

Open Source Software in Software Intensive Industry - A Survey

Øyvind Hauge

Master of Science in Computer Science
Submission date: September 2007
Supervisor: Reidar Conradi, IDI
Co-supervisor: Carl-Fredrik Sørensen, IDI

Problem Description

The industrially controlled research project ITEA COSI has observed a shift in the software intensive industry. Open Source Software (OSS) and open source development practices are used more often. As a participant in the ITEA COSI project, NTNU contributes in the development of a baseline description of the industry's use of OSS.

The student should conduct a study of how and why European software intensive industry uses OSS and development practices from the open source community.

Assignment given: 29. January 2007
Supervisor: Reidar Conradi, IDI

Abstract

The use of Open Source Software (OSS) has increased in both the industry and the public sector. The software intensive industry integrates OSS into their products, participates in the development of OSS products, and develops its own OSS products. The understanding of how and why the industry is approaching OSS is so far limited.

To help fill this gap, this thesis intends to explore how and why the software intensive industry approaches OSS. This is done by performing an extensive literature study and by executing a web-based survey. This survey is distributed to a near representative sample of companies from the Norwegian software intensive industry and to a convenience sample of participants in the ITEA 2 research program.

The research presented here shows that OSS components are widely used in the software intensive industry. Close to 50% of the Norwegian software intensive industry uses OSS in its development. The industry is mainly motivated to use OSS by practical reasons. OSS components provide functionality of high quality and the industry is satisfied with its use of these components. When using OSS, the industry benefits from the availability of source code, and easy access to components and information about these components.

Companies participate in OSS projects because they use the software and because of the learning effect of this participation. The participation is however limited. However, some companies provide commercial services related to the OSS projects they participate in.

Releasing a product as OSS attracts more users and customers to a product. These community members may contribute with implemented code, feedback, and requirements. There are, however some side-effects related to releasing an OSS product and companies should be aware of these consequences.

The main contributions of this thesis are new understanding of how and why companies approach OSS, a reusable research design, and experiences performing survey research.

Preface

The use of Open Source Software (OSS) in software intensive industry has increased the last few years. Major industrial actors as IBM, Sun Microsystems, Novell, and several others are participating in the OSS revolution.

The ITEA COSI is a European industry-driven research project. COSI tries to help the industry to benefit from OSS. To create a baseline description of the use of OSS in the COSI project we performed a survey within the project [Hauge and Røsdal 2006]. The results from this survey were presented to the ITEA reviewers who encouraged us to distribute a new survey to all ITEA members.

This Master thesis extends the work of [Hauge and Røsdal 2006] and concludes my MSc in Computer Science at the Department of Computer and Information Science (IDI) at the Norwegian University of Technology and Science (NTNU). It consists of an extensive literature study and a survey of industrial OSS involvement in the ITEA project and in the Norwegian software-intensive industry. The current survey has been improved based on feedback and results from the first survey and new findings in literature.

I would like to thank ITEA, COSI, and ICT Norway for assisting the realization of this survey, my supervisors Professor Reidar Conradi and Dr. Carl-Fredrik Sørensen for their advice and support, Professor Ola Listhaug for his feedback on the research design, Professor Tor Stålhane for his advice on statistical analysis, the students and professionals who helped with the pre-test, and Andreas Røsdal for our collaboration on the previous survey. Last but not least, I would like to thank the respondents who participated in the survey.

Trondheim September 29, 2007

Øyvind Hauge

Contents

I	Introduction	1
1	Introduction	3
1.1	Rationale and Background	3
1.2	Problem Statement	4
1.3	Contributions	5
1.4	Content	6
II	Prestudy	9
2	What Is Open Source?	11
2.1	The History of Open Source	11
2.1.1	Hacker Culture	11
2.1.2	The Internet	12
2.1.3	UNIX	12
2.1.4	GNU and the Free Software Foundation	13
2.1.5	Linux	14
2.1.6	Apache	14
2.1.7	Mozilla and Open Source	15
2.1.8	The Eclipse Foundation	15
2.1.9	Recent Industrial OSS Involvements	15
2.2	Open Source Software	16
2.3	Open Source and Intellectual Property Rights	16
2.3.1	Intellectual Property	16
2.3.2	Open Source Licensing	17
2.4	Open Source Community	20
2.5	Open Source Development Practices	20
2.6	Summary	20
3	Motivations for Approaching OSS	23
3.1	Motivations for Individual OSS Involvement	23

3.2	Motivations for Industrial OSS Involvement	24
3.2.1	Providing Supplementary Products and Services	25
3.2.2	Providing Commercial OSS	25
3.2.3	Using OSS	28
3.2.4	Supporting OSS Communities	29
3.2.5	Other Motivations	30
3.3	Summary of Motivations for Approaching OSS	30
4	OSS Development Practices	33
4.1	Keeping it Public	34
4.1.1	Improved Communication	34
4.1.2	Feedback from Community	36
4.1.3	Peer Review	36
4.1.4	Increase Trust	36
4.2	User Involvement and Domain Expertise	37
4.3	Implementation	37
4.3.1	Code Repository	37
4.3.2	Code Ownership	38
4.3.3	Build and Release Frequently	38
4.3.4	Modularity and Clear Interfaces	38
4.3.5	Informal Design	39
4.3.6	Freedom of Choice	39
4.4	Industrial Use of OSS Development Practices	39
4.5	Summary of OSS Development Practices	40
5	Industrial Providers of OSS	43
5.1	CommSy	43
5.2	Open Source Component Artifact Repository	44
5.3	Mozilla	44
5.4	Lessons Learned	46
5.5	Other Evidence	46
5.6	Summary	46
6	Industrial Selection of OSS Components	49
6.1	The Selection Process	49
6.2	Finding OSS Components	50
6.3	Evaluating OSS Components	50
6.3.1	The Product	50
6.3.2	Community and Position of OSS	51
6.3.3	Reputation and Further Plans	51
6.3.4	Skills and Experience	52
6.3.5	Summary of Evaluation Criteria	52
6.4	Use of OSS Components	52
6.5	Summary	53

7	Industrial Participation in OSS Projects	55
7.1	Industrial Contributions	55
7.2	Summary	56
III	Research	57
8	Research Design	59
8.1	Research Process	60
8.2	Literature Review	61
8.2.1	Finding Publications	61
8.2.2	Selection and Evaluation of Publications	63
8.2.3	Summary of the Literature Study	63
8.3	Research Questions	64
8.4	Questionnaire Design	66
8.5	Challenges with OSS Related Survey Work	67
8.5.1	Population: Not Defined	67
8.5.2	Population: Too Particular	69
8.5.3	Erroneous Population Description	69
8.5.4	"Unreachable" Study Objects	70
8.5.5	The Gatekeeper Problem	71
8.5.6	Low Response Rates	72
8.5.7	Summary of Challenges with OSS Related Survey Work	73
8.6	Population and Sample	73
8.6.1	ITEA 2 Participants	74
8.6.2	Software Intensive Companies in Norway	75
8.6.3	Response Rates	87
8.7	Summary of the Resararch Design	87
9	Data Collection Using A Questionnaire	89
9.1	The Basis	89
9.2	Development	90
9.2.1	Structure	91
9.2.2	Logic Flow	91
9.2.3	Mapping to Research Questions	91
9.3	Implementation	92
9.3.1	The Web-Questionnaire	92
9.3.2	Question Types	93
9.4	Quality Ensuring Measures	98
9.5	Increasing the Response Rate	99
9.6	Summary	100
10	Data Analysis	101
10.1	Re-Coding of Scales	101
10.2	Statistics	102

10.2.1	T-test	102
10.2.2	ANOVA	103
10.3	Tools and Data Formats	103
IV	Results and Conclusions	105
11	Results	107
11.1	Response Rates: Phone vs Email Invitation	107
11.2	Use of OSS in the Norwegian Software Industry	109
11.3	Motivations for Approaching OSS Development	111
11.3.1	Integrating OSS Components	111
11.3.2	Interaction with and Participation in OSS Projects	112
11.3.3	Releasing OSS Licensed Products	113
11.4	OSS Integrators	113
11.4.1	The Share of OSS in Software Development	114
11.4.2	The Use of OSS Components	116
11.4.3	Selection of OSS Components	118
11.4.4	Participation in OSS Projects	123
11.5	Experiences Providing OSS Products	124
11.5.1	Community Contributions	124
11.5.2	Side-effects of Releasing an OSS Products	126
11.5.3	Product and Community Statistics	127
11.6	Demographic Data	129
11.6.1	The Companies	129
11.6.2	The Respondent	130
12	Discussions and Implications	133
12.1	Implications	133
12.1.1	Research Method	134
12.1.2	A Need for More Research on Industrial OSS	135
12.1.3	The Use of OSS Development Practices	135
12.1.4	The Amount of OSS in Software Products	136
12.1.5	High Quality OSS	137
12.1.6	Releasing OSS Products is No Free Lunch	137
12.2	Validity and Reliability	137
12.2.1	The Literature Study	137
12.2.2	Instrument Construction	138
12.2.3	Population and Sampling	139
12.2.4	Review and Reporting	140
12.2.5	Reliability	141
12.2.6	Summary of Validity and Reliability	141
13	Conclusions	143
13.1	Summary	143

13.2 Contributions	143
13.3 Limitations	144
13.4 Further Research	145
Index	146
Glossary	149
Bibliography	152
V Appendices	163
A The Invitation Letter and The Reminder	165
B The Questionnaire	171
C On-going ITEA 2 Projects	207
D Surveying Roles Industrial Roles in Open Source Software Development	211
E How to Perform a Web Survey	219

List of Tables

1.1	Research questions	5
1.2	Research questions and findings in literature	6
3.1	Individual motivations for open source participation	24
5.1	Lessons learned from companies providing OSS products	47
8.1	Research questions	59
8.2	Research process	60
8.3	Journal databases	62
8.4	Special Issues on OSS	63
8.5	Initial research questions	64
8.6	Mapping from initial to new research questions	65
8.7	Online open source studies	68
8.8	Description of the population and samples	74
8.9	Economic classification codes	77
8.10	The Norwegian ICT Sector in 2005	78
8.11	Distribution of entities in the 72.00 category, 2007	78
8.12	Distribution of active companies over size, 2007	79
8.13	Enterprise types	80
8.14	Randomly selected companies	83
8.15	Randomly selected companies	85
8.16	Invitations	86
9.1	Changes from the old questionnaire	90
9.2	Mapping of questions to research questions Part 1	93
9.3	Mapping of questions to research questions Part 2	94
10.1	Re-Coding of Likert scales	102
10.2	Calculation of the t-values	103
11.1	Number of responses from companies invited by email and phone	108

11.2	Use of OSS components in software development	110
11.3	Question 2.10 Motivations for using OSS components in SW development	112
11.4	Question 2.23 Motivations for interacting with or participating in OSS Projects	113
11.5	Question 2.23 Motivations for providing OSS products	114
11.6	Questions 2.17-2.19 Experiences using OSS components	119
11.7	Question 2.13 Search activities	120
11.8	Question 2.14 Evaluation activities	121
11.9	Question 2.15 Evaluation criteria - product	122
11.10	Question 2.16 Evaluation criteria - community	122
11.11	Questions 2.24 Contributions to OSS projects	123
11.12	Questions 2.25 Satisfaction with OSS project	124
11.13	Question 1.12 Experiences providing OSS products	125
11.14	Question 1.16 Community contributions	126
11.15	Question 1.19 Attracting a community	127
11.16	Question 1.17 Services provided to the community	128
11.17	The number of companies from the different industry sectors	130
12.1	Answers to research questions	133

List of Figures

2.1	How academic licenses influence the surrounding software	18
2.2	How the first category copyleft licenses influence the surrounding software	19
2.3	How the second category copyleft licenses influence the surrounding software	19
4.1	The bug tracker Bugzilla	35
8.1	Unreachable study objects	71
8.2	The sampling process	76
8.3	Selected companies and discovery rates	83
9.1	Logical flow of the questionnaire	92
9.2	A question with two options	95
9.3	Question with answers from a drop down list	95
9.4	A quesiton where one or more answers may be checked	96
9.5	A free text question	97
9.6	Grid with several questions and several answers	97
9.7	End of Part 1 of the questionnaire	99
11.1	Question 2.6 Development effort in person-months	115
11.2	Question 2.7 Functionality provided by OSS components	115
11.3	Question 2.8 Number of OSS components in products	116
11.4	Question 3.7 Turnover from OSS activities	117
11.5	Question 2.8 a/b Parts of code read/changed	117
11.6	Question 2.8c Number of fixed defects, added glue, returned changes	118
11.7	Question 2.12 Selection effort in hours	120
11.8	Question 1.18 Services provided the community	128
11.9	Question 1.13-14 Number of community contributors	129
11.10	Question 1.9 Development effort in person-months	129
11.11	Question 3.5 Number of employees	131
11.12	Question 3.11 Years of experience	131

11.13 Question 3.12 Involvement in integration of OSS and OSS commu-
nities 132

List of Acronyms

ANS	General partnership (Ansvarlig selskap)
AS	Limited Company (Akjeselskap)
ASA	Public Limited Company (Alment aksjeselskap)
CCRLE	The Central Coordinating Register for Legal Entities (Enhetsregisteret)
COSI	Co-development using inner & Open source in Software Intensive products
DA	Partnership with apportioned liability (Ansvarlig selskap, delt ansvar)
ENK	Sole proprietorship (Enkeltmansforetak)
FLOSS	Free/Libre Open Source Software
ISIC	International Standard Industrial Classification
ITEA	Information Technology for European Advancement
NACE	Nomenclature Generale des Activites Economiques dans L'Union Europee
NTNU	The Norwegian University of Science and Technology
NUF	Norwegian division of a foreign entity (Norskregistrerte utenlandske foretak)
OSS	Open Source Software
SE	Software Engineering
SSB	Statistics Norway (Statistisk Sentralbyrå)

Part I

Introduction

Chapter 1

Introduction

Open Source Software (OSS) is becoming relevant for everyone due to its rapid diffusion, the significant investments made by several companies, and the new ways of organizing software development [Lerner and Tirole 2005].

The industry is clearly starting to enter the OSS arena. [Ghosh 2006] shows widespread use of OSS in the industry and estimates that companies have invested 1.2 billion Euro in the development of OSS that has been made freely available. NASA used OSS in the systems controlling the Mars robots [Norris 2004]. Companies such as MySQL, Trolltech, Red Hat, and Sun Microsystems do business by giving away their software as OSS. IBM claims to have contributed more than \$ 1 billion to Linux development and participated in more than 100 OSS projects [IBM 2005]. The Eclipse IDE is one of these projects where IBM contributes alongside others like CA, Intel, Nokia, BEA, Borland, and Motorola [The Eclipse Foundation 2007].

Other actors are also getting their eyes up for OSS. Gartner mentioned open source as one of the top ten strategic technologies of 2007. The Norwegian government wants to stimulate industrial and public use of open source [AP SV og SP]. The number of publications on OSS within academia and in media, has increased significantly the last years.

1.1 Rationale and Background

When big industrial actors and governments start using OSS, and when we even go to March using OSS, it will affect everyone. However, we know little about the industrial participation in OSS [Bonaccorsi et al. 2004]. There are large gaps related to both why and how the industry approaches OSS. More research is therefore clearly needed to understand the impact of the industry's use of OSS.

Information Technology for European Advancement (ITEA) acknowledges the

success of several open source products but notices that the advantages and drawbacks of integrating OSS into software are not well understood [Lacotte 2004]. COSI, an industry driven research project, was established as part of the ITEA 2 programme. COSI has observed a shift in the software intensive industry; OSS and OSS development practices are used more often and participation in distributed co-operations is becoming every day practice in the industry.

The overall goal of the COSI project is to understand how industry can benefit and learn from open source software and open source development practices. To complete this goal, it was decided to create a baseline description of the OSS practices used in European software intensive industry.

One of NTNU's responsibilities as sub-contractor to the Norwegian COSI project is to contribute to this baseline description. As an input to this baseline, we performed a survey of the OSS practices used within the COSI project. We interviewed representatives from the three industrial participants in the Norwegian COSI project and used the results to construct a questionnaire. This questionnaire was distributed to a convenience sample consisting of the participants in the COSI project. The results of this survey are available through COSI deliverable D2.1.2 and [Hauge and Røsdal 2006]¹. The results from this survey were well accepted by the ITEA reviewers. Rudolf Haggemüller, the Chairman of the ITEA 2 Board, invited NTNU to redistribute the questionnaire to all the participants in the ITEA projects. This gave us an excellent opportunity to improve our work with considerable high-level support from ITEA. This invitation was the starting point of the survey performed in this thesis.

1.2 Problem Statement

The context of this thesis is given and the solution space is fairly constrained. The overall goal of this work is to contribute to the understanding of how the European software intensive industry approaches OSS. This insight will be gained by performing a web-based survey within the projects under the ITEA umbrella. This following is the original problem description (translated from Norwegian):

The industrially controlled research project ITEA COSI has observed a shift in the software intensive industry. Open Source Software (OSS) and open source development practices are used more often. As a participant in the ITEA COSI project, NTNU contributes in the development of a baseline description of the industry's use of OSS.

The student should conduct a study of how and why European software intensive industry uses OSS and development practices from the open source community.

¹<http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2006/hauge-fordyp06.pdf>

This project description will be addressed by performing a review of relevant literature and by extending and improving the survey performed by [Hauge and Røsdal 2006]. Based on the literature survey and feedback on the first survey we have defined the following research questions, Table 1.1, which will be addresses by performing a survey within the ITEA programme and the Norwegian ICT sector.

RQ1.	Why do companies integrate OSS components into their software, distribute their software as OSS, and interact with and participate in OSS projects?
RQ2.	Which OSS development practices are used when industry develops software?
RQ3.	How does an industrial OSS provider attract and sustain a community of users and developers?
RQ4.	To what extent does the industry integrate OSS components into their software and how does it find, select, and evaluate OSS components?
RQ5.	To what extent is the industry participating in OSS projects?

Table 1.1: Research questions

The previous survey had some drawbacks; we had low control over the respondent in each of the companies, with only 24 responses the statistical strength of the results was quite low, completing the questionnaire was too time-consuming, and it was too difficult to answer questions related to processes. These shortcomings will be addressed and the new survey will help us to fill some of the gaps in the OSS research. The survey focuses on why companies approach OSS, and their experiences related to selection of OSS components and industry-provided OSS products. Samples from two populations were investigate in the survey. However, the number of responses from the ITEA sample was low and the findings are primarily based on results from the Norwegian ICT sector.

1.3 Contributions

This thesis has two main contributions *knowledge related to how and why companies approach OSS development*, and *a re-usable research design and experiences from the execution of two large industrial surveys*.

Three sources provide this knowledge. First, a paper accepted at The Third International Conference on Open Source Systems summarizes the most important findings from the previous study, see Appendix D. Second, the pre-study presents a theoretical basis from the OSS research community, see Part II. Third and most importantly, this thesis presents the results from two large industrial surveys, see Part IV.

Experiences from the accomplishment of the surveys are first of all provided through a re-usable and well documented research design. The research design is described in Part III and the survey is available as Appendix B. In addition to designing a survey we discuss several challenges related to survey research. Possible solutions are provided to some of these challenges. Furthermore, we give advice to researchers who are performing their first survey(s), see Appendix E. This paper is targeted at students and researchers primarily at NTNU and it is therefore written in Norwegian.

In addition to this we have also developed a list of Norwegian IT companies with about 1000 companies, more than 200 companies using OSS in their software development, and more than 500 contact persons. This list can be used as a basis population for future studies.

1.4 Content

This thesis consists of five parts. **Part I** consists of this introduction.

Part II makes up the pre-study and presents findings from the literature study. Chapter 2 introduces OSS and puts it into context. This is done by discussing what OSS is, providing a short overview of its history, and discussing some legal aspects of OSS licensing. The remaining chapters of Part II provide a theoretical basis for each of the research questions, see Table 1.2.

RQ	Chapter	Content
1	3	Motivations for approaching OSS
2	4	OSS development practices and the use of these practices in industry
3	5	Commercial OSS providers
4	6	Industrial selection of OSS components
5	7	Industrial OSS participation

Table 1.2: Research questions and findings in literature

Part III contains an extensive documentation of the research design. Chapter 8 is started by a presentation of the research process, the procedures for the literature study, and the development of the research questions. Next, we discuss the questionnaire design and challenges related to performing survey research in software engineering. The last part of Chapter 8 describes the populations in this study and the sampling procedures.

Based on the literature study and the research questions, we developed a web-based questionnaire. The development of this questionnaire is presented in Chapter 9. The third part is concluded by Chapter 10. This chapter gives an overview

of the tools and procedures used to analyse the data from the survey and the screening process.

Part IV presents and discusses the results from the questionnaire. The results are presented in Chapter 11 before the validity and significance of these results are discussed in Chapter 12. Chapter 13 concludes this thesis by summarizing the contributions and presenting possible future work.

Part V Contains all the appendices for the thesis; an overview of the ITEA projects, the questionnaire, the invitation letters, the paper accepted to The Third International Conference on Open Source Systems, and an introductory paper about performing surveys using a questionnaire. This paper is based on our experience performing surveys. Its target group is primarily students here at the university and it is therefore written in Norwegian.

Part II

Prestudy

Chapter 2

What Is Open Source?

The term 'Open Source' may be understood in different ways. Several authors have tried to define open source, for instance [Gacek and Arief 2004]. Brown and Booch state "...describing a laundry list of different definitions of open-source and the posting a new one is not particularly fruitful." [Brown and Booch 2002; 125]

This section will discuss some important issues related to open source rather than trying to find an exact definition of it. First and most importantly, it is software (OSS). Secondly, it can be defined by the communities which make this software. Thirdly, open source consists of the set of development practices used to make this software. Fourthly, open source includes the people and organizations behind these communities and this software. We will in this section take a brief look at important aspects of open source. An historical overview of important people, events, and organizations in the Open Source history will open this chapter before we discuss some properties of open source software, including OSS licenses. Open source practices will be discussed in further detail in Chapter 4.

2.1 The History of Open Source

This section will provide a brief overview of the history of open source and some of its most important moments and products.

2.1.1 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 all kinds of research results as well as software [von Krogh and von Hippel 2003]. It was hard to get a program to work effectively and sharing was therefore normal. 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 of the hacker¹ culture. This openness makes the foundation of the open source community we know today.

2.1.2 The Internet

One of the motivations behind the Internet was to be able to share knowledge and programs between the participants [Tuomi 2003]. In 1968, Pentagon's Defence Advanced Research Project Agency (DARPA) started the Advanced Research Project Agency Network (ARPANET). One year later, 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 and software platforms. This incompatibility, lead to the development of the platform independent TCP/IP. In 1983, TCP/IP replaced the old communication protocols in ARPANET.

Internet provided scientist and engineers a communication platform where they could cooperate over distances. Internet also allowed them to share both research results and software, and it has provided researchers and open source enthusiasts the necessary infrastructure for collaboration and sharing. This infrastructure gives them a place where they can work and share their results. The open source movement would probably not have existed without the Internet.

2.1.3 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 wanted to play his favourite computer game 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

¹A hacker is not a person who breaks security measures, but "someone who loves to program and enjoys being clever about it." [Stallman 1999; 53]

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 TCP/IP took place. A UNIX implementation of TCP/IP was developed at Berkeley and became part of the BSD distribution. As a result of AT&T's commercial actions, Berkeley, re-released the TCP/IP protocol stack in a release called Networking Release 1 [Weber 2004]. This was a true open source product with a liberal license, later called a 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.

2.1.4 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 GNU² project. He later wrote the GNU Manifesto explaining the goals of the GNU project and in 1985 he formed the Free Software Foundation³. The Free Software Foundation is an idealistic organisation supporting the free software movement, especially the GNU project. Stallman wanted to use existing copyright law to guarantee some

²GNU is a recursive acronym and it stands for "GNU's Not Unix"

³<http://www.fsf.org>

basic rights to all future users of software. Free, as in freedom not as in free beer, 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.

2.1.5 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 perhaps sociological. Linux was developed with a new and unique development model. Huge numbers of volunteers coordinated their development through Internet [Raymond 2001].

2.1.6 Apache

The Apache HTTP Server is another of the most important developments in the open source history. 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 web 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. They 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.

The Apache Group later became the Apache Software Foundation (ASF)⁴. The ASF is a non-profit corporation which oversee several important open source projects like the HTTP Server, ANT, Tomcat, Spring, Struts, and several others.

⁴<http://www.apache.org/>

2.1.7 Mozilla and Open Source

The Open Source Initiative came as a reaction to the (mis)understanding that the free software had to be gratis. Free software was not an industrial success and the term open source was coined by Bruce Perens related to Netscapes's kick-off of the Mozilla project in 1998. 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 2001], Netscape released the code open to everyone and named the project Mozilla⁵. Within hours after the release, people around the world were submitting patches [Feller and Fitzgerald 2002].

The definition of open source was later formed by Bruce Perens [Perens 1998], currently available in version 1.9 at <http://opensource.org/>. Perens later formed the Open Source Foundation. The Open Source Foundation acknowledges mostly the same licenses as the Free Software Foundation. The two organizations differ primarily in philosophical matters. The Open Source Foundation emphasizes more on the practical benefits of open source licenses instead of the moral issues [von Krogh and von Hippel 2003]. Since they differ on moral issues, a piece of free software will always be open source but open source will not always be free software [Scacchi et al. 2006].

2.1.8 The Eclipse Foundation

The Eclipse Foundation⁶ is one of the first big industrial initiatives in the OSS arena. It was formed in 2001 by Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, and Webgain. It was three years later reorganized as a not-for-profit corporation and it has today more than 150 members from the industry and academia. The Eclipse Foundation controls the development of the Eclipse platform and hosts other OSS projects and more than 50 subprojects.

2.1.9 Recent Industrial OSS Involvements

Eclipse is one of the most famous industrial OSS initiatives but there are many others. Many of these industrial initiatives are covered in media. JBOSS was recently purchased by Red Hat, Novell sells support to Linux, Sun Microsystems released Java as OSS, and many companies participate in the development of several other OSS projects. OSS has become an important part of the software development environment of today.

⁵<http://www.mozilla.org/>

⁶<http://www.eclipse.org>

2.2 Open Source Software

We have already seen that OSS has been given several names due to the (small) differences between each part of the Open Source Software/Free Software movement. Open Source (OS), Free Open Source Software (FOSS), or Free/Libre⁷ Open Source Software (FLOSS) are other names that are found.

The term *Open Source Software* is most correctly used about software licensed with a license approved by the *Open Source Initiative*⁸ according to *The Open Source Definition*. The Open Source Definition contains a list of 10 requirements to open software and the rationale behind these requirements. *Free Software* on the other hand is software approved according to *The Free Software Definition*. The Free Software Definition provides four freedoms to free software. These are the freedom to run, study, redistribute, and to improve the software. Most open source licenses are also acknowledged by the Free Software Foundation and vice versa, making the differences more ideological than practical.

This thesis will however continue to use Open Source Software or OSS to cover OSS, FS, FOSS, and FLOSS. The idealistic differences between Free Software and Open Source are not important here and this simplification will ease both the reading and the writing of this thesis. The different terms will be used only if it is important to differentiate them.

2.3 Open Source and Intellectual Property Rights

Both the Free Software Foundation and The Open Source Initiative maintain extensive lists of software licenses compatible with the Free Software Definition⁹ and the Open Source Definition¹⁰. Going into detail on all these licenses is out of scope of this thesis. Nevertheless, it is important to discuss some properties of open source licensing. This will help the reader to better understand the challenges the industry meets when working with OSS, especially relate to derived works.

2.3.1 Intellectual Property

To understand intellectual property (IP), it is important to understand the two terms 'idea' and 'expression'. An idea could for instance be the concept of a circle. An expression, on the other hand, could be a drawing of that idea (the circle). An expression is often called an intellectual or creative work.

⁷Libre is Spanish for free, as in freedom

⁸<http://opensource.org>

⁹http://www.fsf.org/licensing/licenses/index_html

¹⁰<http://opensource.org/licenses/>

The rights to expressions are covered by intellectual property law, contract law, and licensing. The right to ideas is covered by patenting. Software patenting exists at least in the U.S. but it will not be discussed here.

Ownership of a an Expression

It is easy to understand who owns a car or a book. Ownership to something intangible like source code is perhaps not that easy to understand.

Intellectual property law generally says that the creator of an expression is the owner of that expression [Olson 2006]. The owner controls all the rights to the expression and he can decide what to do with it.

We can conclude that the original author of some piece of code is the owner of that code. When you own a piece of software you can decide how to use it and how others are allowed to use it. You can in other words grant people certain rights to you software by licensing it to other people.

Derived Work

A derived work is a work which includes or extends another work. If you take my drawing of the circle and draw a square inside the circle you have extended my work. You have made a derived work. In the world of software development, a derived work is a piece of software which includes or extends another piece of software.

If we copy some code from somewhere else, we are clearly not the owner of that code because we were not the original author. We do not possess the intellectual property rights (IPR) to this code and we cannot grant any rights to a piece of code someone else owns. It is important that the original author has granted us the right to use his code inside our software.

2.3.2 Open Source Licensing

Licenses in software development are normally used to protect creative work by making it illegal to (mis) use, copy, or change a piece of software. In open source, the purpose of licensing is to deny anybody the right to exclusively exploit a work [Laurent 2004; 4]. Licensing is in other words about preserving the user's freedom to do what he or she wants with the software and about granting the user rights.

Open source licenses can be divided into two broad categories, reciprocal and academic licenses [Olson 2006]. These two license types differ on how they grant rights to redistribution of derived works. Reciprocal or copyleft licenses require some contribution back to the community. Academic licenses usually just require some acknowledgement of the original author. These two categories of licenses will be explained next.

Academic Licenses

[Webbink 2003], calls academic licenses for non-protective open source licenses. They are non protective because the author retains his copyright, but they grant all other rights to the licensee (user). The user can in other words use the software in any way he wants.

This can be illustrated with an example, as shown in Figure 2.1. Imagine a piece of software with an **OSS component** with an academic license at the bottom. The OSS component has been **changed** a bit and it is part of a larger **software** product. The original license of the OSS component does not restrict the developer when he chooses a license for the changes he made and the rest of his software.

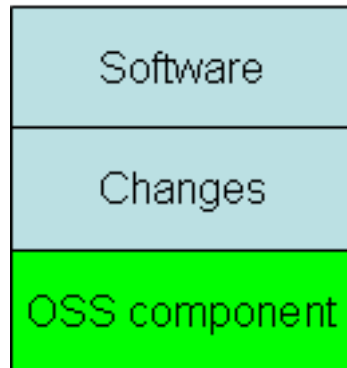


Figure 2.1: How academic licenses influence the surrounding software

The Berkeley Software Distribution (BSD) license is the most used academic license. It contains only three requirements to future modifications of the code. The two first say that the copyright notice must be kept in the source code and binaries. The third says that neither of the names of the authors can be used to promote derived products without permission.

Copyleft

Copyleft is a way of ensuring the same freedom to all users of all future versions of a piece of software. 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].

If a piece of software is licensed with a copyleft license it forces all derived works to use the same copyleft license as well. The freedom to run, study, redistribute, and modify are thereby preserved for all future users. A piece of software licensed with a copyleft license is protected from becoming someone's private intellectual property [Mustonen 2003].

There are basically two categories of copyleft licenses. The first category requires only the **changes** made to the original **OSS component** to be distributed with the same license as the component. The rest of the **software** can be distributed with any license, see Figure 2.2.

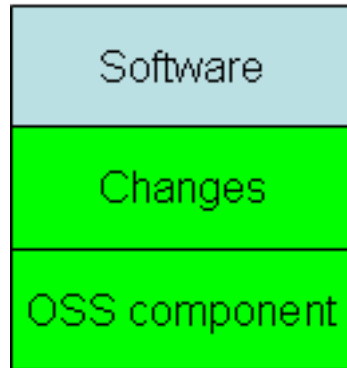


Figure 2.2: How the first category copyleft licenses influence the surrounding software

The second category requires a software developer which makes a derived work to distribute both the **changes** and the rest of the **software** with the same license as the original **OSS component**, see Figure 2.3.

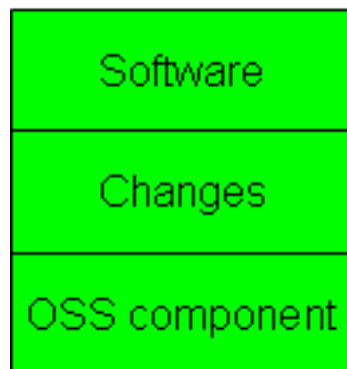


Figure 2.3: How the second category copyleft licenses influence the surrounding software

The copyleft principle originates from the GNU project by Richard Stallman and the GNU General Public License (GPL). The GPL license is still one of the most used open source licenses together with the less restrictive GNU Lesser GPL. Contrary to the GPL, the LGPL allows software licensed under this license to be used as libraries in proprietary products, see Figure 2.2.

Critics of copyleft describe this behaviour as viral or contaminating because it does not leave a developer with the choice of keeping his modifications proprietary.

2.4 Open Source Community

Open source licenses are merely a tool used by the people developing open source to ensure certain freedoms to future users of the software. These people who develop and use OSS make up the OSS communities around the many different OSS projects.

In the world of OSS, we find products with several thousands or maybe millions of users like the Apache web server, Linux, BIND, Sendmail, and the FreeBSD TCP/IP implementation. We also find many small projects with few or no users [Madey et al. 2004, Krishnamurthy 2002]. Because of this diversity it is hard to say something general for all communities participating in OSS projects.

Some researchers have however tried to describe the OSS communities and the participants in these communities. This is neither necessary nor particularly fruitful for this thesis and it is most likely impossible to give a general description of all the hundred of thousands of OSS communities out there.

2.5 Open Source Development Practices

The development practices used to develop the many successful OSS products we know are many and diverse. Some are new and unique while others are borrowed from traditional software engineering. It is important to remember that there is a great variety of OSS projects and that there is no set of practices which is used in all OSS projects. We will come back to this discussion in Chapter 4 and we will look at some of the practices which have made the distributed development of OSS possible.

2.6 Summary

There is no single, simple answer to what open source is. Open Source is the software but the software cannot be removed from the ecosystem surrounding it and the history accompanying it. We have seen that the open source movement has its roots back to sixties when scientists and engineers shared results and software. Open source has developed into a complex phenomenon which includes different people, organizations, license types, development practices, and so on. We will not try to define it clearer but we will discuss some aspects of OSS in the next chapters.

Chapter 4 will discuss some of the development practices reported in research literature. These practices are highly relevant to the survey. Furthermore, it is also important to understand the motivations behind both individual and industrial participation in OSS. To shed light over this issue, Chapter 3 will present some

of the research done in this field. The remaining chapters of Part II will look into industrial approaches to OSS development.

Motivations for Approaching OSS

This chapter will present the most important motivations companies are driven by when they approach OSS. It will start by giving a brief overview of personal motivations for participating in OSS projects. The remaining parts of this chapter will discuss several motivations behind industrial involvement in OSS related software development.

3.1 Motivations for Individual OSS Involvement

Most open source communities have so far consisted of individuals. Developers in OSS 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 individuals while the benefit is given to the public. What makes people spend their time and resources working for others?

There are several explanations to this phenomenon. Participants get private benefits from their contribution. [von Hippel and von Krogh 2003] argue that the benefit has to exceed the effort a developer puts into the work done. [Lakhani and von Hippel 2002] show that community members of the Apache community value the learning benefits of providing online support to other users. These learning benefits out-weight the work of helping. Other benefits from working in open source could for instance be money, status, enjoyment, or affiliation with the community.

Studies as [Lakhani and Wolf 2005, Ghosh 2002, Hars and Ou 2002, Hertel et al. 2003, Ye and Kishida 2003] and several others give us valuable indications of the motives driving individuals who participate in open source projects.

The motivational factors mentioned 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 (learning), or to get respect and status. *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.

A summary of the most important motivational factors from [Lakhani and Wolf 2005, Ghosh 2002, Hars and Ou 2002, Hertel et al. 2003, Ye and Kishida 2003], is presented in Table 3.1.

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

Table 3.1: Individual motivations for open source participation

These studies also report that about 20 % to 40 % of the responding developers got 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.

The industry is as we see already participating in OSS through paying individuals but is the industry also participating as companies? We look into this in the next section.

3.2 Motivations for Industrial OSS Involvement

To understand why the industry approaches OSS, we will look at the different roles a company may have and the possible motivations for undertaking these roles. A company may sell products and services related to an OSS product, provide its own OSS product, use OSS, and contribute to OSS communities. These roles are closely related to the business model of the company. The roles are also related and a company will most likely have more than one role at the time.

3.2.1 Providing Supplementary Products and Services

A company can provide services and products in addition to the ones provided by the community [Lerner and Tirole 2005]. These products and services are in most cases provided to make profit. A company can of course sell products and services related to their own OSS products as well. These supplementary products and services can take several forms and we will take a quick look at some of them here.

One example of such a service is *software distribution* [Krishnamurthy 2005]. Linux distributors like Red Hat interact with the OSS community, collect software, package the software as a product, and sell the package [Karels 2003]. In addition to selling the package they may also *guarantee the quality* of the software, provide *upgrade services*, and *installation or deployment services*.

To be able to create a package with high quality and to guarantee the quality of such a distribution, the integrator needs to know the software and the community providing the software. Guaranteeing the quality can only be done if they are able to influence the OSS community to either correct problems or to include corrections provided by others.

By having knowledge about OSS products they may also sell *user support and user training*. The Norwegian company Linpro is one example of such a company. The commercial support provided by Linpro and other companies offers an alternative to the support provided by the community itself. Having commercial support can lower the risk for other companies adopting OSS.

Yet, another example is Sun Microsystems. Several OSS projects can be run on Sun *hardware*. By contributing to the development of this software, Sun increases the value of their own hardware and thereby their sales. [Bonaccorsi et al. 2004] also mention selling accessories like gadgets, books etc, as another possible business approach to OSS.

3.2.2 Providing Commercial OSS

Companies like MySQL, Sun Microsystems, IBM, JBoss¹ have all given away (some of) their software as OSS. There are several possible motivations behind this decision.

Development Support

One reason for releasing software as OSS is to get development support from the community [Rossi and Bonaccorsi 2005, Henkel 2006, West and O'Mahony 2005]. Adopters of the software may contribute with bug reports, bug fixes and so on. These contributions could *increase the quality* of the product. Working with the

¹A division of Red Hat

community can also *reduce the effort* a company has to spend detecting bugs and writing patches [Lin 2006b].

Attracting Customers

Still, the professional open source business model is not really about development savings. Rather, it is about maximizing distribution of one's product: getting it beyond the purchasing firewall/bureaucracy bottleneck to plant the product in the hands of its developer end users so that they can try and then revisit the professional open source vendor for support/service contracts. [Asay 2006; 116]

Giving away the source code "is similar to the strategy of giving away the razor (the released code) to sell more razor blades (the related consulting services that [company name deleted] will provide)." [Lerner and Tirole 2005; 68]. We have seen that a company may base their business on selling OSS related products and services, Section 3.2.1. These companies might as well *sell products and services* related to their own OSS. By releasing the product as OSS, you may attract a large pool of adopters [West 2007, West and O'Mahony 2005]. Some of these adopters could be potential customers.

Developing the OSS themselves gives a company a clear competitive edge over other companies. They know the software very well and they control the future of it.

Increasing Product Whole Value

Releasing a product as OSS enables other software developers to provide complementary products and services. These products and services contribute to the 'total' or 'whole' value of the product [West 2007]. The existence of such an ecosystem will therefore *increase the value of the product*. This can be illustrated by the Linux platform. The fact that more and more vendors are providing software and hardware drivers to the platform has increased the value of Linux.

Dual Licensing

If the company providing an OSS product controls the intellectual property (IP) rights to the code base, it may sell licenses to the product. Anyone controlling the IP rights to a work may license it with as many licenses as he likes. Releasing a software product with an OSS license, typically a copyleft license, and a proprietary license is called *dual licensing*. The dual licensing model allows a company to gain profit from selling licenses to a product which is also released under an OSS license.

If a company includes software licensed with a copyleft license like GPL into their software, they have to distribute their software as OSS too. If they want to create

a commercial software and sell licenses to it rather than releasing it as OSS, they may buy a proprietary license from the OSS provider. Other customers could buy proprietary licenses if they have not legally cleared OSS licenses in their organization or if they want guarantees and user support not covered by an OSS license.

There are some challenges related to IP-rights. The company must have all rights to the software's code base to be able to release it with two licenses. Contributions from the community cannot be included into the product unless the author of the contribution shares his IP-rights with the company. Michael Olson² claims that because of this restriction, open source is primarily used as a distribution strategy, not as a development model [Olson 2006].

MySQL, eZ Systems, Trolltech are some examples of companies which have undertaken the dual licensing approach.

Including Other OSS Products

Copyleft OSS licenses require, as we have seen in Chapter 2.3, adaptors to release their derivative software as OSS as well. If a company decide to include a copyleft OSS component they have to release their product as OSS as well [Henkel 2006]. If a company decides to release their software under the GPL license, they are able to include all other software released with GPL as well. Re-use of other components can lead to significant development savings.

