Adoption of Open Source Software in Software-Intensive Industry

Øyvind Hauge

Doctoral Thesis

Submitted for the Partial Fulfilment of the Requirements for the Degree of

philosophiae doctor

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering
Norwegian University of Science and Technology

June 17th, 2010

ISBN 978-82-471-2112-2 (printed version)

ISBN 978-82-471-2113-9 (electronic version)

ISSN 1503-8181

NTNU Serial number 2010:76

Printed in Norway by NTNU-trykk, Trondheim

Abstract

Context: Open source software (OSS) has seen significant adoption, and it has changed not only how software-intensive companies develop and make money on software, but also how they select and acquire software. As a consequence of these changes there has been confusion about how organizations may benefit from OSS.

Objective: Through answering the following research questions (RQ1-RQ3), this thesis aims to explore and describe (1) the different ways in which organizations adopt OSS and (2) how these organizations select OSS components.

RQ1: How and to what extent are software-intensive organizations currently adopting OSS?
RQ2: What is the current status of research on OSS adoption in organizations and how may this research benefit practitioners?
RQ3: Which strategies and resources do software developers use to identify, evaluate, and select OSS components?

Method: This thesis consists of six related studies, all focused on the adoption of OSS in software-intensive organizations. These studies embody case studies, systematic literature reviews, and surveys using face-to-face interviews, e-mail, and web-based questionnaires.

Results: Based on these six studies, this thesis provides the following five contributions (C1-C5) through eight papers (P1-P8):

C1 Empirically grounded descriptions of how several organizations adopt OSS.
C2 A systematic review of the literature on OSS adoption in organizations.
C3 A classification framework presenting six ways of organizational OSS adoption, each with its particular benefits and challenges. The six ways include: deploying OSS products, using OSS CASE tools, integrating OSS components, participating in the development of OSS products, providing OSS products, and using OSS development practices.
C4 Descriptions, based on empirical evidence, of the strategies and resources practitioners actually use to identify, evaluate, and select OSS components.
C5 A model for situated software selection and its constraints, indicating why formalized selection methods have failed to see significant adoption.

Conclusion: Practitioners should observe that they have several possibilities and take this into account when adopting OSS. Research on OSS adoption should focus on a few topics, borrow more support from related fields within software engineering and information systems research, and extend the foundation offered by this thesis. Research on software selection should put stronger emphasis on the situation the selection is conducted in and the rich (text) experience developers benefit from when selecting (OSS) components.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfilment of the requirements for the degree of philosophiae doctor.

This doctoral work has been performed at the Department of Computer and Information Science, NTNU, Trondheim, under the supervision of Professor Reidar Conradi as main supervisor and Professor Letizia Jaccheri and Adjunct Associate Professor Torgeir Dingsøyr as co-advisors.

This PhD thesis is financed as an integrated-PhD study by an internal scholarship from the Department of Computer and Information Science and the Faculty of Information Technology, Mathematics and Electrical Engineering at NTNU. Some of the work has been financed and performed as part of the NorskCOSI project.

NorskCOSI was a research project funded by the Norwegian Research Council (2006-2008) through project number 174417. The project was managed by ICT Norway, and it was part of the European Co-development using inner and Open source in Software Intensive products (COSI) project under the Information Technology for European Advancement (ITEA) 2 programme.

Acknowledgements

This thesis is financed by an internal scholarship from the Department of Computer and Information Science and the Faculty of Information Technology, Mathematics and Electrical Engineering at NTNU, and to some extent by the NorskCOSI project. I would like to acknowledge both NTNU, NorskCOSI, and the Norwegian Research Council for their financial support.

Next, I would like to thank Professor Reidar Conradi for his supervision during my time as a PhD student. Furthermore, I would like to thank the participants in the COSI project who have provided very valuable input. While I have been deeply involved in all the work reported in this thesis, much of it has been done in collaboration with colleagues. I would therefore like to thank my colleagues at IDI/NTNU for our collaboration, in particular Claudia Ayala, Thomas Østerlie, Sven Ziemer, Daniela S. Cruzes, Carl-Fredrik Sørensen, and everyone participating in "Forskerfabrikken". I would also like to thank Juho Lindman, Jingyue Li and Xavier Franch for our co-authorship, and Andreas Røsdal, Marinela Gerea, Pål Haug, Ketil Sandanger Velle and Tron André Skarpenes for the collaboration related to their Master's theses.

The many anonymous individuals who have participated in our interviews and surveys have contributed more than they know and I would like to show my gratitude for sharing some of your precious time with us. Thank you!

Finally, I would like to thank my family, in particular Constanza, for their support.

NTNU, May 2, 2010
Øyvind Hauge

Contents

List of Tables

List of Figures

Abbreviations

API	Application programming interface
CBSE	Component Based Software Engineering
COSI	Co-development using inner and Open source in Software Intensive products. The COSI project was part of the ITEA 2 programme.
COTS	Commercial Off-The-Shelf
EBSE	Evidence Based Software Engineering
FLOSS	Free/Libre Open Source Software
FOSS	Free Open Source Software
FS	Free Software
FSF	The Free Software Foundation
ISS	Inner Source Software
ITEA	Information Technology for European Advancement.
NACE	Statistical Classification of Economic Activities in the European Community
NTNU	Norwegian University of Science and Technology
OSI	The Open Source Initiative
OSS	Open Source Software
SE	Software Engineering
SLR	Systematic Literature Review

Chapter 1

Introduction

This thesis is a paper collection consisting of eight papers (**P1-P8**). This chapter provides a brief overview of the reported research, including its background and scope, and its research questions. The chapter furthermore gives an overview of the papers included in the thesis and the thesis' contributions. Finally, it presents the structure of the thesis.

1.1 Background and Scope

Open Source Software (OSS) has over the last decade had a significant impact on software-intensive organizations. With software-intensive organizations we think of any public or private institution, company, or similar organizations which develop, maintain, or make heavy use of software. OSS offers these organizations a variety of possibilities and benefits. The collaborative and distributed development paradigm associated with some large OSS projects has inspired organizations to evolve their own development processes (Wesselius, 2008), and increasingly collaborate across company borders (Ågerfalk and Fitzgerald, 2008). Moreover, Fitzgerald (2006) describes how OSS has contributed to changing how software companies make money by provoking a shift from traditional license-based models into service-based business models. Finally, Ghosh (2006), shows that the easy and massive access to OSS products has changed how organizations acquire software, and has enabled significant adoption of OSS products in many domains.

Even though the adoption of OSS has been significant, there has, according to an ITEA Report on Open Source Software (2004), been much confusion around what OSS is, and how organizations may benefit from it. ITEA 2 COSI, a European research project, was therefore established to understand **how software intensive industry may benefit from OSS** and from distributed collaborative software development. Understanding how organizations benefit from or adopt OSS is also the main topic of this thesis.

One of the key challenges for the COSI project and the software-intensive industry is the continuous commodification of software (van der Linden et al., 2009). Increasingly large parts of most software systems are becoming commodity, and provide no or little advantage over competitors. Hence, it is usually better to reuse existing commodity components or to collaborate with others, than to develop new components (see Figure 1.1). At the same time, we must be aware of not giving away the intellectual property which ensures

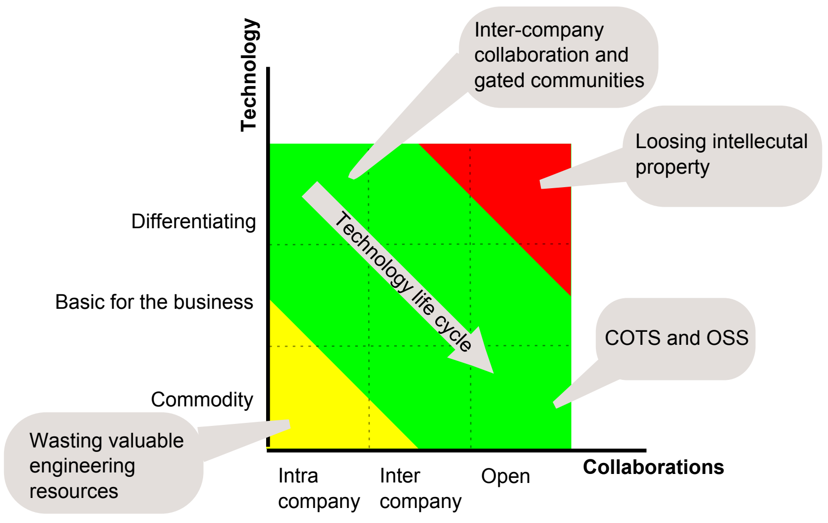


Figure 1.1: Effective and efficient software development

a competitive advantage.

