selecting and Evaluating OSS Components

Guidelines for Selection and Evaluation of OSS Components

These preliminary guidelines are meant to help companies which use open source components in their software development. These guidelines are not final, they are meant as contribution to the work in the European ITEA project, COSI.

Selection process for OSS components

The following selection process is recommended as a way to efficiently select OSS components:



Possible sources for OSS components

Perhaps the most used source for OSS components is previous knowledge to a component. You may know of components yourself, you may ask colleagues, or former colleagues, or you may ask expert networks[1], or mailing lists. Some other information sources are listed in the table below[2].

Site	Url
Freshmeat.net	http://freshmeat.net/
IBM's resource for developers	http://www-128.ibm.com/developerworks/
Javalobby	http://www.javalobby.org/
Java-Source.net	http://java-source.net/
koders	http://koders.com/
Slashdot	http://slashdot.org/
SourceForge.net	http://sourceforge.net/
TheServerSide.com	http://www.theserverside.com/
Tigris.org	http://www.tigris.org/

To increase the knowledge about components and the open source community, it is important that participation in open source activities is accepted, and perhaps also rewarded.

[1] A list of expert networks and possible mailing lists online should be gathered.
[2] This list is no way near complete, or final. The list should be extended and reviewed.

Information sources

To evaluate a component you need information. In addition to the web resources in the table above, you may find information on community mailing lists and web forums, project web sites, the source code, the project documentation, online statistics from for instance SourceForge, books and magazines concerning open source, and from peers, or colleagues.

Evaluation criteria

Possible evaluation criteria for OSS components will be presented next.

Evaluation criteria	Explanation/metric
Functionality	Does the component provide the functionality you require?
Licenses	Is the license compatible with the license you use?
Maturity and state of the component	Does the component have a stable release? Does the component many open bugs-reports?
Reputation	Does the component have a track record? Is it used by many others? Is the provider reliable?
Roadmap	What are the future plans of the component? Will they stop supporting the current version? Are new features about to be implemented?
Documentation and information	Is there any information available on the component? Is the component well documented?
Community development process	Do the community providing the component have procedures for testing, building, common code standards, and so on?
Activity	How many times is the component downloaded? How many active users does the community have? How many messages are posted on the mailing lists or forum? How often is the component upgraded?
Support	Does the community provide some support? Do the community members help each other with user-support?
Implementation	Is the component implemented in a confidence-inspiring way? Are code-standards used? Is the code well documented? Is the structure of the component sound? Is the component easy to extend?

Motivations for using OSS components

Motivation	Explanation
No cost	Most open source components are downloadable for free
Availability of the component	Open source components are easy to evaluate. Fully featured components are normally available on the components web site.
Availability of the source code	The source code is available with the component. It allows you to read, and possibly change the code if necessary.
Availability of information, openness, and honesty	Open source communities are normally quite open about the status of the product. All information about the product is normally available to everyone.
Standard compliance	Many open source products implement (open) standards as well.

D.2 Initiating and Managing an OSS Product

Guidelines for Initiating and Managing an Open Source Project

These preliminary guidelines are meant to help companies which think about licensing one of their products with an open source license. These guidelines are not final, they are meant as contribution to the work in the European ITEA project, COSI.

Should we release this product as an open source product?

The more of the following statements that are true, the more likely are you at succeeding with an open source release of your product.

Statement
You can make money having this product as an open source product.
You will not lose sales, or give away important business secrets by releasing this product as an open source product.
You need the product internally in the company.
Your developers are users of the product.
There is a potentially big market for the product, with many potential customers and community members.
An active community with technically competent people exits around the product.
No economical benefits are required from the product.
The product is commodity software.
There are no or few similar open source products.
You have sufficient resources to continue the development of this product.

How to get a community?

The more of the following statements that are true, the more likely are you to attract a community of users, and possible customers.

Statement
You know how to, and have the resources to make the product available, and widely know to a big group of potential community members.
The product provides value to its users.
You are active in other, related open source communities.
You publish information and articles about your product.
You have an infrastructure which supports the distribution of your product, and the activity in an open source forum.

How to benefit from having a community?

The more of the following statements that are true, the more likely are you to benefit from having a community.

Statement
You have clear procedures for community contributions.
You have informed the community about these procedures.
You communicate openly with your community.
You have an infrastructure, bug-reporting systems, forum, communication channels, etc, in place to take care of the community.
You have resources to handle the contributions and the feedback from the community.
The product has few bugs.
The product and its code is well documented.
The product is easy to use.

D.3 Inner Source Development Projects

Guidelines for Inner Source Development Projects

These guidelines are meant to help companies choose development practices from OSS projects, which can be used in Inner Source development projects. Inner source development uses aspects of open source software engineering inside a business unit, company, or consortium of cooperating companies. This will enable development to be distributed at several physical locations, and allow companies to take advantage of development practices which have proved to be successful in open source projects.