Image Building

Companies can also release their software as OSS *to appear as good OSS players* or to *show their technical excellence* [Henkel 2006]. Releasing software to the OSS community could be seen something good and thus improve the reputation of the company releasing it. The OSS provider can also do it to increase the trust to their products. When the source code is available, everyone can see how good (or bad) the product and the provider of the product are.

Not for Profit

[West and O'Mahony 2005] also mention the wish to create software as a public good as one motivation for releasing software as OSS. This is however more likely to be the case for non-profit or governmental sponsors than for companies.

Some commercial companies may however release software as OSS if they see no possibility for making money having it as proprietary software. Many tools and

²The former president of Sleepycat Software which has developed the Berkley DB. Sleepycat was purchased by Oracle in 2006

development tools have been developed by companies and released as OSS. If a tool is not part of a company's business they might as well release it as OSS.

3.2.3 Using OSS

Many companies use OSS as part of their business. They may use OSS components as part of their software or they may use OSS tools or infrastructure in their software development. Yet another approach is to use OSS platforms to extend their market. Companies can for instance provide Linux versions of their software to reach new customers [Bitzer 2004]. Only the motivations for companies which use OSS components in their software development will be discussed here. The use of OSS office tools, development tools, infrastructure etc, is out of scope of this thesis.

The Reasons Why the Industry Use Components

The reasons why components are reused are well established. [Li et al. 2005] mention lower cost, shorter time-to-market, higher quality, adherence to standards, and so on. The motivations behind why companies choose OSS over COTS components are not explored that well.

Access to Innovation at Reduced Costs

Reduced costs due to the lack of license fees is often mentioned as one reason [West 2007, Dedrick and West 2004]. OSS allows small companies to afford (faster) *access to cutting edge innovation* through "free" OSS components [Rossi and Bonaccorsi 2005, Goldman and Gabriel 2005]. Many of the OSS components available contain contributions from several large contributors. These contributions give smaller companies access to innovation they never could have developed or acquired on their own.

The quality and reliability of many OSS products is perceived to be high [Ruffin and Ebert 2004]. Many adopters are motivated by this [Rossi and Bonaccorsi 2005, Ajila and Wu 2007] and they gain *access to quality software* for free by using OSS.

Independence and Reduced Overhead

Independence from large software companies or reduced lock-in is often mentioned as a reason for adoption of OSS components [West 2007, Rossi and Bonaccorsi 2005]. [Serrano et al. 2004] mention reduced administrative delays as another reason why OSS is used instead of COTS. When a company is harvesting COTS, they have to purchase them. This purchase may need approval and include administrative overhead.

Availability and Openness

The availability of the components, its source code, and information about them is clearly another important reason why OSS is used [Hauge et al. 2007]. Together with the honesty about the true status of the software, availability was found to be the most important motivations for using OSS [Hauge and Røsdal 2006]. OSS communities are not afraid to tell about the shortcomings and the problems with their software. Most communities have web pages with bug-trackers, downloadable binaries and source code, information, and documentation. It is easy for an adopter to download and test the software without any limitations.

The availability of skilled personnel is also mentioned as another motivation for using OSS components [Glynn et al. 2005]. Glynn et al. also suggest that the awareness that other companies are adopting it, and the network externality effects that could be achieved through collaboration with community as reasons why companies adopt OSS.

The Reasons Why the Industry Does not Use OSS Components

Not all companies are using OSS and there may be several reasons why they do not. The lack of a vendor, the perceived lack of support, lack of liability, and the learning costs might frighten them [Glynn et al. 2005, Goode 2004, Giacomo 2005]. Furthermore, they may fear legal implications or security related aspects [Giacomo 2005], the vast number of different versions of the software [Krishnamurthy 2005], or poor usability [Nichols and Twidale 2003].

The cost of using OSS could also stop companies from adopting it. It is only the license which is free. The total cost of ownership is clearly not equal to zero [Giacomo 2005]. Many companies have OSS incompatible software, infrastructure [Glynn et al. 2005, Goode 2004], or skill set [Dedrick and West 2004]. The cost of replacing software and infrastructure and training employees may exceed the saved license fees.

3.2.4 Supporting OSS Communities

Several companies are also supporting different OSS communities. They support communities through paid development efforts, marketing, distribution, code contributions, etc.

The most important reason why companies do this is most likely because *they need the software* themselves. Developers benefit from fixing bugs and customizing the software because they use the software themselves privately or in their company [Lerner and Tirole 2005]. [Bitzer 2004] argues that some commercial software vendors support the development of Linux of two reasons. *It is cheaper* than the continued development of their own operating systems and because they *can*

adapt Linux to their needs. The companies thereby benefit from the development support in the community.

One of biggest examples of a company supporting a community is IBM and the Eclipse platform. IBM donated the source code of its development platform to a non-profit foundation. IBM continued to use it and to participate in the development of the tool. They have however managed to create a community around the platform and they benefit from the contributions from other community members. IBM could not or did not benefit from keeping the Eclipse tool internally. By supporting an OSS community which maintains the product they save costs and possibly sell related products and services.

Another well know example is the Mozilla project and Netscape. Netscape was loosing the browser war against MS Internet Explorer. Netscape created the Mozilla Organization and released the code base of its browser in 1998 [Baker 2006].

A company will also benefit from participation in a community through *increased skills* and the *influence* they might have on the OSS product.

3.2.5 Other Motivations

Other motivations may also matter when companies approach OSS related software development. This section will give a brief overview of such motivations.

[Rossi and Bonaccorsi 2005] reported that intrinsic community based incentives, as *agreement with the values* of the OSS community, was said to be important for companies. However, they were not put into practice.

The *skill set* of the employees is another reason why some companies decide to adopt OSS [Lin 2006a]. If the company have developers with extensive knowledge about OSS, it is wise to benefit from these skills. [Lerner and Tirole 2005] mention the "alumni effect" as one reason why people have good OSS skills. The software is freely available and it can easily be used in education or by students as a free alternative to proprietary software.

[Lerner and Tirole 2005] also argues that visibility in the OSS community allows a company to attract talented individuals.

3.3 Summary of Motivations for Approaching OSS

This chapter has provided an overview of the many possible motivations the software industry may have for approaching OSS. However, we have seen little empirical research of what the most important motivations are. Are they all equally important? The most important motivations are perhaps; reduced costs of developing software, cheap access to innovation, and access to new markets and customers. Furthermore, the business model of a company will most likely

influence its motivations for approaching OSS but we do not know. More research is needed to discover the most important motivations behind why companies approach OSS. If we understand why the software intensive industry approaches OSS we may understand how OSS could be approached most effectively.

Chapter 4

OSS Development Practices

Open source software development (OSSD) has been described to be something different than traditional software engineering (SE) [Feller and Fitzgerald 2002, Mockus et al. 2002, Raymond 2001]. Other writers have questioned this and said that the difference between OSSD and SE is exaggerated [Fuggetta 2003] and that OSS is wrongly described as a homogeneous phenomenon in literature [Østerlie 2007]. Other researchers have provided evidence that the difference between OSSD and SE is not that big [Paulson et al. 2004, Samoladas et al. 2004].

We have already seen that many of the OSS participants have a day-time job as a programmer [Hertel et al. 2003, Hars and Ou 2002, Lakhani and Wolf 2005]. With the entry of more and more companies into the OSS arena the differences between SE and OSSD are likely to be reduced. Developers will probably start mixing best practices from both camps and companies will most likely introduce practices and development processes from industry into the OSS development. [Fitzgerald 2006] predicts that 'OSS 2.0' will be more formal and have more similarities to more traditional software development.

The homogeneity mentioned by Østerlie could be caused by the fact that much of the descriptions in research literature have so far come from a few big and successful OSS project [Christley and Madey 2007]. This is projects like Mozilla and Apache [Mockus et al. 2002], Apache [Lakhani and von Hippel 2002], Linux [Hertel et al. 2003], and FreeBSD [Dinh-Trond and Bieman 2005]. However, not all OSS projects are big and successful. Studies have shown that most OSS projects are small with one or two developers [Krishnamurthy 2002, Madey et al. 2004]. The variety is great. [Michlmayr et al. 2005] were surprised by how much the development practices varied between different OSS projects.

To illustrate that some of the development practices described in the literature cannot be used in all OSS projects an example is given below. Several authors have reported that OSS projects have a hierarchy of participants with a pyramidal shape. This pyramid consists of core developer, developers, and active and passive

users [Mockus et al. 2002, Dinh-Trond and Bieman 2005, Crowston et al. 2004]. While this is true for some OSS projects, it cannot be true for all of them. Most OSS projects have as mentioned one or two developers and it is hard to imagine a pyramid with just one person.

To provide a basis for **RQ2**, we will discuss some development practices known from OSSD which could be applied in the industry. These practices are not used in all OSS communities but at least in some of them. We will also see that some of these practices are reported to be used in the software intensive industry as well.

4.1 Keeping it Public

OSSD has a reputation of keeping information public. However, some discussions and some information is kept private as well [Divitini et al. 2003, von Hippel and von Krogh 2003] but a clear trend is to keep it public. Most OSS projects make mailing lists, forums, trackers, web sites, and of course source code available to everyone.

4.1.1 Improved Communication

Keeping discussions and information public have positive effects. People (should) choose their words more carefully when they know that what they write is made public. Maintaining the information public allows people to catch up on discussions and refer to old documents. By referring to old documents they do not have to explaining the same things all over.

The OSS communities use several tools to simplify their communication and coordination. Public information in issue trackers, e-mail etc, can be used to coordinate the development of the software [Persson et al. 2005]. Keeping this communication public can also help a project to keep awareness [Gutwin et al. 2004].

Mailing Lists and Forums

Perhaps the most used communication tool in the OSS communities are the mailing lists and the web-forums. These tools may have many purposes. Users may ask for help at public forums or mailing lists [Singh et al. 2006]. The following discussions will first of all aid the help-seeker but if they are kept public they may also help other people. People who read the forums or mailing lists may learn or get help to solve similar problems. [Lakhani and von Hippel 2002] reported that this learning was one of the reasons why users helped others in online forums. The value of learning outweighed the effort of answering requests from other users.

The mailing list can also be used in both bug fixing, code review, and technical discussions related to for instance the inclusion of a new code modules [Ducheneaut 2005].

Bug Trackers

Bug or issue trackers are databases containing problem reports or feature requests for an OSS project. These reports normally contain an id, a description, status, assigned developers, history, time-stamp, comments etc, as shown in Figure 4.1. The databases containing these reports allow users to communicate their problems and requests to the developers. These databases play an important role in many OSS project [Anvik et al. 2006]. The developers will learn about the user's problems and wishes and they are thereby able to address their requests and problems. The developers may update the tracker with information and change the status of the reports.

Bug 8527 – New status "ASLEEP" would be cool

Status: NEW
Severity: enhancement
Keywords:
Whiteboard: [blocker will fix]
URL:

Product: Bugzilla
Component: Creating/Changing Bugs
Version: 2.10
Hardware: All
OS: All

Reported: 1999-06-18 17:41 PDT by [Hixie \(not reading bugmail\)](#)
Modified: 2007-02-12 12:27:13 PDT ([View Bug Activity](#))
Votes: 14 ([show](#))
CC: [bross2@stanford.edu](#), [bugzilla-mozilla@bkor.dhs.org](#), [bugzilla@shaw.ca](#), [jouni@heikniemi.net](#), [jrudeman@gmail.com](#)

Assigned To: [Nobody's working on this, feel free t...](#)
QA Contact: [default-qa@bugzilla.bugs](#)
Priority: P4
Target Milestone: ---

Depends on: [101179](#)
Blocks: [Show dependency tree](#) - [Show dependency graph](#)

Attachments
[Add an attachment](#) (proposed patch, testcase, etc.)

[Hixie \(not reading bugmail\)](#) 1999-06-18 17:41:35 PDT [Description](#)

Now that Bugzilla supports the really cool dependency stuff, it would be useful if a bug could be resolved as DEPENDS. This resolution would remain until such time as all bugs that the bug in question depended upon had resolutions of their own (whether FIXED, INVALID, DUPLICATE or whatever), at which time the bug would magically REOPEN itself.

Figure 4.1: The bug tracker Bugzilla

Resolving bugs is perhaps not as easy as it seems. [Østerlie and Wang 2006] describe the bug resolving as a process of ambiguity and negotiation. [Anvik et al. 2006] report that during a four month period of 2005, more than one third of the bug reports in the Eclipse project were invalid or duplicate reports. Furthermore, a lot of work is related to understanding and resolving these reports.

The bug trackers are normally public and they serve another important function;

they communicate the true state of the OSS product. Having a bug database without any bugs is most likely a sign of no activity rather than high quality. A database with many resolved bugs may indicate an active community and developers who do their job of resolving bugs. [Hauge et al. 2007] reported that knowing the true status of a component was an important input in the decision process when selecting components.

4.1.2 Feedback from Community

The community is only able to provide feedback if it has something to comment on. It is therefore important for an OSS provider to keep information and artefacts public. Feedback as bug reports, bug fixes, requirements requests, comments on design, and so on can be provided through mailing lists, forums or bug trackers. The developers behind an OSS can encourage the community to use these channels by actively using them themselves. "If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource." [Raymond 1998]

Community participation is one of the main strengths of OSS development. This participation would have been almost impossible if the information was hidden and if communication was kept private. Public information and communication invite people to participate.

4.1.3 Peer Review

Public source code allows everyone who is skilled enough in programming to inspect the code and possibly find defects in it. This can happen after the code is released or before the code is checked into the source repository. This is peer review. According to Raymonds and his Linus' Law "[g]iven enough eyeballs, all bugs are shallow." [Raymond 1998] Code inspection is a way of ensuring the quality of the code by giving a large number of developers access to the code and requesting them to review it. Peer review is used in several OSS communities like WINE [Lussier 2004] and Mozilla [Baker 2006]. Studies have showed that peer review reduces the time it takes to find and fix bugs [Mockus et al. 2002, Paulson et al. 2004].

4.1.4 Increase Trust

Public information gives better transparency in the development of a OSS product and about the product. This transparency could help to build trust to both the OSS product and the community providing the product.

4.2 User Involvement and Domain Expertise

[Scacchi 2002b] describes the requirements elicitation in OSS projects as an informal process. Requirements may be captured through a public e-mail or a discussion. "The next best thing to having good ideas is recognizing good ideas from your users." [Raymond 1998] These requirements are later made available through the web site to allow the community to review and refine them.

Many OSS projects have a similar proprietary software solution. OpenOffice has its counterpart in MS Office, Linux in UNIX and so on. The requirements are in these cases mostly clear because the functionality of the proprietary solution is known [Scacchi 2004]. When both the developers and the users have a-priori knowledge of what the system should be like, it is a lot easier to design and develop the software.

The developers have in many cases knowledge of the system they are developing through their own use of the system [Mockus et al. 2002, Dinh-Trond and Bie-man 2005]. They can therefore spend less time on eliciting requirements than in traditional software development [Scacchi 2002a]. In fact, [Potsar and Chang 2004] state that the requirements are already known by many open source developers before they start an OSS project. Many OSS projects are started because individual developers discover areas where software can improve a situation, and develop a solution themselves.

Another advantage of having developers which are expert users is that documentation does not need to be that detailed. You can expect the other developers to understand the system because they are also domain experts.

4.3 Implementation

The OSS communities have used several techniques to implement OSS products. We will now take a look at some of these implementation practices.

4.3.1 Code Repository

OSS is often implemented through a process called continuous integration [Holck and Jorgensen 2004]. Continuous integration is when developers integrate new functionality into the system at the same time. Easy access to the latest version of the source code at any time is therefore important. To provide the developer with such access, many OSS projects use code repositories like Concurrent Versions System (CVS) or Subversion (SVN). According to [Scacchi 2004] the version control system coordinates development and manages which additions that should be made. It gives control over configurations and it also prevents conflicting modifications to an extent. [Christley and Madey 2007] provide a list over many

of the activities in an OSS community. Many of these activities are related to code repositories. The repository tool is clearly a very central development tool¹.

The version control systems allow the developers to work in several branches of the development at the same time. They may have a stable and an experimental branch of the code tree.

4.3.2 Code Ownership

The code repositories allow individual developers or module owners to be responsible for parts of the code base. These developers may control the code which is checked into their part of the code base. To ensure the quality of the code, OSS projects may have reviewers which review the code and a committer which adds the code to the source tree. However, this also results in additional work, so it is not practiced in all OSS projects [Holck and Jorgensen 2004].

The developer responsible for a part of code may also have other responsibilities too. [Jørgensen 2001] describe how maintainers are responsible for fixing bugs and answering to questions for each their section of the code in the FreeBSD project.

4.3.3 Build and Release Frequently

Gathering the latest version of the code in one place has several advantages. Any developer may download the latest snapshot of the code and implement new features or bug fixes into the latest version of the software. He can then build and test this version of the software before he checks in his changes.

When the code is collected at one place, it is also easy to often release new versions of the software. Frequent releases are made in order for users to get access to new functionality as early as possible and to allow the user to report defects or improvements back to the developers as soon as possible [Hang et al. 2004].

4.3.4 Modularity and Clear Interfaces

A study by [Holck and Jorgensen 2004] shows that most work in the Mozilla and FreeBSD project is performed as one-man projects. To make this possible, it is important to use standards and modularity. Clear interfaces reduce the coupling between software elements. They improve the interoperability between different pieces of software and they make it possible to develop software in a highly distributed manner. Other examples of this are for instance standard compliance in ArgoUML [Persson et al. 2005], and modularity in UNIX [Weber 2004].

¹It is easy to capture data from a CVS tool. This could perhaps contribute to create the results and make the CVS tool look more important than it really is.

4.3.5 Informal Design

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 projects have in many cases no clear design phase and one can question how these projects are able to successfully create software. 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.
- The architecture can evolve from a draft into a useful modular architecture over time.
- Developers can systematically improve the code in order to improve the architecture.

4.3.6 Freedom of Choice

Developers participate in OSS development by using their own resources [von Hippel and von Krogh 2003]. When they provide their own resources they are in many cases able to choose which tools they use in their work with the OSS product.

While some developers are paid for their participation in OSS project, most developers participate on their own initiative. This has two important implications. First of all, a developer is attracted towards a project he is interested in. He is not put on that project. This must clearly influence the developer's motivations for contributing to the project. Secondly, a developer cannot be told what to do because he is participating on his own initiative. He is in some sense free to choose his own tasks. There are ways of making a developer do what the community wants him to do but because the developer is not paid to work in the community, they must accept that he rejects certain tasks or prioritize other tasks.

4.4 Industrial Use of OSS Development Practices

We basically leveraged our rather extensive experience with the open source communities and we have borrowed many of their philosophies, strategies, tools and a lot of their culture to transform IBM's internal development practices to support global component development and promote collaboration and reuse of technology. Doug Heintzman, IBM Software Group's VP of Strategy and Technology in [Worthington 2005]

Studies on the benefits of using OSS development practices in industry are missing [Gurbani et al. 2005]. Nevertheless, the use of OSS development practices can be found several places in industry. This is evidence of the increasing influence OSS has on the industry and perhaps vice versa.

One example of where OSS development practices have been applied is in the Telecom business. Lucent Technologies has developed an Internet telephony server using several OSS practices [Gurbani et al. 2006]. The authors show how they implemented an open source cross business-unit strategy to implement the server. To allow the open sharing of code between the units, they used a code repository. The repository and the open code encouraged feedback and it allowed branching, easy patching, and relatively frequently releases.

[Lussier 2004] reports how the development in his company was influenced by OSS development. Lussier's company participated for some time in the WINE² project. They saw advantages of several of the practices used in the WINE project. As a consequence they implemented peer review, source code repositories, single committer (module ownership), mailing lists, and standards for bug reporting in their company. These practices had several benefits like time savings, improved quality, and improved skills.

[Martin and Hoffman 2007] explain how they use development practices inspired by agile development and OSS in a small organization. Martin and his colleagues use practices and tools like CVS, tools generating documentation from code, mailing lists, Wikis, issue trackers, and they keep the status of the product available to the developers.

Paech and Reuschenbach describe how they have used user-driven just-in-time requirements engineering inspired by OSS development for selection of an E-Learning tool [Paech and Reuschenbach 2006].

[Bolado et al. 2004] report how OSS practices have influenced platform development around an open source CPU core. They wish to learn from software development where a developer can use, modify, debug, and improve the code. The authors hope that this can be done for hardware platforms as well.

While there are many advantages related to OSS practices, there are some potential disadvantages too. The use of OSS development practices may make organizations pay less attention to strategic planning, detailed requirements elicitation, testing, and organized support [Spinellis and Szyperski 2004].

4.5 Summary of OSS Development Practices

The software industry may learn many things from OSS practices? Perhaps the most important thing they can learn from OSS is how OSS projects manage to attract and take advantage of large numbers of volunteers. The involvement of

²<http://www.winehq.com/>

users is closely related to the openness of the OSS communities. Sharing information and encouraging users to provide feedback have several positive effects. Increased transparency could lead to increased trust and increased quality. Everyone is allowed to see how the software is constructed and they may provide feedback if something is wrong, missing, or can be improved.

Even though the findings in literature are scarce, it is evident that OSS practices are used in the industry. It is hard to distinguish OSS practices from software engineering (SE) practices, especially with the entrance of companies into the world of OSS. Communities where the industry is heavily involved use these practices and they use practices from SE. Both The Eclipse Foundation and The Apache Foundation have incorporated process descriptions and several guidelines of how the development should work using OSS and SE practices, see for instance <http://apache.org/foundation/how-it-works.html> and http://www.eclipse.org/org/documents/Eclipse_Development_Process_2006.pdf.

Companies can benefit from practices originating from the OSS community, but they must be aware of potential pitfalls when implementing these practices. To understand these advantages and pitfalls, more research is clearly needed.

Chapter 5

Industrial Providers of OSS

Even though companies like Sun, MySQL, eZ Systems, Trolltech and others have released their products as OSS, we discovered little empirical evidence in the research literature. This is evidence of how companies provide OSS products, and attract and sustain a community around these products. Some evidence exists but it is scarce and dominated by personally experience stories rather than empirical research. However, some evidence related to the start-up and release of commercial OSS products can be found in the literature. This chapter will focus on these few examples of commercial OSS products.

5.1 CommSy

CommSy is a web-based groupware system developed at the University of Hamburg since 1999 [Bleek and Finck 2004]. The development of the software has been supported by a government funded research project until late 2003. When the funding ended, the researchers aimed to move the project to the open source community to ensure its continuity. Several of the original developers were interested or eager to continue working on the project [Bleek and Finck 2005].

At the end of the government funded research project, they started branding the software as a product and no longer as part of a research project. They decided to license the software under the GPL license and make it easy for new developers to join. They also decided to open up and move the development to SourceForge¹, including bug-tracking, feature requests, task tracking, and documentation. The inclusion of new members and the scattering of the original developers lead to reduced use of face-to-face meetings and increased use of electronic communication [Bleek and Finck 2005].

¹<http://sourceforge.net/projects/commsy>

The authors describe some challenges they have to deal with. How are new participants integrated into the team and culture? The statistics at the projects home page at SourecForge shows a pretty stable history without any dramatic growth after its move to SourceForge². The development has slowed down because the original developers now had new day-time jobs and because they were distributed [Bleek et al. 2005].

A positive side-effect of the move to SourceForge was increased code and documentation quality. This happened due to increased visibility and because the developers were anxious about their reputations [Bleek et al. 2005].

The team identified some challenges related to their move from closed to open source development [Bleek and Finck 2005]. These challenges are summarized in Table 5.4.

5.2 Open Source Component Artifact Repository

Open Source Component Artefact Repository (OSCAR) was intended to be a tool for supporting storage and retrieval of large collections of heterogeneous software artifacts [Boldyreff et al. 2004]. The development of this product ran in to several problems as described by Boldyreff et al. The project group did not agree on a configuration management strategy and they did not use the same tools for version control of source code. This made the development very difficult. The project team wanted to deliver a mature product to the open source community and they decided to develop it internally and then release a mature version of it. This was reported to be a big mistake because communication with the community was very scarce during the development phase. External users were because of the lack of a product and the lack of communication, not particularly interested in the project.

From the unlucky outcome of the OSCAR project, we can learn that it is important to use a common set of tools/protocols and practices for development and versioning. It is also extremely important to communicate with possible users. Unless you do that, they will just leave the project.

5.3 Mozilla

The Mozilla project is perhaps one of the most famous commercial OSS projects known. When Netscape released the source code to their Navigator browser in 1998, Mozilla became one of the first projects to originate from a company. The release of the source code and the browser came as a response to that the Netscape was loosing market share to Microsoft's Internet Explorer [Feller and Fitzgerald 2002].

²Last accessed 2007-08-10

One of the main issues with the release of the source code was the *third-party code* included in the product [Hamerly et al. 1999]. Netscape had to work hard to convince the third-parties to allow them to release their product as binary and/or with the source code. If a third-party vendor did not agree, their code had to be re-implemented.

Another issue which had to be solved was *licensing*. Netscape considered both the BSD and the GPL license. The BSD license was found insufficient and the GPL license was because of its viral behavior not desirable [Hamerly et al. 1999]. Netscape decided to create the Netscape Public License (NPL) and the Mozilla Public License and license the code under these licenses. NPL gave Netscape rights to use the code as they wanted. Community members were not happy with this and they required Netscape to differentiate between new functionality and bug fixes [Hamerly et al. 1999]. People were willing to give away their bug fixes but a bit more reluctant to give away new functionality. They did not necessarily want someone else to make money based on their hard work.

The browser was earlier developed in-house with a more traditional development model. Releasing the source code as open source was clearly not enough. The development of the product had to be made open as well [Baker 2006]. The control over the code needed to be distributed outside Netscape. With people from all over the world participating in the development, both *infrastructure* and *people* were needed to coordinate these development efforts. Netscape solved these challenges by founding the Mozilla Organization and registering the `mozilla.org` domain [Baker 2006].

To avoid quality and legal problems, it was important to review both code quality and the licenses used with code contributions. These issues were partly solved through *code ownership* [Hamerly et al. 1999]. A module owner would be a person who knew the code very. The owner would also decide what code to include or not into that module.

Opening up the development model also meant that the development crew had to release more information to the public and to use public information channels to include the community members. Storing and sharing information about the code allow people to keep track of who do what. The Mozilla project tracks quite a lot of information related to every piece of code in the code tree, like: who checked it in and who reviewed it, when it was checked in, what problem it was addressing and the history of that problem, any new problems related to the code and so on [Baker 2006; 8].

There are several lessons that can be learned from the Mozilla story. Some of these are summarized in Table 5.4. [Baker 2006] rises several interesting questions which should be considered when creating an open source project.

- Code review for everyone?
- Who should have write access to the code repository?
- How to decide the release schedule? Should it be fixed dates, when it's

done, maintenance releases?

- How should quality assurance activities in the community be organized?

A last important observation from the Mozilla case is that other things than the product itself matters. When Firefox started to get popular, a wave of viruses and security issues came across the Internet. These problems created a need for a new browser and Firefox was there to fill that need [Baker 2006]. The release of a new open source product is not just left up to chance but when the chance is there, you have to take it.

5.4 Lessons Learned

Some lessons learned from the three cases described above are summarized in Table 5.4. These lessons may not be applicable in every setting and they should be evaluated carefully before they are implemented.

5.5 Other Evidence

The literature study did not discover many other publications about OSS. Our previous study found that attracting, sustaining, and benefiting from a community is hard work [Hauge and Røsdal 2006]. The community needs attention to grow and to contribute. Contributions from the community will also generate extra work because they have to be reviewed and in many cases rewritten or rejected. A company can however benefit from all the activity around the product in form of add-ons, free publicity, feedback, and increased sales. One example is Activmedia's Pioneer robot. The sales of the robot increased after the release of its software as OSS [Barrera et al. 2005].

5.6 Summary

More research is clearly needed to understand how companies are able to release OSS and benefit from doing so. In an OSS community driven by a company, we will find code sharing, most likely some kind of peer-review, and it is likely that the company's development processes are influenced by the presence of a community. The company will hopefully benefit from having a community and vice versa.

Case	Lessons
CommSy	<p>Make it simple for people to join</p> <p>Think about branding of the product</p> <p>Roles and their tasks should be documented publicly</p> <p>Keep track of the people involved in the project and what they do</p> <p>Include team members or at least make the decision processes transparent</p> <p>To keep the transparency of communication, limit the number of communication means, copy communication to several means or store all communication in a common place</p> <p>To avoid getting a project with an endless scope, make sure to have a commonly know goal and mission statement</p> <p>Identify conflicts early and have transparent decision processes</p>
OSCAR	<p>Make agreement on how to perform configuration management</p> <p>Use the same tools or tools with the same interfaces/protocols</p> <p>Communication with the community must be kept alive at all stages</p>
Mozilla	<p>Use licenses which are appropriate both for you and the community</p> <p>Review both contributed code and its licenses</p> <p>Create a meeting place for your community</p> <p>Provide the necessary infrastructure</p> <p>Store and share information</p> <p>Open up your development</p> <p>Consider letting go of (some) control</p>

Table 5.1: Lessons learned from companies providing OSS products

Chapter 6

Industrial Selection of OSS Components

OSS influences software development primarily by offering reusable components with higher quality and more functionality, than if you wrote them yourself [Spinellis and Szyperski 2004]. In 2004, [Madanmohan and De' 2004] claimed that there existed no empirical analysis of the use of OSS components and how these components were selected. However, in the last few years we have seen several publications shedding light on the use of OSS components. This chapter will present some of this work.

6.1 The Selection Process

Reuse of OSS can be a systematic or a more informal developer-dependent process. [Morad and Kuflik 2005] shows how they implemented systematic reuse of OSS using a central reuse team and a repository of reusable OSS components. The reuse team is responsible for the components which are selected based on real needs and clearly defined requirements. All components are evaluated to ensuring that they meet a minimal level of quality, documentation, reusability and scalability.

[Hauge and Røsdal 2006] describe OSS reuse as an informal process where the selection is depending heavily on the individual developer. A developer discovers a need, searches for a long-list of candidates before he briefly evaluates these candidates and reduces the list to a short-list. The candidates on this short-list are evaluated or tested further before one component is selected and implemented.

6.2 Finding OSS Components

COTS components are most commonly found through a familiarity-based selection process [Li et al. 2006a], combined Internet search with hands on trials [Li et al. 2006a, Chen et al. 2007], or search for components or comparisons of components [Hauge and Røsdal 2006]. These searches are performed using search engines, and repositories like SourceForge, Freshmeat etc.

[Madanmohan and De' 2004] describe similar search methods for OSS components. Most developers used manual methods with search on for instance Google or Freshmeat. The authors also mention formal methods, artificial intelligence methods, ontological approaches, and corporate maintained portals with information about OSS components as other possible search approaches.

6.3 Evaluating OSS Components

The evaluation of an OSS component involves many different criteria. It requires a special approach because it is different from proprietary software in other ways than just the availability of the code [Cruz et al. 2006]. It is of course important to evaluate the component itself, but properties of the community and licensing issues must be taken into consideration as well. The following subsections will discuss several evaluation criteria which are special for OSS products.

6.3.1 The Product

OSS products must be evaluated like any other software product. Coverage of required functionality, extra functionality, how well it responds to changing needs, how easy it is to start using it, design and architectural fit, ease of integration, and quality attributes like security, reliability, performance, modularity, dependencies, etc, should be evaluated.

Furthermore, due to the availability of the source code, it is possible to perform *code reviews*. Code reviews can give extra information about the quality of the code and the product itself. Structured, effective, and easy-to-understand code is an indication of quality. Good code quality shows that the developers have time to perform quality assuring activities. If they have time to organize the code, they have most likely had time to do other quality assuring work as well. Secondly, it is a lot easier to find and remove problems in structured code. Hence, structured code reduces the risk in case there are defects in the software.

Many of the close to 150 000¹ OSS projects registered at SourceForge are immature and unstable. [Norris 2004] recommends *evaluating the maturity* and how established an OSS product is before using it. The number of stable versions,

¹149 391 registered projects on the 2nd of June, 2007

conformance to open or industry standards, the number of adopters, etc, can be indicators of the maturity of the product.

Another very important property of the OSS product is the *license*. The license must be compatible with your intended use. See Chapter 2 for a discussion of the different license types. It is also important that the product has a clear legal status [Merilinna and Matinlassi 2006]. There should not be any uncertainty related to the ownership of the code base, existence of third party code liability should be clear (a company may not want to be liable for code they have not written), and there should not be any patent restrictions [Cruz et al. 2006].

6.3.2 Community and Position of OSS

When using an OSS product, it is important to determine if it is a long life survivor [Norris 2004]. The position of an OSS product is primarily determined by its community but also by its competitors.

It is an advantage that the OSS product has *support from a strong community* and that it is supported by several software vendors [Merilinna and Matinlassi 2006]. An OSS product with a large and active community has many developers which can move the product forward. Small communities are clearly more vulnerable if one or two developers decide to quit. Companies involved in OSS communities have most likely an economic interest in the existence and evolution of the OSS product. It is therefore likely that they participate in the development of the product and drive the product forward.

An active community will develop the product further and it is more likely to fix bugs and respond to your requests. Having someone who can provide support in case of problems is important [Cruz et al. 2006]. The activity in a community can be evaluated by the number of active developers, response time to bug reports and requests on mailing lists or forums, the number of open versus fixed bugs, the number of releases, the number of developers and users, the number of updates to the code tree, and other measures. The strength of the community can further be assessed by the presence of corporate stakeholders or commercial support, quality of documentation, appearance of web site, adherence to standards and so on [Madanmohan and De' 2004].

The existence of other dominant competitive open or closed source software products is clearly a disadvantage [Krishnamurthy 2005]. It is therefore important to assess the OSS products competitive position before you start using it. The development of several OSS products has been discontinued because of other better or wider adopted products.

6.3.3 Reputation and Further Plans

The reputation of both the OSS product and the community providing it can be important input in the evaluation of an OSS product. A product, for instance

sponsored by IBM, lead by Linus Torvalds², with many adopters, and with a good reputation is most likely a safe choice. It is also easier to trust OSS product provided by for instance The Apache Foundation or The Eclipse Foundations. As these foundations are sponsored by industry and have routines and quality processes for approving new products, see for instance The Apache Foundation Incubator at: <http://apache.org/foundation/how-it-works.html#incubator>.

The OSS products should have a clear direction of evolution [Cruz et al. 2006]. This is particularly important if you plan to live with this product and if you want to update to new versions of it. The evolutionary direction of the OSS product should be compatible with your plans for your product. If the future plans of the OSS product are non-existing or incompatible with your plans you should evaluate whether you are able to influence the community.

6.3.4 Skills and Experience

Previous experience with an OSS product should be used as an evaluation criteria [Madanmohan and De' 2004]. Prior experience or familiarity is probably one of the evaluation criteria which influence the decision the most. It is easy to reuse something which has been used successfully before and it is very easy to reject unsuccessful OSS products. Prior experience also means that you have knowledge of the component which also reduces the risk of using it.

The experience added to your company by using a certain OSS product could also be taken into consideration. Using a particular product can be a move to get access to new knowledge. New knowledge can give you access to new customers.

6.3.5 Summary of Evaluation Criteria

OSS components must be evaluated like other components. [Cruz et al. 2006] recommend making a simple list of functional requirements ordered by priorities, and to define and describe several evaluation criteria. Among these requirements we should find criteria like licenses, total cost of ownership, and the strength of the community,

6.4 Use of OSS Components

The actual use or integration of OSS components is not covered by this thesis. It is not covered here to limit the scope of the thesis and because the construction of software using OSS components is described elsewhere in the literature. Several of these report that OSS components are treated like any other component. They use the component without reading the code [Madanmohan and De' 2004, West 2007, Li et al. 2006b].

²The initial developer of the Linux core

6.5 Summary

The selection of OSS components can be completed through different processes. However, several researchers have reported that selection of OSS components is an informal activity based on internet searches, familiarity, and informal evaluation and testing. One of the advantages and challenges related to selection and evaluation of OSS components is the availability of information. There are many thousands OSS components available, all with many different information sources. These information sources should be used in the selection process to simplify the evaluation.

Nevertheless, more research is needed to understand how the software intensive industry can perform more effective selections of OSS components. To do this it is necessary to increase the understanding of how selection of OSS is performed today.

Chapter 7

Industrial Participation in OSS Projects

Several major and minor companies are daily participating in OSS projects. We have the big companies like IBM¹, Sun Microsystems², Oracle³, and Novell⁴, and we have the many smaller and not so famous companies which also participate and contribute to the OSS community.

The research literature has paid little attention to the relationship between commercial organizations and OSS communities [Østerlie 2007]. Some evidence is however found. This chapter will present the scarce evidence discovered in our literature study.

7.1 Industrial Contributions

The most common contribution is bug reports and bug fixes [Merilinna and Matinlassi 2006]. Sometimes new features are also provided. If a company finds a defect in one of the OSS products they are using, it is in their best interest to at least report the defect or possibly also correct it. [Hauge and Røsdal 2006] observed similar behavior. If a company found a bug they needed to have corrected, they reported the bug or fixed it themselves and returned the correction to the community. Embedded Linux is one example. Many hardware providers are using it and they benefit from the improvement of the Linux version. A considerable and increasing share of the companies involved in this development, share their own code with the community [Henkel 2006].

¹<http://www-128.ibm.com/developerworks/opensource>

²<http://www.sun.com/software/opensource/>

³<http://oss.oracle.com/>

⁴<http://developer.novell.com/opensource/>

The openness towards the community was strongly heterogeneous, but driven by cost/benefit evaluations [Henkel 2006]. If a company benefits from sharing their code with the community, they are much more likely to do it. If the benefits do not out-weight the inconvenience of releasing it they will most likely not do it. The more important it is to obtain external development support from the community, the more code will the company reveal. Small companies reveal more, maybe because they need or expect more external development support. At the same time they do not reveal everything, they protect their IP.

The influence a company may have on an OSS component is clearly related to the interest they have in the component [Merilina and Matinlassi 2006] and the resources they use on its development. Company sponsored participants can through to their available resources, primarily time, get important positions in a community [Dahlander and Wallin 2006]. The sponsoring company can influence the development in the community through these individuals. However, in most cases a company cannot influence a community as much as they would like [Merilina and Matinlassi 2006].

7.2 Summary

It is quite evident that commercial organizations are contributing to the OSS communities. This participation has however gained little attention in the literature so far. We have seen some evidence that companies participate with bug reports, bug fixes, and code contributions when they benefit from their efforts, but the literature does not describe this phenomenon in detail. More research is clearly needed.

Part III

Research

Chapter 8

Research Design

The overall purpose of this thesis is to:

To understand how and why the software intensive industry is approaching OSS.

Based on this research goal we have defined five research questions RQ1-5 which are summarized in Table 8.1.

RQ1	Why do companies integrate OSS components into their software, distribute their software as OSS, and interact with and participate in OSS projects?
RQ2	Which OSS development practices are used when industry develops software?
RQ3	How does an industrial OSS provider attract and sustain a community of users and developers?
RQ4	To what extent does the industry integrate OSS components into their software and how does it find, select, and evaluate OSS components?
RQ5	To what extent is the industry participating in OSS projects?

Table 8.1: Research questions

These research questions were answered by undertaking an extensive literature review and by performing a web-based survey. This research design was partly given by the problem description. This thesis should improve and reproduce the survey in [Hauge and Røsdal 2006] by distributing a questionnaire to the projects in the ITEA 2 program, where 93 ITEA Project and Work Package Leaders were invited to participate. 204 representatives from Norwegian software intensive companies using OSS components to build software were also invited to participate in the survey. The next chapters will present the elaboration of the

research design and the questionnaire.

8.1 Research Process

This thesis is a continuation and extension of previous work performed by Andreas Røsdal and the author, see [Hauge and Røsdal 2006]. A review of the results and the feedback from the previous study was therefore a natural starting point of this thesis as illustrated in Table 8.2.

Activity	Reference
1. Review previous study and feedback	
2. Report previous study	Appendix D
3. Develop problem description	Chapter 1
4. Review literature	Section 8.2
5. Define research questions	Section 8.3
6. Develop questionnaire	Section 8.4 and Chapter 9
7. Identify challenges related to survey research	Section 8.5
8. Define population and draw sample	Section 8.6
9. Data collection	Chapter 9
10. Analyse data	Chapter 10
11. Discuss results and validity	Chapter 12
12. Report finding	This thesis

Table 8.2: Research process

Reporting results and getting feedback is an important part of research. The most important findings from the previous study were summarized into a paper at The Third International Conference on Open Source Systems in Limerick [Hauge et al. 2007], see Appendix D.

The problem description of this thesis was simultaneously developed in collaboration with my supervisors, see Chapter 1.

The results and the feedback from the first survey provided us with new understanding of the topic. To complement this understanding we performed an extended literature study reviewing more than 200 OSS-related publications. This literature study is explained in detail in Section 8.2.