In a review of several empirical studies, Mohagheghi and Conradi (2007) show how Component Based Software Engineering (CBSE) and software reuse may reduce the effort needed to develop software systems and to increase the quality of the end products. Most software systems are therefore built through integration of reusable software components (Yang et al., 2005). Given the significance of CBSE and the valuable resource OSS components constitute, it is evident that reuse of OSS has made a significant impact on the software industry.

However, even though software reuse has potential advantages, it is not unproblematic. The selection of components is according to Gorton et al. (2003), a part of the reuse process where failure can have significant consequences. The vast numbers of poorly described OSS components, and fragmented, incomplete, and untrustworthy information available over the Internet is not making component selection easier. To be able to support practitioners through improved tools and practices, it is important to understand their current practice. It is in other words important to understand **how software-intensive organizations currently select OSS components**. This leads us to the second topic of this thesis, selection of software components.

1.2 Research Questions

This thesis has three research questions aiming to explore how software-intensive organizations adopt OSS (**RQ1**), to assess the status of research on OSS in organizations (**RQ2**), and in particular how these organizations select OSS components to be integrated into software systems (**RQ3**). These are:

RQ1: How and to what extent are software-intensive organizations currently adopting

OSS?

RQ2: What is the current status of research on OSS adoption in organizations and how may this research benefit practitioners?

RQ3: Which strategies and resources do software developers use to identify, evaluate, and select OSS components?

With adoption of OSS we consider software-intensive organizations at any of the five stages of the adoption process (Rogers, 2003). This includes both organizations that plan (knowledge, persuasion, and decision) to adopt OSS and organizations that have already included OSS as part of their software development (implementation and confirmation). However, we will mainly focus on organizations that have already adopted OSS.

1.3 Conducted Research

This thesis has had a focus on empirical studies, using both qualitative and quantitative methods in the context of various software-intensive organizations from the ITEA 2 COSI project (2006-2008) and the Norwegian software industry (see Section 3.1). Moreover, in collaborations with our colleagues at the Technical University of Catalunya, Barcelona, one study was also conducted with a sample of the Spanish organizations.

The investigations in this thesis have mainly been conducted through six different studies (**S1-S6**) (see Figure 1.2). These studies have been conducted in collaboration with colleagues and Master's students at NTNU and the Technical University of Catalunya, partners in the COSI project, Telenor IT Norway, and other software-intensive organizations. A brief overview of these studies is presented below (see also Chapter 3).

S1: A survey (2006) conducted in the context of the COSI project, consisting of interviews with five employees from three COSI companies and a web-based questionnaire with 24 responses from the industrial partners in the COSI project. The study was aimed at defining a base-line for how the COSI partners adopted OSS.

S2: A survey (2007) conducted in the Norwegian software industry. The survey consisted of a large e-mail survey with more than 700 responses, a web-based questionnaire with 66 usable responses, and an additional 16 interviews with developers from different companies. The study was initiated to assess the extent of OSS adoption in the software sector and to understand how organizations adopted OSS.

S3: A case study (2006-2008) based on material from the COSI project. This material consisted of several interviews, project deliverables, informal conversations with project members, workshops, and field notes from project meetings and company visits. This study was conducted to help the COSI partners in reaching their individual goals and to increase our understanding of OSS adoption.

S4: A systematic literature review (2008-2009) focusing on OSS adoption and software engineering. The goal of this study was to create a platform for future research on OSS adoption and to use evidence from the literature to understand the different ways in which organizations adopt OSS.

S5: A survey (2009) in Telenor IT Norway, the IT department of a large telecom company, consisting of four interviews, a questionnaire with more than 80 responses, and two workshops. This study was conducted to understand the perceived advantages and risks of OSS adoption, and to identify steps to reduce these risks.

S6: A survey (2008-2009) conducted in the Norwegian and the Spanish software-intensive industry consisting of interviews with 23 software developers from different organizations. The goal of this study was to understand the actual practices used to select OSS components.

1.4 Contributions

This section gives an overview of the papers included in this thesis and the thesis' contributions. The relation between the papers, research questions, studies, and the thesis' contributions is found in Figure 1.2 and Table 7.1.

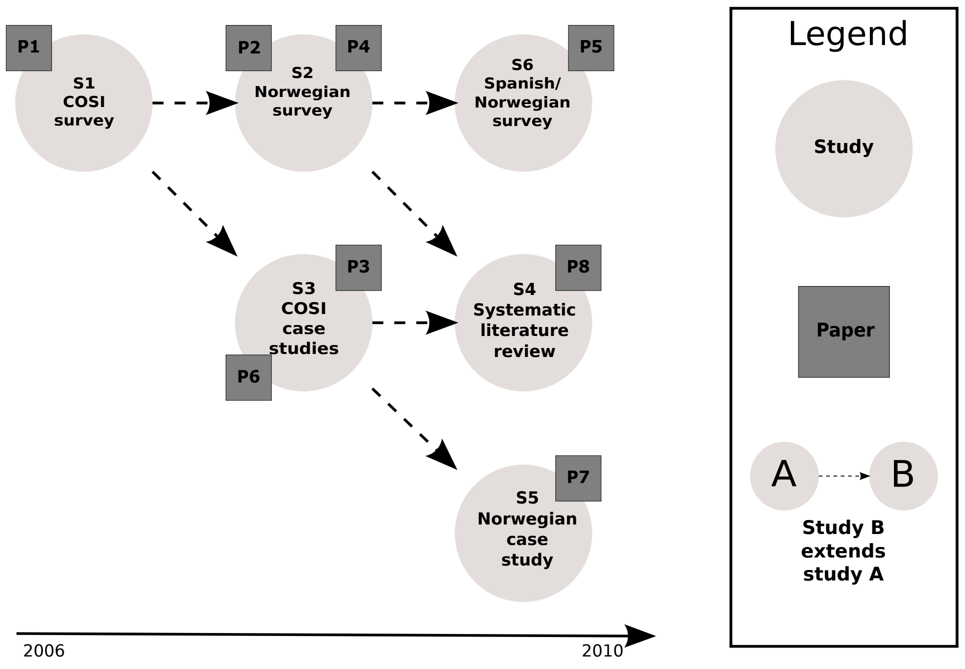


Figure 1.2: Relation between studies and papers

1.4.1 Selected Papers

This thesis includes eight papers that will be labeled **P1-P8** throughout the thesis. The papers are added verbatim as attachments in Appendix A and they are available from the Software Engineering group's web page: `http://www.idi.ntnu.no/grupper/su/`

P1 **Øyvind Hauge**, Carl-Fredrik Sørensen, and Andreas Røsdal. *Surveying Industrial Roles in Open Source Software Development.* In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors, Proceedings of the 3rd IFIP WG 2.13 International Conference on Open Source Software (OSS2007) - Open Source Development, Adoption and Innovation, June 11th-14th, Limerick, Ireland, volume 234/2007 of IFIP Advances in Information and Communication Technology, pages 259-264, 2007. Springer.
My contribution: I was involved in all stages of the paper, including being the leading author. Røsdal contributed to design and data collection.

P2 **Øyvind Hauge**, Carl-Fredrik Sørensen, and Reidar Conradi. *Adoption of Open Source in the Software Industry.* In Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors, Proceedings of the 4th IFIP WG 2.13 International Conferences on Open Source Software (OSS2008) - Open Source Development Communities and Quality, September 7th-10th, Milano, Italy, volume 275/2008 of IFIP Advances in Information and Communication Technology, pages 211-222, 2008. Springer.
My contribution: I was involved in all stages of the paper, including being the leading author. Sørensen contributed to design and data collection.

P3 Sven Ziemer, **Øyvind Hauge**, Thomas Østerlie, and Juho Lindman. *Understanding Open Source in an Industrial Context.* In Albert Dipanda, Richard Chbeir, and Kokou Yetongnon, editors, Proceedings of the 4th IEEE International Conference on Signal-Image Technology & Internet-Based Systems (SITIS 2008), November 30th-December 3rd, Bali, Indonesia, pages 539-546, 2008. IEEE Computer Society.
My contribution: The paper was a collaborative effort based on discussions and data from the four authors. I wrote mainly the literature section, took care of submitting the paper, and prepared the camera-ready version.

P4 **Øyvind Hauge**, Thomas Østerlie, Carl-Fredrik Sørensen, and Marinela Gerea. *An Empirical Study on Selection of Open Source Software - Preliminary Results*. In Andrea Capiluppi and Gregorio Robles, editors, Proceedings of the ICSE 2009 Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS 2009), May 18th, Vancouver, Canada, pages 42-47, 2009. IEEE Computer Society.
My contribution: I was involved in all stages of the paper, including being the leading author. However, Gerea contributed significantly to data collection, and Østerlie contributed to analyzing and presenting the findings.