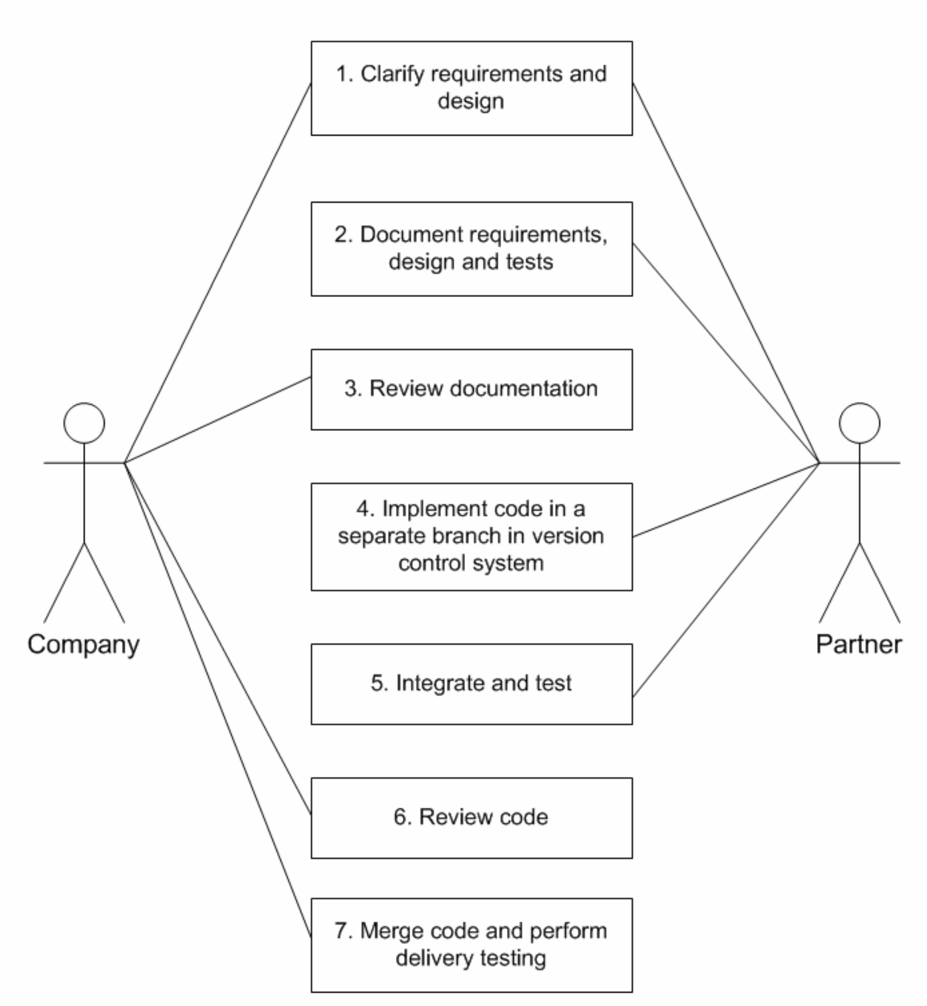
Inner Source Development Practices

The following development practices are typically used in OSS projects, and can be adapted to be used inside a company, as inner source development.

Development practice	Explanation
Share source code with partners	Source code can be shared with the partners in the inner source project. This will enable partners to participate more efficiently in development and improve code developed by partners.
Development process influenced from OSS projects and Agile Software Development	Many inner source projects use methodologies which have similarities with Agile software development, which is a characteristic which is similar to development in OSS projects. This can be referred to as emulating OSS practices on the inside of the company. For example, frequent releases, and close involvement of customers, users and the community.
Community	Inner Source projects have the possibility to involve community members from other companies. This includes core-developers, developers, active users, and passive users, which all can contribute in some way to the inner source project.
Code ownership	Code ownership means that developers are given responsibility for developing and maintaining specific modules in the system. The owner of a particular module is responsible for committing changes to that module, based on contributions from participants in the inner source project.
Communication methods	Many inner source projects use asynchronous communication methods like email and mailing lists. This will enable a timeline of discussions, and the possibility to communicate in a distributed setting. However, some companies are cautious about using only email and mailing lists.
Development tools	Version control systems such as Subversion and CVS are frequently used to coordinate development, and mediate changes to the source code.

Inner Source Development process

The following development process can be used in inner source projects. The development process is based on cooperation between developers at the company, and developers at a partner company. These entities are represented in the figure as ‘Company’ and ‘Partner’. The process involved 7 steps, which are shown in the figure. The steps are self explanatory.
The process is iterative, where steps 2 to 7 are repeated several times during the development.


1. Clarify requirements and design
2. Document requirements, design and tests
3. Review documentation
4. Implement code in a separate branch in version control system
5. Integrate and test
6. Review code
7. Merge code and perform delivery testing
Company
Partner

Note that development processes used in inner source projects have to be adapted to the company and environment they are used in. Some companies have used and adapted Agile development, and Rational Unified Process, to be used in projects where inner source development is used.