The research questions found in Table 8.1 were developed based on the problem description and findings from the literature study. The development of these research questions is discussed in Section 8.3.

Input from the literature study was later used in the development of a web-survey. The detailed development of this tool is described in Chapter 9.

Researchers performing survey research face several different challenges in their work. In Section 8.5 we will look at some of these challenges and discuss what we can do to avoid them. One of these challenges is the lack of a well defined population. Our populations and samples will be defined in Section 8.6.

The data collection was done by using a web-based survey tool. The development of this tool is discussed in Chapter 9. The data from this tool were analyzed using the methods described in Chapter 10.

The results from the data analysis are presented in Chapter 11. A discussion of the validity and any limitations of the study can be found in Chapter 12 before the thesis is concluded by Chapter 13.

8.2 Literature Review

To define interesting research questions, to find evidence discovered by other researchers, and to construct the questionnaire, it was necessary to perform a literature survey. The literature study in this thesis builds on a pre-study by [Hauge and Røsdal 2006] and extends it in two ways. The current literature study includes newer publications, and is more focused than the previous pre-study. Having clearly defined the purpose of this thesis allowed the literature study to be more focused. The relatively broad literature study we performed in [Hauge and Røsdal 2006], allowed us to extend our previous review, to focus more on OSS in an industrial context, and to concentrate primarily on more recent publications, after 2005. The literature study was primarily used to increase the knowledge about industrial OSS and to focus the questions in the questionnaire.

The literature study showed that empirical studies of industrial involvement in OSS development are generally missing. Some studies were found and relevant results from these publications are included in Part II. The literature study also functioned as a basis for a catalogue of relevant OSS publications. This catalogue is constructed using an OSS web-tool and it is available at <http://www.idi.ntnu.no/~oyvh/bibadmin/>. The catalogue contains around 150 publications which were found relevant to this thesis. Publications which were rejected in the literature study were not included into this catalogue.

8.2.1 Finding Publications

Literature included in this thesis were primarily found through two different search strategies, focused searches in publication databases and transitive exploration. By transitive exploration we mean transitively exploring the references in other publications.

Searching Publication Databases

Electronic publication databases have been the main source of literature for this thesis. See Table 8.3 for a list of the most used publication databases. These publication databases have been searched with phrases like 'open source', 'industrial open source', 'commercial open source', 'open source components' etc.

Journal Database	URL
The ACM Digital Library	http://portal.acm.org
EBSCOhost Electronic Journals Service	http://ejournals.ebsco.com
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
SpringerLink	http://springerlink.com
ISI Web of Knowledge	http://portal.isiknowledge.com

Table 8.3: Journal databases

Transitive Exploration of References

Good publications contain references to other interesting publications. Using publications as a source for new references is very important and quite common. The initial literature review provided us with several good publications and these publications have given us a lot of interesting references. Publications providing an overview of the research field like [Fitzgerald 2006, Scacchi et al. 2006, Scacchi 2007] contain several references to other publications.

Other Sources

Other sources were also used to some extent. Stefan Koch and Joseph Feller each maintain research bibliographies at http://wwwai.wu-wien.ac.at/~koch/forschung/sw-eng/oss_list.html and <http://opensource.ucc.ie/biblio.html>. Furthermore, MIT hosts a selection of online publications at <http://opensource.mit.edu/>.

Some publications were also found through special OSS-related issues in journals, see Table 8.4. Other papers have been found through workshops and conferences on open source like the *Workshop on Open Source Software Engineering*, the *International Conference on Open Source Systems*, and others.

Journal	Volume	Issue	Year
Knowledge, Technology, and Policy	18	4	2006
Software Process: Improvement and Practice	11	2	2006
IEEE Software	21	1	2004
Research Policy	32	7	2003
IEE Proceedings - Software	149	1	2002
Information Systems Journal	11	4	2001

Table 8.4: Special Issues on OSS

8.2.2 Selection and Evaluation of Publications

To narrow the literature study we primarily focused on publications relevant to software engineering. There are many publications about OSS or community-driven activities in economics, sociology, psychology etc., but it is impossible to cover all these research fields.

The selection of publications was done manually, based on an informal evaluation process. The title was in most cases evaluated first. If it was found interesting, the abstract, introduction, and perhaps the conclusion were read. If the paper still was interesting, the pdf-file was saved and printed for more thorough review.

There were several factors influencing the relevance of the papers. The most important one was of course the topic. The publication had to be relevant to OSS, and to either some of the background topics included in Part II of this thesis or to industrial approaches to OSS. Publications published in peer-reviewed journals or known conferences were preferred but some publications from other sources were included as well. Empirical studies were preferred over publications containing only discussions or evaluations without any empirical research. The references included in the paper were also considered in the evaluation.

8.2.3 Summary of the Literature Study

The literature study is unfortunately not a comprehensive review of all OSS related literature. This would have been impossible within the time frame of this thesis. However, several trustworthy sources were used, some hundred publications were reviewed, and all the publications have been evaluated as objectively as possible. It is nevertheless an informal and subjective process. Other researchers may have ended up with a slightly different result but we are quite confident that the literature review has high validity.

8.3 Research Questions

The research questions of this thesis have their origin in the previous study. The initial research questions (I-RQ) from [Hauge and Røsdal 2006] are listed in Table 8.5. These initial research questions were refined based on results and feedback from the previous study. Table 8.1 contains the new research questions.

I-RQ1	Why do companies use OSS components, and how do they find and evaluate these components?
I-RQ2	Why do companies choose to make their products Open Source, and how can this be done successfully?
I-RQ3	Why and how do companies use development methodologies from Open Source, in ISS development?
I-RQ4	How does the use of OSS influence the development processes of companies?
I-RQ5	How and why do companies participate in the development of OSS projects controlled by a community outside the company?

Table 8.5: Initial research questions

I-RQ1, I-RQ2, I-RQ3, and I-RQ5 contain a part about motivation or why companies undertake each of the four roles described in [Hauge and Røsdal 2006]. The part about motivation from these four questions, were contracted into one new research question, RQ1.

I-RQ1 was rephrased to keep the same content but to also include experiences with the use of OSS components. Questions related to the experiences with components were asked in the previous study and in studies by [Li et al. 2005]. A similar research question related to how components are selected is also found in [Chen et al. 2007]. Experiences related to the actual implementation of OSS components were left out to limit the scope of the study.

I-RQ2 was slightly changed from, how a company could successfully make a product an open source product, to focus more on how to attract a community to a commercial OSS project and what to expect from that community. Attracting a community and benefiting from it is not a trivial matter [Hauge and Røsdal 2006]. Having a community is of course one of the success criteria for an OSS product.

I-RQ3 was rephrased to focus more on development practices from OSS development rather than methodologies. Furthermore, Inner Source Software (ISS) development was removed from the research question. ISS did not seem to be a well established term in industry and it was therefore hard to get relevant responses to this question.

The creation of the research questions was somewhat constrained because the problem description stated that the thesis should distribute a survey (question-

naire) to the participants in the ITEA program. I-RQ4 was not included into this study because of this limitation. It is hard to answer questions related to (development) processes in a questionnaire. This was one part of the feedback we got on the initial study. I-RQ4 was also dropped to limit the scope of this thesis.

I-RQ5 was kept with the same content as before.

Table 8.6 contains an overview of the changes done to the research questions. The first column describes the relevant fragments of the initial research questions and the second column contains the new research questions.

Initial Research Question	New Research Question
<p>I-RQ1 Why do companies use OSS components</p> <p>I-RQ2 Why do companies choose to make their products Open Source</p> <p>I-RQ3 Why do companies use development methodologies from Open Source</p> <p>I-RQ5 Why do companies participate in the development of OSS projects controlled by a community outside the company?</p>	<p>RQ1 Why do companies integrate OSS components into their software, distribute their software as OSS, and interact with and participate in OSS projects?</p>
<p>I-RQ3 How do companies use development methodologies from Open Source, in OSS development?</p>	<p>RQ2 Which OSS development practices are used when industry develops software?</p>
<p>I-RQ2 How can companies make their products Open Source successfully?</p>	<p>RQ3 How does an industrial OSS provider attract and sustain a community of users and developers?</p>
<p>I-RQ1 How do companies find and evaluate OSS components?</p>	<p>RQ4 To what extent does the industry integrate OSS components into their software and how does it find, select, and evaluate OSS components?</p>
<p>I-RQ5 How do companies participate in the development of OSS projects controlled by a community outside the company?</p>	<p>RQ5 To what extent is the industry participating in OSS projects?</p>
<p>I-RQ4 How does the use of OSS influence the development processes of companies?</p>	<p>Dropped</p>

Table 8.6: Mapping from initial to new research questions

8.4 Questionnaire Design

A questionnaire is a suitable tool for gathering well structured data from respondents located at different locations. Information can easily be collected from a large number of respondents. It is cheaper and easier than for instances structured interviews and the respondent is not disturbed in his work for a long time.

A survey can be either descriptive, explanatory, or explorative [Wohlin et al. 2000]. This study is primarily a descriptive study where we try to describe characteristics of the population. The characteristics we want to describe are found in the research questions.

The questionnaire is the most common quantitative method [Jacobsen 2005]. However, it can be used to capture both quantitative and qualitative data. In this study, it is primarily used to capture quantitative data. Quantitative research has the goal of measuring, counting, or comparing objects. Information is primarily gathered through pre-coded questions with pre-defined answer alternatives.

Deduction is a form of top-down reasoning where the conclusion follows from the reasons given, as opposed to more bottom-up induction, where the conclusions are drawn from facts or evidence [Cooper and Schindler 2006]. Deductive research is based on existing experiences and theory, while inductive studies create theories from their own observations. Research using questionnaires is often based on deductive reasoning because information must be ordered before the study is performed. For example, when using closed questions the answers must be prepared beforehand. Therefore, deductive reasoning can only create a limited amount of new knowledge within the scope decided by the researchers [Jacobsen 2005]. Thus by using closed questions, knowledge outside the given alternatives will not be discovered.

One major limitation of questionnaires is that questions can only be asked about topics which the researcher finds important and relevant. 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].

The study was performed as a cross sectional study, capturing one moment of history. 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 2006]. The respondents were asked to report their experiences with the development of a software product. Performing a longitudinal study is out of scope of this thesis, but the research design and the questionnaire may be reused at a later stage.

The actual development of the data collection tool is discussed in Chapter 9 and the final questionnaire is available in Appendix B.

8.5 Challenges with OSS Related Survey Work

By looking at the work of others we may identify some of the specific challenges of performing OSS-related surveys in software engineering (SE). By identifying these challenges we would be able to reduce the threat they pose to the validity of our results. Our intention is not to criticize the work of other researchers but to highlight some of the challenges they have met when performing survey work in SE. We have identified a list of challenges which will be discussed in the next sections. One or more of these challenges are present in most OSS-related surveys we know of.

- The population is not defined
- The population is very particular
- Descriptions of the population are lacking or have errors
- It is hard to reach the study object
- The gatekeeper problem
- Low response rates

8.5.1 Population: Not Defined

In 2002, Italian researchers performed a survey on companies' attitude towards OSS [Bonaccorsi et al. 2004]. As they did not have any list of (OSS) companies in Italy they adopted a snowball sampling procedure. A set of known companies were asked to refer to other OSS companies they knew about. This forward referring continued until no more companies were found. In total 275 companies were found and contacted, and 146 valid answers were obtained (53 %).

[Hars and Ou 2002] performed a web survey on the motivations for participating in OSS projects. The survey was performed by inviting 389 people whose email addresses were found through the Internet from open-source discussion lists and news groups. The survey received 81 responses, a response rate of 21 %.

The Free/Libre and Open Source Software (FLOSS): Survey and Study was performed as an online survey of OSS developers [Ghosh 2002]. 2784 individual developers which were reached through a snowball sampling procedure completed the survey. The researchers posted the survey in a few OSS communities and it was distributed within the whole OSS community by the developers themselves. Similar studies have also been performed to survey motivations and demographic information about OSS developers. The intended population in these surveys is all OSS developers world wide. Reaching all OSS developers world wide is rather unlikely. Table 8.7 has an overview of some of these online studies.

Even though most studies mentioned here receive many responses, they suffer from their sampling method. The infamous story about George Gallup and the

Name	Url	Year	Respondents
FLOSS-US	http://www.stanford.edu/group/floss-us/	2003	1588
FLOSS-JP	http://oss.mri.co.jp/floss-jp/index-en.html	2003	547
FLOSS	http://www.infonomics.nl/FLOSS/report/index.htm	2002	2748
Who Is Doing It?	http://widi.berlios.de/paper/study.html	2001	5593

Table 8.7: Online open source studies

U.S presidential elections in 1936 clearly shows that only having a lot of respondents is not enough¹. Performing surveys using convenience or snowball sampling without having a well defined population have two major validity problems. One, the researchers do not know if the respondents actually belong to the population they want to study, because anyone can respond. Two, they do not know if the actual sample of respondents is representative for the population they want to study, because they have no defined population to compare the sample with. As a consequence, they will have a hard time claiming that the results are externally valid for other populations than the actual respondents. Convenience and snowball sampling was in the mentioned studies selected, because there did not exist any complete description of the population.

One study dealt with this challenge by using data from the Norwegian and German Census Bureau, and the Italian Yellow Pages [Li et al. 2005]. The Census Bureaus have complete lists over all legal entities in the country and thus the whole population of software companies. The Yellow Pages do not contain complete overviews of all companies, but they would probably have most companies. At least larger companies should be well represented. From this population they constructed a close-to representative stratified random sample.

Having a complete overview of the population available is not always the case. In these instances it is perhaps better to define a subset of the population you originally wanted to study and look at this subset. Lakhani and Wolf defined their population as all developers participating in an OSS project at the SourceForge.net rather than all the OSS developers in the world [Lakhani and Wolf 2005]. By looking at a well defined subset it is at least possible to discuss response rates and to ensure the internal validity of the results from this subset. While generalizing from a sample drawn from a subset can be hard it is better than having a sample from an ill-defined population. With a sample from an

¹Gallup predicted the outcome of the elections in 1936 based on a representative random sample of 5 000 respondents. The magazine Literary Digest, performed an extensive poll with a de-facto biased sample of more than 2 000 000 respondents. Literary Digest predicted the wrong candidate to win.

ill-defined population it becomes hard both to generalize and to say something about how representative the results are for the intended population.

8.5.2 Population: Too Particular

When defining a limited subset of a larger population it is important not to make the limited population too particular. If the population is too particular it is perhaps not representative for the larger population and it is therefore hard to generalize the results. The results can only be said to be representative for the part of the population which is investigated.

One example is a survey on motivations performed in the Linux kernel community [Hertel et al. 2003]. Hertel et al. invited an unknown number of people who read the Linux kernel mailing-lists to participate in a web survey. The survey got 141 responses but the participants of the Linux kernel mailing-lists are hardly representative for all OSS developers.

Another example is our study performed within the COSI research project where we got 24 responses [Hauge et al. 2007]. Companies participating in a research project focusing on OSS are perhaps not representative for all companies using OSS. Yet another example is a survey of Australia's top 500 companies about why they do not adopt OSS [Goode 2004]. The biggest companies in a country are not necessarily representative for the many smaller companies. The Australian survey got 108 responses (21.6%).

8.5.3 Erroneous Population Description

Using membership list and data from Census Bureaus is far better than not defining the population. However, there are some problems related to using data from such sources, even though they are supposed to be complete. Statistics from Statistics Norway about the Norwegian industry can be generated with as much as a two-year delay. As a result of this delay and the high paced turnover in the ICT industry, these statistics are often outdated.

To define a population of companies in Norway, it is possible to use The Central Coordinating Register for Legal Entities (CCRLE). The CCRLE contains all legal entities in Norway. These entities are registered with a sector code (NACE). The CCRLE is further described in Section 8.6. Even though the CCRLE is complete there is some delay related to changes.

Several problems are also related to the NACE sector codes. Enterprises are registered with more than one sector code because they are involved in several sectors. Enterprises are registered under a sector they are no longer involved in. This could happen if they change their business focus. Enterprises are not registered under a sector which they are active in. Companies with a large IT department may develop a lot of software but they can for instance be registered as an oil company because this is their main focus.

Furthermore, a company which has one contact point may consist of several legal entities/enterprises. "Microsoft Holdings Norge AS" and "Microsoft Norge AS" are two legal entities but they are related to the same company, "Microsoft". For such companies, a small legal entity may for instance own the commercial property of the company and have a very large turnover compared to the number of employees. When later talk about companies they may consist one or more legal entities hidden behind the same contact point or company name.

Being aware of these limitations is important. However, using such data sources to define the population is clearly recommended. With the many thousand ICT companies in Norway the error is in percentage perhaps not very large. It is perhaps more concerning that you can miss large companies which are involved in several sectors but registered under a sector outside your scope.

8.5.4 "Unreachable" Study Objects

When surveying individuals, the study object and the respondent is the same and the respondent answer for himself. When studying companies, products, or projects the respondent is not answering for himself. It is necessary to get one or more individuals to act as a proxy and answer on the behalf of the study object. Different employees in the same company will most likely answer differently about the same study object. The impact of this error is reduced if respondents are randomly selected from the different companies. Reaching random respondents with deep knowledge about the unit of study, in different companies can however be difficult.

It is as we have seen possible to define a population of companies or individuals if there exist, records containing information about the population. Normally, when studying projects or products no such records exist. It is possible to find a representative sample of companies. It is however far more difficult to ensure a truly representative sample of software products [Conradi et al. 2005]. To do this we would have had to survey all software products ever made and then selected a representative sample of these software products. Conradi et al. conclude that the best one can achieve is a stratified random sample of ICT companies and a convenience sample of software projects. In these cases, the study object is hidden behind the front of the companies. This is illustrated in Figure 8.1 where the companies form an interface towards the researchers.

It is possible to draw a stratified random sample of companies. However, behind the interface of companies we will find the population of projects which is the unit of study. These are not directly reachable for the researcher. The researcher depends on a respondent in each company to select a representative project and answer based on his experiences with this project. This gives us a double mismatch. First, we would like to sample projects but we can only sample companies. Secondly, we would get answers from projects but we can clearly only get answers from individuals.

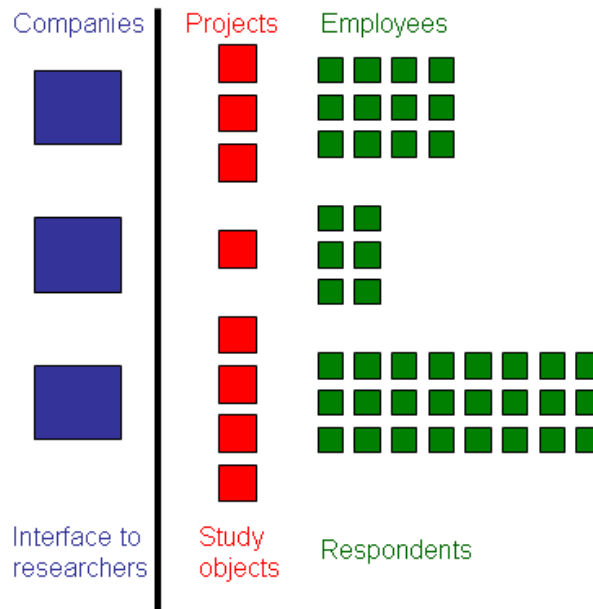


Figure 8.1: Unreachable study objects

8.5.5 The Gatekeeper Problem

When performing research on companies, products, projects, or a certain role in the company, researchers are faced with the gatekeeper problem. Most companies make a generic email address, a switchboard telephone number, and a postal address for their main office available through their websites. This forces researchers to go through one or more gatekeepers to reach the individual they are interested in.

A switchboard operator does in most cases not have deep insight in i.e. the software development in the company. Furthermore, he does not have the motivation to answer a questionnaire about something he knows little about. It is therefore important to get beyond the gatekeeper when performing survey research. Succeeding can be difficult because you do not know whom you want to reach. Thus, you may have to bounce back and fourth between different people before you reach the right employee who is both able and willing to answer your request.

When studying only one organization or a community where membership lists are available the problem is not present. [Dyba 2000] avoided this problem by using contact information of quality managers from a membership list of two ICT associations. This solves the gatekeeper problem but it may result in a particular population where the results are valid only for the members of the two associations.

8.5.6 Low Response Rates

In a Chinese study on the use of OSS components, the researchers used membership data from a national Chinese software organization to create a sample [Chen et al. 2007]. The membership data included about 6000 companies. In a screening process, they selected 2000 of these 6000 and contacted them by email. About 200 companies (10%) responded in this screening process. These 200 companies were invited to participate by email. The companies who responded were promised access to the results of the survey or an annual membership in the Chinese software organization. In total 40 companies responded (20%) to the invitation. However, 10 of the responses had to be excluded because of poor data quality. The 160 remaining companies were contacted by phone and another 17 companies returned the questionnaire. The survey ended with in total 37 valid responses (18.5%) or only 1.9% of the original 2000 companies.

A survey on the use of Off-The-Shelf components used central registers and random sampling from these registers [Li et al. 2005]. The researchers in used the Census Bureau in Norway and German, and the yellow pages in Italy to create a population of ICT companies. Random and stratified random sampling was used to draw a sample from these populations. In total 365 companies were selected in Norway, 196 companies in Italy, and an unknown number of companies in Germany. At the time the paper was written they had 115 responses from the three countries. Even though the data collection was still on-going in Germany and Italy this is a fairly low number. Only 47 of the 365 (12,9 %) Norwegian companies responded.

In 2006, the OSS Watch conducted a survey of attitudes and policies towards open source software (OSS) in UK Higher and Further Education institutions [Helsper 2006]. They composed a list of names of the individuals most likely to be in charge of ICT at each institution. These 637 individuals were first contacted by post. The invitation letter contained an URL to the web page where the questionnaire was located. The first letter was followed up with an email. Further reminders were sent by email two three weeks later to the people who had not yet completed the survey. This survey got a response rate of 18 %. A similar study performed in 2003 only gained a response rate of 6 %.

Even though the studies mentioned here have good research designs they struggle with response rates below 20 %. Increasing the response rates is clearly a huge challenge when performing survey research. To increase responses it is important to acknowledge that responding to a survey involves both cognition and motivation [Dillman 2007]. The respondent must be both able and willing to answer the questions in the questionnaire. However, multiple reminders are critical to achieving satisfactory response rates see the book [Dillman 2007], which contains lengthy discussions of how to achieve this. The main points are to make a questionnaire which is easy to comprehend and respond to, to motivate the respondent, and to remind him over and over again.

A Norwegian survey on success factors for software process improvement got an impressive 77.9% response rate [Dyba 2000]. This was achieved by drawing a sample from memberships in two national IT interest organizations, inviting the 154 respondents by phone, and keeping the questionnaire short (about 10 minutes).

The survey by [Bonaccorsi et al. 2004] achieved a very decent response rate of 53 %. The description of the research method in this study does not contain supplementary information. It is therefore hard to say with certainty what caused the high response rate. The fact that the respondents were referred to by another respondent could perhaps increase their involvement or their feeling ownership in the survey.

A survey performed in the US and Canada about the use of OSS in companies, got a total response rate of 50% [Ajila and Wu 2007]. The researchers randomly selected 120 software-intensive organizations. These organizations were contacted by email to request their participation in the study before the survey was distributed. 85 organizations responded positively but 10 organizations had less than 20 software developers and fell outside the scope of the study. Out of the remaining 75 organizations, 60 completed and returned the survey.

In all the three studies mentioned above, the participants were informed about the survey or asked to participate prior to the distribution of the survey itself. The survey using phone contact had the highest response rate by far.

8.5.7 Summary of Challenges with OSS Related Survey Work

So, when performing a survey, you should draw a representative sample from a well defined population. We recommend contacting the sample and inviting them to participate, preferably by phone. Then the sample should be given a questionnaire which is easy and motivating to complete. After the survey has been distributed, it is important to follow up with reminders to increase the response rates.

8.6 Population and Sample

We wish to investigate how and why the software-intensive industry approaches OSS. With the software-intensive industry we think of software houses, software consulting companies, or companies developing software for internal use. We will refer to these companies as *software developing companies*.

To investigate this kind of companies we first define a population of companies and draw a sample from this population. Then we ask the selected companies to pick a typical OSS product or a typical software product containing OSS components and to answer based on this product. These software products are the study objects of this survey. Two samples taken from the Norwegian ICT

sector and the ITEA 2 program, and the study objects are summarized in Table 8.8. The terminology is adopted from <http://www.socialresearchmethods.net/kb/sampterm.htm> and the presentation from [Conradi et al. 2005; 217].

	Sample One	Sample Two
Theoretical population	All software products developed by companies which either contains OSS or is licensed as OSS	
Study or target population	OSS or software products containing OSS components developed by companies in any of the ITEA 2 projects.	OSS or software products containing OSS components developed by Norwegian ICT-companies.
Sampling frame	OSS or software products containing OSS components developed by companies on lists provided by ITEA 2, see Section 8.6.1	OSS or software products containing OSS components developed by Norwegian ICT companies registered in The Central Coordinating Register for Legal Entities, see Section 8.6.2.
Sample	OSS or software products containing OSS components developed by the 93 companies of ITEA Work Package and Project Leaders.	OSS or software products containing OSS components, developed by 204 Norwegian companies selected from The Central Coordinating Register for Legal Entities.
Sub-sample	OSS or software products containing OSS components developed by the companies which responded to the survey	

Table 8.8: Description of the population and samples

8.6.1 ITEA 2 Participants

ITEA 2² is a European research program. ITEA 2 supports R&D projects which can give European industry an advantage within the Software-intensive Systems and Services (SiS) sector. This is done by bringing together partners from industry, universities, and research organization and providing links between funding, technology, and engineering skills. The research projects are funded by the participating organizations or countries. ITEA has, as of February 2007, 85 projects with more than 500 partners from 25 countries [ITEA 2007].

The ITEA 2 website lists 45 on-going projects related to different aspects of research on software and/or hardware. A list of these 45 projects can be found

²Information Technology for European Advancement: <http://www.itea-office.org>

in Appendix C. Each of these projects has a number of academic and industrial participants from many of the European countries.

The Sampling Frame and The Sample

Due to legal reasons, ITEA could not give third parties access to their email lists. We were thereby not allowed direct access to the population and we were therefore unable to control the population and the selection of respondents. Furthermore, we were unable to directly remind each individual subjects of the survey. However, ITEA did assist us in the selection of respondents. When the Chairman of the ITEA 2 invited NTNU to perform a new survey within the ITEA projects we hoped to get better access to the population and more high-level support than we actually got in the end.

ITEA performed a first screening of their database and found a list of 42 Project Leaders and 135 Work Package Leaders from the industry in the ITEA projects. Thereafter, they corrected overlap between these two lists and removed some people from the same company and department. A second screening resulted in a list of 30 Project Leaders and 63 Work package leaders from in-launch and on-going projects. These 93 contact persons constitute the sampling frame for the first sample. Through these contact persons we reached the first sample of software products.

Invitation to Participate

The respondents were invited to participate through an invitation letter which was prepared by NTNU in co-operation with ICT Norway, see Appendix A. The letter was signed by the Programme Co-ordinator of ITEA Erik Rodenbach and the Project Manager in the COSI project Dr. Frank van der Linden. Thereafter, it was sent from the ITEA office to the participants by email. 93 Project and Work Package Leaders received the invitation letter. These 93 were asked to answer personally or, if they were not involved in software development, to ask someone else from their local business unit to answer.

The questionnaire was ready to be distributed in mid/late March but due bureaucracy internally in the Norwegian COSI project and ITEA, it was not distributed until the 21th of May. A reminder was distributed to all 93 Project and Work Package Leaders June 11th, see the second part of Appendix A.

8.6.2 Software Intensive Companies in Norway

The sample of companies from ITEA was supplemented with a sample of Norwegian ICT companies. This sample which constitutes the sampling frame for our study was constructed in a five step process. An overview of this process is illustrated in Figure 8.2 and the process itself is explained below.

1. Start with a convenience sample of about 430 companies from earlier studies.
2. Merge this list with a stratified random sample of about 1250 legal entities drawn from The Central Coordinating Register for Legal Entities.
3. Find electronic contact information for all companies. Contact information for about 1000 companies was found.
4. Contact 949 companies by email asking whether they develop OSS or use OSS in their development. 558 companies responded.
5. Invite 204 companies involved in OSS development to participate in the survey.

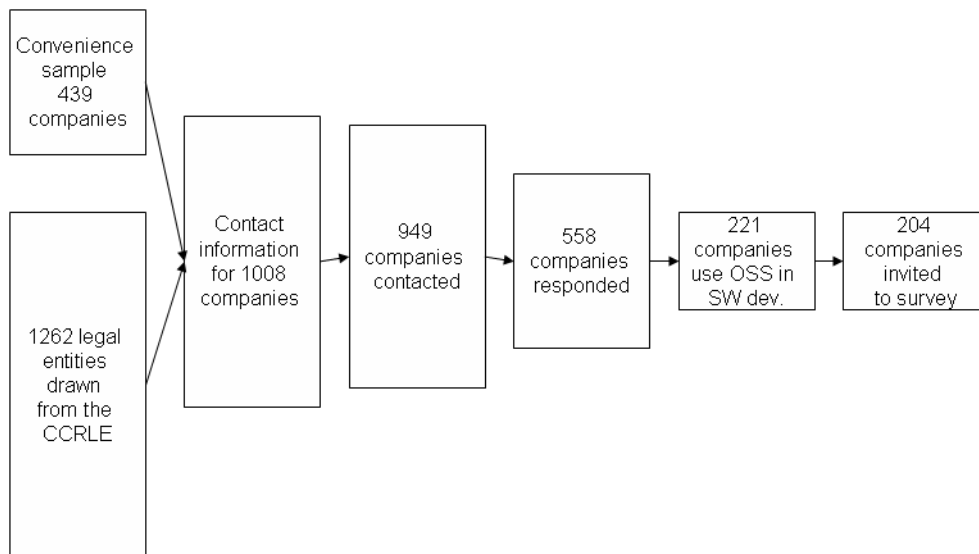


Figure 8.2: The sampling process

To be able to draw a sample of software developing companies from the Norwegian ICT sector it has to be defined. This is done by giving an overview of data sources used to define it. Then, the population is described before we go through the sampling process in detail.

Data Sources

Statistics Norway³ (SSB) is the national census bureau in Norway. SSB provides several statistics about the Norwegian trade and industry. Several of these statistics are based upon data from the official register authority, The Brønnøysund Register Centre⁴.

³<http://ssb.no/>

⁴<http://www.brreg.no/>

One of these registers is The Central Coordinating Register for Legal Entities (CCRLE) (Enhetsregisteret). The CCRLE collects information on all Norwegian legal entities (enterprises). All entities in this register are associated with one or more sector codes. The official register authority has adopted the Nomenclature Generale des Activites Economiques dans L'Union Europee (NACE) which is the European Union's extension of revision 3.1 of United Nation's International Standard Industrial Classification (ISIC). Both Nace and ISIC are used in classifying data according to kind of economic activity, see Table 8.9.

Name	Url
NACE	http://ec.europa.eu/comm/competition/mergers/cases/index/nace_all.html
ISIC	http://unstats.un.org/unsd/cr/registry/regcst.asp?Cl=17&Lg=1 .

Table 8.9: Economic classification codes

In addition to the sector code, the CCRLE contains information about the enterprise type, contact information, the number of employees (only for AS/ASA), start-up-date, and some financial information. The contact information is unfortunately incomplete and outdated in most cases. See Section 8.5.3 for a discussion of other possible problems related to databases like the CCREL.

The Norwegian ICT Sector

In 2005, the Norwegian ICT sector had about 70 000 employees or 4.7 % of all employees in Norway [SSB 2005]. The number of employees includes both the owner(s) and hired personnel. The ICT sector includes ICT manufacture industry (Nordic Semiconductor, Alcatell Bell Telephone), ICT wholesale and retail trade (Elkj p, Komplet), telecommunication (Telenor, Netcom), Maintenance of databases (DB Medialab, Sesam.no), and other computer related activities as consultancy (Tietoenator, Accenture), and software houses/vendors (Microsoft, Visma). These sectors, their NACE code, number of employees, and turnover in millions NOK are given in Table 8.10. The database sub sector will most likely be moved from ICT to content management in the new ISIC standard in 2007.

The ICT trade sector is not likely to develop large amounts of software. Software development is not the main focus for the ICT manufacture industry and the telecommunications either. However, the sector for other computer related activities including ICT consultancy and software houses has its focus on developing software and providing related services. For these reasons, we decided to focus on this sector. The "Computer and related activities" sector with NACE code 72.xx can be broken down further using data extracted from the CCRLE. The number of legal entities registered for each sub sector, are shown in Table 8.11⁵.

⁵Data extracted July 4th 2007

Sector	NACE	Employees	Turnover mill. NOK
The whole ICT sector		72 918	175 106
Manufacture of ICT related equipment and materiel	3x.xx	10 324	24 462
ICT wholesale and retail trade	5x.xx	10 870	41 328
Telecommunications	64.2x	13 019	60 072
Computer and related activities	72.xx	36 311	49 244
Database activities	72.40	2 394	

Table 8.10: The Norwegian ICT Sector in 2005

Sub sector	NACE	Number of enterprises
Computer and related activities	72.00	26105
Hardware consultancy	72.10	251
Software consultancy and supply	72.20	21559
- Publishing of software (software houses/vendors)	72.21	1295
- Other software consultancy and supply (consultancy)	72.22	20 264
Data processing	72.30	489
Database activities	72.40	2916
Maintenance and repair of office, accounting and computing machinery	72.50	733
Other computer related activities	72.60	163

Table 8.11: Distribution of entities in the 72.00 category, 2007

Data from SSB shows that only 49.0% of these legal entities are active enterprises and that most of these enterprises are very small. 72.2 % of the active enterprises have no employees and 17.9 % have 1 to 4 employees. Table 8.12 shows the distribution of active enterprise over the number of employees [SSB 2007]. An active enterprise is an enterprise which is registered in the CCRLE, the Central VAT Register (Momsregisteret), and the Employer/Employee Register (Arbeidsgiver/arbeidstagerregisteret). While the CCRLE may have unreliable data, the Central VAT and Employer/Employee Register are considered to only contain reliable data. Inactive companies are therefore held outside the statistics in Table 8.12. Some of these "inactive" companies may report turnover or pay employees for instance every second year. So even if they are reported as inactive they may have some, very limited activity.

By using data from Table 8.10 and 8.11 gives us an average of 1.4 employees per registered legal entity in the 72.00 sector. If we use the number of active compa-

Category	Number of active enterprises
Total	12 798
No employees	9 238
1-4 employees	2 294
5-9 employees	558
10-19 employees	353
20-49 employees	230
50-99 employees	75
100-249 employees	41
250 employees and more	9

Table 8.12: Distribution of active companies over size, 2007

nies from Table 8.12 we get an average of 2.8 employees per active enterprise.

There are several possible types of companies in Norway. A list of the most common types and the number of enterprise registered under NACE code 72.00 are shown in in Table 8.13⁶.

We see that most of the enterprises are sole proprietorship (ENK: Enkeltmannsforetak) which is the same as in all other branches and countries. The sole owners of such enterprises have full responsibility of the enterprise's economy and debt. The owner is not very likely to undertake the economic responsibility of producing (large) software systems. Furthermore, this enterprise type is most normal for very small enterprises without the resources to develop (large) software products. Many of these small enterprises are inactive or just run as a hobby. The business form sole proprietorship is for these reasons not that interesting for this survey.

(Public) limited companies (AS/ASA: Aksjeselskap/Alment Aksjeselskap) is the most common enterprise type for larger companies because the owners do not have personal (economic) responsibility if the enterprise goes bankrupt. The AS/ASA is also the second largest category of enterprise.

We decided to keep our primary focus on AS/ASA with more than 10 employees because of two reasons. First, an enterprise needs to have a certain size before it is able to develop (large) software products and reuse OSS components. Secondly, AS/ASA is the most common business type for companies who are able to produce software. However we will include both smaller enterprise and companies with other enterprise types in our sample.

⁶The total number of companies in this Table 8.13 is barely smaller than in Table 8.11. This is because some companies are registered with more than one sector code.

Enterprise type	Number of enterprise
Sole proprietorship (ENK)	17953
Private/public limited company (AS/ASA)	5236
Partnership with apportioned liability (DA)	1049
Norwegian division of a foreign entity (NUF)	742
General partnership (ANS)	728
Other	146

Table 8.13: Enterprise types

Step 1: A Convenience Sample of Norwegian ICT Companies

The basis for the second sample in this survey was a list of Norwegian ICT compiled by Dr. Carl-Fredrik Sørensen and Marinela Gereá for Gereá's Master thesis. Sørensen and Gereá's list was based on list created by [Li et al. 2005]. Li's list was again based on data from 2002 provided by ICT Norway and Statistics Norway. The list contained the 100 largest IT companies, the 15 IT departments in the largest 3 companies in 5 other sectors, 150 medium-sized software companies (20-99 employees), and 100 small-sized companies (5-19 employees). This summary of companies was unfortunately outdated. It contained several non-existing companies and it missed several new ones. Companies which have ceased to exist were removed from the list. This was done by looking them up on the Internet through Google searches and company urls. If a company was not found it was assumed to be non-existing. Some company names lead to new companies and these new companies were added to the list, while the old ones were removed.

Sørensen and Gereá supplemented the list from other sources. The Norwegian "Yellow Pages"⁷, LinkedIn⁸, and magazines like ComputerWorld, Teknisk Ukeblad, and some newspaper were used to find companies to complement the list.

The information in the list from Sørensen and Gereá was verified and updated as a collaborative effort with Sørensen. This included updating all company names with the correct names registered in The Central Coordinating Register for Legal Entities and removing some duplicate entries. The list was further extended with about 60 new companies from a list of 200 companies generated by the FAMILIER research project together with the Norwegian ICT industry. The resulting list with about 430 Norwegian ICT companies functions as a starting point for this survey.

⁷Gule Sider <http://www.gulesider.no/>

⁸LinkedIn is a web-service which allows you to publish your own CV and create a network of contacts. It is available at <http://www.linkedin.com/>

Step 2: Extending the List

The convenience sample of Norwegian ICT companies was a bit biased because the companies in the list were included based on knowledge about their existence. To complement this sample and to reduce possible bias we decided to draw a random sample from The Central Coordinating Register for Legal Entities.

The most relevant sector code for this survey is as mentioned "72.00 Computer and related activities" and especially the sub sector "72.20 Software consultancy and supply". 72.20 is also the largest sub sector. However, from the convenience sample described in the previous section we saw that about 20 % of the companies were registered under other sub sectors. These companies are mainly large companies which are involved in several sectors. From Table 8.14 we will also see that 244 of the companies from the convenience sample were selected in the sample from the CCRLE.

We decided to include all companies with more than 10 employees from all the sub sectors under "72.00 Computer and related activities" and to draw a random sample from other categories. This selection was done using a small tool built on the Java `Math.random()`⁹ method. We selected in total 1262 companies from the CCRLE from different strata. From the 72.2x sub sector we selected companies from the following strata: AS/ASA with more than 25 employees, AS/ASA with 10-24 employees, AS/ASA with 5-9 employees, AS/ASA with 2-4 employees, AS/ASA with 0-1 employees, AS/ASA with an unknown number of employees, ANS, DA, and NUF. From the other 72.xx sub sectors (not 72.2x) we selected companies from the two groups; more than 10 employees and less than 10 employees. Table 8.14 contains an overview of these strata.

Step 3: Finding Electronic Contact Information

The 1262 legal entities selected from CCRLE were matched with the companies in the convenience sample. The results from this merge are listed next and the procedures for this merge are explained further below.

- We did not find contact information for 395 entities.
- 244 of the 1262 legal entities were already included in the convenience sample.
- 56 entities were not included in the list as they had a common contact point with one or more entities which were already included in the list.
- 569 entities were added to the list from the convenience sample, resulting in a list of 1008 companies.

We first looked up all companies not included in the sample using data from CCRLE, searching on Google and company databases like Bedriftsdatabasen AS,

⁹[`http://java.sun.com/javase/6/docs/api/java/lang/Math.html#random\(\)`](http://java.sun.com/javase/6/docs/api/java/lang/Math.html#random())

and by trying possible urls. I.e. the company "Daldata AS" will most likely have the domain name "daldata.no". If we found any web-site we recorded the url, contact e-mail, and phone number. When searching for contact information we preferred company contact information rather than personal contact information, i.e. "info@microsoft.com" rather than "bill.gates@microsoft.com". In those cases where no such company address was available we preferred lists or persons related to software development or at least IT.

Companies, for which we could not find any web-sites, would either have closed down or decided to keep a very low profile. Keeping a low profile for a company making money of software, sounds odd. Companies which want to sell their products or services will need to do some marketing and a web-site is one very common way to do so. Companies without web-sites and contact information were therefore not included in the list. If a company had recently merged with another company, it was not included in the list. This was done because the company we were looking for did not exist. As we see of Table 8.14 and Figure 8.3 we see that small AS/ASA and non AS/ASA were harder to find than the large AS/ASA.