P5 Claudia P. Ayala, **Øyvind Hauge**, Reidar Conradi, Xavier Franch, Jingyue Li, and Ketil Sandanger Velle. *Challenges of the Open Source Component Marketplace in the Industry.* In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, Proceedings of the 5th IFIP WG 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3rd-6th, Skövde, Sweden, volume 299/2009 of IFIP Advances in Information and Communication Technology, pages 213-224, 2009. Springer.
My contribution: The paper is the result of a collaborative effort led by Ayala and Hauge. I was involved in the design of the study, transcription, translation, analysis of the results, and writing of the paper. Data collection was done by Ayala and Velle.

P6 **Øyvind Hauge** and Sven Ziemer. *Providing Commercial Open Source Software: Lessons Learned.* In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3rd-6th, Skövde, Sweden, volume 299/2009 of IFIP Advances in Information and Communication Technology, pages 70-82, 2009. Springer.
My contribution: The paper is the result of a collaborative effort where I was the leading author.

P7 **Øyvind Hauge**, Daniela Soares Cruzes, Reidar Conradi, Ketil Sandanger Velle, and Tron Ándre Skarpenes. *Risks and Risk Mitigation in Open Source Software Adoption: Bridging the Gap between Literature and Practice*. In Pär J. Ågerfalk, John Noll, and Cornelia Boldyreff, Proceedings of the 6th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2010), May 30th-June 2nd, Notre Dame, USA, volume 319/2010 of IFIP Advances in Information and Communication Technology, pages 105-118, 2010. Springer.
My contribution: I was involved in all stages of the paper, including being the leading author. However, Velle and Skarpenes contributed significantly to the design of the study and to the data collection, and Cruzes contributed to the presentation of the finding.

P8 **Øyvind Hauge**, Claudia P. Ayala, and Reidar Conradi. *Open Source Software in Organizations - A Systematic Literature Review*. Submitted to Information and Software Technology (IST) on December 3rd 2009.
My contribution: The paper is the result of a collaborative effort led by Hauge and Ayala. I was involved in all stages of the paper, including being the leading author.

1.4.2 Contributions of this Thesis

This thesis contributes to filling two gaps in OSS research. First, the majority of the research on OSS focuses on communities of volunteers and the activities going on in these communities (Scacchi et al., 2006; von Krogh and von Hippel, 2006; Stol and Babar, 2009). In contrast, this thesis has focused on the adoption of OSS in organizations. Second, most of the research on software selection has focused on suggesting formalized methods that prescribe how selection should be done (Mahmood et al., 2007; Mohamed et al., 2007). In contrast, this thesis has focused on describing actual selection practice.

More precisely the thesis has five main contributions (C1-C5). First, C1-C3 offer insight as to how organizations actually adopt OSS. These three contributions construct a solid platform for future research on OSS adoption. Based on this platform, we offer implications for research, in particular, but also for practice. The contributions are:

C1 Through empirically grounded descriptions, we offer insight as to how several organizations actually adopt OSS, and we show that these organizations adopt OSS in significantly different ways.

C2 We offer a systematic review of the empirical evidence in the literature on OSS adoption. We moreover organize this empirical evidence according to how the involved organizations adopt OSS.

C3 Based on C1 and C2, we have developed a classification framework consisting of six ways in which organizations adopt OSS. This framework offers researchers increased precision when talking about OSS adoption. It may also serve practitioners in identifying the benefits and challenges related to their own adoption of OSS.

Second, contributions C4 and C5 increase the understanding of software selection. Based on these contributions we draw implications for both practice and research. We suggest in particular that research on software selection should increase its attention towards the situation the selection is conducted in and the rich (text) experience available from both people and across the Internet. The contributions are:

C4 We provide empirically grounded descriptions of the practices software developers actually use when selecting OSS components.

C5 Based on this empirical foundation, we offer a model for situated[1] software selection. This model puts the practices that are used to select components into a context. By adding the situation the selection is conducted in as a new dimension to software selection, the model offers and explanation as to why formalized selection methods have had limited influence on practice.

1.5 Thesis Structure

This thesis is structured into seven chapters, including this introduction. These are:

Chapter 2 presents the background for the work conducted in this thesis, including software engineering, OSS, and integration of software components. Then, it focuses on the two main topics of this thesis (1) adoption of OSS in organizations and (2) selection of software components. Readers who are familiar with these topics may skip this chapter. In addition, the chapter provides an overview of related topics which had to be left out of the thesis' scope.

Chapter 3 presents the research conducted in this thesis and the context of this work.

Chapter 4 and 5 give an overview of the main results provided by this research. These results are organized into the thesis' two main topics. Chapter 4 focuses on OSS adoption and presents contributions C1-C3. Chapter 5 focuses on software selection and presents contributions C4 and C5.

Chapter 6, evaluates and discusses the research with respect to the research literature, research questions, COSI goals, validity, and the scope of the thesis. In addition, it presents implications for both research and practice. These implications include input for future research on both OSS adoption and software selection.

Finally, Chapter 7 concludes the thesis and discusses a few possible extensions to this thesis. Readers who are mainly interested in this thesis' contributions should read this introduction together with Chapters 4, 5, and 7.

In addition, Appendix A presents the papers that are published as part of this work, and Appendix B gives an overview of the interview guides and questionnaires used in this thesis.

[1]The term situated action "underscores the view that every course of action depends in essential ways upon its . . . circumstances" (Suchman, 1987, p. 50). Software selection practices are heavily depending on the context in which they are performed and on the developer performing them.

Chapter 2

Background and Related Work

This chapter provides a background for the topics discussed in this thesis. The chapter is divided into four sections. The first section gives a brief background for software engineering, OSS, and software integration. The second and the third section present related research on the thesis' two main topic (1) OSS adoption and (2) selection of OSS components. Finally, the fourth section summarizes the research challenges and gives a short overview of topics that had to be left outside the scope of this thesis.

2.1 Background

Software engineering, OSS, and software selection constitute a basis for the work conducted in this thesis. In this section, we give a brief overview of software engineering and software engineering research. We describe OSS as a multifaceted phenomenon, give a short historical background for OSS, and present the view that OSS is something different from software engineering. While we disagree with this view, it has dominated much of the research on OSS. Finally, we give an introduction to software reuse and CBSE.

2.1.1 Software Engineering and Empirical Research

Software Engineering

As software development became increasingly complex, many software development projects faced premature cancellations, delays, and cost overruns. The application of engineering practices was believed to remedy these problems, and *software engineering* (SE) was the response to what Dijkstra (1972) described as the "*software crisis*". Boehm (1976, p. 1226) defined software engineering as:

> The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them.

Even though the severity of this "crisis" has been questioned (Jørgensen and Moløkken-Østvold, 2006), there is no doubt that software engineering is a complex endeavor covering a large span of topics. According to the Software Engineering Body of Knowledge (SWEBOK) edited by Abran et al. (2004), software engineering covers topics such as software requirements, design, construction, testing, maintenance, configuration management, and quality, together with software engineering management, processes, and tools and methods. Software engineering is furthermore significantly different from other kinds of engineering (Kruchten, 2004), and it is characterized by the unique, complex and ever changing nature of software development projects (Basili et al., 1986; Mohagheghi, 2004).

Software Engineering Research

The overall goal of research on software engineering is to inform practitioners through creating new or revised knowledge related to any of the areas which constitutes software engineering (Osterweil, 2007). According to Sjøberg et al. (2007, p. 358) software engineering (SE) research concerns:

> (1) the development of new, or modification of existing, technologies (process models, methods, techniques, tools or languages) to support SE activities, and (2) the evaluation and comparison of the effect of using such technology in the often very complex interaction of individuals, teams, projects and organisations, and various types of task and software system *[sic]*.

Much of this research was for a long time motivated by solving the "software crisis". However, in failing to solve this crisis, research ended up in what Glass (1994) criticized as the "research crisis". Research failed to influence practice because it was troubled by:

- **Narrow Focus:** While software engineering research covers a wide span of topics, much of this research has had a rather narrow focus (Glass et al., 2002; Segal et al., 2005; Höfer and Tichy, 2007). Topics like conceptual analysis, measurement and metrics, and tools, methods, and frameworks frequently occur. Researchers have furthermore favored a limited number of research methods, included few references to other disciplines, and conducted a limited number of longitudinal studies. Research has been heavily influenced by formal and mathematical approaches (Boehm, 2006b), it has focused too much on proposing new methods (Fenton, 1993), and has had a bias towards normative research and prescriptive contributions (Glass et al., 2002; Hansen et al., 2004).
- **Lack of Empirical Validation:** In their eagerness to propose new methods, many researchers have made claims and promises about their newly developed methods without backing them up with solid empirical data (Fenton, 1993; Tichy et al., 1995; Zelkowitz and Wallace, 1998). Even though the level of validation has increased, the lack of empirical validation is still a significant challenge to the credibility of software engineering research (Zelkowitz, 2009).
- **Limited Relevance to Practice:** Many of the problems researchers have studied are of little significance to practitioners, while the overlooked problems are often the ones which turn out to be important (Potts, 1993). Frustrated by the research

community's constant focus on proposing new methodos, Glass (2004) requested advice on how to use current methods rather than an continuous stream of new ones.

The challenges to research on software engineering, motivated some changes to software engineering research. Potts (1993) requested a more practical approach to research, Tichy et al. (1993) asked for a scientific basis for the software engineering research, Zelkowitz and Wallace (1998) motivated for more empirical research, and Basili (1996) discussed the need for real world studies and the use of the industry as a "laboratory". Many of these requests from the nineties were repeated by, for example Sjøberg et al. (2007). All in all there was, and still is, an agreement that there is a need for:

- More relevance to practice through more varied empirical research in real contexts
- More rigor in the planning, execution, and reporting of this research

Inspired by evidence based medicine, Kitchenham et al. (2004), and Dybå et al. (2005) proposed evidence based software engineering (EBSE) as an evolution of empirical software engineering. By using systematic literature reviews they intend to integrate the evidence from many empirical studies with current best practices. Having a base of high quality studies focusing on the same research question(s) is thus a prerequisite for EBSE.

2.1.2 Open Source Software

Open Source Software Is a Multifaceted Phenomenon

OSS products cover almost everything from operating systems (Linux, OpenSolaris), through object-relational mapping libraries (Hibernate, Apache Torque), to desktop tools (OpenOffice.org, Thunderbird). Users of these products are through each individual products' license granted the freedom to run, study, redistribute, and improve the software. Access to the software's source code is a prerequisite for these freedoms (Rosen, 2005). An OSS product can be defined as:

> A piece of software released with a software license approved by either the Open Source Initiative (OSI) or the Free Software Foundation (FSF).

However, the OSS phenomenon is much more than just software products. While there have been several attempts at defining OSS, "[w]e do not have a universally accepted definition of OSS" (Wang and Wang, 2001, p. 90). We agree with Brown and Booch (2002, p. 125) in that "describing a laundry list of different definitions of open-source and then positing a new one is not particularly fruitful". We will rather reflect on the fact that there are several understandings of what OSS is. Brown and Booch (2002),and Gacek and Arief (2004) illustrate that OSS has a multidisciplinary nature and that it may be understood as software products, communities, software development processes, release management processes, business models, and so on.

The diversity of OSS is also seen in the many different communities supporting OSS products and the practices these communities use. For instance, while Robles et al. (2007)

show that close to 1500 companies had contributed to Debian, Capiluppi et al. (2003a) provide evidence that most OSS communities have only one or a few developers. Moreover, Noll (2009) contrasts evidence in the widely cited paper by Mockus et al. (2002), and shows that the development practices in OSS communities are significantly heterogeneous. We therefore consider that:

> Open source software (OSS) is a multifaceted phenomenon consisting of a wide spectrum of software products provided by heterogeneous communities using a variety of software development and maintenance practices.

Not only are there different understandings of what OSS is, there are also several other similar terms. Despite some of their historical and ideological differences, this thesis considers Free Software (FS), Free Open Source Software (FOSS), and Free/Libre Open Source Software (FLOSS) to be equivalent with Open Source Software (OSS). No distinctions will be made unless absolutely necessary.

The multidisciplinary nature of OSS can also be seen by looking at the many different perspectives which have been used to study OSS. In their effort to define OSS, Gacek and Arief (2004) consider research fields such as computer science, management and organization science, social science, psychology, economics, and finally law.

A Brief Historical Background

This section gives a simplified view on the OSS history and illustrates how software has gone from being open and available, through being closed and unavailable, to again becoming more open (see Figure 2.1).

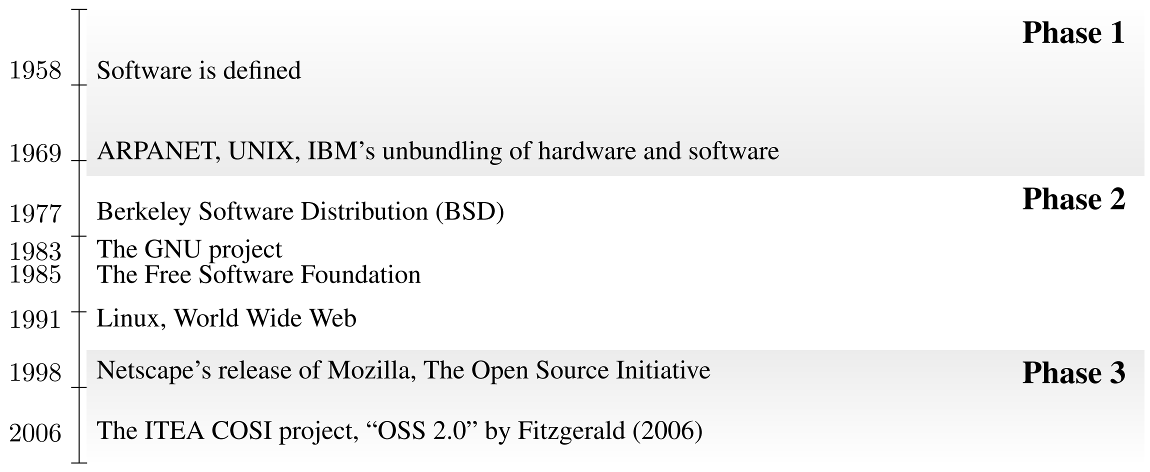

Phase 1
1958 Software is defined
1969 ARPANET, UNIX, IBM's unbundling of hardware and software
Phase 2
1977 Berkeley Software Distribution (BSD)
1983 The GNU project
1985 The Free Software Foundation
1991 Linux, World Wide Web
Phase 3
1998 Netscape's release of Mozilla, The Open Source Initiative
2006 The ITEA COSI project, "OSS 2.0" by Fitzgerald (2006)

Figure 2.1: A brief timeline of relevant events in the OSS history

Phase 1: The Hacker Era: Many of the programmers in the period after the Second World War had engineering or physics backgrounds and were using computers mainly to solve problems from their own domain (Raymond, 1999). Software was at the time mainly shared freely between researchers and engineers who had common interests. Many of these programmers were often called hackers[1] because of their interest in software programming. Companies within the early computer industry were furthermore mainly focusing on developing and selling hardware (von Krogh and von Hippel, 2003). This phase ended when IBM decided to unbundle their software from their hardware in 1969 (Grad, 2002).

Phase 2: The Growth of Proprietary Software: In the late seventies and early eighties the use of software increased quite dramatically, and AT&T and other companies started to see the commercial value of software. To exploit this commercial potential, AT&T released Unix as a for-fee product and restricted access to its source code (Weber, 2004). Other companies followed AT&T in their commercialization of software, and the eighties witnessed the rise of software companies like Microsoft, Oracle, and SAP AG.

In contrast to the companies refusing to release their products' source code, Richard Stallman announced the GNU (GNU is Not Unix) project in 1983 and the Free Software Foundation in 1985 (Stallman, 1999). The goal of the GNU project and FSF was to create a new Unix-like operating system and promote free software. This operating system was never completed, but with Linus Torvalds' Linux kernel, GNU/Linux has grown to become perhaps the most well known OSS product ever.

Phase 3: The Commercialization of OSS: Even though Stallman (1999) and the Free Software Foundation emphasized that software should be free as in free speech, not as in free beer, "Free" did not correspond very well with commercial companies. As a response to this, the term "open source" was coined in relation to Netscape's release of its Mozilla web browser under an OSS license in 1998 (Hamerly et al., 1999). Later the same year, Bruce Perence and Eric Raymond initiated the Open Source Initiative as an organization for education and advocacy of OSS (Perens, 1999).