Some of the 1262 enterprises are registered as several different legal entities but they have one common contact point. One example is: Visma IT AS, Visma Retail AS, Visma Software Norge AS, and Visma Unique. These are four different legal entities in the CCRLE but they use the same contact point. In such cases only one of such entities were included as a new company and the rest were added to the "duplicate list", see Table 8.14.

It was no surprise that we could not find contact information for 395 of the 1262 randomly selected companies (31.3 %) when data from Statistics Norway showed that only 49.0 % of the enterprises registered in the CCRLE are active. The reason why our number (31.3 %) is lower than the data from Statistics Norway is probably because we have focused on larger AS/ASA, which we assumed to be the most active and important company type.

Table 8.14¹⁰ describes the strata we defined, the total number of companies in that strata, the number of (randomly) selected companies, the number companies already included in the convenience sample, the number of legal entities with the same contact point, the number of companies for which contact information was not found, and the number of new companies which were added to the list. The companies in the "Non 72.2" categories are companies under the 72.00 group which are not part of the 72.20 sub-group.

Not surprisingly, the convenience sample contained primarily large AS/ASA. This shows that the AS/ASA category is clearly the most active and prominent company type in the market. As illustrated in Figure 8.3, it was also easier to find contact information for the larger companies.

¹⁰Note that all the companies with 10 or more employees in the 72.xx sector were included in the sample. 3 NUFs and 1 ENK were included as a consequence of this.

Company Type	Total	Selected	Already in the list	Duplicate legal entity	Contact info NA	Added to the list
AS/ASA 25+	186	186	135	13	2	36
AS/ASA 10-24	300	300	80	23	19	178
AS/ASA 5-9	460	100	13	0	14	73
AS/ASA 2-4	1072	52	5	0	11	36
AS/ASA 0-1	1981	52	1	0	35	16
AS/ASA unknown	360	50	0	0	36	14
ANS	480	100	0	0	75	25
DA	707	100	0	0	72	28
NUF	576	103	0	5	62	36
ENK	15152	1	0	0	0	1
Other	32	0	0	0	0	0
Non 72.2 <10	4441	100	0	0	60	40
Non 72.2 >=10	118	118	10	13	9	86
Total Sum	25854	1262	244	56	395	569

Table 8.14: Randomly selected companies

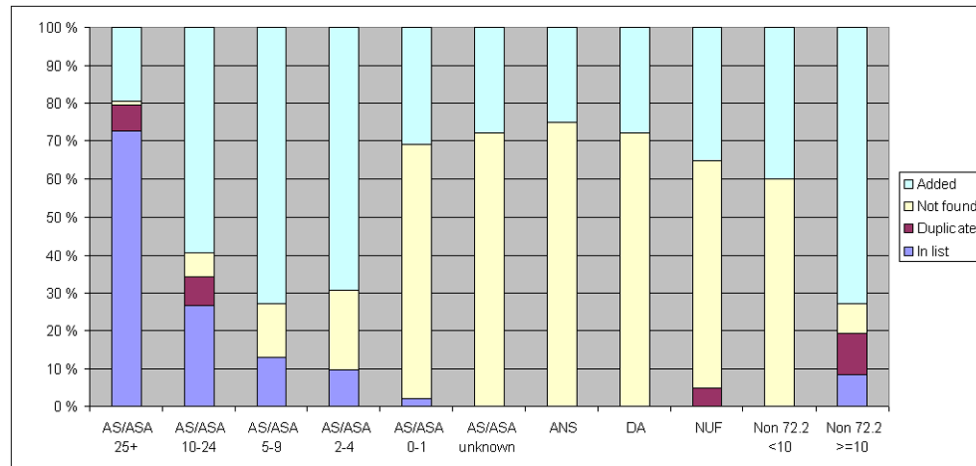


Figure 8.3: Selected companies and discovery rates

Step 4: Screening of Sample

In the end the list contained 1008 unique companies. To be certain about the size of the companies and to find out whether they were involved in OSS development

we contacted them. Most companies were contacted through e-mail, using the request below (in Norwegian). However, information about a small number of companies was also recorded through phone, personal, or previous contact. Close to 400 companies had already been contacted by Sørensen and Gereia in their work. Sørensen and Gereia did unfortunately not include the question related to running or participating in an OSS project. Therefore, we do not have data from some companies on this question.

Hi,

I am a researcher at NTNU, working with a survey on the use of open source in Norwegian ICT-companies.

We are now in the middle of a screening process and we hope that you are able to answer four questions:

1. How many employees do you have in Norway?
2. Are you involved in software development in Norway?
3. Do you use OSS in your products or services?
4. Do you run or participate in any OSS projects?

Best Regards,

Øyvind Hauge/Carl-Fredrik Sørensen

IDI, NTNU

mailto: <mail-address>

tlf: <phone-number>

This short request had two other advantages in addition to getting answers to our questions. By contacting the companies we were able to make our survey known to them before it was even initiated. Based on the discussion in the last section, this would increase the response rates. Secondly, we got a contact person within the companies. It is more likely to get a response to a personalized request rather than a request directed to a company contact point

59 of the 1008 companies we contacted were removed from the list of various reasons. 34 companies did not have a working email. 8 companies reported that they were closing down or had been acquired by other companies. 1 company responded that they did not want to participate in our screening. Another 16 companies with duplicate contact information were found. The remaining 949 companies constitute the basis of our further research.

558 of these 949 companies or 58.8% responded to our email request. Table 8.15 shows the response rates of the different categories. The response rates are fairly similar except from the ANS group. ANS are in most cases very small companies and we did only find contact information for about one forth of the randomly selected ANS. This indicates that this companies registered as ANS do not have high activity.

Company Type	Contacted	Responses	Response Rate
AS/ASA 100+	36	21	58.3 %
AS/ASA 25-99	128	81	63.3 %
AS/ASA 10-24	252	144	57.1 %
AS/ASA 5-9	118	80	67.8 %
AS/ASA 2-4	75	49	65.3 %
AS/ASA 0-1	31	21	67.7 %
AS/ASA unknown	14	9	64.3 %
ANS	26	5	19.2 %
DA	26	13	50.0 %
NUF	37	15	40.5 %
ENK	3	1	33.3 %
Non 72200 ≥ 10	129	77	59.7 %
Non 72200 < 10	74	42	56.8 %
Total Sum	949	558	58.8%

Table 8.15: Randomly selected companies

Their answers were coded and stored together with other company information. We were only interested in companies which used OSS components actively in their software development. Therefore, companies which answered that they developed software only to a very limited extent were coded as non-software developers. These were companies as hardware vendors, internet service providers, technical operators, graphical designers, project managers etc. Secondly, companies which stated that they only used OSS tools or which only used OSS components very rarely were coded as non-OSS users. Results from this screening process are discussed further in Chapter 11.

Step 5: Invitation to Participate

221 of the 558 companies answered that they were using OSS in their software development. This sample was divided in two. The division was done by sorting the companies by their number of employees and selecting every second company for each of the two groups.

One group was invited to participate in the survey by email using the Norwegian invitation letter in Appendix A. In the second group was first contacted by phone. Then, we sent them an invitation by email. We contacted the second group by phone because of two reasons. One, we believed this would increase the response rates, see Section 8.5. Two, we could like to test this assumption through the following hypothesis:

- **H1: Companies contacted by phone will have a higher response**

rate than companies contacted only through email.

The first group was invited by email the 8th and 9th of August. One of the 110 companies we contacted by email responded that he could not participate because he was too busy at the moment.

The second group of 95 companies was contacted by phone between the 8th and the 28th of August. This group was a bit smaller than the group contacted by email because of two reasons. One, 16 companies which have been participating in other recent studies by our research group were not contacted. This was avoided to reduce the stress on our contact persons in these companies. The majority of these 16 companies belonged in the group we contacted by phone. Two, some of companies which were contacted in the latest round of our screening process responded after we had divided the sample in two groups. These were primarily contacted by email.

We contacted the companies by phone between 10 to 12 and 13 to 15 either on cell phone or fixed connection. The phone numbers were gathered from the email responses or from the respondents' web sites. Contacting the companies by phone was time quite consuming. An average of at least 10 minutes was spent per phone call and several of the companies had to be contacted more than once. The results from in total three attempts to reach the contact persons are shown in Table 8.16. We see that about 50% of the people were unreachable when we tried to contact them by phone.

Status	1st contact	2nd contact	3rd contact	Total
Total contacted	95	43	19	95
Agreed to participate	47	23	6	76
Could not participate	2	1	2	5
Percentage of contacted companies reached	51.6%	55.8%	43.2%	85.3%
Cumulative percentage of reached companies	51.6%	76.8%	85.3%	85.3%

Table 8.16: Invitations

3 respondents were unavailable until September 3rd. These were only contacted once. Two of the five companies which could not participate were not particularly involved in OSS development. Two companies were involved in a merge and the respondents were too busy to participate at the moment. The last company which could not participate just did not want to. 13 of the 14 companies which were not reached by phone after the third attempt were invited by email on the 28th of August. The contact person in the last company had quit his job and started in a new company.

In total 204 companies were invited to participate. 123 companies were invited by email and another 81 by phone. The way a company was invited to participate

was recorded in a spreadsheet. 5 of the 81 companies invited by phone could not or did not want to participate.

8.6.3 Response Rates

The data collection ended late June for the ITEA smaple and the 12th of September for the Norwegian sample.

The ITEA sample got a total of 9 responses or poor 9.7%. Five respondents completed the survey after invitation and another four completed the survey after the reminder. Six of these completed the first part and three completed the second part of the survey.

71 of the 204 (36.3%) Norwegian companies invited to participate, completed the survey. However, 14 of these did not answer any of the two main parts. These 14 answered only the third part with demographic questions. This problem will be discussed further in Section 12.2.2. This leaves 57 (28.4%) completed and useful responses. 4 of these answered the first part and 53 answered the second part. 26 of these, answered the part about participation in and interaction with OSS projects.

Both the ITEA and the Norwegian sample had few responses for the first part. Because of this we will join data from the two samples when we perform data analysis from the first part. The 5 completed responses from the ITEA sample will be analysed together with the 4 completed responses from the Norwegian sample. In addition, data from 1 respondent who only completed Part 1 but neither Part 2 nor 3, will also be used. Another respondent completed the 11 first questions of Part 1. Data from this respondent will be compared to data from the other respondents.

The data analysis for the second part will be based on these 53 responses. However, data from the 3 ITEA responses and the 7 partially completed responses will be used to verify the results from the Norwegian sample. The respondents who partially complted the second part, completed the first 10 to 16 questions of Part 2.

8.7 Summary of the Resararch Design

The research questions described in earlier in this chapter were answered in a two phase manner. First, the literature study discovered evidence in literature. Second, we performed a survey studying software products using two different samples of companies. In the ITEA sample the survey was distributed to 93 Project and Work Package Leaders in the ITEA 2 Program. From the second sample, 204 Norwegian companies using OSS in their software development were invited to participate.

Chapter 9

Data Collection Using A Questionnaire

The development of the questionnaire was influenced by several factors; results and feedback from the study in [Hauge and Røsdal 2006], results from the literature study, and of course the research questions found in Chapter 8. This chapter presents the process of developing the questionnaire and some of the measures taken to improve the quality of the questionnaire and the validity of its results.

9.1 The Basis

The new questionnaire (Appendix B) is based on the questionnaire in [Hauge and Røsdal 2006]¹ but it has undergone several major and minor changes. Table 9.1 contains an overview of the most important changes. There were two main reasons behind these changes. Firstly, answering the questionnaire was too time-consuming. Secondly, new understanding of industrial OSS approaches has been found. Hence, the objective when developing the second questionnaire was to simplify the job of answering it, to incorporate new findings into the questionnaire, and to reflect the new research questions. Changing the questionnaire reduces the value of comparing the new results to the results from the previous study but the changes were necessary to maintain focus, and to increase the response rates and the validity of the results.

¹The original questionnaire is an appendix of the report which can be downloaded from <http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2006/hauge-fordyp06.pdf>

Change	Motivation
Removed questions related to development processes	Hard to describe development processes in a questionnaire, see Section 8.3 Change of focus
Moved the definitions from the introduction to relevant questions	Reduce the time spent from starting the questionnaire to answering the first question
Reduced the extent of the introduction	Reduce the time spent from starting the questionnaire to answering the first question
Removed questions not directly related to a research questions	Reduce response time
Removed questions not directly related to the unit of study	Reduce response time
Reduced the number of questions requiring text input	Reduce response time
Removed questions providing no interesting answers	Provide more interesting results
Added questions based on new findings	Provide more interesting results
Reorganized the questionnaire to include two instead of four main parts	Inner source software development was not a well established term Make it easier to respond
Focused on terminology and rephrased several questions	Make it easier to understand the questionnaire Ensure readability Reduce ambiguity and confusion
Focused on the invitation letter	Increase response rate

Table 9.1: Changes from the old questionnaire

9.2 Development

The content and the wording of the questions were reviewed several times during the development of the questionnaire. These reviews were performed by the author or by the author in cooperation with professor Reidar Conradi, dr Carl-Fredrik Sørensen and professor Ola Listhaug from the Department of Sociology and Political Science. Professor Listhaug has specialized in development of questionnaires. 3-4 quite extensive reviews were performed with several minor revisions in between.

Reviewing the questionnaire improved it in many ways. The use of language

and terms was made more consistent. The reviews also helped maintaining the focus of the questionnaire. Revisions and changes were recorded using a Word document. All changes were stored in a new version of the document and more than 30 versions of the document were created. Using such a versioning scheme kept track of the history of the questionnaire and it stored all questions including the discarded ones. Discarded questions may be used later.

9.2.1 Structure

The questionnaire was structured in two main parts and one part with questions related to demographic information. The first part concerns companies which develop OSS products or OSS providers. The study unit for Part 1 is an OSS product developed by the respondent's local business unit and the activities around this OSS product.

The second part concerns companies which uses OSS components as part of their software products. The study unit for this part is a software product developed by the respondent's local business unit and the activities related to selecting OSS components for this software product.

The second part also includes some questions related to interaction with and participation in OSS projects. The unit of study is the same software product and contributions from respondent's local business unit to one OSS project.

The last part contains questions related to the company and the respondent. Any respondent may answer one or both main parts.

9.2.2 Logic Flow

The questionnaire was as mentioned structured into two main parts. Both parts included some conditional questions which were only asked dependent on the answer to previous filter questions. Figure 9.1² illustrates the logical flow of the web-questionnaire.

9.2.3 Mapping to Research Questions

Questions can be divided into three categories; administrative, classification, and target questions [Cooper and Schindler 2006]. Administrative questions are used to identify the participant and other conditions. Classification questions cover sociological-demographic variables and filter questions. Target questions directly address the topics of the research questions.

Most questions in the questionnaire are either related to a research question (target questions) or gather important demographic information (classification questions). Part 3 gathers information about the respondent and the respondent's

²LBU is short for Local Business Unit

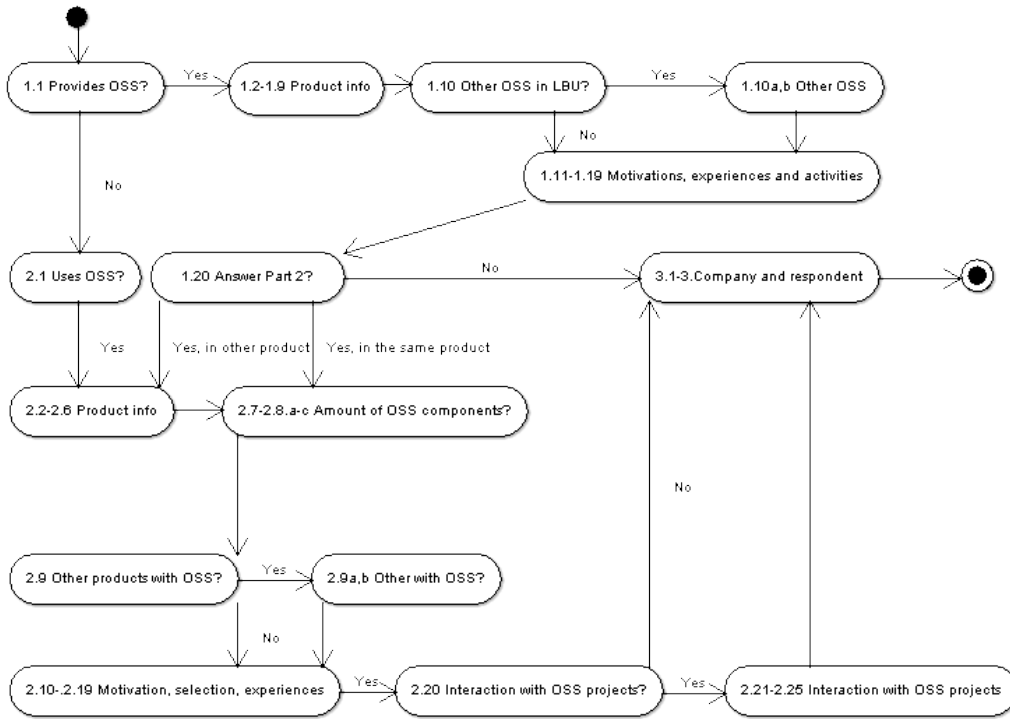


Figure 9.1: Logical flow of the questionnaire

local business unit. The two main parts (1 and 2) gather information primarily related to the research questions. Some demographic questions related to the unit of study are however asked. An overview of the questions in the two main parts and their relation to the research questions and other question types, is shown in Table 9.2 and 9.3.

9.3 Implementation

The questionnaire was implemented and distributed using a web-based survey tool. This section will briefly present the tool, some of the features it offers, and the question types used in the questionnaire.

9.3.1 The Web-Questionnaire

To implement and distribute the questionnaire we used the Confirmit³ web-survey system. Confirmit is a web-based tool made for designing and distributing questionnaires. Implementation of the questionnaire is fairly straight forward. However, there are some deviations from the Word-based questionnaire due to advan-

³More information can be found at <http://confirmit.com/>

Question Number	Question Type		
	Target	Administrative	Classification
1.1			X
1.2		X	
1.3			X
1.4			X
1.5			X
1.6	RQ3		
1.7	RQ3		
1.8			X
1.9	RQ3		
1.10			X
1.10a			X
1.10b			X
1.11	RQ1		
1.12	RQ3		
1.13	RQ3		
1.14	RQ3		
1.15	RQ3		
1.16	RQ3		
1.17	RQ3		
1.18	RQ3		
1.19	RQ3		
1.20			X

Table 9.2: Mapping of questions to research questions Part 1

tages and limitations in the web-tool. In addition to creation and distribution of questionnaires, the tool offers different export and reporting features.

9.3.2 Question Types

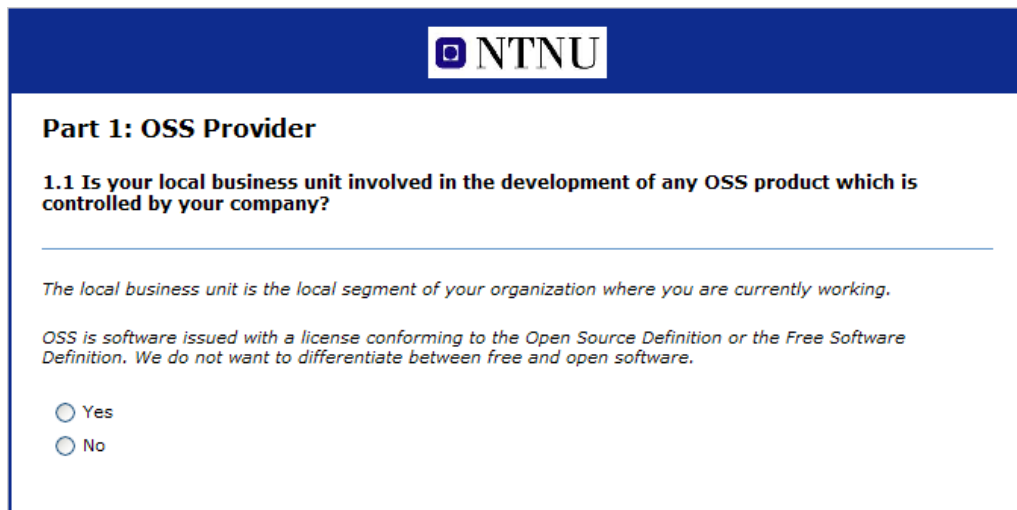
When designing a questionnaire, different questions seek different answers. Most questions were closed offering a range of pre-coded answers. Some questions allowed the respondent to pick one of two or more answers (radio button list), see Figure 9.2. This question type was used when the answers were mutually exclusive.

Two questions allowed the respondent to pick the year of an event. Instead of allowing the respondent to type the year we used a list of alternatives. These lists were chosen because of two reasons. Firstly, selecting an item from a list

Question Number	Question Type		
	Target	Administrative	Classification
2.1			X
2.2		X	
2.3			X
2.4			X
2.5			X
2.6			X
2.7	RQ4		
2.8	RQ4		
2.8a	RQ4		
2.8b	RQ4		
2.8c	RQ4		
2.9			X
2.9a			X
2.9b			X
2.10	RQ1		
2.11	RQ4		
2.12	RQ4		
2.13	RQ4		
2.14	RQ4		
2.15	RQ4		
2.16	RQ4		
2.17	RQ4		
2.18	RQ4		
2.19	RQ4		
2.20			X
2.21		X	
2.22	RQ5		
2.23	RQ1		
2.24	RQ5		
2.25	RQ5		

Table 9.3: Mapping of questions to research questions Part 2

is often faster than typing. Secondly, the answers were already coded with the year as a number. Allowing the respondent to type the year could for instance give the following answers 1990, 90, nineteen ninety, ninety, ninety hundred and ninety etc.



NTNU

Part 1: OSS Provider

1.1 Is your local business unit involved in the development of any OSS product which is controlled by your company?

The local business unit is the local segment of your organization where you are currently working.

OSS is software issued with a license conforming to the Open Source Definition or the Free Software Definition. We do not want to differentiate between free and open software.

Yes

No

Figure 9.2: A question with two options

The list of alternatives for these questions was quite long and would have resulted in a list of too many radio buttons. Therefore, a drop down list was used instead of the normal radio buttons, see Figure 9.3. For all other questions with lists, radio buttons were preferred rather than drop down lists. The use of drop down lists was reduced to a minimum because they have shown to increase the response time and increase the number of questions without a response [Healey 2007].



NTNU

1.6 When was the development of the OSS product initiated?

Please specify the year

Please select your answer

Please select your answer

NA/Don't know

2007

2006

2005

2004

2003

2002

2001

2000

<< >>

Figure 9.3: Question with answers from a drop down list

When the answers were not mutually exclusive, the respondents were allowed to check none, one, or more alternatives using a checkbox list, see Figure 9.4.

The scales for these types of questions could either be nominal (mutually exclusive

NTNU

1.8 Which license types are used for the OSS?

A non-protective license (for instance BSD) applies no restrictions on the distribution of derivate works. A protective license (for instance GPL) does apply such restrictions.

Mark one or more alternatives

Don't know

A protective/viral/reciprocal/copyleft license

A non-protective/academic/non-copyleft license

Proprietary license

Other (please specify)

Figure 9.4: A question where one or more answers may be checked


groups), ordinal (ordered), interval (intervals of equal size), or ratio (with origin) [Cooper and Schindler 2006]. The checklists provide nominal information while the multiple choice questions may also use ordinal or ratio scale. Position in a company (Manager, Developer etc) is clearly on the nominal scale. Age is clearly on the ratio scale. Researchers sometimes disagree whether data is on the ordinal or the interval scale and they thereby treat them differently [Cooper and Schindler 2006]. Lickert scales is often treated as being on the interval scale [Cooper and Schindler 2006; 338] and we treated them as interval scales when performing data analysis.

Other questions required text input, see Figure 9.5. However, these questions were kept to a minimum to reduce the time spent responding to the questionnaire.

Perhaps the most important type of questions was the grid questions. These grids allow a respondent to state how much they agree to a given statement using a Lickert like scale. These questions were also used to show to what extent the respondent performed an activity or to show how true a statement was, see Figure 9.6.

Most of these grid questions incorporated a 5-point Lickert like scale. 5-point scales were chose because of two main reasons. One, they are widely used. Two, 5-point scales were used in the previous study and in other related studies. By using the same scale it is easier to compare the results with these studies.

While designing the questionnaire, we focused on providing a complete set of possible answers which were non-overlapping. However, it is sometimes hard to include all possibilities. Furthermore, the respondent may be unable to respond in some cases. To aid the problem we included '*Don't know/NA*' and '*Other with textual input*' where it was needed. By having such categories in closed questions




Select one OSS product provided by your company, where your local business unit is involved in the development of the product.

Answer all the questions in this part based on your experiences with this OSS product.

We are primarily interested in OSS products which have an ongoing development and at least one running release. Nevertheless, OSS, where the development is discontinued, is also of interest.

1.2 What is the name of the OSS product?

Figure 9.5: A free text question



1.19 How true are the following statements related to how you have tried to attract a community?

If there are other important activities which have helped you attracting a community, please indicate these at the last row.

Mark one alternative per row

	NA/Don't know	Totally false	Mostly false	Neither true nor false	Mostly true	Totally true
A community already existed for the software before it was made open source	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We have attracted community members through advertisements and publications in media and in scientific literature	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We have been very active in other OSS communities	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We have focused on the quality of the software and its documentation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We listen to our community and we allow them to influence us	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We are engaged in partnership with community members	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
We have focused on providing the necessary infrastructure to the community	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other (please specify)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 9.6: Grid with several questions and several answers

the respondent is not forced to answer something which he feels is not correct. The drawback of including this kind of category is that several respondents may

answer 'don't know' because it is easier than making up their minds.

9.4 Quality Ensuring Measures

The quality of the survey tool was ensured through different means. The questionnaire was based on previously performed studies. Questions which provided useful insight were reused from the studies in [Hauge and Røsdal 2006] and [Li et al. 2006b]. These questions had already been tested and quality assured.

Furthermore, the questionnaire was based on a comprehensive literature study including several publications from many researchers. By reviewing the literature we were able to use known terminology and to focus on topics seen as interesting elsewhere.

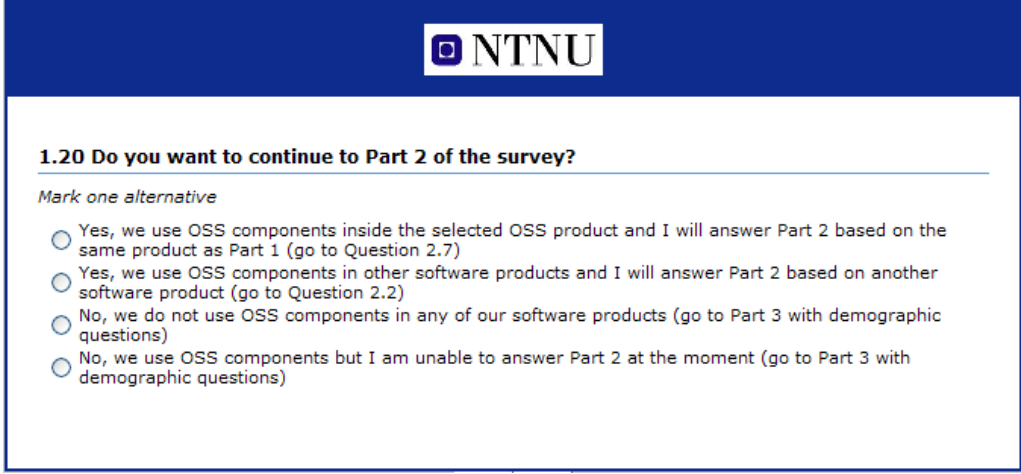
When developing the questionnaire we performed internal reviews of the questionnaire. By having several people reviewing the questionnaire we have hopefully removed any bias and ambiguity in the question wording.

In addition to these internal reviews, a two stage pre-test have been performed. Firstly, three master students read through the questionnaire, gave suggestions to improvements, and commented shortcomings and errors. Through this pre-test we found that the questionnaire was too long. As a consequence of this we were forced to prioritize which questions to be included. Following, all questions related to **RQ2 'Which OSS development practices are used when industry develops software?'** had to be excluded from the questionnaire. This was a hard but important decision to make.

Next, the improvements were included and the second pre-test was undertaken. This pre-test was performed by four former colleagues. They are all working with OSS related development in different companies. Feedback from these developers, lead to minor changes, some rephrasing and more importantly a clearer distinction between the two parts. Answering both the main parts of the questionnaire was seen as quite time consuming. Respondents who had answered the first part (OSS provider) were therefore given the possibility to skip Part 2 even if they used OSS components inside their product, see Figure 9.7.

By giving the respondent the opportunity to skip Part 2 some interesting results may be lost. However, giving the respondent a more pleasant and less time consuming experience was seen as more important. It is better, to allow the user to decide himself rather than forcing the user to answer. A respondent which is forced to answer a series of questions would most likely not consider his answers very closely. Hence, the results would be of lower value.

[Cooper and Schindler 2006] recommend starting easy and with the interesting questions first to get the respondent interested and not to frighten him. The survey was structured into two parts and both parts started with questions about the product. These were fairly easy to answer and it feels like a natural start.



1.20 Do you want to continue to Part 2 of the survey?

Mark one alternative

- Yes, we use OSS components inside the selected OSS product and I will answer Part 2 based on the same product as Part 1 (go to Question 2.7)
- Yes, we use OSS components in other software products and I will answer Part 2 based on another software product (go to Question 2.2)
- No, we do not use OSS components in any of our software products (go to Part 3 with demographic questions)
- No, we use OSS components but I am unable to answer Part 2 at the moment (go to Part 3 with demographic questions)

Figure 9.7: End of Part 1 of the questionnaire

The perhaps least interesting part (demographic information) is put at the end of the questionnaire. We ask for the respondents email address to send him results. When the demographic questions are put at the end of the questionnaire the respondent knows what he is putting his name on.

9.5 Increasing the Response Rate

All respondents in the first sample are affiliated to ITEA. This affiliation and the high-level support from ITEA and COSI should contribute positively to the response rate.

The use of incentives in form of a gift or through lotteries was also discussed but we decided not to initiate such measures. This decision was taken based on two things. One, the incentive would only be a small reward to one (well paid and busy) person answering on the behalf of his local business unit. Two, the reward could contribute to bias the respondents by increasing the number of respondents interested in the gift rather than the study.

To respondents' motivation could also be increased by showing them how the results of the study can be useful to them [Kitchenham and Pfleeger 2002]. We believe that these results are far more valuable to the respondents than a gift. Giving the respondents access to these results will increase the response rates far more than any small gift.

The companies we invited from the Norwegian had responded to our primary inquiry. This gave us a contact person which was personally involved in the survey. Furthermore, we tried to personalize the invitation to the survey by using the name of the respondent and the name of the company. A sample of respondents was also contacted by phone to get them even more involved in the

survey.

We tried to use the fact that several hundred other Norwegian companies had responded, thus giving the respondents a feeling of being part of a group. Next, we tried to show that open source is important to the Norwegian software industry and thereby showing the respondent the importance of our work.

9.6 Summary

This chapter has presented the development of the survey tool used to collect data in this thesis. This tool was developed through several rounds of reviews and it was implemented using a web-tool. This chapter also discusses the measures taken to increase the reliability of the tool like the reviews and pilot tests. The final questionnaire is available as Appendix B.

Chapter 10

Data Analysis

The first step of the data analysis is to describe the data using descriptive statistics. These statistics provide an overview of the data by characterizing fundamental features of the sample. For some variables we performed statistical tests. This chapter discusses and presents important properties of the data analysis in this study.

10.1 Re-Coding of Scales

We used Likert type scales for several of the questions in the questionnaire. It is quite common to do arithmetical calculations like calculating mean and standard deviation on such scales i.e. [Li et al. 2005, Rossi and Bonaccorsi 2005]. Treating the Likert scale as an interval scale is possible because people tend to treat categories in Likert scales as equidistant [Spector 1980]. Furthermore, a precise scale type should not limit the use of statistics as long as the results are used with the underlying approximation in mind [Tukey 1961]. When applying statistics it is important to remember where the data is taken from, what kind of data it is, and any assumptions which have been made.

The Likert scales used in this survey were re-coded to the interval values from 1 to 5, as shown in Table 10.1. This allowed us to calculate and compare means and standard deviations from the different variables. Not applicable (NA) was not included in these statistics.

We then divide the calculated means into three groups:

Code	Scale 1	Scale 2	Scale 3
1	Not important at all	Totally false	None at all
2	Unimportant	Mostly false	Small
3	Neither important or unimportant	Neither true nor false	Some
4	Important	Mostly true	Large
5	Very important	Totally true	Very large

Table 10.1: Re-Coding of Likert scales

Interval	Scale
1.00-2.33	Low/False
2.34-3.66	Medium/Undecided
3.67-5.00	High/True

10.2 Statistics

Basic descriptive statistics will be used to describe properties of the distributions of the variables included in this survey. Statistics as mean, mode, median, standard deviation, variance, and different plots will be used to describe and analyse the data gathered through this survey. We will not discuss these statistics further. In addition to these descriptive statistics, we will perform some simple parametric tests to investigate if there are any statistically significant differences between two or more strata or samples. These tests include the t-test and analysis of variance (ANOVA).

10.2.1 T-test

The t-test is a parametric statistics used to test whether two normally distributed populations are equal by using their means, standard deviations, and number of data points. Normally the null hypothesis $H_0 : \bar{X}_1 = \bar{X}_2$ is tested with a defined alpha-level (α), using a one or two-tailed test. The null hypothesis is rejected if there is a significant difference between the two samples. The formulas from [Walpole et al. 2007] for calculating the t-values are given in Table 10.2.

While calculating the t-test we will use t-test where the variances are assumed to be unequal because this is more conservative than assuming equal variance. Tools in Excel and SPSS will be used to calculate the t-values. In addition to the t-test we will in some cases calculate the effect size or "Cohen's d" to say something about the significance of the difference. The effect size should at least be larger than 0.7 but preferably closer or greater than 1.0. Cohen's $d = \frac{mean_1 - mean_2}{\sqrt{(SD_1^2 + SD_2^2)/2}}$

$n_1 = n_2$	$n_1 \neq n_2$ and $\sigma_1 = \sigma_2$	$n_1 \neq n_2$ and $\sigma_1 \neq \sigma_2$
$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2 + S_2^2}{n}}}$	$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} \left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$	$T' = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$ has an approximate t-distribution with approximate degrees of freedom $v = \frac{(S_1^2/n_1 + S_2^2/n_2)}{\frac{(S_1^2/n_1)^2}{n_1 - 1} + \frac{(S_2^2/n_2)^2}{n_2 - 1}}$

Table 10.2: Calculation of the t-values

10.2.2 ANOVA

Analysis of variance (ANOVA) is a collection of statistical models and methods frequently used to test if more than two populations have the same means. Normally the null hypothesis $H_0 : \bar{X}_1 = \bar{X}_2 \dots = \bar{X}_n$ is tested with a set α and rejected if there is a significant difference between at least two of the samples. The statistical foundation for ANOVA will not be further discussed here. Tools in Excel and SPSS will be used to perform ANOVA statistics.

10.3 Tools and Data Formats

To generate descriptive statistics and to perform statistical tests we used the reporting tools provided by Conformat and tools integrated into MS Excel and SPSS. The Conformat tool is able to produce (simple) reports and to export data in MS Excel, SPSS, and comma/tab-separated formats. This allows the data to be used in a wide variety of tools.

Data from the screening was stored using an MS Excel spreadsheet. We used MS Excel to perform most of the analysis on both the data from the screening and the main survey. Even though Excel is not a statistical program it contains several statistical tools like PivotTables, graph generators etc. These are sufficient for most basic statistical analysis. However, SPSS in many cases a more convenient tool and SPSS were preferred when comparing several strata or many variables.

Part IV

Results and Conclusions

Chapter 11

Results

In the previous part we have seen the research method and the construction of a web-based survey. The purpose of this questionnaire was to investigate how and why the software intensive industry is involved in OSS development. In this chapter, we present the results from this survey and provide answers to the research questions. We will start by looking at the impact phone invitations have on response rates. Secondly, we look at the results from the screening process related to the use of OSS in the Norwegian sample. At last, we look at the results from the survey itself.

11.1 Response Rates: Phone vs Email Invitation

To increase the response rates we invited about half of the Norwegian sample by phone. We assumed that inviting respondents by phone would increase the response rate and we wanted to test this as a hypothesis.

- **H1: Companies contacted by phone will have a higher response rate than companies contacted only through email.**

To test this hypothesis we defined the null hypothesis $H1_0$ and the alternative hypothesis $H1_1$. The probability of getting a response from a company invited by email is p_e and the probability of a response from a company invited by phone is p_p .

- $H1_0 : p_e = p_p$
- $H1_1 : p_e \neq p_p$

The estimated probability of getting a response from a company from a given group, \hat{p} , can be calculated by dividing the number of responses on the number of invitations sent to that group. We will use the number of companies which completed the survey to test this hypothesis. Even though 14 of these companies

only completed the last part, we still believe it is correct to include all 71 respondents in the evaluation of the hypothesis. This is discussed further in Section 12.2.2.

The respondents were asked to give the name of their company in Question 3.1. 69 of the 71 respondents who completed the survey gave the name of their company. This information was used to estimate the probability of getting a response from a company invited by email and phone. Of the 110 companies invited by email, 26 responded. Furthermore, 39 of the 81 companies invited by phone responded. 14 of the 95 we tried to invite by phone were not reached. 13 of these were invited by email and 4 of these responded. 2 respondents did not give their company name so it is unclear whether they were invited by phone or email. The number of companies, the number of responses, and \hat{p} are illustrated in Table 11.1. The first row is for the companies invited by email. The second row is for the companies we reached by phone. The third row is for the companies we did not reach by phone but rather invited by email. The fourth row includes the companies we intended to invite by phone. We will use the scenario from the fourth row in our calculations because these will give the most conservative numbers. The last row is for the two respondents which did not state their company name.

Group	Invited	Reponses	\hat{p}
Email (Sample 1)	110	26	0.24
Phone reached	81	39	0.48
Phone not reached, invited by email	13	4	0.31
Phone intended (Sample 2)	95	39	0.41
Unknown	N/A	2	N/A

Table 11.1: Number of responses from companies invited by email and phone

To test if the hypothesis $H1_0 : p_e = p_p$ we use a binomial distribution. We calculate the probability of getting 39 or more responses from a group of 95 given the following:

1. We draw $n=95$ companies.
2. A company either completes the survey or not.
3. $\hat{p} = 0.24$ is the probability of getting a response from a randomly selected company. This probability is constant.
4. Each of the companies, are independent of each others.

These assumptions give us the following calculation:

$$\begin{aligned}
 &P(X = 39, n = 95 | \hat{p} = 0.24) \\
 &= 1 - B(38; 95, 0.24) \\
 &= 1 - \sum_{x=0}^{38} b(x; 95, 0.24) \\
 &= 1 - 0.99982 \\
 &= 0.00018
 \end{aligned}$$

The probability of getting 39 responses from 95 companies with $p=0.24$ is only 0.018%. We can therefore reject H_{10} and conclude that there is a difference between the two probabilities p_e and p_p . Even if the two respondents who did not give the name of their company were invited by email it would not influence the rejection of H_{10} . This conclusion supports H_1 and that contacting the respondents by phone will increase the response rates.

11.2 Use of OSS in the Norwegian Software Industry

The use of OSS components in the Norwegian software industry is widespread. 558 companies responded in our screening process and 472 of them said they were involved in software development. 221 of these 472 companies (46.8%), use OSS as part of their products or solutions. Furthermore, we see that companies with more than 100 employees use OSS to a very large extent. On the other hand, very small companies are not using OSS that frequently. Another important finding is that consultant companies use OSS to a greater extent than companies developing software.

In the screening process the companies were asked whether they used OSS. The results from this screening process showed that 46.8% of the companies in the Norwegian sample use OSS components in their development. These results are further broken down in Table 11.2. We see that the use of OSS components in software development is fairly evenly distributed in most of the groups with some exceptions.

Most of the large companies use OSS in their software development. The large companies are involved in several markets and projects, and they are therefore more likely to be involved in OSS related development. They have more customers, employees, and most likely projects of different nature.

The opposite argument can be used about the small companies. The groups "AS/ASA 2-4", "AS/ASA 0-1", "AS/ASA unknown" have use OSS less frequently than the other groups. This can also be explained by the increased investment (relative to size) it is for a small company to start using new technology. Small companies have fewer customers and most likely a more uniform project or product portfolio. They are also less likely to hire new employees with OSS experience.

The use of OSS seemed to be fairly evenly distributed among the other strata we tested. However, one interesting thing to notice is that only 38 of the 109 companies or 34.9% registered under sector "72.21 Publishing of software" use OSS components. Whereas, 140 of the 270 companies or 52.9% registered under sector "72.22 Other software consultancy and supply" use OSS in their software development. We see that companies which deliver software use less OSS than companies which deliver solutions and billable hours. It is clearly easier for a company selling hours and solutions (not software licenses) to adapt to the new business opportunities offered by OSS. It is perhaps not that easy to adopt this

for companies making money selling software licenses.