Although open source software is considered a better term for commercial companies, OSS did not immediately see significant commercial adoption. However, in the beginning of the new millennium there has been an increasing interest in OSS, and Fitzgerald (2006, p. 587) writes that OSS has evolved "into a more mainstream and commercially viable form". In 2006, ITEA initiated the industrial research project COSI and a report, edited by Ghosh (2006), illustrates the economic impact of OSS on the ICT sector.

The Alienation of OSS - Views that OSS is Something "Different"

OSS has to a certain extent been characterized by the many conflicting views of what it is and how it is different from "traditional" software. In particular, advocates of OSS have claimed that OSS is cheaper, has fewer defects, gets improvements faster, and is generally better than "other kinds" of software.

> "Both evidence and theory confirm that open source delivers better reliability, lower costs, shorter development times, and a higher quality of code (includ-

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

ing better security)" (Raymond, 2004, p. 88).

At the same time, Østerlie and Jaccheri (2007) state that many researchers have treated and described OSS as being something different from proprietary software products and traditional software development as well. There are conflicting views on open source and free software (Stallman and Lessig, 2002), and contrasts between OSS vs. proprietary or closed source software (Paulson et al., 2004), OSS vs. Commercial-Off-The-Shelf (COTS) (Di Giacomo, 2005b), the cathedral vs. the bazaar (Raymond, 2001), copyleft vs. copyright (de Laat, 2005), OSS development vs. software engineering (Dinh-Trong and Bieman, 2005), and so on. We furthermore see companies which use OSS to differentiate themselves from their competitors by using the OSS brand. For instance, slogans such as "The world's most popular *open source* database" (MySQL) and "global leader in enterprise-class *open source* middleware" (JBoss) are used to illustrate that these companies offer something different than other software vendors.

Fitzgerald (2005), Fuggetta (2003), and others question whether these statements are really true and show that the reality is somewhat more nuanced. For instance:

- The majority of OSS projects struggle to attract contributors (Capiluppi et al., 2003b; Krishnamurthy, 2002).
- Many of the developers participating in the development of OSS are paid by their employers to do so (Hertel et al., 2003; Robles et al., 2007).
- Many OSS products are company initiated (Bonaccorsi et al., 2007).
- The quality of OSS products is not always as good as expected (Stamelos et al., 2002).
- OSS products may also be developed in-house without any community (Noll, 2009).

More recent research is starting to overcome the view that OSS is something different. Fitzgerald (2006) says that OSS has evolved into a more commercially viable form, and we agree with Baird (2008) in that pitting OSS up against proprietary software is meaningless. Proprietary software and OSS are rather converging, and we see that organizations most often adopt a hybrid approach to OSS and combine OSS with proprietary software (Bonaccorsi et al., 2006; Baird, 2008).

2.1.3 Integration of Software Components

Software reuse and CBSE have had the significant impact on software development through dealing with some of its complexity (Boehm, 2006a). Software reuse, CBSE, and integration with existing systems have therefore became the preferred way of developing software (Yang et al., 2005).

The basic idea of software reuse is to take existing software components and reuse them in other software systems. While Karlsson (1995) discusses software development *for* reuse and software development *with* reuse, this thesis is only focusing on software development *with* reuse. Software development with reuse is believed to (1) reduce the development time and cost, (2) give a richer feature set than if the required functionality was developed from scratch, and (3) increase the quality as compared to development from scratch (Mohagheghi and Conradi, 2007). Frakes and Kang (2005, p. 529) define software reuse as:

> the use of existing software or software knowledge to construct new software.

We see that software reuse involves more than just the reuse of software artifacts. Software reuse does in fact concern reuse of procedures, knowledge, documentation, architectures, design, and code (Rothenberger et al., 2003). However, in this thesis we will mainly concern ourselves with the reuse of software components.

Although we acknowledge that there are several definitions of what a component is (Torchiano and Morisio, 2004), we rely mainly on only one of these definitions. Szyperski et al. (2002, p. 41) define a software component as:

> a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

From the definition of a component we see that it is a piece of source code and it is developed independently from the system in which it is integrated. These pieces of source code can take several forms. The granularity of a component can, according to Mohagheghi (2004), vary from subroutines and classes, through libraries and modules, to frameworks constituting a whole product line. While much of the early research focused on reuse of COTS, a reusable component may also be developed in-house or as OSS.

By decomposing software systems into components, Parnas (1972, p. 1055) expected to “reduce the programming [of the system] to the relatively independent programming of a number of small, manageable programs”. The development of small independent components was expected to:

- Enable parallel development of components
- Increase comprehensibility
- Increased flexibility and reuse
- Simplify (internal) modification of a component

Boehm and Abts (1999), and Vitharana (2003) recognize similar advantages of CBSE, but add that CBSE may increase the quality of a software product through (massive) reuse of its parts and through enabling simple replacement of one component with another. CBSE can therefore be seen as an important enabler of extensive software reuse.

Even though there is some discussion about what CBSE and component based software development (CBSD) are (Brown and Wallnau, 1998), they can be said to concern the construction of a software system through integrating (relatively) independent components. One system can consist of many components and one component can be part of several systems. There are also several different stakeholders involved in CBSE, each of which have different requirements to the components and the final system. Vitharana (2003) identifies (1) the *component provider*[2] which develops the reusable components, (2) the *integrator* which integrates these components into a software system, and (3) the *customer* which sponsors the development of the system as the three most important stakeholders in CBSE. In addition, we may also include the (4) *end user*. Each of these four stakeholders may have multiple relations with other stakeholders.

We have already seen that software reuse and CBSE are the standard ways of developing (all kinds of) software, and that a range of software components of different granularity, from different domains etc. are reused. However, reuse projects and the processes used in these projects can also have significant variations (Morisio et al., 2000; Rothenberger et al., 2003). Li et al. (2009) find furthermore that CBSE can be used with any kind of traditional development process. Software reuse is thus used:

- In all kinds of software products
- With all kinds of reusable software artefacts
- With all kinds of software projects, processes, and practices

2.2 Adoption of OSS in Software-Intensive Organizations

In this section, we give a brief overview of existing work on how organizations approach OSS. First, we illustrate that OSS adoption may give significant advantages. However, there are possible pitfalls related to this adoption, and there are furthermore several ways of adopting OSS. This, and the perception that OSS is something different from traditional software, have made practitioners uncertain about what it means to adopt OSS. If practitioners fail to understand what OSS adoption means, they could miss the benefits OSS adoption may give. Finally, we will show that there is a lack of research investigating this matter.

2.2.1 Significant Potential and Adoption: No Guarantee for Success

Due to the success of big projects like Linux, Apache, and Mozilla, OSS has become interesting to software-intensive organizations on several fronts:

- **As OSS products or components:** SourceForge and other portals host thousands of products spanning not only several domains, but also the whole software stack. Ghosh (2006) estimated the total value of available OSS products to be in the range of billions of Euros. Several empirical studies show that the adoption of such products is significant (Ghosh, 2002; Glynn et al., 2005; Lundell et al., 2006).
- **Through OSS communities:** Many OSS projects have achieved significant diffusion of their products and have attracted a large number of contributions from their communities. This is for instance exemplified in the contributions and user assistance in Apache and Mozilla projects (Mockus et al., 2002; Lakhani and von Hippel, 2003). To achieve similar benefits, several software-intensive organizations

[2]Vitharana (2003) uses the terms “component developer” and “application assembler”.

have also released their own OSS products and contributed to OSS products controlled by others (Henkel, 2006; Bonaccorsi et al., 2007; Stam, 2009).

- **Through "OSS development practices":** Several OSS projects have managed to deal with many of the challenges related to distributed software development (Crowston et al., 2007; Østerlie and Wang, 2007). Even though there is no set of development practices which are universal to all OSS projects, practices such as user participation, short release cycles, and peer code reviews have frequently been associated with OSS projects and are often labeled "OSS practices" (Feller and Fitzgerald, 2002; Scacchi, 2004). The successful application of these practices has inspired several companies to adopt them (Melian and Mähring, 2008; Wesselius, 2008).

Despite offering significant potential, OSS does not come without possible pitfalls. These pitfalls may stop organizations from exploring the opportunities offered by OSS.

The amount of OSS products available is considerable, and many of these products are high quality products like Linux, Apache HTTP Server, and MySQL. While these high profile products have attracted a lot of attention in the research literature e.g. (Mockus and Herbsleb, 2002; Dahlander and Magnusson, 2005; Yu, 2007), the products at the other end of the scale have attracted next to no attention. Despite this lack of attention, the quality of many OSS products is, according to Stamelos et al. (2002), lower than expected. Moreover, many products are struggling to attract attention or simply do not work at all (Krishnamurthy, 2002; Capiluppi et al., 2003a). However, it is not only the community-driven OSS projects which struggle, commercially initiated OSS products are also having problems attracting a community and getting contributions from this community (Boldyreff et al., 2004; Bleek et al., 2005; Järvensivu and Mikkonen, 2008).