Company Type	Responses	Sw. dev.	OSS in dev.	OSS in dev. rate
AS/ASA 100+	21	19	16	84.2%
AS/ASA 25-99	81	78	35	44.9%
AS/ASA 10-24	144	128	58	45.3%
AS/ASA 5-9	80	65	30	46.2%
AS/ASA 2-4	49	40	16	40.0%
AS/ASA 0-1	21	18	5	27.8%
AS/ASA unknown	9	6	1	16.7%
ANS	5	4	2	50.0%
DA	13	10	9	90.0%
NUF	15	10	6	60.0%
ENK	1	1	0	0.0%
Non 72200 ≥ 10	77	58	26	44.8%
Non 72200 < 10	42	35	17	48.6%
Total Sum	558	472	221	46.8 %

Table 11.2: Use of OSS components in software development

More than half of the companies were also asked if they participated in, or provided OSS projects. The companies participating in the screening performed by Sørensen and Gereaa were unfortunately not asked this question. We do therefore not have data for all the companies in the sample. However, 130 of the companies using OSS in their software development answered this question. 39 of them said they participate in OSS projects. Some of the companies commented that they returned some bug fixes and others that some employees were participating on their own time. The participation did not seem to be extensive nor an important part of their business. We would estimate that perhaps 30% of the companies using OSS in their development participate in some of the OSS projects they use. This is further supported by Question 2.8c. 36.7% of the companies answered that they had reported changes for at least a few OSS components back to the community.

15 companies or roughly 10% of the 130 companies answered that they have published their own OSS products. However, it seemed to be an important part of the business for only two or three of these 15 companies. Both participation in OSS projects and distribution of commercial OSS products happened less frequently than expected.

11.3 Motivations for Approaching OSS Development

In Chapter 3 we discussed possible motivations for approaching OSS development. To investigate why companies approach OSS development, we focused on one or more research questions on this. We asked the respondents why they integrate OSS components into their products, why they interact with and participate in OSS projects, and why they distribute their own software as OSS. Results to these questions are provided below.

11.3.1 Integrating OSS Components

Companies developing software are primarily motivated by the practical benefits of having OSS components, their source code, and information about these components available at no cost. Furthermore, they OSS integrators value the standard compliance of many OSS components. Motivations which are not related directly to the benefit of using OSS are not important. Idealism, the market, political reasons, and the customer did rarely motivate the use of OSS components. The low importance of the customer indicates that the customers do not care or are not aware of the consequences of using OSS.

The mean¹ in Table 11.3 is calculated by taking the average of the respondents' ratings. See Section 10.1 for a brief discussion of how this is done. Then we divided the motivations into three groups (high, medium, low), also described in Section 10.1

The results were fairly equal across most of the strata we tested. The data from the seven partially completed responses were close to equal to the responses in Table 11.3. None of the means or variances was changed by more than 0.01 when data from these partially completed responses were included.

9 respondents mentioned other motivations as important for using OSS components. These other motivations are listed below and we see that functionality is mentioned several times.

- The OSS components provide the functionality required
- The OSS components provide unique functionality
- The OSS component is the best available product
- No bureaucratic involved in licensing/purchasing process
- Community collaboration

¹ \bar{x} is the mean and s is the sample variance

	Motivation	\bar{x}	s
High	Component could be acquired at no cost	4.15	0.77
	OSS components are easily available for test and use	4.08	0.99
	Source code is available and can easily be changed	3.92	0.94
	Compliance to standards	3.91	1.04
	Information about OSS components is highly available	3.87	0.99
Medium	Maintenance costs with OSS are lower (than with other software)	3.57	0.84
	Our company possesses knowledge about OSS	3.47	1.03
	Honesty and openness from the OSS provider about the true status of the component	3.46	0.96
	To become more independent from software vendors	3.40	1.23
	We want to increase our knowledge in the OSS area	3.23	1.01
	Reduced risk of component provider going out of business	3.12	1.06
	Reduced risk of selected component evolving into an unwanted direction	3.06	0.96
	Idealism	2.62	1.19
	The market is looking for OSS	2.54	1.11
	Political reasons (company policy, licensing conditions)	2.41	1.22
Low	To use OSS components was decided by the customer	1.56	0.82

Table 11.3: Question 2.10 Motivations for using OSS components in SW development

11.3.2 Interaction with and Participation in OSS Projects

Companies are primarily motivated by their need for the software and the learning effect from participation, when interacting with and participating in OSS projects. Branding and publicity is perhaps surprisingly not stated as important for participation, see Table 11.4.

Some companies provide commercial services related to an OSS product and the motivation "We provide commercial services related to the OSS" was divided into two groups, one, small group of companies which provide commercial services to OSS product and one larger group which does not.

It is also interesting to notice that idealism is ranked a bit higher here than as a motivation for integrating OSS components. Furthermore, the companies participating in OSS project rank idealism higher on Question 2.10 than other companies. The mean for the companies participating in OSS projects is 2.96 compared to 2.27 for the ones which do not. There is a statistically significant difference between the two groups with a t-value of 2.27 and a fairly small effect size of 0.6. Even though the effect size is small this difference could indicate that

	Motivation	\bar{x}	s
High	The OSS is used inside the company or in the company's software/products	4.07	0.68
	Acquire new knowledge by learning from participation in the OSS project	3.81	1.00
Medium	Improve relationship to the OSS project	3.27	0.83
	Influence the OSS project	3.22	0.80
	Idealism	3.07	1.30
	We provide commercial services related to the OSS	2.44	1.56
Low	The company wants to be identified as an OSS company	2.33	1.18
	Increased publicity	1.89	1.01

Table 11.4: Question 2.23 Motivations for interacting with or participating in OSS Projects

idealism is a factor influencing the decision to participate in an OSS project or not.

11.3.3 Releasing OSS Licensed Products

Companies which distribute their software as OSS seem to be mainly motivated by the opportunity to attract more users/customers to their product. Getting different contributions from the community seems to matter but it is not highly important.

There were differences between the ITEA and the Norwegian sample. The ITEA sample valued code contributions, defects reports, and new requirements from the community a lot higher than the Norwegian sample. The number of responses was low and one or two responses may influence the mean and sample variance quite a lot. From Table 11.5 we see that only one of the motivations stood out. The other motivations were rated similarly and ended up in the "medium" category. With more respondents we would most likely have been able to see trends more clearly.

11.4 OSS Integrators

We have seen that almost 50% of the software developing companies in Norway integrate OSS components into their products. In this section we will look closer at to what extent these companies use OSS, if they benefit from the availability of the source code, how they find and evaluate OSS components, and to what extent they participate in OSS projects outside the company.

	Motivation	\bar{x}	s
High	Attract more users/customers to the software	3.92	1.31
Medium	Increase the number of new requirements and ideas from the community	3.50	1.45
	Allow supplementary extensions, products and services from other providers	3.42	1.38
	Code contributions from the community (other than defect corrections)	3.25	1.29
	Increase attention and publicity	3.25	1.36
	User-to-user support form the community (self supporting community)	3.08	1.00
	Sell related services (integration, deployment, support, quality assurance, packaging, distribution, user training)	3.08	1.31
	More defect reports from the community	2.92	1.24
	More defect corrections from the community	2.83	1.40
	Return value to the OSS community	2.83	1.47
	Idealism	2.82	1.60
	Enable you to sell proprietary add-on software	2.58	1.31
	Enable integration of other OSS into your software	2.50	1.35
	Sell proprietary licenses to customers (dual licensing)	2.42	1.31

Table 11.5: Question 2.23 Motivations for providing OSS products

11.4.1 The Share of OSS in Software Development

Most of the respondents have been involved in the development of quite few products containing OSS the last year. Of the 53 companies, 19 have not been involved in the development of just one product containing OSS components the last year (Question 2.9). While 21 companies answered that they have integrated OSS components into 1 or 2 more products. One company has been involved in the development of more than 50 such products during the last year. These products vary in size, ranging from small products with less than one person-month to large products with more 100 person-months during the last year. Figure 11.1 provides an overview of the development effort spent on the different products the last year.

In these products, OSS components provide some functionality but perhaps not a lot of end user functionality. 38 of the respondents (71.7%) answered that OSS components provided less than 40% of the functionality of the product.

When looking at the functionality provided by OSS components against the effort in person-months, it is interesting to notice that only 1 of the 18 (5.6%) large products, has more than 60% functionality provided by OSS components. While 6 of the 32 (18.8%) small products have more than 60% functionality provided

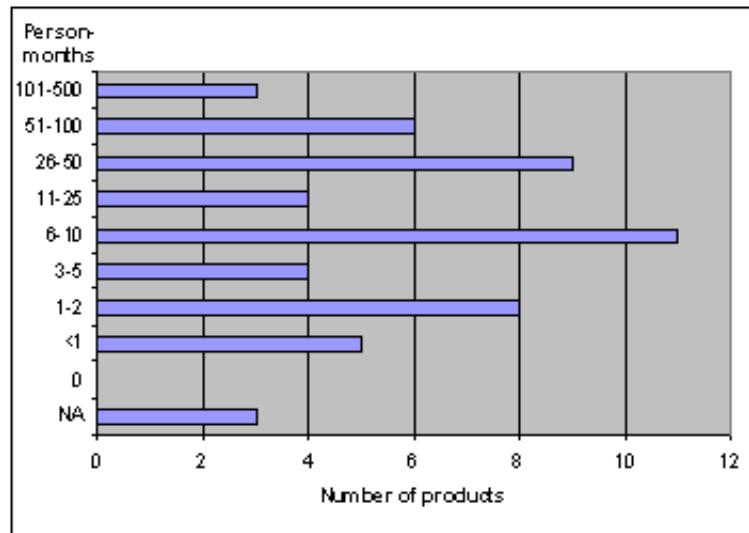


Figure 11.1: Question 2.6 Development effort in person-months

by OSS components. A large product is defined as a product with a development effort of more than 25 person-months during the last year and a small product is a product where the development effort is 25 or less person-months. Figure 11.2 show an overview of the percentage of functionality provided by OSS.

The number of data points here is too low to draw any statistically significant conclusions. However, this observation is supported by the partially completed responses where 2 of 7 small products have more than 81 % of their functionality provided by OSS components.

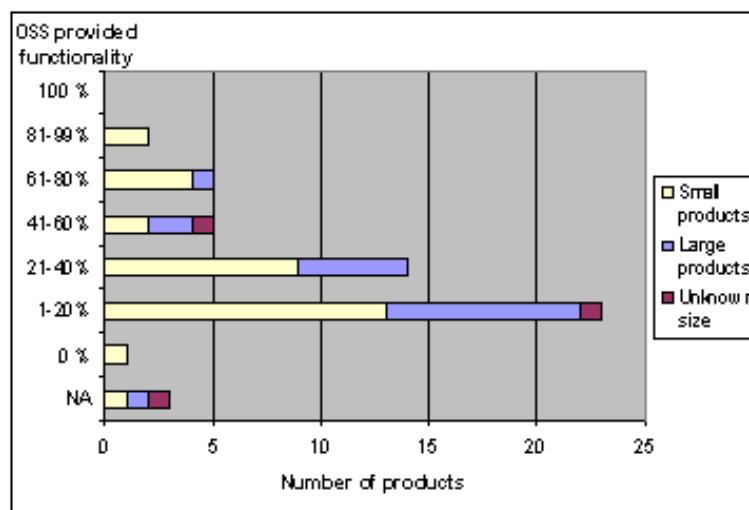


Figure 11.2: Question 2.7 Functionality provided by OSS components

The number of OSS components used in a product is most commonly 5 or below. 66.0% of the respondents answered that their product contained 5 or less components. However, 26.4% answered that their product contained 6 or more OSS components, see Figure 11.3. We did not observe any correlations to neither the size of the company nor the development effort used to develop the products. Not surprisingly, we got very similar but slightly higher results for the number of selections per product (Question 2.11). 62.2% responded that they had performed 5 or less selections and 30.2% responded that they had performed more than 5 selections. 4 respondents answered "don't know" on both questions.

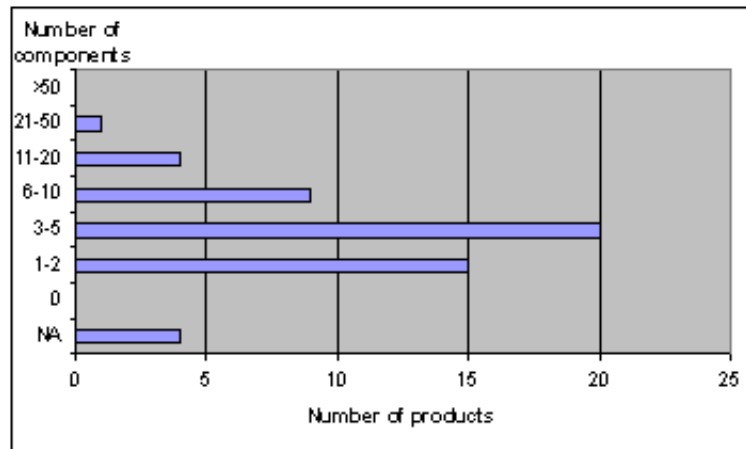


Figure 11.3: Question 2.8 Number of OSS components in products

14 of the respondents answered "n/a" on how much of the turnover which comes from OSS related services. However, 19 of the respondents who answered said that their turnover from OSS related services is below 20% of the company's total turnover. Were on the other hand, 13 respondents answered that more than 60% of their turnover came from OSS related services.

OSS components clearly have a visible place in the software intensive industry. However, the amount of projects containing OSS, the number of OSS component, the amount of functionality provided by OSS, and the turnover from OSS related services are so far not dominating the industry. Some companies have extensive use of OSS but in most cases, OSS is combined with other products and services.

11.4.2 The Use of OSS Components

OSS components are not treated as total black box components. Figure 11.5 shows that parts of the code are read and changed in most cases. Minor to some parts of the code are read and very few parts of the code are changed. Even though this use is not extensive, developers benefit from being able to study and change the source code. Only 6 of the 49 (12.2%) respondents who answered

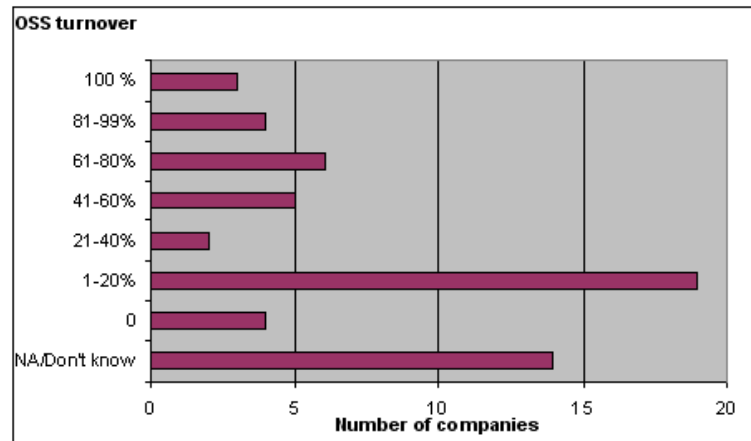


Figure 11.4: Question 3.7 Turnover from OSS activities

Question 2.8a/b said that they did not read any code at all. While 14 (28.6%) answered that they did not change any parts of the code.

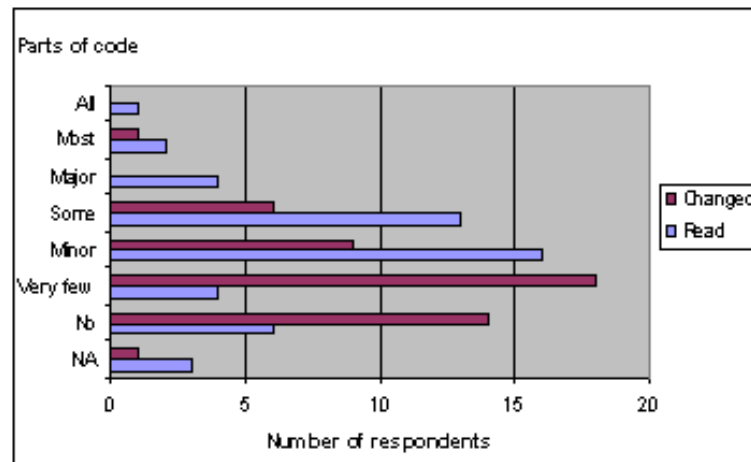


Figure 11.5: Question 2.8 a/b Parts of code read/changed

The observation that developers benefit from the availability of the source code is further supported by the answers to Question 2.8c, see Figure 11.6. 24 respondents (49.0%) answered that they have fixed a defect for at least a few OSS components. 18 (36.7%) said they have reported changes for at least a few OSS components back to the community. Even though OSS components are left untouched to a large extent, software developers clearly benefit from the availability the source code.

OSS integrators are satisfied with their use of OSS components. They believe OSS reduce the total lifetime cost of the software and they have (or admit) very few problems developing with OSS components. Problems with licenses, adoption

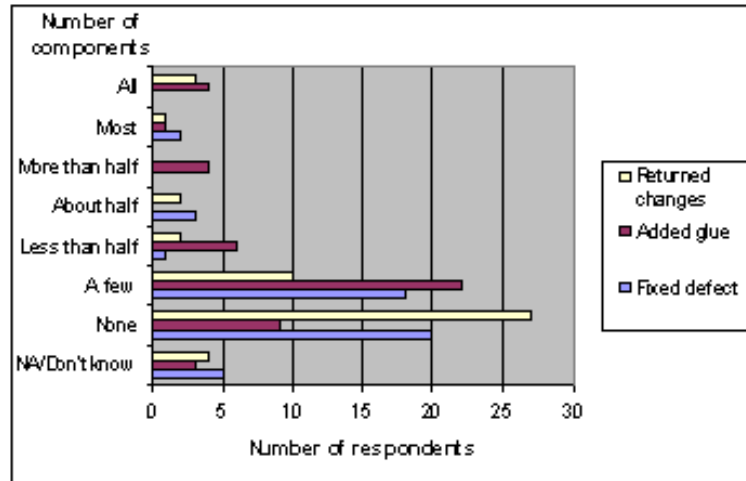


Figure 11.6: Question 2.8c Number of fixed defects, added glue, returned changes

of OSS components or any other matter seems to be very small. This lack of problems can be caused by at least three things. One, the respondents do not admit their problems. Two, the companies' selection processes are good enough. They are able to avoid the bad components and communities and they thereby avoid any problems. Three, they have the knowledge to repair or avoid possible problems.

11.4.3 Selection of OSS Components

The effort spent on selection of OSS components the last year is mainly distributed between 11 and 200 person-hours for most of the products. The effort spent for the large products is somewhat larger than the effort spent on the small products, see Figure 11.7. There seemed to be a weak correlation between the number of components and the selection effort as well, meaning the more components you want to include, the more time you have to spend on selection.

OSS components are primarily found through searches on OSS portals, by using search engines, and by previous experience with the component. Table 11.7 shows an overview of the activities the respondents were asked to rate. We see that none of them get a high mean score, meaning that none of these activities were used to a large extent by all the respondents. No other activities were mentioned as important in the search for OSS components.

After finding components it is necessary to evaluate them. Testing and prototyping seemed to be quite common in most companies. Searching for experiences with the component and reviewing documentation is also common. The evaluation process seems to be quite informal. Almost no companies have a documented selection process and use documented checklists and only few document the choice of OSS component and the rationale behind this choice.

	Statement	\bar{x}	s
True	The total lifetime cost (including maintenance) of the software was reduced due to use of OSS components	3.98	1.06
Undecided	It is difficult to influence the evolution of OSS components	2.57	0.97
	Requirements to your software product were changed a lot (by the customer)	2.41	1.19
	It was difficult to identify whether defects were inside or outside the OSS components	2.30	0.97
False	It was difficult to upgrade the software with the latest version of the OSS component	2.20	1.04
	Integration and testing of OSS component took longer than estimated/expected	2.16	1.19
	Information about the reputation and technical support ability of the community providing the OSS components were inadequate or not available	2.02	0.91
	Community or provider of OSS components did not provide enough technical support/ training	2.00	0.85
	It was difficult to plan and perform maintenance of your software product due to properties of the OSS components	1.92	0.99
	Selection of OSS component took longer than estimated/expected	1.81	1.12
	The software product was delivered long after schedule or had severe delays	1.80	1.13
	Local adaptation of OSS components resulted in high maintenance cost	1.71	0.98
	OSS components negatively affected quality attributes of the software (security, performance, etc.)	1.63	0.80
	OSS licenses restrict us from using the components the way we want	1.49	0.78
	OSS components were not satisfactorily compatible with the production environment when the software product was deployed	1.48	0.87
	OSS components could not be sufficiently adapted to changing requirements from the customer	1.42	0.69
	Use of OSS components lead to legal conflicts	1.36	0.64

Table 11.6: Questions 2.17-2.19 Experiences using OSS components

There are several criteria involved in the evaluation process. Properties of the component itself are the most important evaluation criteria. Most of the criteria

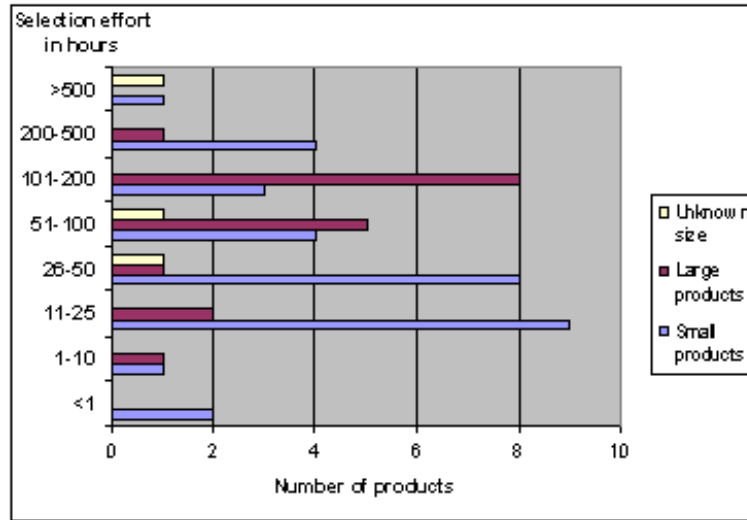


Figure 11.7: Question 2.12 Selection effort in hours

	Activity	\bar{x}	s
Medium	Searched OSS portals (SourceForge.net, tigris.org, apache.org, eclipse.org, etc.)	3.60	0.95
	Used search engines (Google, Msn search, etc.) to search for individual OSS	3.57	1.12
	Selected OSS components based on previous experience	3.52	1.16
	Used search engines (Google, Msn search, etc.) to search for comparisons of several OSS components	3.17	0.98
	Asked friends and colleagues etc. whether they know any OSS candidate components	3.15	1.10
	Requested advice at forums and mailing lists	2.52	1.22
Low	Reviewed books and magazines	2.23	1.08
	Used a company internal "knowledge base"	1.98	1.26
	Used an (external) component broker to find the component for you	1.22	0.76

Table 11.7: Question 2.13 Search activities

related to the component were rated quite highly. When a developer evaluates an OSS component he wants to find a component with functionality which covers and extends his needs. This component should have a stable release, licensed under a suitable license, and it should be implemented in "the right" programming language. If the component is licensed under a too restrictive license or implemented in an unsuitable programming language the component is most likely rejected. Table 11.9 contains an overview of the evaluation criteria related to the component and their scores.

	Activity	\bar{x}	s
High	Performed testing and/or prototyping with OSS components	3.81	1.16
Medium	Looked for references and/or other experiences with the OSS components	3.42	0.96
	Reviewed documentation for OSS components	3.35	1.05
	Assessed the activity within the community around the OSS components	3.08	1.32
	Estimated how much effort you would use on selection and integration of the OSS component and include this estimate into the project plan	2.88	1.21
	Defined a list of requirements to the OSS component before the selection started	2.71	1.27
	Performed architecture reviews of OSS components	2.67	1.16
	Used shortlists (identify several OSS components with similar functionality)	2.48	1.08
	Documented the choice of OSS component and the rationale behind this choice	2.40	1.05
Low	Performed code reviews of OSS components	2.02	1.01
	Used documented checklists to evaluate the OSS components	1.76	1.08
	Used a selection process which is well documented in the company	1.50	1.86

Table 11.8: Question 2.14 Evaluation activities

Properties of the community do not seem to be as important as the properties of the component itself. However, "Momentum of development and deployment" and "future development plans" were mentioned in Question 2.15 as important evaluation criteria. These criteria are more related to the community than the component. Nevertheless, roadmaps and future plans are not mentioned as the most important evaluation criteria related to the community. On the contrary, the reputation of the community and the component and the quality of the documentation are the most important evaluation criteria. If a community with a good reputation delivers reputed software it is clearly easier to use this software.

On the other hand, having a commercial actor guaranteeing the quality of the OSS product was not important, see Table 11.10. This is most likely because the respondents have knowledge about software development. The companies are thereby able to correct any problems themselves.

	Criteria	\bar{x}	s
High	Programming language/environment	4.27	0.85
	Stable releases of the component	4.23	0.68
	Compliance to requirements	4.19	0.86
	Extra functionality/features	4.18	0.65
	License/license type	4.14	0.81
	Integration with other software	4.04	0.74
	Performance	4.02	0.83
	Code quality	3.96	0.85
	Security	3.90	0.76
	Modularity and clear interfaces	3.80	1.04
Medium	Few dependencies to platforms, other components, standards	3.60	0.85
	Architecture	3.56	0.99
	Experience with component	3.56	0.96

Table 11.9: Question 2.15 Evaluation criteria - product

	Criteria	\bar{x}	s
High	Reputation of the OSS component	4.02	0.80
	Quality and availability of documentation	3.83	0.68
	Reputation of the OSS component provider/community	3.69	0.94
Medium	Size of user base/Number of downloads of the OSS component	3.58	0.92
	Activity on mailing lists and in forums	3.54	0.94
	Quality and availability of defect and feature trackers	3.42	0.76
	End user support from the community or OSS provider	3.31	1.05
	Release frequency of the OSS component	3.23	0.94
	Quality and availability of roadmaps and future plans for the OSS component	3.15	0.89
	Response time on requests in forums and on mailing lists	3.04	0.99
	Quality of OSS provider web site	3.04	0.99
Low	Size of developer team developing the OSS component	3.00	0.86
	A commercial actor guaranteeing the product quality of the OSS component	2.12	0.86

Table 11.10: Question 2.16 Evaluation criteria - community

11.4.4 Participation in OSS Projects

Answers to Question 2.22 show that participation in OSS projects is done at an individual level. One of the 26 respondents who completed the part about participation in OSS projects said that participation was organized as an internal project. Two respondents said it was organized through the line organization and the remaining respondents answered that it was done at an individual level.

Companies primarily participate on the mailing lists and in the forums and contribute with defect reports, new requirements and testing. The participation does not seem to be a critical part of any of the respondents' business. We saw that "the need for the OSS product" was among the most important motivation for participation in OSS projects and contributions to such projects are most likely related to their own needs. If a company wants a new feature they request it, if they have a problem with a product they fix it or ask someone on the mailing list for help.

	Contribution	\bar{x}	s
Medium	Participate on mailing lists and forums	3.07	0.87
	Defect-reports	2.92	0.98
	New requirements to the OSS	2.62	0.80
	Testing	2.48	0.94
Low	Defect-fixes	2,29	1,16
	New features (code)	2.19	0.94
	Local adaptations of the OSS	1,92	0,93
	Supplementary or add-on OSS	1,90	1,18
	Architecture or design	1.85	0.88
	Distribution of software	1,58	1,10
	Documentation	1,56	0,82
	Financial support	1,46	0,93
	Integration towards other software (glue ware)	1,45	0,67
	Free support	1,42	0,70
	Commercial supplementary or add-on software	1,30	0,57
	Infrastructure (e.g. servers or storage space)	1,29	0,69
	Commercial support	1,28	0,89

Table 11.11: Questions 2.24 Contributions to OSS projects

Even though the companies' participation in OSS communities seems to be moderate it gives the communities contact with a wider variety of users/developers. These users/developers are able to provide feedback and perhaps more importantly they make the product more known by using it. The contributions from companies are therefor an important part of OSS communities.

Furthermore, the companies are overall very satisfied with the communities they participate in and their software. Most of the properties in Table 11.12 have a high or medium to high rating. The companies are most satisfied with the functionality of the software and its source code. As they are developers this is not a surprise and this satisfaction is probably one of the reasons why they participate in a specific OSS project and use its software.

	Property	\bar{x}	s
High	Functionality of the OSS	4.19	0.48
	Code quality	3.91	0.67
	Security in the OSS	3.75	0.90
	Other non-functional quality attributes of the OSS	3.74	0.65
Medium	Architecture of the OSS	3.64	0.86
	Documentation (availability and quality)	3.50	0.71
	Support (in case of problems)	3.44	1.04
	Response to defect-reports	3.38	0.82
	Response time to requests on mailing lists and forums	3.36	0.91
	Your ability to influence the OSS project	3.27	0.70
	Response to new requirements/feature requests	3.14	0.77

Table 11.12: Questions 2.25 Satisfaction with OSS project

11.5 Experiences Providing OSS Products

Attracting more users and customers to their product was the most important motivation for releasing an OSS product and it seems as they are succeeding to some extent. Table 11.13 contains an overview of some of the experiences the respondents have releasing their products as OSS products. Increased attention and publicity, and code contributions are two of the benefits from releasing a product as OSS.

Two other things are also worth noticing. First, releasing software as OSS does not make it hard to focus on the customers. Even though the software is freely available to everyone, the customers will still need services as related to the product. Secondly, the release of the software as OSS does not make it particularly hard to plan releases. This problem would perhaps have been larger if the companies were more depending on community contributions.

11.5.1 Community Contributions

Code contributions in form of defect corrections and new functionality, is as mentioned above one of the advantages of releasing an OSS product. However, code is

	Statement	\bar{x}	s
True	You got increased attention and publicity	3.89	0.60
	Feedback and requests from the community lead to extra work	3.80	0.63
	Contributions from the community are often included into the software	3.78	1.20
	Contributions from several community members increase the effort to keep a sound architecture, design, and implementation	3.67	1.22
Undecided	You improved your relationship to the OSS community	3.60	0.97
	The openness (of the code) makes it easier to find security holes	3.44	1.01
	Having this software as OSS product enable you to sell related software or services	3.38	1.30
	Code provided by the community has too low quality to be included directly into the software	3.22	0.97
	You got increased number of paying customers (licenses or services)	3.11	1.05
	It is more difficult to protect intellectual property and/or business secrets	3.00	1.58
	It is difficult to accept code contributions from the community due to licensing and intellectual property rights	2.88	1.55
	Other OSS have been integrated into your OSS product	2.71	1.70
	Selling licenses to the software is difficult because the OSS is freely available	2.50	1.60
False	It is hard to focus on one group of customers because the OSS is available to everyone	2.33	0.71
	It is more difficult to plan releases because of constant input from the community	2.33	0.87

Table 11.13: Question 1.12 Experiences providing OSS products

not the most important community contribution. The community provides new requirements to the product, test the software, and provide defect reports. The contributions from the communities around the different OSS products seem to be limited. Table 11.14 lists these contributions and to what extent the community provides them.

There were some differences between the two samples. The communities in the ITEA sample provided more add-ons, support, and input on architecture and design than the Norwegian communities. This may be cause by the fact that the ITEA products seemed to be somewhat better established.

	Contribution	\bar{x}	s
High	Provide new requirements	3.80	1.32
Medium	Test the software	3.50	1.18
	Report defects	3.40	0.97
	Fix defects	2.60	1.07
	Implement new functionality	2.60	1.43
	Develop supplementary or add-on OSS	2.50	0.97
Low	Provide local adaptations of the product	2.30	1.06
	Provide free support to community members	2.20	1.03
	Develop commercial supplementary or add-on software	2.00	1.25
	Develop architecture or design	1.90	1.10
	Create and maintain documentation	1.70	0.82
	Provide commercial services (integration, deployment, support etc.)	1.67	0.71

Table 11.14: Question 1.16 Community contributions

11.5.2 Side-effects of Releasing an OSS Products

The respondents claim that releasing the products as OSS have increased the attention and publicity. Even though, as we will see below, the products have not attracted vast number of users they may have attracted some media attention and some customers and users. Getting the attention of customers and users is a good thing, as code contributions and feedback may improve the product. However, attention generates feedback and requests, and answering these requests leads to extra work. This extra work was one of the most important side-effects of releasing an OSS product, see Table 11.13. Contributions from the community are supposed to reduce the effort to develop the software. However, code contributions can also increase the architectural drift and make it harder to maintain a clean and well structured design.

A few differences between the two samples were noticed. The most important one is the problems related to accepting code from the community. The Norwegian sample answered that accepting code from the community could sometimes be a problem because of licensing. The ITEA sample on the other hand did not see this as a problem. The number of respondents is too low to say anything conclusive about this and the difference could be leveled out with more respondents.

In addition to these side-effects it is important to remember that it is necessary to provide services to attract and sustain the attention from a community. However, from Table 11.15 it seems as the respondents have primarily focused on their own product and not as much on other activities to attract a community. We noticed one difference between the two samples. The ITEA sample had engaged in partnership with community members to a far greater extent than the Norwegian

sample.

	Statement	\bar{x}	s
High	We have focused on the quality of the software and its documentation	3.73	1.35
Medium	We listen to our community and we allow them to influence us	3.45	0.80
	We have focused on providing the necessary infrastructure to the community	3.45	0.82
	We are engaged in partnership with community members	3.36	1.43
	We have attracted community members through advertisements and publications in media and in scientific literature	2.60	1.51
Low	A community already existed for the software before it was made open source	2.10	1.29
	We have been very active in other OSS communities	1.82	1.08

Table 11.15: Question 1.19 Attracting a community

In addition to attracting community members it is important to provide them the necessary infrastructure to participate and contribute. Table 11.16 and Figure 11.8 provide an overview of the services and artifacts the respondents offer their communities. We were a bit surprised that not all of the companies provided mailing lists and code repositories to their community. The extent of services provided to the community was generally lower than expected. This could be related to the size of the communities and their age. We will see that most of the products are recently released as OSS and that the communities around these products are fairly small.

11.5.3 Product and Community Statistics

The communities around these products are as mentioned relatively small. Figure 11.9 shows the number of active community members outside the companies. These community members post messages on the forums and contribute with code. These numbers are quite small, except for one of the products. However, it seems a bit strange that as many as 5000 people have contributed code to the product.

The number of downloads of the products is also relatively low. 1 respondent said that their product was downloaded between 1001 and 10000, another said it was downloaded between 10001 and 100000, 4 respondents said they did not know, and the rest answered that their product was less than 1000 times (Question 1.15). Most of the products have not succeeded at attracting a large community, except perhaps the product with more than 10 000 downloads. These numbers

	Statement	\bar{x}	s
Undecided	You provide up-to-date installation guides and tutorials	3.64	1.03
	You provide an updated web site with news about the software	3.55	0.93
	Internal developers use public mailing-lists or forums to discuss important issues related to the software	3.55	1.29
	You provide up-to-date roadmaps	3.45	1.04
	You encourage community members to provide feedback on features, specifications, design, architecture etc	3.45	1.04
	You provide up-to-date developer documentation	3.36	1.12
	Everyone in the community is free to contribute to the development of the product (requirements, patches, proposals etc.)	2.82	0.98
	Employees in your company respond to all requests in forums and mailing lists	2.78	1.30

Table 11.16: Question 1.17 Services provided to the community

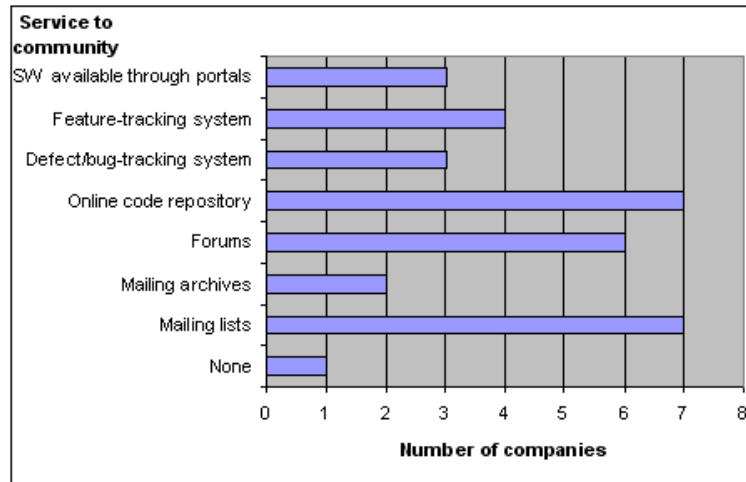


Figure 11.8: Question 1.18 Services provided the community

and the number of community members show that the communities have limited activity.

This limited activity could be caused by the relatively short life-time of these communities. The development of most of the products started between 2004 and 2006 (Question 1.6). However, more than half of them were released as OSS the following year or later (Question 1.7). If this is because the companies want to release a "finished" product or if it is because they did not plan to release it as OSS in the first place is hard to say.

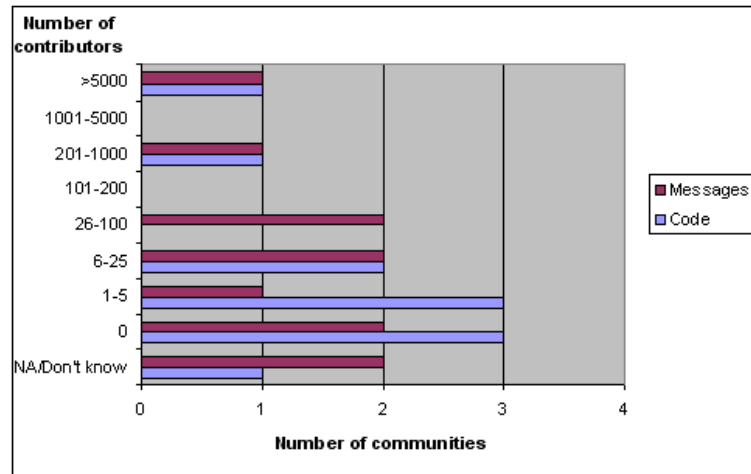


Figure 11.9: Question 1.13-14 Number of community contributors

The effort spent by the companies developing these products is noticeable for small companies but not really large. Between 3 and 25, person-months have been spent during the last year for most products. However, more than four person-years have been spent on one of the products, see Figure 11.10.

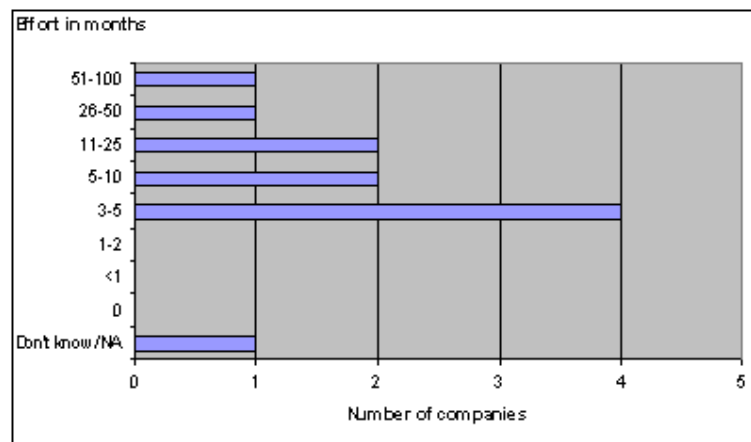


Figure 11.10: Question 1.9 Development effort in person-months

11.6 Demographic Data

11.6.1 The Companies

Two of the companies in the Norwegian sample did not state their name and we were therefore unable to find their company type. Of the companies which gave

their name, 50 were AS/ASAs, 3 were DAs, and 2 were NUFs. Most of the respondents came from the 72.22 sector but other sectors are also represented, see Table 11.17. The companies answered that they were software consulting companies, software houses, or other software intensive companies. In the last category the respondents mention primarily software/hardware solutions for different niches and web services.

Sector	Companies
32.10 Manufacture of electronic valves and tubes and other electronic components	1
51.874 Wholesale of machines and equipment	1
51.84 Wholesale of computers and accessories	1
70.12 Buying and selling of own real estate	1
72.21 Publishing of software (software houses/vendors)	5
72.22 Other software consultancy and supply (consultancy)	40
72.30 Data processing	1
72.40 Database activities	4
74.209 Other technical consultancy	1
Unknown	2

Table 11.17: The number of companies from the different industry sectors

The distribution of the number of employees from the Norwegian sample is shown in Figure 11.11. By combining the company names and the data from the screening we are able to find more exact data for the number of employees in the Norwegian companies. These data give us average number of employees of 20.9 a mode of 14 and a median of 16.5 employees.

The ITEA sample is represented with companies come from different business sectors. These companies have not surprisingly a lot more employees than the Norwegian companies. However, the percentage of turnover from OSS is lower than in the Norwegian sample.