While it is possible to read about organizations struggling with their adoption of OSS products in the media, it is rare to see studies of failed or at least problematic OSS adoption endeavours in the research literature. Fitzgerald (2009) reports one story from an Irish hospital where they succeeded at adopting an OSS e-mail solution, but failed to adopt an OSS office application suite. Even though the research literature has focused little on OSS adoption failures, failure is a possible scenario for organizations that consider OSS adoption.

2.2.2 Ways of Leveraging OSS

Organizations adopt OSS in different ways. For instance, Henkel (2006) discusses participation in the Linux embedded community, Chen et al. (2008) investigates the integration of OSS components, and Fitzgerald (2009) presents two cases where OSS products are deployed in a hospital environment. However, to our knowledge no attempt has been made to get a complete overview of these differences in a software engineering perspective.

The ways software-intensive organizations adopt OSS are typically influenced by a desire to either make or save money, or in other words their business model. Even though business models are outside the scope of this thesis, they illustrate that organizations have several ways of benefiting from OSS. The business models also influence how these organizations approach OSS in software development. Based on Hecker (1999), Raymond (2001), Krishnamurthy (2005), and Fitzgerald (2006), we present an overview of some OSS business models in Table 2.1. However, these business models focus on on creating revenue stream, and less on how these organizations actually do software development.

Table 2.1: OSS business models

Business model	**Description**
Loss-Leader/Support Sellers/Brand Licensing	Release products as OSS to retain market position and to sell related products, services, or brands. May be combined with a dual licensing schema.
Distributor	Package and distribute OSS products developed by someone else and charge for these and other services.
Service Provider	Provide services of any kind related to any OSS product provided by someone else, and charge for these services.
Accessorizing	Sell accessories to OSS products ranging from mugs and T-shirts, to professionally-edited and produced documentation.
Widget Frosting	Provide software products as OSS to attract contributions or to share development costs. This is particularly relevant for many hardware manufacturers that require software to operate their products, but make no money selling this software.
Software Producer	Integrate existing OSS products or the source code of these products into their own software.

Although Hecker (1999), Raymond (2001), Krishnamurthy (2005), and Fitzgerald (2006) use different names for each of the business models, most of them have one thing in common: most companies provide and make money on related services and/or products. According to Fitzgerald (2006), OSS and its general lack of license fees is in fact contributing to shifting the software industry's traditional license-based business models towards service-based models. Despite this shift towards more service-based models, most organizations which adopt OSS business models are still providing proprietary products in what Bonaccorsi et al. (2006) call a "hybrid business model". OSS seems to have found a place in the market together with, rather than instead of, proprietary products (Baird, 2008).

The four-level ladder of resource allocation by Grand et al. (2004) illustrates another dimension of how organizations approach OSS. The model shows how organizations dedicate (sparse) resources to leveraging OSS:

Level 1 The organization is mainly a user of OSS products and needs to dedicate resources to deploy OSS products within its own organization.

Level 2 The organization uses OSS as a complementary asset, though for instance delivering OSS products together with its hardware products or as part of their software products.

Level 3 The organization actively contributes to the development of OSS products controlled by itself or by others.

Level 4 The organization bases its whole business around providing services related to existing OSS products.

Finally, Dahlander and Magnusson (2005) present three different relationships an organization may have to an OSS community. The authors illustrate how the organization may (1) exploit and harm the community as a parasite, (2) benefit from the community in a commensialistic way, or (3) live in a symbiotic relationship with the community. In a more recent paper, Dahlander and Magnusson (2008) show that an organization may use different strategies for making use of OSS communities, either though accessing, aligning, or assimilating the communities.

We see that organizations may have different business approaches to OSS, different levels of resource allocations, and different relationships to OSS communities using different strategies for benefiting from them. These are but a few of the possible options an organization has when adopting OSS. However, in the remainder of this thesis we will focus on how software-intensive organizations face OSS in their own software development, and the benefits and challenges they meet when doing so.

2.2.3 Uncertainty about What OSS Adoption Is

The OSS phenomenon has been characterized by several contrasts (see Section 2.1.2). These contrasts and the view that OSS is something "different" have contributed to alienating OSS and making practitioners uncertain about how they may benefit from it. Organizations have faced OSS with skepticism and confusion. Goode (2005) showed that practitioners did not see the relevance of OSS and were therefore skeptic about adopting it. According to an ITEA Report on Open Source Software (Daclin, 2004), there has been a lot of confusion around what OSS is and how organizations may benefit from it. There are several factors which could have contributed to creating this confusion and skepticism:

- Organizations are used to proprietary software provided by a vendor. The presentation of OSS as something different than proprietary software, has made organizations uncertain about its relevance.
- OSS advocates have created unrealistic expectations to OSS products by portraying them as being better, faster, cheaper, and so on. When OSS products fail to live up to this hype, practitioners' skepticism towards OSS is increased.
- The research literature has been unclear about what it actually means to adopt OSS. Several publications discuss adoption of OSS without clarifying how, or what, the involved organizations actually do related to OSS. Özel et al. (2007) discuss "F/OSS usage and adoption" in public administration, Bonaccorsi et al. (2006) talk about companies which "have entered the open source field", and Ravesteyn and Silvius (2008) discuss organizations which "are active in the OSS domain". However, none

of these really state clearly what these organizations actually do in relation to OSS. This unclarity further increases the confusion around OSS adoption.

We fear that this cofusion around OSS adoption may prevent software-intensive organizations from leveraging OSS, and it is therefore important to understand what OSS adoption means to organizations.

2.2.4 The Lack of Empirical Research on OSS Adoption

The views that OSS is something different has contributed to creating an excessive focus in the research literature on communities of volunteers and the activities taking place in these communities (Feller et al., 2006; Stol and Babar, 2009). von Krogh and von Hippel (2006) categorize the research on the OSS phenomenon into three areas: (1) motivation of OSS contributors, (2) governance, organization, and the process of innovation in OSS projects, and (3) competitive dynamics enforced by OSS. In another overview of OSS literature, Scacchi et al. (2006) focus on the processes found in OSS projects. By overlooking organizations and their approaches to OSS, both these overviews illustrate the excessive focus on OSS as a community-driven phenomenon.

Even though there are several studies which focus on different aspects of OSS adoption, (Henkel, 2006; Ågerfalk and Fitzgerald, 2008; Dahlander and Magnusson, 2008), the majority of these are published relatively recently. The research community is furthermore lacking an overview of, and a clear direction for, this research.

2.3 Selection of Software Components

Software reuse and CBSE span a large number of topics, stakeholders, and challenges (Vitharana, 2003; Crnkovic, 2001). However, most of these are outside the scope of this thesis. This thesis will mainly focus on the **integrator** and one of the most important parts of the integrator's job, **selection of components**. In this section we illustrate why selection is both important and challenging. We show that research has mainly focused on proposing formalized methods for selection of components, but that these methods have seen little adoption.

2.3.1 The Practical Selection Problem

A simplified view of the integrator's responsibilities can be divided into three main activities (1) determining the customer's requirements, (2) **selecting components matching these requirements**, and (3) integrating the selected component(s) into the resulting system. We will hereafter mainly focus on the selection of components. This selection consists of (a) identifying candidate components, (b) evaluating them, and (c) choosing one or more of them. Solving this practical selection problem is imperative to successful software reuse and CBSE (Kunda and Brooks, 2000; Mahmood et al., 2007). However, there are several challenges involved in selection. For instance, there is a large number of (evolving) components, attributes, features, and combinations of both hardware and software that have to be considered (Ncube and Dean, 2002).

The components should match the customer's requirements, but they should also also be *reusable*, meaning that they should solve a common problem, be of good quality, and be easy to understand (Mili et al., 1995). However, mismatch between the functionality offered by the component and the customer's original requirements, may influence these requirements (Alves and Finkelstein, 2003). If the components miss functionality, the customer may be convinced to make (significant) cost savings by accepting a solution with somewhat reduced functionality. If the components have extra functionality, they may inspire the customer into using the software in new ways. Selection therefore concerns an evaluation of technical (functional and non-functional) requirements, but also commercial and organizational issues (Brereton and Budgen, 2000).

The consequences of selecting a poor component are serious, as reusable components "will often have shortfalls in usability, dependability, interoperability, and localizability to different countries and cultures" (Boehm, 2006a, p. 21). These shortcomings may create significant challenges related to maintenance of the system and thus increase maintenance costs (Reifer et al., 2003; Boehm, 2006b). The integrator and/or the integrator's customers have to live with the consequences of their choices. Assessment and selection of components is because of these long term consequences, one of the most critical phases of CBSE (Ochs et al., 2001).

However, selecting a component is far from simple in the large, uncontrolled, and complex OSS marketplace. This virtual marketplace is where OSS products are made available, typically through the Internet. With OSS and the general evolution of the Internet, web-based services, and user contribution (Web 2.0), the marketplace has over the last years evolved and become even more complex. It contains a large number of components, a lot of information from a number of stakeholders, and several new roles and actors undertaking these roles. These actors provide different kinds of information and have their own motivations for doing so. While this commercial marketplace gives integrators access to vast number of reusable components, it is not problem-free. Ayala (2008) describes the marketplace as:

- Uncontrolled
- Constantly changing and increasing in size
- Plagued by information with unclear trustworthiness
- Dominated by components with (strong) inter-dependencies
- Held back by the lack of standard descriptions for the components provided in the marketplace, and by the marketplace's lack of reuse mechanisms

Even though new roles or services like code-search engines and software repositories attempt to reduce the marketplace's complexity, there is still no complete overview of everything, and the marketplace is still plagued by unreliable information and poor components. This complex marketplace has made it difficult for integrators to find the components they need (Brereton and Budgen, 2000; Kunda and Brooks, 2000). We agree with

Wang and Wang (2001, p. 90) in that "the myriad number of OSS packages make actual adoption a real challenge."

2.3.2 Research: Focus on Formalized Selection Methods

The importance of selecting the "right" component from this complex marketplace made Brereton and Budgen (2000) identify selection and evaluation of components as an one of four key issues for research on component based development. Selection of COTS and OSS has therefore received a lot of attention within the software engineering community. However, this research has some of the same problems as the literature on software engineering (see Section 2.1.1):

- A bias towards suggesting formalized and prescriptive methods and evaluation schemes for selecting components
- Little or no empirical validation of the suggested methods and schemes
- Limited reported use of these methods and schemes in practice

The Bias Towards Formalized Selection Methods

Ayala (2008) shows that the literature on software selection has had a focus on analysis, evaluation, and decision techniques for selection of components, while minor attention has been drawn towards identification of components and knowledge management. Ayala also points out that there has been a focus on one-time selection of components. Further reuse of the component and the knowledge gained in the selection process has attracted less attention. In reviews of the CBSE literature, several authors identify a considerable bias towards (proposing new) systematic or formalized methods for selection and evaluation of components (Mohamed et al., 2007; Land et al., 2008; Birkmeier and Overhage, 2009). Fitzgerald (1996, p. 4) defines formalized methods as:

> formally-defined, brand-named or published . . . methodologies, of which there are many examples in the literature

Even though the selection methods identified by these reviews are different, they share the same basic principles and are built with the same natural science or rational engineering mindset to problem solving. The methods are in most cases strictly systematic, and they frequently expect that all the requirements already are defined and stable, and that several components and plenty of (trustworthy) information is available. Based on a review of 18 of these selection methods, Mohamed et al. (2007) describe a general COTS selection method. Figure 2.2 illustrates this process where the integrator (1) defines the customer's requirements, (2) identifies candidate components, (3) filters these candidates based on the most important requirements, (4) evaluates the remaining components, and (5) selects the best component. Even though these methods share many of the same concepts, the structure of their activities differ. Land et al. (2008, p. 104) group a number of the proposed methods into the following four groups: (1) sequential with branches, (2) iterative, (3) situation-driven/opportunistic/flexible, and (4) concurrent and interrelated processes.

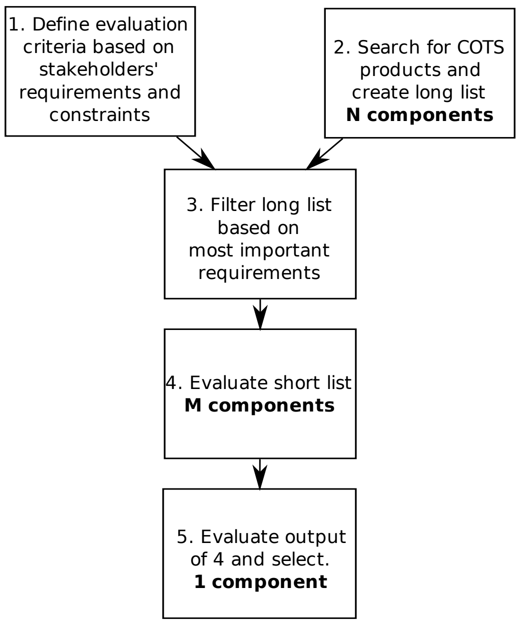

1. Define evaluation criteria based on stakeholders' requirements and constraints
2. Search for COTS products and create long list **N components**
3. Filter long list based on most important requirements
4. Evaluate short list **M components**
5. Evaluate output of 4 and select. **1 component**

Figure 2.2: A general COTS selection process

The more recent work on the selection of OSS components shows the same trends. Researchers continue proposing and reviewing formalized selection methods and frameworks, together with evaluation criteria and scheme. In just a few of the examples, Alfonzo et al. (2008) provide 102 metrics for evaluating OSS tools for analysis and design of software systems, Ardagna et al. (2007) present the FOCSE metrics framework which contains 13 general purpose metrics and 5 specific metrics for evaluating security aspects, and Johansson and Sudzina (2009) propose and discuss 17 criteria for evaluating OSS ERP systems. In addition, Cruz et al. (2006) offer a systematic approach to evaluation of OSS products consisting of eleven usage scenarios and a number of functional, technical, organizational, legal, economical, and political requirements. del Bianco et al. (2009) present the QualiPSo model of OSS trustworthiness consisting of three qualities and 11 sub-qualities together with a (large) number of metrics. Finally, Majchrowski and Deprez (2008) establish an operational method for selection of OSS components. There are in also other initiatives like the Open Source Maturity Model (OSMM) (Golden, 2004), the Open Business Readiness Rating (OpenBRR, 2005), and the Qualification and Selection of Open Source software (QSOS) method (Semeteys et al., 2006).

Limited Empirical Validation

In a review of 15 selection methods, Birkmeier and Overhage (2009) finds that few of the reviewed methods are validated empirically. While there are empirical studies on COTS selection (see Section 2.3.3), this lack of validation is common to much of the research on software engineering (Fenton, 1993). Besides a few exceptions, very few of the proposed methods and evaluation scheme for OSS selection are actually validated empirically. The validation which has been performed is furthermore rather limited. There are only a few papers reporting cases where the authors have selected an OSS product (Di Giacomo, 2005a; Goh et al., 2006, 2008). However, these authors use methods and evaluation schemes which they have developed themselves. Therefore, Li (2006) and Ayala (2008) have identified a need for more empirical research on selection of components.

Problems with the Formalized Approaches

Formalized selection methods have only been able to influence practice to a very limited extent (see Section 2.3.3). This very limited influence may be caused by several problems. Birkmeier and Overhage (2009) suggest that the lack of tool-support and the lack of operative descriptions on how to use the proposed methods could contribute to hindering the adoption of methods for COTS selection. Others suggest that there is a mismatch between requirements and evaluation criteria (Lewis and Morris, 2004), and that the information needed to evaluate a component is often not available (Bertoa et al., 2003). Gorton et al. (2003) state that it may be impractical to conduct complete evaluations with respect to time and cost. Ncube and Dean (2002) state that the use of weighted calculations based on a set of individual scores is misleading. Finally, Li et al. (2009) claim that formalized selection methods are not used because of the integrator having strong relationships with one vendor or because selection takes place during all stages of a development project, not only during a selection stage of the project. The literature on OSS selection highlights several problems with many of the proposed methods as well:

- Have too strong focus on technical issues, and too little on the impact on the business including costs of adoption and extension (del Bianco et al., 2009; Lavazza, 2007)
- Lack relevant evaluation criteria, or have unclear or overlapping ones (Cabano et al., 2007; Deprez and Alexandre, 2008; Taibi et al., 2007)
- Ignore important software artefacts like the code (Samoladas et al., 2008)
- Are highly subjective (Samoladas et al., 2008)
- Lack possibilities for automation (Samoladas et al., 2008)
- Cannot be used with several stakeholders (Majchrowski and Deprez, 2008)
- Lack of advice on how to use the methods (Majchrowski and Deprez, 2008)
- Lack context sensitivity (Deprez and Alexandre, 2008)
- Information is not available (Deprez and Alexandre, 2008; Taibi et al., 2008)
- Unclear scoring rules (Deprez and Alexandre, 2008)