11.6.2 The Respondent

The respondents are primarily males (1 female) between 30 and 50 holding an IT related Bachelor or Master Degree. They have a wide variety of different positions in their companies but IT managers and Project managers are by far the most common positions. The respondents have from very short to very long experience in the field. Figure 11.12 shows the respondents' years of experiences with OSS, with software development, and their current business unit.

Furthermore, the respondents have participated in none to more than 50 development projects where they have integrated OSS components into the product.

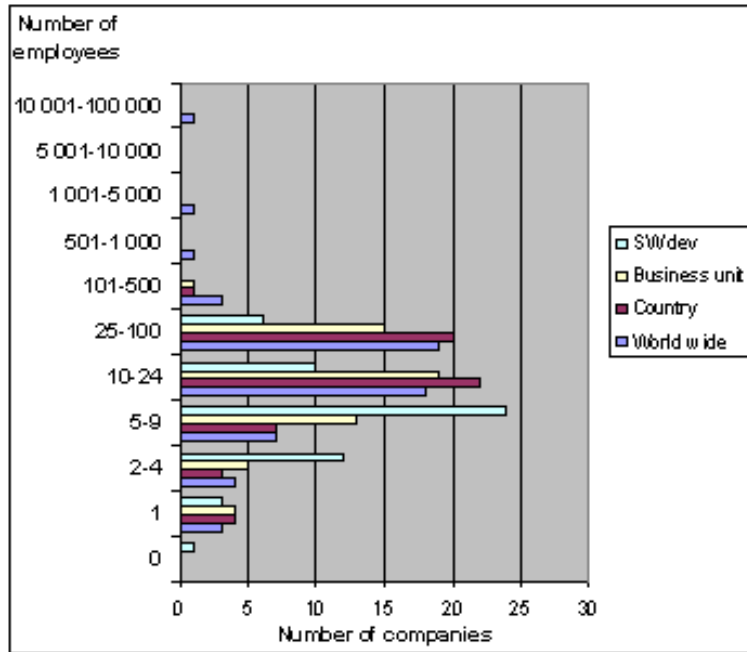


Figure 11.11: Question 3.5 Number of employees

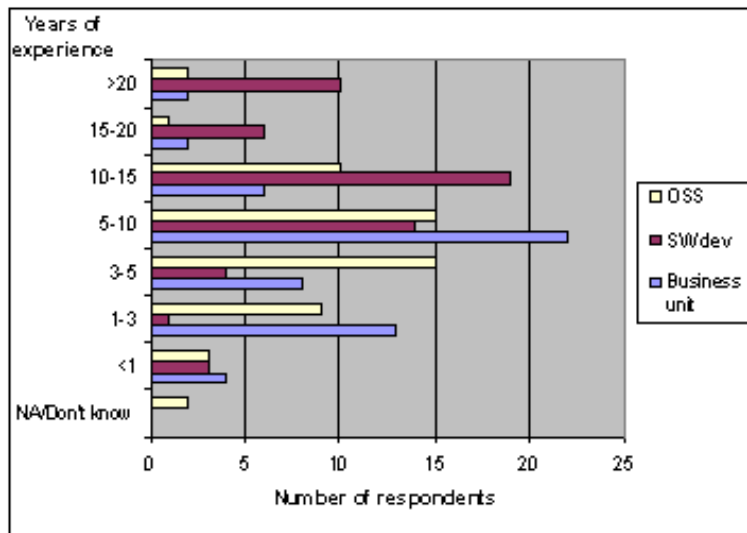


Figure 11.12: Question 3.11 Years of experience

Figure 11.13 shows the distribution of the respondents' participation in projects using OSS. The figure also shows how many OSS projects (outside the company) the respondents have been involved in, ranging from none to more than 25 OSS projects.

The respondents from the ITEA sample have similar characteristic as the Nor-

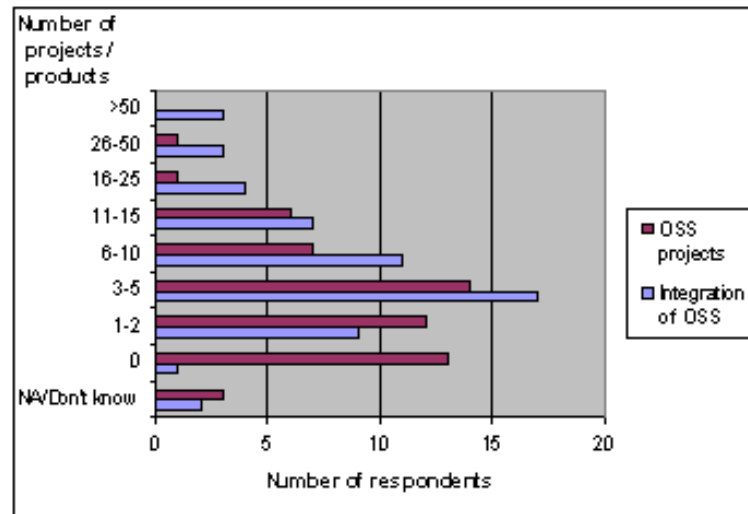


Figure 11.13: Question 3.12 Involvement in integration of OSS and OSS communities

wegian respondents except from experience with OSS. The respondents from the ITEA sample had been involved in fewer OSS projects and projects where OSS was integrated into the product.

Chapter 12

Discussions and Implications

The current study has gathered new answers to all but one of our research questions. Questions related to Research Question 2 had to be excluded from the questionnaire to reduce its extent. The rationale behind this decision is further described in Chapter 9. Table 12.1 provides an overview of where to find results to the different research questions.

Objective	Status
H1	The hypothesis was tested and supported, see Section 11.1.
RQ1	Results provided in Section 11.3.
RQ2	Results provided only from the literature study.
RQ3	Results available in Section 11.5.
RQ4	Results to this question are found in Section 11.4 and 11.2.
RQ5	Results are found in Section 11.4.4 and 11.2

Table 12.1: Answers to research questions

This chapter start by discussing the significance and implications of some of these results. The last part of the chapter will discuss reliability and threats to validity.

12.1 Implications

This section discusses some of the implications of our results and gives some advice to both practitioners and researchers. The survey we performed is a fairly large scale survey and we start by discussing some of our experiences performing this survey.

12.1.1 Research Method

The evaluation of H1 in Section 11.1 supported our hypothesis. We can therefore recommend contacting the respondents by phone to increase the response rates. However, calling close to 100 contact persons is quite time consuming, especially when a lot of people are unavailable. We spent an average of approximately 10 minutes per call. In each contact session, about 50% of the people we tried to contact were unavailable, see Table 8.16. Similar results are achieved in American household studies [Groves et al. 2004; 171]. Given that this success rate is stable at 50 %, 175 (100+50+25) phone calls would be needed to reach 87 of 100 contact persons using three attempts. If an average of 10 minutes is used per phone call it would be necessary to use at least 29 hours to contact those 87 contact persons.

Regardless of increasing the response rates or not, contacting the companies by phone had other advantages. We were able to increase our understanding of the Norwegian software intensive industry through phone contact with representatives from this industry. Even though the conversations were short we got a somewhat better impression of what the companies were doing. Two, we were able to respond to possible questions the respondents may have related to the survey or other issues. Three, we were able to make our work better known in industry and show that we are interested in the use of OSS in the industry. Some companies were interested in future cooperation on the topic.

Our first email enquiry was sent to about 1000 companies. It was a very brief request but as a consequence of being brief, it was easy to answer. Even though, getting the email addresses to all the companies was quite time consuming it was fairly easy to send them the request by email. We got a response rate of close to 60 % without sending any reminders. The enquiry enabled us to say something about the use of OSS in software development in the Norwegian software intensive industry. Secondly, it gave us contacts which knew we were about to launch a survey about OSS. Thirdly, the relatively cheap screening process allowed us to use more resources on the companies which actually use OSS in their software development.

Even though we were quite satisfied with this design it had some drawbacks. One, in larger companies it was in some cases the Chief Executive Officer, Chief Information Officer, Chief Marketing Officer, or some other high ranking officer who replied to our request. When these were contacted by phone they were positive. However, we got the impression that they had little time to answer our survey and some cases also limited insight in the actual software development in the company. The advantage of having a high ranking contact person is that he or she can have someone else answering the questionnaire. Two, some companies did not have a working email address or contact form on their web-site. Especially large, multi national companies did not have electronic contact information on their web-sites. Some companies (primarily small ones) did not have a working web-site or a working email address.

Close to 400 of the 949 companies were contacted in March by Sørensen and Gereaa. The remaining companies were contacted in July. When contacting the sample of 95 companies by phone we observed the following: The contacts who were contacted in March had in most cases forgotten our inquiry. While, most of the ones contacted in July remembered it and showed interest in our work. We do not have any data supporting this observation but it indicates that it is important to do follow up studies in a reasonable short amount of time after the first inquiry.

These experiences can be summarized into some recommendations for researchers performing survey research where many companies are involved:

- Use a fast and simple screening process to reach your focus group, to get contact persons, and to make the contact persons aware of your forthcoming survey.
- Do not allow the time between the screening and the main survey to be too large.
- If possible, invite the sample by phone to increase response rates and to get more information.

12.1.2 A Need for More Research on Industrial OSS

Close to 50 % of the software intensive industry uses OSS in its software development. This extensive use of OSS must influence both developers and customers. How use of OSS in industry affects it is still fairly unknown. The literature study uncovered some evidence of industrial involvement in OSS development. However, there is not much published empirical evidence. The evidence which exists is dominated of reports from a handful of different OSS projects and most often case studies or personal experiences. Other authors have also reached similar conclusions ie [Gurbani et al. 2005, Christley and Madey 2007, Østerlie 2007].

The research community needs to continue its research to be able to understand how OSS affects the software industry and its customers, and how the industry can benefit from OSS. New knowledge must be shared with the industry.

OSS is also an imprecise term covering everything from one-developer-projects to big multi-developer-projects with heavy industrial participation. This confusion or lack of consciousness about what OSS is or is not, is heavily influencing existing literature. It is important that we increase our awareness of the diversity of OSS and OSS development and stop treating OSS as if it was a homogeneous phenomenon.

12.1.3 The Use of OSS Development Practices

To reduce the extent of the questionnaire we were forced to remove questions related to Research Question 2 from the questionnaire. Therefore, the survey did

not uncover any new evidence related the use of OSS development practices in industry.

However, some observations were made through the literature study. One, because of the variety in the world of OSS it is difficult to clearly define a set of OSS development practices. Two, the number of industrial participants in OSS development is increasing and the industry is already benefiting from lessons learned in the OSS communities. Three, companies are also more and more involved in global and multi-company co operations. It is therefore difficult to talk about any clear distinction between traditional "software engineering" and "OSS development". These two camps have started to blend and they will most continue likely learn from each other's best practices.

12.1.4 The Amount of OSS in Software Products

71.7% answered that OSS provided less than 40% of the functionality of their product. This is perhaps a bit less than expected. Nevertheless, we still believe that functionality is a better measure of the amount of OSS in a product than for instance lines of code. It is easier to count lines of code than it is to estimate functionality. However, one large OSS component may for instance constitute 90% of the code base even though only a small fraction of its functionality is used. While the code developed by the company only constitutes i.e. 1% of the code base it may provide all the end-user functionality.

There are two possible problems with using functionality. One, it is hard to estimate functionality exactly. Two, the functionality of an OSS component may be under estimated. We believe that under estimating the value of a component is more likely than over estimating it. A developer is not as deeply involved in the code of a component as he is in his own code. The effort used developing own code is expected to be greater than the effort of using a component. The weight of own code is therefore emphasized more than the component. Furthermore, OSS components provide in many cases frameworks which are quite large i.e. Hibernate, Spring, Struts. Even though these frameworks provide an important infrastructure they do not provide a lot of end user functionality and it may be easy to overlook the value of these components.

Our results implied that OSS components provide less functionality in products which require a large amount of development effort. If this is verified it could be explained by the lack of available OSS components or increased need of control over the code base. First, if there does not exist any appropriate (OSS) components the developer is either forced (or chooses) to develop the needed functionality himself. This would of course increase the development effort. Second, if the product is large and complex, it is perhaps necessary to increase the control over the code base. This means using less components developed elsewhere and thus a reduced amount of functionality provided by OSS components.

12.1.5 High Quality OSS

OSS is quality software. All OSS products are of course not quality products but many OSS products are quality products. The respondents reported very few problems using and integrating OSS into their products. The lack of reported problems could be explained by lowered expectations to something which is free (OSS) or resistance to admitting problems. However, we strongly believe that many companies are successfully integrating OSS components into their products with few or no problems at all. In addition to the lack of problems, the respondents agree that the use of OSS reduces the total lifetime cost of the software product.

Furthermore, the companies do seldom use glue code to wrap the OSS components. This means that they interface directly towards the component and they have to live with any changes in the OSS components. We see this as a sign that they the software industry trusts the OSS components they use.

12.1.6 Releasing OSS Products is No Free Lunch

We saw that releasing OSS products gave some benefits to the provider. The provider gets increased attention and he may get contributions from the community. However, releasing OSS has some side-effects as well. One, it is necessary to provide an infrastructure to support the community. Two, requests and feedback from the community generates extra work. Three, code contributions increase the effort needed to maintain a clean and structured architecture and design. We do not wish to discourage anyone from releasing their software as OSS but it is important to be aware of the requirements to an OSS provider.

12.2 Validity and Reliability

Validity is related to how representative the results are for the population of interest [Wohlin et al. 2000]. It is always important to assess the quality of both a study and its results. This section will discuss possible validity threats and some of the measures taken to increase this validity.

12.2.1 The Literature Study

Even though the literature study cannot be claimed to be neither totally complete nor totally objective, we are fairly confident that it has covered relevant literature. It has been an ongoing process which started the spring 2006 and it has covered several hundred articles from sources like journals, workshops, conferences, and web sites.

The only possible way to increase the reliability of the literature study is to use rigor procedures for systematic literature reviews, i.e. [Kitchenham 2004]. Such a systematic review would have needed the participation from several people and it would have needed a lot more time than available in this thesis work.

12.2.2 Instrument Construction

There are several common human factors which influence the responses to a questionnaire. It is hard to control these factors. Human memory, knowledge, experience, motivation, personality, and the fact that people may want to be seen in a good light are possible threats to internal validity with questionnaires [Robson 2003].

All these issues could be influenced by the questionnaire itself. Measuring something else than what you want is a possible problem with questionnaires [Jacobsen 2005]. The conclusion validity may be influenced by poor question wording and bad questions [Wohlin et al. 2000]. If conclusions are drawn based on questions which are misunderstood they may be erroneous. Whether the questionnaire measured and collected the data we wanted is related to the construct validity.

The best ways of ensuring construct validity is through testing and review [Robson 2003]. Our questionnaire has undergone several reviews which involved experienced researchers and a two-phase pre-test. These measures have removed several issues which could have lead to misunderstandings.

However, in our pursuit for rigor we may have reduced the relevance of the questions. We wanted to have a consistent use of terms and to reduce the length of the questionnaire to a minimum. This may have caused that 14 respondents did not complete any of the two main parts. Furthermore, we got some feedback that some of the questions were hard to understand, especially since we were only interested in the subset of OSS components, not all OSS. One respondent wrote "the way some OSS is not considered applicable for the survey makes answering the questions more difficult than necessary".

We contacted six of the respondents who only completed the part with demographic questions and asked why they only completed that part. Two reasons were given. One, consultancy companies do not deliver products or solutions they deliver personnel to development projects. Even though these companies use OSS in their development, some seemed to be confused by the question text in question 2.1 *"Has your business unit been involved in the development of any software products or solutions where OSS components have been integrated into the product?"*. The question text focuses on products and solutions and they decided not to respond this part because of this. Two, some companies mostly use OSS infrastructure and programming languages, or interfaces towards OSS. They do not actively integrate OSS components into their products to any particular extent. Therefore, they decided not to respond to the second part.

The second group was not the intended target group of our survey and these companies could perhaps have been removed in the screening process. However, they did use OSS in their software development, but they did not actively integrate OSS components into their products.

The different interpretations of "product" and "solutions" show one of the challenges with questionnaires. The interpretation of a term depends on the context it is used in. Our understanding of the terms "product" and "solution" was unfortunately not equal to the respondents understanding of these two terms. To reach a common understanding it would have necessary to provide further explanations either in the invitations or in the questionnaire itself. This again would have increased the extent of the questionnaire and perhaps made it both harder to read and to complete.

12.2.3 Population and Sampling

The first population in this study consists of participants in the ITEA Programme. Companies participating in the ITEA Programme share an interest in research which other companies might not have. This threatens the external validity of the results from this population.

The survey was distributed through ITEA to the Project and Work Package Leaders of ongoing ITEA projects. This sample may be biased because it is most likely only larger organizations which have sufficient resources to take the position as Project or Work Package Leaders in ITEA projects. Together with being unable to gather demographic information about the population and threatens the internal validity. Many Project and Work Package Leaders are perhaps not directly involved in software development. This could further threaten the internal validity of the study. To aid this problem we asked them to find someone else in their local business unit to answer the questionnaire.

To increase the number of possible respondents it would have been possible to ask the 93 Project and Work Package Leaders to forward the invitation to their project participants. This would have increased the number of possible respondents but it would have been impossible to control response rates.

The last but perhaps biggest validity threat from the second sample was the response rates. Only 9 responses and a response rate under 10% makes it impossible to provide any conclusive finding. This is most likely caused by two factors. One, we were unable to contact the sample directly and send them reminders. Two, the sample of Project and Work Package Leaders was not a suitable sample because they have a very busy schedule and they are most likely not involved in software development.

The second sample is a fairly representative sample of the software developing companies in Norway. It was based on a convenience sample of about 400 companies. However, a stratified random sample of more than 1200 companies was

merged with this convenience sample. The stratified random sampling procedure focused on AS/ASA with more than 10 employees as these are more likely to participate in the development of (larger) software solutions. The screening process could slightly increase the bias towards larger companies because it was harder to find contact information for the smaller companies, see Figure 8.3. This can most likely be explained by the fact that many of the smaller companies do not have any active operation.

Even though the sample of companies was fairly representative we did not have control over the selection of the contact person in these companies. Furthermore we were unable to control the selection of the product the respondents answered for. This could influence the results but having a stratified random sample of companies and a convenience sample of products is the best one can expect [Conradi et al. 2005].

The response rate of 36.3 or **28.4%** is decent compared to other OSS related surveys. Several other studies have response rates below 20 %. However, the response rate is lower than the 60 +/- 20 % recommended for academic studies by [Baruch 1999]. To increase the response rates reminders have been sent to both the companies which did not respond in the screening process and to the companies which did not complete the questionnaire. It was unfortunately not possible to include results from these reminders in this report. Updated results and response rates will be published later. Analysis of non-responses is also planned and will be performed after the reminders have been sent.

In total only 11 respondents from the two samples completed Part 1. We were forced to analyze data from the two samples together as a consequence of the low number of responses. Therefore, the results from Part 1 cannot be claimed to be anything more than explorative.

12.2.4 Review and Reporting

Two measures can ensure the internal validity of a qualitative study; validation through review by others and validation through critical self-review [Jacobsen 2005]. These two measures can also be used to ensure the validity of quantitative studies.

The results discovered in this study have been presented to and discussed with professors here at the university, and fellow PhD students and researchers. All of these have provided valuable feedback.

The results from the previous study was well accepted in the ITEA review and they were summarized into a paper which was accepted at the Third International Conference on Open Source Systems, Limerick Ireland, see Appendix D. This acceptance shows the high relevance of this work. The results from the current study will also be reported both to the COSI project, to ITEA, an industrial seminar held by Tekna, and hopefully through international publications.

12.2.5 Reliability

Reliability is related to stability and consistency to reduce the errors and the biases in a study. 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 work performed in this study is transparent and very well documented. This documentation includes a complete research design, the questionnaire, and all the data files. Reliability is primarily about performing the same work, not about replicating the same results in another setting [Yin 2003]. It should be perfectly possible to replicate this study in another setting to verify our results.

12.2.6 Summary of Validity and Reliability

We have discussed some possible threats to validity and the measures we have taken to reduce the impact of these threats. The screening process was performed using a large sample from the Norwegian software sector. The response rates are high and we have not discovered any bias in the sample which responded to our request and we are therefore quite confident of the validity of the results from the screening process. We are also confident that inviting the sample by phone increases the response rates.

We used a tested survey tool to gather completed and useful answers from in total 66 companies. The second part of the survey has been completed by a representative sample of Norwegian companies which integrate OSS components into their product. The results from this part are reliable. However, results from the first part can only be seen as explorative. The number of respondents from the two samples is too low to give us any conclusive findings from Part 1.

Chapter 13

Conclusions

13.1 Summary

The literature study discovered limited evidence of industrial involvement in OSS development. However, it is evident that the software intensive industry is benefiting from OSS. The industry benefits from the software, the development practices, and the new markets which have opened up.

The current study has provided us new understanding of how companies approach OSS. Furthermore, it has shown that the software intensive industry benefits from OSS. The results are primarily related to the selection and use of OSS components but also to commercial participation in OSS communities and commercial OSS products. Together with this new understanding, we have reported several experiences and lessons learned from two large scale industrial surveys.

13.2 Contributions

The purpose of this thesis was to investigate how and why the software intensive industry approaches OSS and to reproduce the survey from [Hauge and Røsdal 2006]. The main contribution of this thesis can be divided in two.

First, this thesis provides knowledge related to how and why the software intensive industry approaches OSS development. One, findings from [Hauge and Røsdal 2006] are summarized into a paper which was accepted at The Third International Conference on Open Source Systems. Two, a fundamental understanding of key issues related to OSS is provided through the pre-study in Part II. Three, this thesis reports new findings from a large-scale survey. The most important findings are listed below:

- 46% of the Norwegian software intensive industry integrates OSS components into its products. They are mainly motivated to integrated OSS into

their products by practical reasons as the lack of license fees and the high availability of source code, components, information, and functionality.

- The companies are satisfied with their use of OSS components and report few problems with this use.
- Software developers take advantage of the availability of the source code by reading and changing it. However, it is done only to a limited extent.
- OSS components are selected using an informal selection process based on searches and previous experiences.
- About 30%, of the companies integrating OSS components participates in OSS projects outside the company. They are mainly motivated by their need for the software and the learning-effect from this participation. They participate primarily through individuals by requirements, bug report, some bug fixes, and activity in forums and on mailing lists.
- About 10% of the companies integrating OSS components, have their own OSS products. Possible advantages of releasing an OSS product are increased attention and contributions from the community. However, side-effects must be considered when releasing an OSS product. Providing an infrastructure to the community and responding to the requests from the community, generates extra work.

Second, this thesis provides a re-usable research design and experiences from the accomplishment of a large industrial survey. One, the survey tool is well documented and it can be reused in another context, see Appendix B. Two, we have discussed several challenges with industrial survey research. Some possible solutions for these problems were provided. Furthermore, we provide experiences from the process of conducting a large industrial survey, see Chapter 8, Chapter 11, and in Appendix E. In addition to this we have also developed a list of Norwegian IT companies with about 1000 companies, more than 200 companies using OSS in their software development, and more than 500 contact persons. This list can be used as a basis population for future studies. The most important findings related to the research design are listed below.

- It is imperative to have direct contact with the population.
- Inviting respondents by phone increases the response rates in a survey.
- An email screening process with few questions has high response rates, provides answers to the questions, and informs the respondents of a forthcoming survey.

13.3 Limitations

The lack of direct contact with and knowledge about the ITEA population made it difficult to draw a representative sample from this population. Furthermore, it

resulted in a low response rate. In future studies having such a weak link with the population must be avoided.

These problems were avoided to great extent with the Norwegian sample. More information was available about the population and it was possible to contact them directly. As a consequence of this direct contact we increased the response rate to an acceptable level. To increase the response rate even further it is necessary to send the sample reminders. Reminders should be sent to both the companies which did not respond in our screening process and companies which were invited to participate in the survey but did not complete the questionnaire. This work is ongoing but it has not been finished. The results of this ongoing process have therefore not been included in this report.

Data from the respondents who answered Part 1 were analysed together. Only a few respondents from each of the two samples completed this part. Results from this part must therefore be considered exploratory.

13.4 Further Research

There is clearly a need for more work related to how the software intensive industrial approaches OSS. It is possible to pose several new questions based on the results.

- *How do developers use the code from OSS components?* We saw that developers benefit from the availability of the code but how do they really use it?
- *Does the use of OSS have any consequences for the customer?* What is the consequences of the widespread use of OSS components for the customers? The role of the customer seemed unimportant in technical decisions. Should the customer care about the technical choices made by the developers?
- *How do a company successfully distribute software as OSS?* Releasing a piece of software as OSS has some benefits but it also has some side-effects. Understanding how to maximize these benefits and reduce the side-effects could help several companies to increase their benefit from a community of users and customers.
- *Do large software systems contain less OSS than smaller systems (compared to size)?* Our results indicated that software requiring more effort contains less OSS. Do they contain less OSS because of their complexity or does it take longer to develop them because OSS is not used?

In addition to new questions it is also possible to replicate and verify the results from this study by using other research methods or by replicating this study in another setting.

- *Use other research methods to verify and explain our results.* Getting richer data from a small sample of companies could increase the understanding of their use of OSS. Other research methods should also be used to answer the questions which were left unanswered. The number of companies providing OSS products is low compared to the number of companies using OSS in their software development. A questionnaire is not the best tool to use when investigating a small population.
- *Replicate the survey.* The survey could be replicated in a year or two here to see if here has been any changes in how and why companies approach OSS. The survey could also be replicated in another country to see if there are any geographical differences.
- *Created guidelines for selection of OSS components* documented processes and checklists were not widely used. Creating easy-to-use guidelines for software developers could help them in their selection and evaluation of OSS components.
- *Perform a systematic literature review.* To create a solid basis of knowledge about industrial approaches to OSS, a systematic literature review could be performed, i.e. using guidelines from i.e. [Kitchenham 2004].

Index

- Apache, 14
- ARPANET, 12
- Berkeley Software Distribution, 13, 18
- BSD, *see* Berkeley Software Distribution
- Code ownership, 38
- Code repositories, 37
- CommSy, 43
- Copyleft, 18, 27
- Cross sectional, 66
- Deductive reasoning, 66
- Dual licensing, 26
- Eclipse, 15
- Forums, 34
- Free Software Definition, 16
- Free Software Foundation, 13, 16
- Free/Libre Open Source Software, 16
- GNU, 13, 19
- GPL, 19
- Inductive reasoning, 66
- Internet, 12
- LGPL, 19
- Licenses, 17
- Linux, 14
- Literature review, 61
- Mailing lists, 34
- Measurement scales, 95
- Mozilla, 44
- Open Source, 16
- Open Source Definition, 16
- Open Source Initiative, 16
- Open Source Software, 11
- OSCAR, 44
- OSSD, 33
- PACT, 12
- Peer review, 36
- Population, 74
- Quantitative study, 66
- Questionnaire
 - Changes, 89
 - Mapping to research questions, 91
 - Pre-test, 98
 - Question types, 93
 - Review, 98
 - Structure, 91
- Questionnaire Design, 66
- Research design, 65
- Research process, 60
- Research questions, 59, 64
- Response rates, 87
- Richard Stallman, 13, 19
- Sample, 74
 - Procedurecs, 75, 80
- Survey
 - Invitation, 75, 85
- Survey research, 67
 - Challenges, 67
- TCP/IP, 13

The Early Stages, 11
The Norwegian ICT Sector, 77
Trackers, 35

UNIX, 12

Validity, 137

Glossary

ANT	a tool for automatic building of software
Apache HTTP server	Earlier known as the Apache web server. It is the most used web server on the Internet today
ARPANET	Advanced Research Project Agency Network
BIND	(Berkeley Internet Name Domain) BIND is the most commonly used DNS server on the Internet. A DNS server translates domain names to IP-addresses.
BSD	Berkeley Software Distribution
Company	A company has one brand and contact point but it can consist of one or more legal entities.
Construct validity	Construct validity is related to generalizing the results to more a general underlying theory
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
COSI	(Co-development using inner & Open source in Software Intensive products: http://www.itea-cosi.org) is an industry-driven research project under the ITEA umbrella
COTS	Commercial off-the-shelf
Cross sectional study	A study performed once representing a snapshot in time

CVS	Concurrent Versions System
Dual licensing	Licensing a pieces of software with two licenses (Most normally a copyleft license and one or more proprietary licenses)
External validity	External validity is related to whether the results are valid for other populations
Extrinsic motivation	When a person is rewarded or encouraged by something outside the person for instance rewards as payments promotions praise or public commendation.
FLOSS	Free Libre Open Source Software where Libre is Spanish for free as in freedom
FreeBSD	FreeBSD is a UNIX like operating system descending from AT&T UNIX and BSD
GPL	The GNU General Public License is one of the most popular free software licenses
Hacker	a person who is very skilled at computer programming and spends a lot of time programming
IDI	Department of Computer and Information Science
Internal validity	Internal validity is related to whether the results are valid for the population the sample is taken from
Intrinsic motivation	When a person is involved in some activity without some obvious external incentive present. This could be the feeling of creativity doing something for others and so on
ITEA	(Information Technology for European Advancement) is an interest organization for the European IT industry: http://www.itea-office.org

Legal entity	A legal entity is an entity which is treated by the law as if it was a person i.e. an incorporated organization.
LGPL	GNU Lesser General Public License
NTNU	Norwegian University of Science and Technology
OSS	Open Source Software
OSSD	Open Source Software Development
PACT	Project for the Advancement of Coding Techniques
Population	A set of subjects with some common characteristics. For this set of subjects we want to draw statistical inferences. This is commonly done by drawing a (random) sample of subjects from the population
Quantitative study	Concerned with information as numbers quantifying a relationship or comparing two or more groups
Qualitative study	Concerned with information as text either written or spoken
Sample	A sample is a set of subjects drawn from a population.
SE	Software Engineering
Sendmail	Sendmail is a mail transfer agent written by Eric Allman in the early eighties. It is currently running most of the e-mail traffic of today.
Survey	Surveys are used to collect information about subjects from a population. Surveys often gather quantitative data but they can also gather qualitative data. Common survey types are questionnaires and structured interviews.
SVN	Subversion

Tomcat a Java Server Pages and Java Servlet extension of the HTTP server

Bibliography

- Ajila, S. A. and Wu, D. (2007). Empirical study of the effects of open source adoption on software development economics. *Journal of Systems and Software*, 80(9):1517–1529.
- Anvik, J., Hiew, L., and Murphy, G. C. (2006). Who Should Fix This Bug? In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 361–370, New York, NY, USA. ACM Press.
- AP, SV og SP (2005). Politisk plattform for en flertallsregjering. Online: http://www.dep.no/smk/norsk/regjeringen/om_regjeringen/001001-990342/dok-bn.html, accessed 2007-02-09.
- Asay, M. N. (2006). Open Source and the Commodity Urge: Disruptive Models for a Disruptive Development Process. In DiBona, C., Copper, D., and Stone, M., editors, *Open Sources 2.0*, pages 103–119. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472.
- Baker, M. (2006). The Mozilla Project: Past and Future. In DiBona, C., Copper, D., and Stone, M., editors, *Open Sources 2.0*, pages 3–19. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472.
- Barrera, P., Robles, G., Canas, J. M., Martin, F., and Matellan, V. (2005). Impact of Libre Software Tools and Methods in the Robotics Field. In *5-WOSSE: Proceedings of the Fifth Workshop on Open source Software Engineering*, pages 1–6, New York, NY, USA. ACM Press.
- Baruch, Y. (1999). Response Rate in Academic Studies A Comparative Analysis. *Human Relations*, 52(4):421–438.
- Bitzer, J. (2004). Commercial Versus Open Source Software: the Role of Product Heterogeneity in Competition. *Economic Systems*, 28(4):369–381.
- Bleek, W.-G. and Finck, M. (2004). Migrating a Development Project to Open Source Software Development. In Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K., editors, *Collaboration, Conflict and Control Proceedings of the 4th Workshop on Open Source Software Engineering*, pages 9–13.

- Bleek, W.-G. and Finck, M. (2005). Ensuring Transparency - Migrating a Closed Software Development to an Open Source Software Project. In *IRIS28: Proceedings of the 28. Conference on Information Systems Research in Scandinavia, 6-9 August 2005, Kristiansand, Norway*.
- Bleek, W.-G., Finck, M., and Pape, B. (2005). Towards an Open Source Development Process ? Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model. In Scotto, M. and Succi, G., editors, *OSS 2005: Proceedings of the First International Conference on Open Source Systems, 11-15 Juli 2005, Genova, Italy*, pages 37–43.
- Bolado, M., Posadas, H., Castillo, J., Huerta, P., Sanchez, P., Sanchez, C., Fouren, H., and Blasco, F. (2004). Platform Based on Open-Source Cores for Industrial Applications. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 21014, Washington, DC, USA. IEEE Computer Society.
- Boldyreff, C., Nutter, D., and Rank, S. (2004). Communication and Conflict Issues in Coollaborative Software Research Projects. In Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K., editors, *Collaboration, Conflict and Control Proceedings of the 4th Workshop on Open Source Software Engineering*, pages 14–17.
- Bonaccorsi, A., Rossi, C., and Giannagenli, S. (2004). Adaptive Entry Strategies under Dominant Standards - Hybrid Business Models in the Open Source Software Industry. Working Paper Series, online: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=519842#, accessed 2007-03-25.
- Brown, A. W. and Booch, G. (2002). Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors. In *Software Reuse: Methods, Techniques, and Tools : 7th International Conference, ICSR-7, Austin, TX, USA, April 15-19, 2002. Proceedings*, volume 2319/2002 of *Lecture Notes in Computer Science*, pages 123–136. Springer-Verlag, Berlin Heidelberg.
- Chen, W., Li, J., Ma, J., Conradi, R., Ji, J., and Liu, C. (2007). An Industrial Survey of Software Development with Open Source Components in Chinese IT Industry. In Munch, J. and Abrahamsson, P., editors, *Proceedings of 8th Intl' Conf. on Product Focused Software Development and Process Improvement (PROFES'2007)*, Riga, Latvia, 2-4 July 2007, page 15p. Springer LNCS. To appear.
- Christley, S. and Madey, G. (2007). Analysis of activity in the open source software development community. In *40th Annual Hawaii International Conference on System Sciences, 2007*, page 40th Annual Hawaii International Conference on.
- Conradi, R., Li, J., Slyngstad, O. P. N., Kampenes, V. B., Bunse, C., Morisio, M., and Torchiano, M. (2005). Reflections on Conducting an International Survey of Software Engineering. In Verner, J. and Travassos, G. H., editors,

- Proceedings on International Symposium on Empirical Software Engineering ISESE 05*, pages 214–223.
- Cooper, D. and Schindler, P. S. (2006). *Business Research Methods*. McGraw Hill, 9th edition. ISBN: 007-124430-1.
- Crowston, K., Annabi, H., Howison, J., and Masango, C. (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*, pages 18–26, Newport Beach, CA, USA. ACM Press, New York, NY, USA.
- Cruz, D., Wieland, T., and Ziegler, A. (2006). Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis. *Software Process: Improvement and Practice*, 11(2):107–122.
- Dahlander, L. and Wallin, M. W. (2006). A Man on the Inside: Unlocking Communities as Complimentary Assets. *Research Policy*, 35(8):1243–1259.
- Dedrick, J. and West, J. (2004). An Exploratory Study into Open Source Platform Adoption. In *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, page 80265b, Washington, DC, USA. IEEE Computer Society.
- Dillman, D. A. (2007). *Mail and Internet Surveys The Tailored Design Method*. John Wiley & Sons, Inc., second edition, 2007 update edition.
- Dinh-Trond, T. T. and Bieman, J. M. (2005). The FreeBSD Project: A Replication Case Study of Open Source Development. *IEEE Transactions on Software Engineering*, 31(6):481–494.
- Divitini, M., Jaccheri, L., Monteiro, E., and Troetteberg, H. (2003). Open Source Processes no Place For Politics. In Feller, J., Fitzgerald, B., Hissam, S., and Lakhani, K., editors, *Taking Stock of the Bazaar Proceedings of the 3rd Workshop on Open Source Software Engineering*, pages 39–44.
- Ducheneaut, N. (2005). Socialization in an Open Source Software Community: A Socio-Technical Analysis. *Computer Supported Cooperative Work*, 14(4):323–368.
- Dyba, T. (2000). An Instrument for Measuring the Key Factors of Success in Software Process Improvement. *Empirical Software Engineering*, 5(4):357–390.
- Feller, J. and Fitzgerald, B. (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.
- Fitzgerald, B. (2006). The Transformation of Open Source Software. *MIS Quarterly*, 30(3).
- Fuggetta, A. (2003). Open Source Software an Evaluation. *The Journal of Systems and Software*, 66:77–90.

- Gacek, C. and Arief, B. (2004). The Many Meanings of Open Source. *IEEE Software*, 21(1):34–40.
- Ghosh, R. A. (2002). Free libre and open source software: Survey and study. Technical report, International Institute of Infonomics, University of Maastricht.
- Ghosh, R. A. (2006). Study on the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communitcaion Technologies (ICT) Sector in the EU. Technical report, UNU-MERIT.
- Giacomo, P. D. (2005). COTS and Open Source Software Components: Are They Really Different on the Playground? In *Proceedings of The 4th International Conference on COTS-Based Software Systems, ICCBSS 2005*, volume 3412/2005, pages 301–310. Springer Berlin/Heidelberg.
- Glynn, E., Fitzgerald, B., and Exton, C. (2005). Commercial Adoption of Open Source Software: An Empirical Study. In *Proceedings of International Conference on Empirical Software Engineering, Noosa Heads, Australia*.
- GNU Project (2005). What is Copyleft? Online: <http://www.gnu.org/copyleft/>, accessed 2006-10-16.
- Goldman, R. and Gabriel, R. (2005). *Innovation Happens Elsewhere: Open Source as Business Strategy*. Morgan Kaufman Publishers. ISBN: 1-55860-889-3.
- Goode, S. (2004). Something for Nothing: Management Rejection of Open Source Software in Australia’s Top firms. *Information & Management*, 42(5):669–681.
- Groves, R. M., Jr., F. J. F., Couper, M. P., Lepkowski, J. M., Singer, E., and Tourangeau, R. (2004). *Survey Methodology*. John Wiley & Sons, Inc.
- Gurbani, V. K., Garvert, A., and Herbsleb, J. D. (2005). A Case Study of Open Source Tools and Practices in a Commercial Setting. In *5-WOSSE: Proceedings of the Fifth Workshop on Open Source Software Engineering*, pages 1–6, New York, NY, USA. ACM Press.
- Gurbani, V. K., Garvert, A., and Herbsleb, J. D. (2006). A Case Study of a Corporate Open Source Development Model. In *ICSE ’06: Proceeding of the 28th International Conference on Software Engineering*, pages 472–481, New York, NY, USA. ACM Press.
- Gutwin, C., Penner, R., and Schneider, K. (2004). Group Awareness in Distributed Software Development. In *CSCW ’04: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, pages 72–81, Chicago, Illinois, USA. ACM Press, New York, NY, USA. ISBN: 1-58113-810-5.
- Hamerly, J., Paquin, T., and Walton, S. (1999). Freeing the Source - The Story of Mozilla. In DiBona, C., Ockman, S., and Stone, M., editors, *Open Sources: Voices from the Open Source Revolution*, pages 197–206. O’Reilly Media, Inc.

- Hang, J., Hohensohn, H., Mayr, K., and Wieland, T. (2004). Benefits and Pitfalls of Open Source in Commercial Contexts. In Koch, S., editor, *Free/Open Source Software Development*, pages 222–241. Idea Group Publishing, Hershey, PA.
- Hars, A. and Ou, S. (2002). Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce*, 6(3):25–39.
- Hauge, Ø. and Røsdal, A. (2006). A Survey of Industrial Involvement in Open Source. Master’s thesis, Norwegian University of Science and Technology NTNU.
- Hauge, Ø., Sørensen, C.-F., and Røsdal, A. (2007). Surveying Industrial Roles in Open Source Software Development. In Feller, J., Fitzgerald, B., Scacchi, W., and Sillitti, A., editors, *Proceedings on The Third International Conference on Open Source Systems*, pages 259–264.
- Healey, B. (2007). Drop Down and Scroll Mice: The Effect of Response Option Format and Input Mechanism Employed on Data Quality in Web Surveys. *Social Science Computer Review*, 25(1):111–128.
- Helsper, E. J. (2006). Open Source Software in Higher Educational Institutions in the UK: A Report on the 2006 Survey. Technical report, OSS Watch: Oxford University Computing Services. Online at: <http://www.oss-watch.ac.uk/studies/survey2006/>, last accessed 2007-08-15.
- Henkel, J. (2006). Selective Revealing in Open Innovation Processes: The Case of Embedded Linux. *Research Policy*, 35(7):953–969.
- Hertel, G., Niedner, S., and Herrmann, S. (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 Jorgensen, N. (2004). Do not Check in on Red: Control Meets Anarchy in Two Open Source Projects. In Koch, S., editor, *Free/Open Source Software Development*, pages 1–26. Idea Group Publishing, Hershey, PA.
- IBM (2005). Welcome to the Open source zone. Online: <http://www-128.ibm.com/developerworks/opensource/library/os-welcome.html>, accessed 2007-02-07.
- ITEA (2007). About ITEA. Online: http://www.itea-office.org/about_itea_2, accessed 2007-03-05.
- Jacobsen, D. I. (2005). *Hvordan gjennomføre undersøkelser?* HøyskoleForlaget. 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. A. (2004). Procedures for Performing Systematic Reviews. Technical report, Keele University. Technical Report TR/SE-0401 and NICATA Technical Report 0400011T.1.
- Kitchenham, B. A. and Pfleeger, S. L. (2002). Principles of Survey Research Part 3: Constructing a Survey Instrument. *SIGSOFT Softw. Eng. Notes*, 27(2):20–24.
- Krishnamurthy, S. (2002). Cave of Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, 7(6).
- Krishnamurthy, S. (2005). An analysis of open source business models. In Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K. R., editors, *Perspectives on Free and Open Source Software*, pages 279–296. MIT Press.
- Lacotte, J.-P. (2004). ITEA Report on Open Source Software. Technical report, ITEA - Information Technology for European Advancement.
- Lakhani, K. R. and von Hippel, E. (2002). How Open Source Software Works: 'Free' to User-to-User Assistance. *Research Policy*, 32:923–943.
- Lakhani, K. R. and Wolf, R. G. (2005). Why Hackers Do What They Do: Understanding Motivations and Effort in Free/Open Source Software Projects. In Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K. R., editors, *Perspectives on Free and Open Source Software*, pages 3–23. MIT Press.
- Laurent, A. M. S. (2004). *Understanding Open Source & Free Software Licensing*. O'Reilly. ISBN: 0-596-00581-4.
- Leonard, A. (2000). The Free Software Project. Online: <http://www.salon.com/tech/fsp/contents/index.html>, accessed 2006-02-07.
- Lerner, J. and Tirole, J. (2005). Economic Perspectives on Open Source. In 1st ed., editor, *Perspectives on Free and Open Source Software*, pages 47–78. MIT Press.
- Li, J., Bjørnson, F. O., Conradi, R., and Kampenes, V. B. (2006a). An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry. *Journal of Empirical Software Engineering*, 11(3):433–461.
- Li, J., Conradi, R., Slyngstad, O. P. N., Bunse, C., Khan, U., Torchiano, M., and Morisio, M. (2005). An Empirical Study on Off-the-Shelf Component Usage in Industrial Projects. In Bomarius, F. and Komi-Sirvio, S., editors, *Proc. 6th International Conference on Product Focused Software Process Improvement (PROFES'2005)*, pages 54–68. Springer Verlag.
- Li, J., Torchiano, M., Conradi, R., Slyngstad, O. P. N., and Bunse, C. (2006b). A State-of-the-Practice Survey of Off-the-Shelf Component-Based Development Processes. In et al., M. M., editor, *Proc. 9th International Conference on Software Reuse (ICSR'06) Torino, 12-15 June 2006*, pages 16–28. Springer Verlag.

- Lin, L. (2006a). Impact of users' expertise on the competition between proprietary and open source software. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, volume 8, pages 166a–166a.
- Lin, Y. (2006b). Hybrid Innovation: How does OSS Firms Collaborate with the FLOSS Community. *Knowledge, Technology and Policy*, pages 86–100.
- Lussier, S. (2004). New Tricks: How Open Source Changed the Way My Team Works. *IEEE Software*, 21(1):68–72.
- Madanmohan, T. and De', R. (2004). Open Source Reuse in Commercial Firms. *IEEE Software*, 21(6):62–69.
- Madey, G., Freeh, V., and Tynan, R. (2004). Modeling the F/OSS Community: A Quantitative Investigation. In Koch, S., editor, *Free/Open Source Software Development*, pages 203–221. Idea Group Publishing, Hershey, PA.
- Martin, K. and Hoffman, B. (2007). An Open Source Approach to Developing Software in a Small Organization. *IEEE Software*, 24(1):46–53.
- Merilinna, J. and Matinlassi, M. (2006). State of the Art and Practice of Open-Source Component Integration. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, pages 170–177, Los Alamitos, CA, USA. IEEE Computer Society.
- Michlmayr, M., Hunt, F., and Probert, D. (2005). Quality Practices and Problems in Free Software Projects. In Scotto, M. and Succi, G., editors, *Proceedings of the First International Conference on Open Source Systems*, pages 24–28.
- Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346.
- Morad, S. and Kuflik, T. (2005). Conventional and Open Source Software Reuse at Orbotech - an Industrial Experience. In *IEEE International Conference on Software - Science, Technology and Engineering. 2005*, pages 110–117.
- Mustonen, M. (2003). Copyleft - the Economics of Linux and Other Open Source Software. *Information Economics and Policy*, 15:99–121.
- Narduzzo, A. and Rossi, A. (2004). The Role of Modularity in Free/Open Source Software Development. In Koch, S., editor, *Free/Open Source Software Development*, pages 84–102. Idea Group Publishing, Hershey, PA.
- Nichols, D. M. and Twidale, M. B. (2003). The Usability of Open Source Software. *First Monday*, 8(1).
- Norris, J. S. (2004). Mission-critical Development with Open Source Software: Lessons Learned. *IEEE Software*, 21(1):42–49.
- Olson, M. (2006). Dual Licensing. In DiBona, C., Copper, D., and Stone, M., editors, *Open Sources 2.0*, pages 71–90. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472.

- Paech, B. and Reuschenbach, B. (2006). Open Source Requirements Engineering. In *International Conference on Requirements Engineering*, pages 257–262, Minneapolis, Minnesota. IEEE.
- Paulson, J. W., Succi, G., and Eberlein, A. (2004). An Empirical Study of Open-Source and Closed-Source Software Products. *IEEE Transactions on Software Engineering*, 30(4):246–256.
- Perens, B. (1998). The Open Source Definition. Online: <http://www.perens.com/Articles/OSD.html>, accessed 2006-01-31.
- Persson, A., Lings, B., Lundell, B., Mattsson, A., and Årliig, U. (2005). Communication, Coordination and Control in Distributed Development: an OSS Case Study. In Scotto, M. and Succi, G., editors, *Proceedings of the 1st International Conference on Open Source Systems, Genoa, Italy*, pages 88–92.
- Potsar, V. and Chang, E. (2004). Open Source and Closed Source Software Development Methodologies. In *Collaboration, Conflict and Control – Proceedings of the 4th Workshop on Open Source Software Engineering*, pages 105–109, Edinburgh, Scotland. IEEE Computer Society.
- Raymond, E. (1998). The Cathedral and The Bazaar. *First Monday*, 3(3). Online, http://www.firstmonday.org/issues/issue3_3/, accessed 2007-04-19.
- Raymond, E. (2001). *The Cathedral & The Bazaar - Musings On Linux And Open Source By An Accidental Revolutionary*. O’Reilly, revised edition edition. ISBN: 0-596-00108-8.
- Robson, C. (2003). *Real World Research*. Blackwell Publishing. ISBN: 0-631-21305-8.
- Rossi, C. and Bonaccorsi, A. (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*, pages 1–5, St. Louis, Missouri. ACM Press, New York, NY, USA. ISBN: 1-59593-127-9.
- Ruffin, M. and Ebert, C. (2004). Using Open Source Software in Product Development: A Primer. *IEEE Software*, 21(1):82–86.
- Samoladas, I., Stamelos, I., Angelis, L., and Oikonomou, A. (2004). Open Source Software Development Should Strive for even Greater Code Maintainability. *Communications of the ACM*, 47(10):83–87.
- Scacchi, W. (2002a). Is Open Source Software Development Faster, Better, and Cheaper than Software Engineering. In *ICSE Woekshop on Open Source Software Engineering, May, 2002 Orlando, FL*.
- Scacchi, W. (2002b). Understanding the requirements for developing open source software systems. *Software, IEE Proceedings-*, 149(1):24–39.
- Scacchi, W. (2004). Free and Open Source Development Pratices in the Game Community. *IEEE Software*, 21(1):59–66.

- Scacchi, W. (2007). Free/Open Source Software Development: Recent Research Results and Methods. In Zerkowitz, M. V., editor, *Advances in Computers*, volume 69, pages 243–269. Academic Press.
- Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S. A., and Lakhani, K. R. (2006). Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice*, 11(2):95–105.
- Serrano, N., Calzada, S., Sarriegui, J. M., and Ciordia, I. (2004). From Proprietary to Open Source Tools in Information Systems Development. *IEEE Software*, 21(1):56–58.
- Singh, V., Twidale, M. B., and Rathi, D. (2006). Open Source Technical Support: A Look at Peer Help-Giving. In *HICSS '06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, page 118.3, Washington, DC, USA. IEEE Computer Society.
- Spector, P. E. (1980). Ratings of Equal and Unequal Response Choice Intervals. *Journal of Social Psychology* 112 Issue 1, p115, 5p; (AN 5388125), 112(1):115–119.
- Spinellis, D. and Szyperski, C. (2004). How Is Open Source Affecting Software Development. *IEEE Software*, 21(1):28 – 33.
- SSB (2005). Table 04466: Informasjonssektoren. Sysselsetting, omsetning og verdiskapning, etter sektor. Online, <http://statbank.ssb.no/statistikkbanken/>, accessed 2007-08-01, A description of these data can be found at <http://www.ssb.no/vis/iktoms/om.html>.
- SSB (2007). Tabell: 03109: Bedrifter, etter næring og antall ansatte (K). Online, <http://statbank.ssb.no/statistikkbanken/>, accessed 2007-08-01, A description of these data can be found at <http://www.ssb.no/vis/iktoms/om.html>.
- Stallman, R. (1999). The Gnu Operating System and the Free Software Movement. In DiBona, C., Ockman, S., and Stone, M., editors, *Open Sources: Voices from the Open Source Revolution*, pages 53–70. O'Reilly Media, Inc.
- The Eclipse Foundation (2007). Strategic Members. Online: <http://www.eclipse.org/membership/members/strategic.php>, accessed 2007-02-09.
- Tukey, J. W. (1961). Data Analysis and Behavioral Science or Learning to Bear the Quantitative Man's Burden by Shunning Badmandments. In *The Collected Works of John W. Tukey, vol III*, pages 187–389. Chapman & Hall, Belmont CA, USA.
- Tuomi, I. (2003). Internet, Innovation and Open Source: Actors in the Network. *First Monday*, 6(1):1–1. Online journal: <http://www.firstmonday.org/>.
- von Hippel, E. and von Krogh, G. (2003). Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science. *Organization Science*, 14(2):209–223.

- von Krogh, G. and von Hippel, E. (2003). Editorial Special Issue on Open Source Software Development. *Research Policy*, 32(7):1149–1157.
- Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K. (2007). *Probability & Statistics For Engineers & Scientists*. Pearson Prentice Hall, Upper Saddle River, NJ 07458.
- Webbink, M. H. (2003). Understanding Open Source Software. *New South Wales Society For Computers and the Law*, 51. Available online: http://www.nswscl.org.au/journal/51/Mark_H_Webbink.html, accessed 2006-10-26.
- Weber, S. (2004). *The Success of Open Source*. Harvard University Press Cambridge. ISBN: 0-674-01292-5.
- West, J. (2007). Value Capture and Value Networks in Open Source Vendor Strategies. In *40th Annual Hawaii International Conference on System Sciences*, pages 176–176.
- West, J. and O'Mahony, S. (2005). Contrasting Community Building in Sponsored and Community Founded Open Source Projects. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05*, pages 196c–196c.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, Boston / Dordrecht / London. ISBN: 0-7923-8682-5.
- Worthington, D. (2005). IBM Turns to Open Source Development. Online: http://www.betanews.com/article/IBM_Turns_to_Open_Source_Development/1118688437, accessed 2007-03-24.
- Ye, Y. and Kishida, K. (2003). Toward an Understanding of the Motivation of Open Source Software Developers. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 419–429.
- Yin, R. K. (2003). *Case Study Research Design and Methods*. Sage Publications, 3rd. edition. ISBN: 0-7619-2553-8.
- Østerlie, T. (2007). A Critical Review of Software Engineering Research on Open Source Software Development. In *Proceedings of AIS SIGSAND European Symposium on Systems Analysis and Design Gdansk*. University of Gdansk Press.
- Østerlie, T. and Wang, A. I. (2006). Establishing maintainability in systems integration: Ambiguity, negotiation, and infrastructure. In *22nd International Conference on Software Maintenance, 24-27 September 2006, Philadelphia, Pennsylvania*.

Part V

Appendices

Appendix **A**

The Invitation Letter and The Reminder

The first invitation letter and the first reminder were sent to the ITEA Workpackage and Project Leaders by the ITEA Office. The second invitation letter was sent to the sample of Norwegian ICT companies by NTNU.

Survey on Industrial Open Source Software Development

Dear ITEA project partner,

The ITEA COSI project invites you to participate in a survey on industrial Open Source Software (OSS) development. Your contributions are of great value to ITEA and COSI. **The results will be shared with participants in all ITEA projects.**

The use of Open Source Software (OSS) in software intensive companies is rapidly increasing. The ITEA project COSI, wants to understand how the European software intensive industry can benefit from OSS. To achieve this goal we have already performed a survey within the COSI project. After the first survey, ITEA encouraged COSI to distribute a survey to the ITEA participants.

ITEA and COSI hope that you see the benefit of these initiatives and encourage you to participate in this second survey. Results from the first survey are available at the COSI web-page: <http://www.itea-cosi.org> as part of deliverable D2.1.2. Results from this second survey will also be made available from the COSI web-page.

If your local business unit either develops OSS or re-uses OSS components in its software development please answer the survey here: <http://survey.confirmit.com/wix/p416962745.aspx>

If you are not personally involved in such software development, please ask someone else in your local business unit to complete the survey.

The survey is a web-based questionnaire with one part for OSS providers, one part for OSS component integrators, and one part for demographic information. It will normally take between 15 and 20 minutes to complete the survey.

If your business unit is unable to participate in the survey, we would appreciate if you could explain why by answering a few questions here: <http://survey.confirmit.com/wix/p432206056.aspx>

Information about particular companies and respondents will be treated strictly confidential and made anonymous. All knowledge gathered through this survey will be anonymized, and reported to ITEA.

We hope that you forgive us if you receive this message more than once.

Thank you for participating in this survey.

Erik Rodenbach, Programme Co-ordinator, on behalf of ITEA and

Dr. Frank van der Linden, Project Manger, on behalf of the COSI project

The survey is conducted by COSI partner ICT-Norway through their sub-contractor The Norwegian University of Science and Technology (NTNU). If you have any questions or comments, please feel free to contact us.

Contact information for the survey:

Dr. student Øyvind Hauge,
oyvind.hauge@idi.ntnu.no,
Phone: +47 97 71 22 52

Dr. Prof Reidar Conradi,
reidar.conradi@idi.ntnu.no,
Phone: +47 73 59 34 44

Address: IDI, NTNU, NO-7491 Trondheim, Norway
Fax: +47 73 59 44 66

Reminder for Open Source Survey

Dear ITEA project partner,

The ITEA COSI project would like to remind you of its ongoing web-survey on industrial Open Source Software (OSS) development. ITEA and COSI encourage you to participate in the survey.

If your local business unit either develops OSS or re-uses OSS components in its software development please answer the survey here:
<http://survey.confirmit.com/wix/p416962745.aspx>

You may answer the survey personally or you may ask someone else in your local business unit to answer it. If your business unit is unable to participate in the survey, we would appreciate if you could explain why by answering a few questions here:
<http://survey.confirmit.com/wix/p432206056.aspx>

This will be the first and only reminder we send you.

Thank you for participating in this survey.

Erik Rodenbach, Programme Co-ordinator, on behalf of ITEA and
Dr. Frank van der Linden, Project Manger, on behalf of the COSI project

The survey is conducted by COSI partner ICT-Norway through their sub-contractor The Norwegian University of Science and Technology (NTNU). If you have any questions or comments, please feel free to contact us.

Dr. student Øyvind Hauge,
oyvind.hauge@idi.ntnu.no,
Phone: +47 97 71 22 52

Professor Reidar Conradi,
reidar.conradi@idi.ntnu.no,
Phone: +47 73 59 34 44

Address: IDI, NTNU, NO-7491 Trondheim, Norway
Fax: +47 73 59 44 66

Om bruk av open source i <BEDRIFT>

Hei <NAVN>,

Tusen takk for at du deltok i screeningprosessen på vegne av <BEDRIFT>. Over 500 andre norske bedrifter har også svart, og nær 50 % av de som driver med programvareutvikling bruker open source-produkter aktivt i sin utvikling. Med en så utbredt bruk er det viktig å forstå hvorfor og hvordan open source-produkter blir brukt, både for norsk programvareindustri og for deres kunder. Økt kunnskap kan bidra til mer bevisste valg både blant kunder og utviklere. Økt forståelse blant kundene kan også gi de bedriftene som benytter seg av open source et konkurransefortrinn.

NTNU ønsker å bidra til å sette fokus på open source og øke kunnskapen om bruk av open source-produkter i norsk programvareindustri. I den sammenheng håper vi du kan delta i en spørreundersøkelse om deres bruk av open source. Ditt svar vil være et veldig viktig bidrag til vårt arbeid.

Selve undersøkelsen er delt i to uavhengige deler. En del for de bedrifter som har egne open source-produkter og en del for de som bruker open source-produkter i sin utvikling. Hver del tar 15-20 minutter og du kan velge å svare på en eller begge delene. Undersøkelsen er utarbeidet på engelsk og den er tilgjengelig her: <http://survey.euro.confirmit.com/wix/p70815360.aspx>

All informasjon vil selvfølgelig bli behandlet konfidensielt. Alle publikasjoner basert på resultater fra denne undersøkelsen vil kun inneholde informasjon om norsk programvareindustri og undergrupper av denne industrien, ikke om den enkelte bedrift eller respondent. Resultater fra undersøkelsen vil bli delt med de respondentene som ønsker det.

Vi håper du har mulighet til å besvare undersøkelsen. Hvis du selv ikke har nær kjennskap til bedriftens programvareutvikling, håper vi at du kan få en av utviklerne deres til å besvare undersøkelsen. Hvis dere ikke har mulighet, eller hvis dere ikke ønsker å svare på undersøkelsen ber vi deg om å gi en kort tilbakemelding om dette ved å svare på denne invitasjonen.

Har du spørsmål om eller kommentarer til undersøkelsen eller vårt arbeid må du gjerne ta kontakt. Tusen takk for deres bidrag.

Med vennlig hilsen
Øyvind Hauge
oyvind.hauge@idi.ntnu.no
Tlf: 97 71 22 52

Dr. Carl-Fredrik Sørensen
carl-fredrik.sorensen@idi.ntnu.no

Tlf: 95 11 96 90

Professor Reidar Conradi
reidar.conradi@idi.ntnu.no
Tlf: 73 59 34 44

Postadresse:
Institutt for Datateknikk og Informasjonsvitenskap
Sem Sælands vei 7-9
NO-7491 Trondheim

Appendix **B**

The Questionnaire

The enclosed Word-version of questionnaire was developed using Word but implemented using a Web-tool called Conformat, see Chapter 9 for more information. The Web-version has some minor deviations from the Word-version due to properties of the Web-tool.

Introduction to the survey (Norwegian ICT)

Thank you for deciding to participate in this survey of industrial involvement in Open Source Software (OSS) development.

Results from this survey will be anonymized and used as part of a Phd thesis. If you provide give us your e-mail address we will also share these results with you. The information provided in this survey will be treated strictly confidential.

The survey contains three parts. One part is about development *of* OSS, one part about development *with* OSS, and one part with demographic questions. Each of the first two parts will take about 15 minutes to complete. However, you are not required to answer both of them.

If you have any questions or comments, or if you want to know more about our work, do not hesitate to contact us.

Thank you again for your participation.

Contact information for the survey:

Dr. student Øyvind Hauge
oyvind.hauge@idi.ntnu.no
Phone: +47 97 71 22 52

Dr. Carl-Fredrik Sørensen,
Carl-Fredrik.Sorensen@idi.ntnu.no,
Phone: +47 73 59 07 31

Dr. Prof Reidar Conradi
reidar.conradi@idi.ntnu.no
Phone: +47 73 59 34 44

Address: IDI, NTNU, NO-7491 Trondheim, Norway
Fax: +47 73 59 44 66

Introduction to the survey (ITEA)

Thank you for deciding to participate in this survey of industrial involvement in Open Source Software (OSS) development.

This survey is the second survey initiated by ITEA and COSI related to industrial OSS development. Results from the first survey are available at the COSI web-page: <http://www.itea-cosi.org> as part of deliverable D2.1.2.

All results from this survey will be reported to ITEA and shared with you if you give us your e-mail address. The information provided in this survey will be treated strictly confidential and made anonymous.

The survey contains three parts. One part is about development of OSS, one part about development with OSS, and one part with demographic questions. Each of the first two parts will take about 15 minutes to complete. However, you are not required to answer both of them.

If you have any questions or comments, or if you want to know more about our work, do not hesitate to contact us.

Thank you again for your participation.

ICT-Norway is the responsible COSI partner for the survey which is organized by The Norwegian University of Science and Technology (NTNU).

Contact information for the survey:

Dr. student Øyvind Hauge
oyvind.hauge@idi.ntnu.no
Phone: +47 97 71 22 52

Dr. Carl-Fredrik Sørensen,
Carl-Fredrik.Sorensen@idi.ntnu.no,
Phone: +47 73 59 07 31

Dr. Prof Reidar Conradi
reidar.conradi@idi.ntnu.no
Phone: +47 73 59 34 44

Address: IDI, NTNU, NO-7491 Trondheim, Norway
Fax: +47 73 59 44 66

Part 1) Open Source Software Provider

1.1 Is your local business unit involved in the development of any OSS product which is controlled by your company?

The local business unit is the local segment of your organization where you are currently working.

OSS is software issued with a license conforming to the Open Source Definition or the Free Software Definition. We do not want to differentiate between free and open software.

	Yes	(Go to Question 1.2, page 4)
	No	(Go to Part 2, page 14)

Select **one OSS product** provided by your company, where your local business unit is involved in the development of the product. Answer all the questions in this part based on your experiences with this OSS product.

We are primarily interested in OSS products which have an ongoing development and at least one running release. Nevertheless, OSS, where the development is discontinued, is also of interest.

Information about the chosen OSS Product

This part concerns information about the selected OSS product.

1.2 What is the name of the OSS product?

Name:

--

1.3 In what business area are most of the OSS's users?

[Mark one or more alternatives]

<input type="checkbox"/>	Consulting companies
<input type="checkbox"/>	Software companies
<input type="checkbox"/>	Manufacturing Industry
<input type="checkbox"/>	Telecom industry
<input type="checkbox"/>	Telecom service provider
<input type="checkbox"/>	Public/Health
<input type="checkbox"/>	Educational institutions
<input type="checkbox"/>	Banking
<input type="checkbox"/>	Non-profit organizations
<input type="checkbox"/>	Media
<input type="checkbox"/>	Private/domestic
<input type="checkbox"/>	No particular area. Our product serves several business areas.
<input type="checkbox"/>	Other (please specify):

1.4 What is the main functionality of the OSS?

[Mark one or more alternatives]

<input type="checkbox"/>	Database
<input type="checkbox"/>	Desktop/Office tools
<input type="checkbox"/>	Software development tool or component
<input type="checkbox"/>	Enterprise solutions
<input type="checkbox"/>	Financial/Banking
<input type="checkbox"/>	Games
<input type="checkbox"/>	Multimedia
<input type="checkbox"/>	Networking
<input type="checkbox"/>	System administration tool
<input type="checkbox"/>	Web/Portals
<input type="checkbox"/>	Operating system
<input type="checkbox"/>	Middleware
<input type="checkbox"/>	Server component
<input type="checkbox"/>	Other (please specify):

1.5 What kind of software is this OSS?

[Mark one alternative]

<input type="checkbox"/>	Infrastructure: It is commodity software or infrastructure and has basic functionality common to software in several market segments (operating systems, databases, web-server etc)
<input type="checkbox"/>	Domain product: It has functionality which is basic for a certain business and has similar functionality to software in the same market segment (office tools, web browser, mail client, etc)
<input type="checkbox"/>	Component: It has functionality which is limited to one problem domain and can be used inside other software solutions (GUI component, XML library etc)
<input type="checkbox"/>	Differentiating product: It has differentiating functionality which is unique for this product in the market segment (Google Earth, Second Life, Photoshop, etc)

1.6 When was the development of the OSS product initiated?

Please specify the year.

1.7 When was the OSS product released with an OSS license?

Please specify the year.

1.8 Approximately how many person-months were spent by the employees in your local business unit developing this OSS product during the last year?

[Mark one alternative]

<input type="checkbox"/>	NA/Don't know
<input type="checkbox"/>	0
<input type="checkbox"/>	>1
<input type="checkbox"/>	1-2
<input type="checkbox"/>	3-5
<input type="checkbox"/>	5-10
<input type="checkbox"/>	11-25
<input type="checkbox"/>	26-50
<input type="checkbox"/>	51-100
<input type="checkbox"/>	101-500
<input type="checkbox"/>	>500

1.9 Which license types are used for the OSS?

A non-protective license (for instance BSD) applies no restrictions on the distribution of derivate works. A protective license (for instance GPL) does apply such restrictions.

[Mark one or more alternatives]

<input type="checkbox"/>	NA/Don't know
<input type="checkbox"/>	A protective/viral/reciprocal/copyleft license
<input type="checkbox"/>	A non-protective/academic /non-copyleft licenses
<input type="checkbox"/>	Proprietary license(s)

Other OSS Products Developed in the Local Business Unit

1.10 In the development of how many other OSS products controlled by your company has your local business unit been involved in during the last two years?

[Mark one alternative]

	NA/Don't know	(Go to Question 1.11, page 7)
	0	(Go to Question 1.11, page 7)
	1-2	
	3-5	
	6-10	
	11-20	
	21-50	
	>50	

1.10a Compare the selected OSS product to the other OSS products your local business unit is involved in.

[Mark one alternative per row]

	Much less	Less	The same	More	Much more	NA
a. Does the selected product have the same number of users?						
b. Does the selected product have the same number of installations?						
c. Has the development of the selected product had the same resources (person months) during the last year?						
d. Has the development of the selected product had the same resources (person months) during the lifetime of the product?						

1.10b Compare the selected OSS product to the other OSS products in your local business unit, how typical is:

[Mark one alternative per row]

	Very atypical	Atypical	Neither atypical or typical	Typical	Very typical	NA
a. The development process and practices used to develop this product?						
b. The customers or target domain for the product?						

Motivations behind Releasing the Software as an OSS

1.11 How important were the following motivational factors to your company when turning this software product into an OSS?

“The community” means the community of users and developers around your OSS product.

“The OSS community” means the community of all users and developers of OSS.

We use the term “defect” instead of “bug” through this survey. The term “bug” is more often used in OSS communities.

[Mark one alternative per row]

	Not important at all	Unimportant	Neither important or unimportant	Important	Very important	NA
a. Idealism						
b. Increase attention and publicity						
c. Return value to the OSS community						
d. Allow supplementary extensions, products and services from other providers						
e. Attract more users/customers to the software						
f. Increase the number of new requirements and ideas from the community						
g. User-to-user support form the community (self supporting community)						
h. Sell related services (integration, deployment, support, quality assurance, packaging, distribution, user training)						
i. Sell proprietary licenses to customers (dual licensing)						
j. Enable you to sell proprietary add-on software						
k. More defect reports from the community						
l. More defect corrections from the community						
m. Code contributions from the community (other than defect corrections)						
n. Enable integration of other OSS into your software						
o. Other (please specify)						

Experiences Related to Providing an OSS

1.12 How true are the following statements as a consequence of releasing the software as OSS?

The community means the community of users and developers around your OSS product. “The OSS community” means the community of all users and developers of OSS.

If you have observed other important consequences of releasing of this software as OSS, please identify them in the last row?

[Mark one alternative per row]

	Totally false	Mostly false	Neither true nor false	Mostly true	Totally true	NA
a. You got increased attention and publicity						
b. It is hard to focus on one group of customers because the OSS is available to everyone						
c. You got increased number of paying customers (licenses or services)						
d. You improved you relationship to the OSS community						
e. It is more difficult to plan releases because of constant input from the community						
f. Feedback and requests from the community lead to extra work						
g. It is more difficult to protect intellectual property and/or business secrets						
h. It is difficult to accept code contributions from the community due to licensing and intellectual property rights						
i. Having this software as OSS product enable you to sell related software or services						
j. Selling licenses to the software is difficult because the OSS is freely available						
k. The openness (of the code) makes it easier to find security holes						
l. Contributions from several community members increase the effort to keep a sound architecture, design, and implementation						
m. Code provided by the community has too low quality to be included directly into the software						
n. Contributions from the community are often included into the software.						
o. Other OSS have been integrated into your OSS product						
p. Other (please specify)						

Activity in the Community around your OSS

1.13 Approximately how many people have contributed code which has been integrated into the OSS product?

Affiliation	Number of people
a. In the local business unit	
b. In other business units in the company	
c. Outside the company/the community	

1.14 Approximately how many people have posted messages on the forum/ mailing lists in the community around the OSS product?

Affiliation	Number of people
a. In the local business unit	
b. In other business units in the company	
c. Outside the company/the community	

1.15 Approximately how many times has the software been downloaded?

	NA/Don't know
	0
	1-100
	101-1 000
	1 001-10 000
	10 001-100 000
	100 001-1 000 000
	>1 000 000
	Our software is available for downloaded from many locations. I would estimate that it is downloaded at least (please specify):

1.16 To what extent do members of the community perform the following activities related to your OSS product?

We think of users and developers, not hired by your company.

We use the term “defect” instead of “bug” through this survey. The term “bug” is more often used in OSS communities.

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. Test the software						
b. Report defects						
c. Fix defects						
d. Provide new requirements						
e. Implement new functionality						
f. Develop supplementary or add-on OSS						
g. Provide local adaptations of the software						
h. Create and maintain documentation						
i. Provide free support to community members						
j. Develop architecture or design						
k. Develop commercial supplementary or add-on software						
l. Provide commercial services (integration, deployment, support etc.)						
m. Other (please specify)						

1.17 How true are the following statements related to the community around your OSS?

[Mark one alternative per row]

	Totally false	Mostly false	Neither true nor false	Mostly true	Totally true	NA
a. Employees in your company respond to all requests in forums and mailing lists						
b. Everyone in the community is free to contribute to the development of the product (requirements, patches, proposals etc.)						
c. You provide up-to-date developer documentation						
d. You provide up-to-date installation guides and tutorials						
e. You provide up-to-date roadmaps						
f. You provide an updated web site with news about the software						
g. You encourage community members to provide feedback on features, specifications, design, architecture etc						
h. Internal developers use public mailing-lists or forums to discuss important issues related to the software						

1.18 Which of the following services or artefacts do you provide to the community around your OSS product?

We use the term “defect” instead of “bug” through this survey. The term “bug” is more often used in OSS communities.

[Mark one or more alternatives]

<input type="checkbox"/>	No services or artefacts are provided to our community
<input type="checkbox"/>	Mailing lists
<input type="checkbox"/>	Mailing archives (mail is stored and made publicly available)
<input type="checkbox"/>	Forums
<input type="checkbox"/>	Online code repository
<input type="checkbox"/>	Defect/bug-tracking system
<input type="checkbox"/>	Feature-tracking system
<input type="checkbox"/>	The software is available through portals like SourceForge.org or freshmeat.net
<input type="checkbox"/>	Other (please specify):

1.19 How true are the following statements related to how you have tried to attract a community?

If there are other important activities which have helped you attracting a community, please indicate these at the last row.

[Mark one alternative per row]

	Totally false	Mostly false	Neither true nor false	Mostly true	Totally true	NA
a. A community already existed for the software before it was made open source						
b. We have attracted community members through advertisements and publications in media and in scientific literature						
c. We have been very active in other OSS communities						
d. We have focused on the quality of the software and its documentation						
e. We listen to our community and we allow them to influence us						
f. We are engaged in partnership with community members						
g. We have focused on providing the necessary infrastructure to the community						
h. Other (please specify)						

Part 1 completed

You have now completed Part 1 of the survey and you may continue to answer Part 2. Part 2 concerns use of OSS components in the software development in your local business unit.

A software component is a program unit of independent production, acquisition, and deployment which can be used as part of a software system. We do not consider infrastructure as operating systems, databases, programming languages or similar to be components.

1.20 Do you want to continue to Part 2 of the survey?

[Mark one alternative]

	Yes, we use OSS components inside the selected OSS product and I will answer Part 2 based on the same product as Part 1 <p style="text-align: right;">(Go to Question 2.7, page 18)</p>
	Yes, we use OSS components in other software products/solutions and I will answer Part 2 based on another software product <p style="text-align: right;">(Go to Question 2.2, page 15)</p>
	No, we do not use OSS components in any of our software products <p style="text-align: right;">(Go to Part 3 with demographic questions, page 31)</p>
	No, we use OSS components but I am unable to answer Part 2 at the moment <p style="text-align: right;">(Go to Part 3 with demographic questions, page 31)</p>

Part 2) Integrator of Open Source Software Components

2.1 Has your business unit been involved in the development of any software products or solutions where OSS components have been integrated into the product?

The local business unit is the local segment of your organization where you are currently working.

OSS is software issued with a license conforming to the Open Source Definition or the Free Software Definition. We do not want to differentiate between free and open software.

A software component is a program unit of independent production, acquisition, and deployment which can be used as part of a software system. We do not consider infrastructure as operating systems, databases, programming languages or similar to be components.

<input type="checkbox"/>	Yes	
<input type="checkbox"/>	No	(Go to Part 3, page 31)

Select a **typical software product**, finished or at least with a running release, where you have been involved in the selection, evaluation, and integration of OSS components. The product may have been developed by your business unit alone or in collaboration with other business units or companies. Answer based on your experiences with the development of this software product.

Information about the Software

2.2 What is the name of the software product?

Name:

--

2.3 In what business area are most of the product's users?

[Mark one or more alternatives]

<input type="checkbox"/>	Consulting companies
<input type="checkbox"/>	Software companies
<input type="checkbox"/>	Manufacturing Industry
<input type="checkbox"/>	Telecom industry
<input type="checkbox"/>	Telecom service provider
<input type="checkbox"/>	Public/Health
<input type="checkbox"/>	Educational institutions
<input type="checkbox"/>	Banking
<input type="checkbox"/>	Non-profit organizations
<input type="checkbox"/>	Media
<input type="checkbox"/>	Private/domestic
<input type="checkbox"/>	No particular area. Our product serves several business areas.
<input type="checkbox"/>	Other (please specify):

2.4 What is the main functionality of the software product?

[Mark one or more alternatives]

<input type="checkbox"/>	Database
<input type="checkbox"/>	Desktop/Office tools
<input type="checkbox"/>	Software development tool or component
<input type="checkbox"/>	Enterprise solutions
<input type="checkbox"/>	Financial/Banking
<input type="checkbox"/>	Games
<input type="checkbox"/>	Multimedia
<input type="checkbox"/>	Networking
<input type="checkbox"/>	System administration tool
<input type="checkbox"/>	Web/Portals
<input type="checkbox"/>	Operating system
<input type="checkbox"/>	Middleware
<input type="checkbox"/>	Server component
<input type="checkbox"/>	Other (please specify):

2.5 What kind of software is this software product?

[Mark one alternative]

	Infrastructure: It is commodity software or infrastructure and has basic functionality common to software in several market segments (operating systems, databases, web-server etc)
	Domain product: It has functionality which is basic for a certain business and has similar functionality to software in the same market segment (office tools, web browser, mail client, etc)
	Component: It has functionality which is limited to one problem domain and can be used inside other software solutions (GUI component, XML library etc)
	Differentiating product: It has differentiating functionality which is unique for this product in the market segment (Google Earth, Second Life, Facebook, etc)

2.6 Approximately how many person-months were spent by the employees in your local business unit developing this software during the last year?

[Mark one alternative]

	NA/Don't know
	0
	>1
	1-2
	3-5
	5-10
	11-25
	26-50
	51-100
	101-500
	>500

2.7 Approximately how much of the functionality in your software product is provided by OSS components?

[Mark one alternative]

	NA/Don't know
	0%
	1-20%
	21-40%
	41-60%
	61-80%
	81-99%
	100%

2.8 How many OSS components are included as part of your product?

A software component is a program unit of independent production, acquisition, and deployment which can be used inside a software system. We do not consider infrastructure as operating systems, databases, programming languages or similar to be components.

[Mark one alternative]

	NA/Don't know	(Go to Question 2.9, page 18)
	0	(Go to Question 2.9, page 18)
	1-2	
	3-5	
	6-10	
	11-20	
	21-50	
	>50	

2.8a How much of the OSS components' code has your local business unit reviewed or read?

[Mark one alternative]

	NA/Don't know
	No parts of the code
	Very few parts of the code
	Minor parts of the code
	Some parts of the code
	Major parts of the code
	Most parts of the code
	All the code

2.8b How much of the OSS components' code has your local business unit changed or rewritten?

[Mark one alternative]

	NA/Don't know
	No parts of the code
	Very few parts of the code
	Minor parts of the code
	Some parts of the code
	Major parts of the code
	Most parts of the code
	All the code

2.8c For how many of the OSS components has your local business unit:

	NA/Don't know	None	A few	Less than half	About half	More than half	Most	All
Fixed a defect?								
Added glue or wrapping?								
Returned your changes to the OSS provider or community?								

Other Software Products Developed by the Local Business Unit

2.9 How many other software products containing OSS has your local business unit been developing during the last year?

[Mark one alternative]

	NA/Don't know	(Go to Question 2.10, page 19)
	0	(Go to Question 2.10, page 19)
	1-2	
	3-5	
	6-10	
	11-20	
	21-50	
	>50	

2.9a Compare the selected product to the other products containing OSS in your local business unit:

[Mark one alternative per row]

	Much less	Less	The same	More	Much more		NA
a. Does the selected product have the same number of users?							
b. Does the selected product have the same number of installations?							
c. Has the development of the selected product had the same resources (person months) during the last year?							
d. Has the development of the selected product had the same resources (person months) during the whole lifetime of the product?							
e. Does the selected product contain the same number of OSS components?							

2.9b Compare the selected product to the other products containing OSS in your local business unit, how typical is:

[Mark one alternative per row]

	Very different	Different	Somewhat Similar	Very Similar	Identical	NA
a. The development process and practices used to develop this product?						
b. The customers or target domain for the product?						

Motivations behind Re-Use of OSS Components

2.10 How important were the following motivational factors to your company when deciding to use OSS components in the development of this product?

If there are other important motivations, please indicate these in the last row.

[Mark one alternative per row]

	Not important at all	Unimportant	Neither important or unimportant	Important	Very Important		NA
a. Reduced risk of component provider going out of business							
b. Reduced risk of selected component evolving into an unwanted direction							
c. Component could be acquired at no cost							
d. Maintenance costs with OSS are lower (than with other software)							
e. Source code is available and can easily be changed							
f. Compliance to standards							
g. OSS components are easily available for test and use							
h. Information about OSS components is highly available							
i. Honesty and openness from the OSS provider about the true status of the component							
j. To use OSS components was decided by the customer							
k. Political reasons (company policy, licensing conditions)							
l. Idealism							
m. To become more independent from software vendors							
n. Our company possesses knowledge about OSS							
o. We want to increase our knowledge in the OSS area							
p. The market is looking for OSS							
q. Other (please specify)							

Selection of OSS Components

Selection Processes

2.11 In the development of this product, how many selections of OSS components has your local business unit been involved in?

[Mark one alternative]

	Don't know
	0
	1-2
	3-5
	6-10
	11-20
	21-50
	>50

2.12 In the development of this product, how much effort (in person hours), has your local business unit spent to find, evaluate, and select OSS components?

[Mark one alternative]

	NA/ Don't know
	0
	>1
	1-2
	3-5
	5-10
	11-25
	26-50
	51-100
	101-500
	>500

2.13 In the development of this product, to what extent did your local business unit perform the following activities while searching for OSS components?

If you performed other important activities in your search for OSS components, please indicate this in the last row.

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. Searched OSS portals (SourceForge.net, tigris.org, apache.org, eclipse.org, etc.)						
b. Used search engines (Google, Msn search, etc.) to search for individual OSS components						
c. Used search engines (Google, Msn search, etc.) to search for comparisons of several OSS components						
d. Requested advice at forums and mailing lists						
e. Asked friends and colleagues etc. whether they know any OSS candidate components						
f. Selected OSS components based on previous experience						
g. Used a company internal "knowledge base"						
h. Reviewed books and magazines						
i. Used an (external) component broker to find the component for you						
j. Other (please specify)						

2.14 To what extent did your local business unit perform the following activities when evaluating OSS components to this product?

If you performed other important activities while selecting OSS components, please indicate this in the last row.

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. Used a selection process which is well documented in the company						
b. Used documented checklists to evaluate the OSS components						
c. Defined a list of requirements to the OSS component before the selection started						
d. Documented the choice of OSS component and the rationale behind this choice						
e. Used shortlists (identify several OSS components with similar functionality)						
f. Performed testing and/or prototyping with OSS components						
g. Performed code reviews of OSS components						
h. Performed architecture reviews of OSS components						
i. Reviewed documentation for OSS components						
j. Assessed the activity within the community around the OSS components						
k. Looked for references and/or other experiences with the OSS components						
l. Estimated how much effort you would use on selection and integration of the OSS component and include this estimate into the project plan						
m. Other (please specify)						

Evaluation Criteria

2.15 How important were the following properties of the OSS components while evaluating components to this product?

If there are other important criteria, please indicate these in the last row.

[Mark one alternative per row]

	Not important at all	Unimportant	Neither important or unimportant	Important	Very Important	NA
a. Compliance to requirements						
b. Extra functionality/features						
c. Few dependencies to platforms, other components, standards etc						
d. Modularity and clear interfaces						
e. Code quality						
f. Stable releases of the component						
g. Security						
h. Performance						
i. Integration with other software						
j. Programming language/environment						
k. License/license type						
l. Architecture						
m. Experience with component						
n. Other (please specify)						

2.16 How important were the following properties of the OSS communities while selecting OSS components to this product?

If there are other important criteria, please indicate these in the last row.

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. End user support from the community or OSS provider						
b. Quality and availability of documentation						
c. Quality and availability of defect and feature trackers						
d. A commercial actor guaranteeing the product quality of the OSS component						
e. Quality and availability of roadmaps and future plans for the OSS component						
f. Activity on mailing lists and in forums						
g. Response time on requests in forums and on mailing lists						
h. Release frequency of the OSS component						
i. Size of user base/Number of downloads of the OSS component						
j. Size of developer team developing the OSS component						
k. Reputation of the OSS component						
l. Reputation of the OSS component provider/community						
m. Quality of OSS provider web site						
n. Other (please specify)						

Development with OSS Components

Experiences from Using OSS Components

2.17 Planning, requirements and deployment: How true are the following statements related to use of OSS components in the development of this product?

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. The software product was delivered long after schedule or had severe delays						
b. Selection of OSS component took longer than estimated/expected						
c. Integration and testing of OSS component took longer than estimated/expected						
d. Requirements to your software product were changed a lot (by the customer)						
e. OSS components could not be sufficiently adapted to changing requirements from the customer						
f. OSS components were not satisfactorily compatible with the production environment when the software product was deployed						

2.18 Maintenance and quality: How true are the following statements related to use of OSS components in the development of this product?

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. The total lifetime cost (including maintenance) of the software was reduced due to use of OSS components						
b. It was difficult to identify whether defects were inside or outside the OSS components						
c. It was difficult to upgrade the software with the latest version of the OSS component						
d. It was difficult to plan and perform maintenance of your software product due to properties of the OSS components						
e. Local adaptation of OSS components resulted in high maintenance cost						
f. OSS components negatively affected quality attributes of the software (security, performance, etc.)						

2.19 Community and licenses: How true are the following statements related to use of OSS components in the development of this product?

	None at all	Small	Some	Large	Very large	NA
a. Information about the reputation and technical support ability of the community providing the OSS components were inadequate or not available						
b. Community or provider of OSS components did not provide enough technical support/ training						
c. Use of OSS components lead to legal conflicts						
d. OSS licenses restrict us from using the components the way we want						
e. It is difficult to influence the evolution of OSS components						

Interaction with and Participation in OSS Projects

2.20 During the development of this software product has your local business unit interacted with or participated in any OSS projects outside the company?

An 'OSS project' spans a bit wider than a normal project, and includes both the development of an OSS and the community of users and software developer around this OSS.

By Interaction and participation we think of: reading forums, posting messages on forums, posting defect reports, sharing code etc.

	Yes (Go to question 2.21, page 27)
	No (Go to Part 3, page 31)

Select one typical **OSS project** where you have been participating in or interacting with within the context of the development of the selected software product. Answer the following questions based on your experiences with this OSS project.

2.21 What is the name of the selected OSS project?

Name:

Motivations behind Interaction with and Participation in OSS Projects

2.22 How does the company or business unit organize participation in or interaction with this OSS project?

[Mark one or more alternatives]

<input type="checkbox"/>	As an internal project in the business unit
<input type="checkbox"/>	Through the line organization
<input type="checkbox"/>	At an individual level
<input type="checkbox"/>	Participation is not organized by the company, but voluntarily by individuals

2.23 How important are the following motivational factors to your local business unit related to participation in or interacting with this OSS project?

If other motivations are important to your local business unit, please indicate them at the last row. [Mark one alternative per row]

	Not important at all	Unimportant	Neither important or unimportant	Important	Very Important	N/A
a. Influence the OSS project						
b. Acquire new knowledge by learning from participation in the OSS project						
c. Improve relationship to the OSS project						
d. The OSS is used inside the company or in the company's software/products						
e. We provide commercial services related to the OSS						
f. Idealism						
g. Increased publicity						
h. The company wants to be identified as an OSS company						
i. Other (please specify)						

Activities in this OSS Project

2.24 To what extent does your local business unit perform the following activities, or contribute the following artefacts or resources to this OSS project?

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. Participate on mailing lists and forums						
b. New requirements to the OSS						
c. New features (code)						
d. Architecture or design						
e. Testing						
f. Defect-reports						
g. Defect-fixes						
h. Documentation						
i. Local adaptations of the OSS						
j. Supplementary or add-on OSS						
k. Commercial supplementary or add-on software						
l. Integration towards other software (glue ware)						
m. Free support						
n. Commercial support						
o. Distribution of software						
p. Infrastructure (e.g. servers or storage space)						
q. Financial support						
r. Other (please specify)						

Satisfaction with the Relationship to the Community

2.25 To what extent is your local business unit satisfied with the following elements in this OSS project?

[Mark one alternative per row]

	None at all	Small	Some	Large	Very large	NA
a. Documentation (availability and quality)						
b. Functionality of the OSS						
c. Security in the OSS						
d. Architecture of the OSS						
e. Other non-functional quality attributes of the OSS						
f. Code quality						
g. Support (in case of problems)						
h. Response to defect-reports						
i. Response to new requirements/feature requests						
j. Response time to requests on mailing lists and forums						
k. Your ability to influence the OSS project						
l. Other (Please specify)						

Part 3) Demographic Information and Comments

Information about the Company

3.1 What is the name of the company/business unit?

--

3.2 Where, is your company/business unit located (country and city)?

--

3.3 What kind of company is your company?

[Mark one alternative]

<input type="checkbox"/>	Stand-alone: The highest reporting entity with no parent organization above it.
<input type="checkbox"/>	Subsidiary: An independent entity with majority interest held by a parent organization.

3.4 What is the ownership of your local business unit?

[Mark one alternative]

<input type="checkbox"/>	Publicly traded on a stock exchange
<input type="checkbox"/>	Privately held company
<input type="checkbox"/>	Government, education, or non-profit organization

3.5 How many people are:

[The staff size may be the same for small and medium sized companies]

Working in your company world wide?	
Working in your company in your own country?	
Working in your local business unit?	
Working with software development in your local business unit?	

3.6 What is the main business area of your company or business unit?

[Mark one alternative]

	IT/ Telecom industry (Ericsson, Nokia etc.)
	Telecom service provider (Telenor, Telefonica, Voodaphone etc.)
	Software House / Software Vendor (SAP, Microsoft etc.)
	Software / IT consulting company (Accenture, CapGemini, TietoEnator etc.)
	Retail product with large degree of embedded software (Philips, Sony etc.)
	Research and development
	Education (universities etc)
	Health Care (hospitals etc)
	Public services/government (Ministries, offices etc.)
	Other software-intensive company (please specify)

3.7 Approximately how much of the local business units total turnover comes from OSS related services or software development?

We are interested in services related to OSS products or software development either of OSS or where OSS is included as part of your products.

Information about the Respondent

3.8 What is your gender?

	Male
	Female

3.9 How old are you?

	<31
	31-40
	41-50
	51-60
	>60

3.10 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 (please specify)

3.11 For how many years

have you been working in the current business unit?	
have you been working with software development?	
have you been working with OSS?	

3.12 How many

projects using OSS components have you been involved in?	
OSS communities have you been involved in (as a developer or active user)?	

3.13 What is your highest completed education?

[Mark one alternative]

<input type="checkbox"/>	Bachelor (BSc)
<input type="checkbox"/>	Master (MSc)
<input type="checkbox"/>	Ph.D.
<input type="checkbox"/>	Other education (please specify)

3.14 Is your degree software-related for instance informatics, computer science or telecommunications?

[Mark one alternative]

<input type="checkbox"/>	Yes
<input type="checkbox"/>	No

3.15 If you are interested in the results of this survey, please provide your e-mail address.

This information will of course be treated strictly confidential.

--

3.16 If applicable for the study, could you be interested to participate in an interview about OSS related software development?

If you are interested, please provide us your email address above.

<input type="checkbox"/>	Yes
<input type="checkbox"/>	No

Comments

Thank you for taking the time to answer this questionnaire. If you have any feedback on this questionnaire then please provide it in the field below. If you had to make any assumptions on any on the questions these can also be provided here.

Appendix C

On-going ITEA 2 Projects

This list of on-going ITEA 2 projects has gathered from the ITEA 2 Programme's home page at: <http://www.itea-office.org/projects>.

Short name	Name/description
3D Testbench	
AMIE	Ambient Intelligence for the Elderly
ANSO	Autonomic network for SOHO users. Pervasive computing for digital homes
BOON COMPANION	Investigation and demonstration of an autonomous cognitive system (ACS) integrating perception, reasoning, and learning
CAM4Home	Collaborative Aggregated Multimedia for Digital Home
CANTATA	Content Analysis & Networked Technologies towards Advanced and Tailored Assistance.
COSI	Co-development with inner and Open source in Software Intensive products
D-MINT	Deployment of Model-Based Technologies to Industrial Testing
E-CONFIDENTIAL	Trusted Security Platform.
EASY WIRELESS	Allow seamless roaming between wireless networks while maintaining Quality of Service
EMODE	Enabling Model Transformation-Based Cost Efficient Adaptive Multi-modal User Interfaces
ENERGY	Empowered NETwoRk manaGement
EPAS	Electronic Protocols Application Software.
ESNA	European Sensor Network Architecture.
ES_PASS	Embedded Software Product-based ASSurance
EUROSYSLIB	European Leadership in System Modeling and Simulation through advanced Modelica Libraries
EVOLIFE	Evolutionary Life Cycle Management
FLEXI	Flexible Global Product Development and Integration
GENE-AUTO	Automatic Software Generation for Real-Time Embedded Systems.
GGCC	Global GNU compiler collection.
HD4U	High definition TV for Europe
LINDO	Large scale distributed INDexation of multimedia Objects
LOMS	Locale Mobile Services
MARTES	Model driven approach to Real-Time Embedded System Development
MERLIN	Embedded Systems Engineering in Collaboration
MoSiS	Model-driven development of highly configurable embedded Software-intensive Systems
NUADU	The goal of NUADU (Celtic god of healing) is to explore the opportunities for providing 'healthcare and wellness' services and applications that facilitate more cost effective and efficient solutions.
OSIRIS	Open Source Infrastructure for Run-time Integration of Services
PASSEPARTOUT	Exploitation of advanced AV content protocols (MPEG 4/7)
PELOPS	Networked Media for Sport production Workflow
ParMA	Parallel Programming for Multi-core Architectures
S4ALL	Services for all, an implementation of the concept of Ambient Service Space
SEMEASY	Semantic makes Middleware EASY.

SERIOUS	Software evolution, refactoring, improvement of operational and usable systems
SERKET	Open platform as a global approach to security provision of places, locations and events
SMARTTOUCH	Browsing Through Smart Objects Around You.
SODA	Service-Oriented Device & Delivery Architectures
SPICES	Support for Predictable Integration of mission Critical Embedded Systems.
SUMO	Service Ubiquity in Mobile and Wireless Realm
TECOM	Trusted Embedded Computing
TIMMO	Timing Model
TWINS	Optimizing hw-sw Co-design flow for software intensive systems
Trust4All	Trust For All
UseNet	Ubiquitous M2M Service Networks
Wellcom	Deployment / management of services / applications in a Distributed Home Environment

Appendix **D**

Surveying Roles Industrial Roles in Open Source Software Development

The following paper has been accepted as a short-paper for the Third International Conference on Open Source Systems June 11-14 2007, Limerick, Ireland. The paper will appear in the conference's proceedings, published by Springer.

SURVEYING INDUSTRIAL ROLES IN OPEN SOURCE SOFTWARE DEVELOPMENT

Øyvind Hauge, Carl-Fredrik Sørensen, Andreas Røsdal
Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway

Abstract: Industry uses Open Source Software (OSS) to a greater and greater extent. We have defined four industrial OSS roles; OSS provider, OSS integrator, OSS participant and Inner Source Software (ISS) participant. Based on these four roles we have performed a survey in the ITEA COSI project. We provide initial answers to what motivates companies to undertake these roles, what are the advantages and challenges of undertaking them, and which development practices they use while undertaking these roles.

Key words: Open Source, Industry, Roles, Survey, Motivations, Development Practices

1. INTRODUCTION

The cost of producing software from scratch goes hand in hand with the steadily increasing size and complexity of the software. Reuse of standard components has been seen as one solution to keep costs down. Reusable components have been developed in-house or acquired from other vendors.

OSS provides quality software, enables new ways of developing software, and makes new business strategies possible. OSS can be important in the battle against constantly larger and more complex software. Several major industrial actors like Sun Microsystems, Oracle, IBM, and Novell, have already started to benefit from OSS.

The entry of industry into the OSS field opens up a new research arena. The ITEA COSI project wants to increase the understanding of how industry can benefit from OSS. As part of the ongoing work in the ITEA COSI project we have performed a survey of current OSS development practices in

parts of the European software industry. The survey gave several interesting indications. The availability of OSS is perhaps the most important reason behind use of OSS. The main advantages for a company having an OSS product come from, value added by supplementary products and community innovation. Attracting and supporting an OSS community requires hard work and there are challenges related to community contributions.

We start by presenting the four industrial OSS roles and the applied research method before we present our results and sum up with a discussion and conclusions.

2. RELATED WORK AND INDUSTRIAL ROLES

Our literature survey did not discover many empirical studies of industrial OSS involvement. However, examples can be found e.g. [1-5].

We want to highlight the need for more varied and reproducible empirical research. The majority of the publications we found were case studies or experience reports which are hard to reproduce. The work is in many cases performed in only one setting, most often in a non-industrial setting.

Based on literature and conversations with the industrial partners of the ITEA COSI project we defined four industrial roles: OSS Provider, OSS Integrator, OSS Participant, and Inner Source Software (ISS) Participant.

An *OSS provider* is a company which controls the code base of an OSS product. MySQL, Trolltech, and Sun Microsystems are some examples. The *OSS integrator* is a company which, uses OSS components in their products or build their products on top of OSS infrastructure. The *OSS participant* is a company actively interacting with one or more OSS projects. IBM and SUN are for instance participating in the development of the Apache DB. The *ISS participant* is a company participating in an inter department or inter company collaborative development using OSS development practices.

3. RESEARCH METHOD

In the first phase of the ITEA COSI project, we wish to create a baseline description of the industrial OSS related development. The following questions were based on a literature review and in conversations with project partners: Why do industrial actors undertake the four OSS roles? What are the advantages and challenges related to undertaking them? Which development practices are used in these roles?

Based on these questions, we created an interview guide which was used in semi-structured interviews with Norwegian COSI partners. The interviews were performed at the offices of the industrial partners and all of them were recorded and later transcribed.

We interviewed two developers in company A, one developer in company B, and one developer and one CEO in company C. Company A is a small company which uses OSS in their development. Company B is a medium sized consulting company delivering services and products based on OSS. Company C is a medium sized company which provides an OSS product.

The interview guide and the results from the interviews were used as a basis for a web-survey. The survey had one part for each OSS role.

The ITEA COSI project consists of big companies from telecom and embedded software, but also smaller and more traditional software companies. Selection of the respondents was because of the composition of the project, unfortunately out of our hands. We distributed the survey to the all of the project partners and encouraged them to respond at least once. The companies selected their respondent(s) themselves and we received the following number of responses; OSS provider: 3, ISS participant: 6, OSS participant: 6 and OSS integrator: 9, in total 24 responses.

4. RESULTS

OSS providers are motivated to release their products as OSS of several factors. The community can perform testing and provide new functionality, bug-fixes, bug-reports, and translations. This may enhance the functionality and increase the quality of the product. The community members may contribute to the innovation of the product in form of new ideas and new requirements. They can also provide supplementary products and services.

Releasing a product as OSS is a way to make it available to a large user group. If the community is satisfied with the product, it will most likely share its experiences with others and thereby give the OSS provider free marketing and increased publicity.

Increased value, availability and publicity, boost the possibility of attracting new users. This is important because many industrial OSS providers sell services related to their OSS products. The more users, the more potentially paying customers and the more likely it is that someone will contribute to the development of the product.

We believe that the innovation and the supplementary products and services which increase the value of the product are more important than

code contributions. This is because the OSS provider has to review contributions in form of code, requests, and opinions.

Maximizing community contributions and reducing the work related to these contributions is one of the challenges an OSS provider faces. Attracting a community is another major challenge for an OSS provider and according to our respondents, hard work.

It is important to offer the community a piece of quality software they need, infrastructure to support the community, enough documentation and information to get the community members going and to make them feel involved. However, it is important not to involve the community too much because involvement will create overhead and delays.

The OSS integrator is motivated by the low purchase price of the OSS products. Perhaps even more important is the high availability of OSS. Standard compliance was also mentioned as a reason why people use OSS.

Many OSS products are available through project web sites containing documentation, forums and mailing lists, bug and feature trackers, road maps, developer info and so on. The honesty about the true status of the OSS product and the availability of information make it easier for the OSS integrator to understand and evaluate it.

OSS components are primarily selected through informal processes. The OSS integrator discovers a need for a component. He forms an initial idea of what the software should do. Based on these initial requirements he performs an informal search to create a long-list. This long-list is later reduced to a short-list. The components on the short-list are tested or evaluated closer before one product is selected.

The candidate components may be found through many sources; prior experience, friends or co-workers, request for help on forum or mailing-list, searches in OSS portals or search engines. Search engines are used to find both single components and comparisons of several components.

Missing functionality, incompatible licenses, unfamiliar programming languages, lack of stable releases, no activity in community, bad or no reputation, and absence of documentation, are easy-to-check evaluation criteria. To evaluate the components further the developer may subscribe to mailing lists, study documentation, perform code reviews, and test the software in a small prototype. Plans and roadmaps, compatibility to other software, standard compliance, reputation of the product and the provider, the development process used in the community, and support from community or a commercial provider, were all mentioned as evaluation criteria in this process. This evaluation was mostly informal but some respondents reported that they used checklists.

The OSS integrator is faced with some challenges. There are vast numbers of OSS available out there and finding quality products can be hard.

By changing the source code of the OSS products he uses, the OSS integrator is left with two choices: He can keep the changes to himself or feed the changes back into the product. Convincing the OSS project to include these changes can be hard. If he is unable to make the OSS project include his changes he has to maintain this code himself. This could be time-consuming and it may lead to problems with new releases of the OSS.

Most of the **OSS participants** could not surprisingly be classified as active or passive users. They provide occasional bug fixes and requirements, subscribe to mailing-lists, read news, and primarily use the software.

The respondents were overall satisfied with the OSS products, their communities, information from the community, and their relationship with the community. However, they acknowledged that they would have been able to influence the community more through increased participation.

Participation as a company was not surprisingly rooted in the need for the product. Learning was also mentioned as one important motivation for some companies. On the individual level learning, idealism, and personal interest in the product were mentioned as the most important factors.

The participants in ISS development use some development practices often used in OSS development. The use of e-mail and mailing list was due to the distributed development quite extensive.

To provide the participating developers a shared view of the code, code repositories were used. These repositories were controlled by gatekeepers or module owners. Based on the code base, several pre-releases of the software were made available to give the users an early impression of the product and to allow the users to provide feedback to the developers.

Some of the respondents reported saved development effort and maintenance effort due to ISS cooperation.

5. DISCUSSION AND CONCLUSIONS

In the section about related work we requested more and more varied empirical research related to industrial OSS involvement. We are aware of some of the limitations of our own work and we will discuss some of these here.

The survey was intended to be a baseline for the companies in the ITEA COSI project. The selection of respondents was done from this population and we cannot claim that our results are valid for other populations.

The number of respondents was unfortunately quite low. The selection of respondents was done by convenience sampling. We were, due to the sampling method, unable to control mortality rates and drop out rates for the

questionnaire. These factors reduce the internal validity and the statistical validity of the survey.

We have however increased the validity through interviews with some of the respondents and through expert review. We have presented the results to the ITEA COSI project and to several of the respondents. None of them gave us any indications that the results were flawed.

We believe that our work is a step on the way to understand how industry can benefit from OSS products and development methodologies. The survey has given us initial ideas of what motivates companies to undertake the four roles OSS provider, OSS integrator, OSS participant, and ISS participant. Furthermore, we have described some of the advantages and challenges related to undertaking these roles. At last we have started to describe some of the processes and practices used by these roles.

The work of answering the initial questions about motivations, processes, advantages and challenges are by far not completed. We will continue this work and a second version of the survey is under development. This survey will be distributed to a larger European population through ITEA.

ACKNOWLEDGEMENT

The Norwegian COSI is sponsored by the Norwegian Research Council's IKT-2010 program. The COSI project is part of the ITEA 2 program.

REFERENCES

1. W-G. Bleek, M. Finck, and B Pape, Towards an Open Source Development Process? Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model, *Proceedings of the First International Conference on Open Source Systems*, Genova, Italy, 37–43 (2005)
2. C. Jensen and W. Scacchi, Collaboration, Leadership, Control, and Conflict Negotiation and the Netbeans.org Open Source Software Development Community, *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 196b-196b, (2005).
3. V. K. Gurbani, A. Garvert, and J.D. Herbsleb, A Case Study of a Corporate Open Source Development Model, *Proceeding of the 28th international Conference on Software Engineering ICSE '06*, Shanghai, China, 472–481 (2006)
4. C. Rossi and A. Bonaccorsi, Why Profit-Oriented Companies Enter the OS Field? Intrinsic vs. Extrinsic Incentives. *Proceedings of the fifth Workshop on Open Source Software Engineering*, 1–5 (2005)
5. L. Dahlander and M. G. Magnusson, Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms. *Research Policy*, 34(4), 481–493 (2005)

Appendix **E**

How to Perform a Web Survey

This article contains advices on how to perform a web-based survey. The article is meant to be a guide for studentes and researchers who are in the same situation as I was when I first started to work with web-based surveys. The primary audience is other students here at NTNU and I have therefore written the guide in Norwegian.

The guide is based on experiences from my depth study, my master thesis, from litterature, and advices from my supervisors here at IDI professor Reidar Conradi and dr. Carl-Fredrik Sørensen, and from professor Ola Listhaug at The Department of Sociology and Political Science.

Å gjennomføre sin første spørreundersøkelse

Øyvind Hauge

29. september 2007

1 Introduksjon

Det er viktig å gjennomføre empirisk arbeid relatert til utvikling av programvare [Tichy 1998]. Surveys og spørreundersøkelser er et av verktøyene i empirisk arbeid. Dessverre er gjennomføringen av slike surveys ressurskrevende og utfordrende, spesielt som fersk student uten særlig erfaring med forskningsarbeid.

I løpet av prosjekt og diplomoppgave har vi gjennomført tre forholdsvis store web-baserte undersøkelser i norsk og europeisk programvareindustri. Gjennom dette arbeidet har vi samlet en del erfaringer som kan deles med andre studenter og ferske forskere.

Denne artikkelen skal la andre kunne lære av våre erfaringer. Artikkelen er ment å fungere som et innspill i arbeidet med surveys. Den inneholder på ingen måte fasitsvar og den er heller ikke en komplett innføring i temaet.

2 Planlegning

Å definere formålet med et hvert forskningsarbeid er viktig. Dette er spesielt viktig i arbeid med spørreskjema av flere årsaker. For det første er det ikke sikkert at det er hensiktsmessig å benytte seg av en spørreundersøkelse. Hvis man har liten kunnskap om det man skal spørre om, hvis det er få mulige respondenter, eller hvis emnet man ønsker å studere er komplekst bør man vurdere om det er andre metoder som er bedre egnet.

Videre skal målet skal veilede og avgrense det videre arbeidet med undersøkelsen. Det blir fort tidkrevende å besvare et spørreskjema med mange spørsmål. I tillegg kan det bli en stor jobb å analysere svarene i etterkant. En god regel er å bare spørre om det som er ytterst nødvendig [Cooper and Schindler 2006]. Det er lett å la seg inspirere til å inkludere, bare et spørsmål til. I slike tilfeller bør du spørre deg om spørsmålet er i tråd med formålet med undersøkelsen.

Fravær av et klart og veldefinert formål var kanskje det største problemet vi hadde i vårt arbeid. Dette medførte at det var vanskelig å fokusere på hva vi virkelig skulle spørre om. Det var ikke minst vanskelig å avgrense seg. I en del tilfeller vil interessante ting falle utenfor omfanget av undersøkelsen. Når dette skjer er det godt å ha et klart mål og klare forskningsspørsmål som kan hjelpe deg med å avgjøre hva som er innenfor og hva som er utenfor undersøkelsens omfang.

Definer problemstilling og mål før du starter med selve spørreskjemaet slik at du kan vurdere om et spørreskjema er et egnet verktøy og slik at du har mulighet til å begrense omfanget av undersøkelsen.

Studieenheten er den enheten eller det fenomenet man ønsker å studere. Dette kan for eksempel være et prosjekt, en bedrift, eller et individ. Det er viktig at studieenheten defineres klart før undersøkelsen startes. Studieenheten må komme klart frem av målet med forskningsarbeidet. Spørsmålene i spørreskjemaet må konsentrere seg om studieenheten, samtidig som det er viktig at studieenheten kommer klart frem i spørreskjemaet. Respondenten vil da vite hva eller hvem han skal svare for. **Definer studieenheten tidlig og vær tydelig på hva studieenheten er gjennom hele spørreskjemaet.**

3 Utforming av spørreskjemaet og kvalitetssikring

Spørsmålene i undersøkelsen må være utformet på en slik måte at de oppmuntrer til at respondentene skal bevare dem. Respondenten må både være villig til og i stand til å besvare spørsmålene. Spørsmål som ber om informasjon respondenten ikke er i stand til eller villig til å gi fra seg blir ikke besvart. Videre er det viktig å bruke et lettlest og forståelig språk hvor vanskelige termer unngås. Bruk av vanskelig språk og termer kan virke skremmende, noe som igjen reduserer svarprosenten eventuelt øker antall "vet ikke" svar. Det anbefales å starte med de spørsmålene som er enkle og interessante [Cooper and Schindler 2006]. De vanskelige og mindre interessante spørsmålene bør heller komme lenger ut i undersøkelsen.

Svaralternativene må være komplette og gjensidig utelukkende. Det vil si at alle mulige alternativer må inkluderes og at det ikke kan være flere kategorier som inkluderer de samme alternativene. For å få kompletthet kan man vurdere å inkludere en kategori "vet ikke" eller "annet". "Annet" kan for eksempel gi respondenten muligheten til å skrive en kommentar. For å oppnå gjensidig utelukkende alternativer bør man heller velge alternativer som "1-5" og "6-10" enn "1-5" og "5-10".

I de tilfellene der respondenten kan tenkes å ville svare mer enn et svar må de gis muligheten til det, slik at de ikke tvinges til å svare noe som ikke er helt korrekt. På spørsmålet "hva liker du?" bør det være mulig å svare både "sjokolade" og "is", ikke enten eller.

Utform et spørreskjema som er lett og motiverende å svare på slik at respondenten ikke tvinges til å svare på noe han er usikker på eller ukomfortabel med.

Spørreskjemaet bør selvfølgelig kvalitetssikres. Det er ofte lettere å utarbeide et godt skjema i samarbeid med andre. Da har man muligheten til å forbedre hverandres forslag. Gjennomlesninger bør også gjøres av andre, for eksempel klassekamerater, veiledere, kollegaer osv. I tillegg til slike gjennomlesninger bør det absolutt gjennomføres pretester.

Hvis mulig, velg ut noen få respondenter og få dem til å svare på undersøkelsen samt kommentere spørsmålene og eventuelle problemer under gjennomføringen av undersøkelsen. Slike pretester kan gi det verdifulle tilbakemeldinger i arbeidet med utarbeidelsen av et godt spørreskjema. I tillegg kan de gi deg en pekepinn på hvilke resultater du kan forvente deg og på hvilke analysemetoder du kan bruke.

Få andre til å lese gjennom spørreskjemaet ditt og gjennomfør pretester.

4 Populasjon, utvalg og oppfølging

For å få gode resultater og for å kunne diskutere validiteten av undersøkelsen din, er det viktig å kjenne populasjonen du undersøker. Det beste er å danne seg et så komplett bilde av populasjonen før du starter, og å bruke metoder for tilfeldig utvalg.

I en av undersøkelsene våre hadde vi ikke direkte tilgang til populasjonen og dermed ingen mulighet til å trekke ut et representativt utvalg. Dette gjorde det veldig vanskelig å beskrive populasjonen og det gjorde det veldig vanskelig å si noe om validiteten til resultatene. I en annen undersøkelse var vi ikke flinke nok til å definere utvalget. Dette medførte at vi inkluderte for mange bedrifter i deler av populasjonen, noe som igjen medførte mer arbeid enn nødvendig. I en tredje undersøkelse hadde vi derimot muligheten til å kontakte populasjonen direkte. For å kartlegge denne kontakten vi de via e-post. I denne e-posten stilte vi fire enkle spørsmål og oppnådde en svarrate på mellom 60 og 70 % uten å sende ut påminnere.

Sørg for å ha direkte tilgang til populasjonen du skal undersøke, beskriv populasjonen, og bruk tilfeldig utvalg om mulig.

For å få folk til å svare er det viktig at temaet og spørsmålene interesserer respondentene. Hvis utvalget gjenkjenner seg i problemstillingen eller føler en tilhørighet til temaet er det mer sannsynlig at de svarer. En slik tilhørighet kan oppnås ved at tema er relevant for dem, eller for eksempel for arbeidsgiveren eller prosjektet deres. Hvis undersøkelsen er relevant for en arbeidsgiver eller annen organisasjon hvor respondenten har tilhørighet vil det være veldig positivt å få synlig støtte fra denne organisasjonen. I tillegg kan du involvere utvalget ved å personalisere invitasjonen til den enkelte. Dette er tidkrevende, men det kan bidra til å øke antall svar betraktelig. En invitasjon per telefon har normalt en vesentlig større effekt en invitasjon på e-post.

Å få støtte fra en seriøs organisasjon bidrar også til at du fremstår som seriøs. Dette er viktig. Invitasjonen, spørsmålene, og layouten til undersøkelsen bør også ha et seriøst og profesjonelt preg.

En form for belønning kan også bidra til å øke responsraten. Hvis temaet er interessant for de som svarer, kan tilgang til rapporten din eller en oppsummering av resultatene kanskje være belønning nok.

Involver respondentene og gjør det interessant og enkelt å besvare undersøkelsen.

I etterkant av invitasjonen bør det gjennomføres en oppfølging av de som ikke har svart. Først bør de oppmuntres til å svare. Hvis de ikke gjør dette, bør du undersøke hvorfor de ikke gjør det. Dette kan gjøres ved å plukke ut et lite utvalg av de som ikke svarte og ta kontakt med dem for eksempel per telefon. Denne oppfølgingen vil sannsynligvis bidra til å øke responsraten din samt gi deg verdifull informasjon om ikke-responder.

I noe av vårt arbeid hadde vi beklageligvis ikke direkte tilgang til populasjonen/respondentene. Som konsekvens av dette var det vanskelig å kontrollere hvilke bedrifter som svarte, hvem som svarte i den enkelt bedriften, og hvilke bedrifter som ikke svarte. Dette gjorde det umulig å gjennomføre oppfølging av responder.

Følg opp de som ikke svarer og undersøk hvorfor.

5 Datainnsamling

Et ferdig verktøy for utforming av spørreundersøkelsen og innsamling av data vil antagelig være til stor hjelp. Slike verktøy gir (oftest) et profesjonelt utseende, mulighet for rapportering og for utforming av forskjellige spørsmålstyper osv. Dette kan i de fleste tilfeller være vesentlig lettere enn å utvikle et slikt verktøy selv og det vil spare deg tid og for mye frustrasjon, spesielt hvis omfanget av undersøkelsen er stort.

Bruk et ferdig verktøy for utforming av spørreskjema.

Mange ganger kan det være fristende å sette i gang med datainnsamling, umiddelbart. Det kan ofte gå bra, men det kan også straffe seg. Derfor kan det være lurt å tenke på hva man skal bruke data til før man setter i gang. Vi opplevde beklageligvis at vi satte i gang litt for raskt ved enkelte anledninger. Dette medførte av vi i noen sammenhenger mistet data og i andre sammenhenger at vi skapte merarbeid for oss selv.

Sikkerhetskopiering og versjonskontroll av filer bør være en selvfølge. Hvis det er flere personer som skal jobbe med de samme data er det avgjørende at man har en enighet om hvordan man fører data og hvordan man unngår overskrivninger av data. Bruk nok tid på å avklare hvordan data skal kodes og lagres slik at dere unngår problemer med ukonsistenet eller manglende data.

I tillegg bør man tenke seg om hvis man har tenkt å overskrive eller slette informasjon. Vi kontaktet en rekke bedrifter via deres kontaktadresser (feks info@bedrift.no). Disse adressene lagret vi, men da vi fikk svar fra en ansatt (ole@bedrift.no) overskrev vi de opprinnelige adressene. I ettertid så vi at vi kanskje heller burde ha laget en ekstra kolonne i regnearket vi brukte slik at den opprinnelige adressen ikke gikk tapt. Det samme gjelder når man skal slette ting. Det er kanskje bedre å arkivere enn å slette.

En annen ting man også bør tenke på er hvordan man behandler data fra flere kilder. Vi så for eksempel på antall ansatte i en bedrift. I noen tilfeller fikk vi data fra bedriftens hjemmeside, fra svar på vår forespørsel, og fra Enhetsregisteret (Brønnøysund). Alle tre kildene gir opplysninger om den samme egenskapen, men har i mange sammenhenger forskjellig verdi. Det kan derfor være lurt å lagre data fra forskjellige kilder hver for seg og eventuelt slå de sammen på et nytt sted om nødvendig.

Bli enige om hvordan data skal kodes, lagres, og brukes.

6 Analyse

I tillegg til at spørsmålene i spørreskjemaet skal være relaterte til målet med studien bør man også ha en klar tanke om hvordan man skal bruke dataene i etterkant. Hvis man ikke vet hva man skal bruke data til, eller hvordan man skal bruke dem, bør man vurdere om man i det hele tatt trenger dataene. For spørsmål hvor man ikke har direkte brukt for dataene bør man vurdere om de kan utelates fra spørreskjemaet.

Enhver analyse bør starte med en gjennomgang av datamaterialet og deskriptiv statistikk som grafer, diagrammer, tabeller osv. Se for eksempel [Cooper and Schindler 2006; 472-488] eller [Wohlin et al. 2000; 82-90] for en oversikt av slike metoder. Ikke forkast ekstremverdier umiddelbart, men prøv å finn en bakenforliggende forklaring.

Før du gir deg i gang med annen statistisk behandling anbefales det å diskutere det med noen som har god greie på bruk av statistisk analyse. Verktøy som SPSS kan hjelpe deg med

regningen, men de kan ikke hjelpe deg med å velge riktig metode. Regneark som MS Excel inneholder også en del grunnleggende statistikk som i mange tilfeller vil være tilstrekkelig. Vår erfaring viser imidlertid at en del operasjoner går vesentlig raskere i verktøy som for eksempel SPSS. Hvis omfanget av dataanalysen er stort anbefales det å vurdere verktøy som er laget spesielt for statistisk analyse.

Bruk deskriptiv statistikk for å beskrive data, konsulter erfarne statistikere for bruk av riktig statistiske metoder om nødvendig, og bruk egnede verktøy.

7 Validitet

Diskusjoner om validitet bør alltid være en del av et forskningsarbeid. En slik diskusjon kan inneholde antall spørsmål, responsrater, antall respondenter, utvalgsstørrelse, utvalgsmetode, populasjon, stryken til statistiske tester, trusler mot validitet, og tiltak du har gjort for å forbedre validiteten. Mange lærebøker inneholder oversikt over utallige trusler og forskjellige kategoriseringer av validitet og trusler mot validitet. En studie behøver imidlertid kun å inneholde det som er relevant for studien.

For å øke validiteten ved bruk av spørreundersøkelser bør man gå gjennom resultatene med kritiske øyne og sammenligne dem med resultater fra andre studier. Man bør presentere resultatene for eksperter og/eller respondentene og få deres tilbakemelding på resultatene. I tillegg kan man bruke andre forskningsmetoder for å belyse den samme problemstillingen (trianglering).

8 Oppsummering

Kanskje det viktigste med gjennomføring av spørreundersøkelser er planlegging og utforming av et klart mål. Formålet fokuserer, avgrenser, og det vil hjelpe deg mye i ditt arbeid. Videre vil det være gunstig å kunne støtte seg til andre og deres erfaring i utarbeidelsen av et spørreskjema. I felleskap har man større mulighet til å avdekke og korrigere potensielle svakheter.

9 Litteratur

Bøkene av Ringdal og Jacobsen gir begge gode introduksjoner til forskningsmetodikk, skrevet på norsk. Fra boken til Ringdal er spesielt kapittel 5 Forskningsdesign, 7 Utvalg og enheter, 13 Survey-metoden, og 14 Kvalitativ dataanalyse interessante [Ringdal 2001]. Fra boken til Jacobsen anbefales kapittel 5 Utvikling av problemstilling, 12 om innsamling av kvantitativ informasjon, 13 om utvalg av enheter, 14 om analyse, og 15 om vurdering av konklusjoner [Jacobsen 2005].

En artikkelserie av Shari Lawrence Pfleeger og Barbara A. Kitchenham på seks artikler fra 2001-2003 gir også en del innspill i forhold til gjennomføring av spørreundersøkelser innen software engineering [Pfleeger and Kitchenham 2001, Kitchenham and Pfleeger 2003]. [Conradi et al. 2005] diskuterer en del utfordringer relatert til landsdekkende undersøkelser innen IKT-bransjen, spesielt i forhold til populasjon og utvalg.

For større og mer inngående diskusjoner om hvordan man kan utvikler spørreundersøkelser kan [Dillman 2007] eller [Groves et al. 2004] anbefales. Begge bøkene er omfattende verk som kun tar for seg surveys/spørreundersøkelser. Bøkene er fra samfunnsvitenskap, men de inneholder erfaringer som er overførbare til IT-sektoren.

10 Takk til

Den første surveyen ble gjennomført i samarbeid med Andreas Røsdal, våren 2006, og jeg må rette en stor takk til han. Jeg vil også takke veilederne mine her ved IDI, professor Reidar Conradi og Dr. Carl-Fredrik Sørensen. I tillegg må jeg også takke professor Ola Listhaug ved Institutt for sosiologi og statsvitenskap for all hjelp og alle tilbakemeldinger.

Referanser

- Conradi, R., Li, J., Slyngstad, O. P. N., Kampenes, V. B., Bunse, C., Morisio, M., and Torchiano, M. (2005). Reflections on Conducting an International Survey of Software Engineering. In Verner, J. and Travassos, G. H., editors, *Proceedings on International Symposium on Empirical Software Engineering ISESE 05*, pages 214–223.
- Cooper, D. and Schindler, P. S. (2006). *Business Research Methods*. McGraw Hill, 9th edition. ISBN: 007-124430-1.
- Dillman, D. A. (2007). *Mail and Internet Surveys The Tailored Design Method*. John Wiley & Sons, Inc., second edition, 2007 update edition.
- Groves, R. M., Jr., F. J. F., Couper, M. P., Lepkowski, J. M., Singer, E., and Tourangeau, R. (2004). *Survey Methodology*. John Wiley & Sons, Inc.
- Jacobsen, D. I. (2005). *Hvordan gjennomføre undersøkelser?* HøyskoleForlaget. ISBN: 82-7634-663-4.
- Kitchenham, B. A. and Pfleeger, S. L. (2003). Principles of Survey Research Part 6: Data Analysis. *SIGSOFT Softw. Eng. Notes*, 28(2):24–27.
- Pfleeger, S. L. and Kitchenham, B. A. (2001). Principles of Survey Research Part 1: Turning Lemons Into Lemonade. *SIGSOFT Softw. Eng. Notes*, 26(6):16–18.
- Ringdal, K. (2001). *Enhet og Mangfold*. Fagbokforlaget, 2nd. edition. ISBN: 82-7674-569-5.
- Tichy, W. F. (1998). Should Computer Scientists Experiment More? *Computer*, 31(5):32–40.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, Boston / Dordrecht / London. ISBN: 0-7923-8682-5.