In a related paper on formalized software development methodologies, Fitzgerald (1998) shows that the adoption of formalized methods is slim at best. In the cases these methods are applied, they are not followed rigorously. Furthermore, Fitzgerald (1996, p. 3) discusses several problems with formalized methodologies and claims that “the assumption that increased adoption of methodologies addresses the problems inherent to systems development is by no means proven”. He continues to state that researchers too often try to find one best way to develop a software, but fail. Often, the proposed methods:

- Are very similar

- Lack empirical foundation
- Overestimate peoples' abilities and skills
- Focus too much on the method and too little on the product being developed
- Are assumed to be applied in all contexts
- Do not sufficiently focus on the developer and the constant changes in the environment where the development is taking place

Other Initiatives

While most of the research on OSS selection has focused on proposing new methods and evaluation criteria, there has been a few interesting initiatives. Hummel et al. (2008) suggest a tool which uses unit tests and the Google Code search engine to identify code fragments which can be reused. Automated tool integration is also the goal of the hierarchical quality model proposed by Samoladas et al. (2008). In a position paper, Gallardo-Valencia and Sim (2009, p. 49) propose Internet-scale code search, or "searching the Internet for source code to help solve a software development problem" as an emerging research field, and says that this problem needs a novel solution.

Ayala et al. (2007) suggest initiating an open Wiki-based portal for sharing and reusing information, about and experiences with, components. The idea is that the Wiki should be built from the ground up, support collaboration and knowledge sharing, and enable systematic support for selection and evaluation of components. There are also a wide variety of Internet-based initiatives, which do not necessarily have their origin from the research community (see Table 2.2).

Table 2.2: Internet-based approaches to OSS selection

Service	**Description**
SourceForge	Hosting site which is the home of close to a quarter of a million OSS projects.
Google Code	Both a search engine for components and a hosting site for OSS projects.
ohloh	A large Wiki based software directory for OSS.
Koders	A search engine for OSS code.
Tigris	An OSS community focused on software development tools and an entry point to a large number of such tools.
Apache	The Apache Software Foundation provides organizational, legal, and financial support for a range of OSS projects and serves as an entry point to these projects.
CMS Matrix	One of several matrix sites. This one compares a large number of Content Management Systems (CMS).

2.3.3 Practice: Informal and Based on Familiarity

The aim of the proposed methods for COTS selection is, according to Mahmood et al. (2007), to “identify and rank candidate components; and finally select [the] components which best meet stakeholder requirements”. While this is a notable goal, most or perhaps all of these proposed methods seem to have failed at reaching wide adoption. Instead, developers use informal methods based on the individual developer’s previous experience with the component and testing through prototyping (Torchiano and Morisio, 2004). According to Li et al. (2006a), the selection process has to be fast because of limitations in time and cost. It is therefore not possible to test several components completely.

According to Tran and Liu (1997) and Kunda and Brooks (2000), identification of commercial components has typically been done through conferences, literature reviews, training, and communication with vendors. Familiarity and previous experience is also a very important source of components, and many companies have, according to Li et al. (2006a), internal knowledge keepers. If no one knew of any components they (1) used search engines to find alternatives, (2) selected a couple of them, and (3) downloaded and tested a demo of these. Consequently, Norris (2004) and Chen et al. (2008) describe Internet searches as one of the most important methods for identifying OSS components. These searches are primarily executed through search engines, but also through project hosting sites like SourceForge, code specific search engines like Google Code, and to some extent social tagging sites like delicious[3] (Umarji et al., 2008).

The evaluation of OSS components, and the development of criteria for evaluating OSS components, have also attracted limited attention in empirical studies. However, basic developer dependent rules of thumbs like assessing the vitality of community, listening to the experience of others, and searching for information in mailing lists, forums and so on, are observed (Merilinna and Matinlassi, 2006). Li et al. (2006a) found that practitioners tested key functionality and relied on newsgroups to assess the quality of the components. Respondents in another study said that wide adoption of an OSS component could be a substitute for run-time tests (Maki-Asiala and Matinlassi, 2006). Morisio et al. (2002) found that prototyping, vendor demonstrations, and reviews of material such as manuals and user guides were used to evaluate COTS together with evaluation of vendor availability. Land et al. (2008) categorized these practices into high level evaluation (based on available information), and prototyping (based on using the actual component). Land et al. (2009) found that the establishment of requirements and the actual selection is often interrelated, because it is difficult to break down system requirements to evaluation criteria. They furthermore found and that some of their respondents did not evaluate the components before adopting them.

[3]`http://delicious.com/`

2.4 Summary: Scope and Main Research Challenges

Based on the literature discussed above and our own explorative research, we have identified two main research challenges and three research questions. It is important to point out that when this research was initiated, there existed very few publications on the topic. The research questions presented below are therefore formed throughout the execution of this work. After we had identified how organizations adopt OSS (see Section 4.3), we decided to continue our research group's existing work on CBSE (Torchiano and Morisio, 2004; Li, 2006; Ayala, 2008), and to focus on selection of OSS components.

2.4.1 Adoption of OSS in Organizations

There is, as shown, a significant potential for software-intensive organizations in adopting OSS. OSS has therefore seen quite significant adoption. This adoption is for several reasons expected to increase. However, this adoption is not without pitfalls.

Failed OSS endeavors, and the alienation of OSS, have contributed to creating an uncertainty about what it entails to adopt OSS and how organizations may benefit from it. Many software-intensive organizations have therefore been skeptic about adopting it. Adoption of OSS has furthermore been overlooked by OSS researchers until recently. It is therefore timely to investigate how software-intensive organizations may adopt OSS, and to establish a platform for future research on OSS adoption. This leads us to the following two research questions:

RQ1: How and to what extent are software-intensive organizations currently adopting OSS?

RQ2: What is the current status of research on OSS adoption in organizations and how may this research benefit practitioners?

2.4.2 Selection of OSS Components

We have seen that selection of the right component(s) is a prerequisite for successful software reuse. While the OSS marketplace gives integrators access to a vast number of reusable software assets, it is large, complex, and dominated by a number of stakeholders who provide a lot of information for totally different reasons. Many of these stakeholders have their own agenda, and selection of components in this marketplace is therefore not simple.

Research on the selection of both COTS and OSS has had a predominant bias toward discussing and proposing new methods and evaluation schemes. These methods and schemes have seen limited empirical validation and are rarely used in practice. Based on the research communities, failure to influence selection practice, Mohamed et al. (2007, p. 106) lists the following as two of several challenges for the research community:

- To support the selection of appropriate and effective methods for our context

- To show how COTS selection approaches can be adapted to fit into different contexts

To support selection and to improve practice, it is important to start with empirically studying practitioners' current selection practices. We will therefore address the following research question:

RQ3: What is the current status of research on OSS adoption in organizations and how may this research benefit practitioners?

2.4.3 Related Research Areas and the Scope of this Thesis

Even though OSS is described as something different, we see clear parallels between research on OSS in organizations and several areas within software engineering and information systems research. Gacek and Arief (2004) consider research fields like computer science, management and organization science, social science, psychology, economics, and finally law to be relevant to OSS research.

While we are aware of the research on, for instance, diffusion and adoption of innovations (Rogers, 2003) and other open, grassroot movements like Wikipedia, Creative Commons, open innovation, OpenCourseware, and OpenStreetMap (Chesbrough, 2003), this research had to be left outside the scope of this thesis. This thesis focuses on software engineering and endeavors to put research on OSS in organizations into context, and we will therefore relate it mainly to research areas within software engineering and information systems. Based on our classification framework for OSS adoption (see Section 4.3), we draw parallels between OSS and some related research areas in Table 2.3. However, this is not an exhaustive list.

There are also several parallels which may be drawn between research on software selection and other areas, for instance, decision support systems (Ruhe, 2002), and decision making which includes such elements as social aspects (Munda, 2004) and uncertainty (Begg et al., 2003). However, this research also had to be left outside the scope of this thesis.

Table 2.3: OSS research in relation to other research areas

Way of adopting OSS	Related research areas
Deploying OSS products	Introduction, deployment, diffusion, and acceptance of information systems (IS) and information technology (Fichman, 1992; Karahanna et al., 1999; Vessey et al., 2002; Venkatesh et al., 2003)
Using OSS CASE tools	Computer Aided Software Engineering (CASE) (Fuggetta, 1993; Wicks and Dewar, 2007)
Integrating OSS components	CBSE (Brereton and Budgen, 2000; Li et al., 2009; McIlroy, 1969; Yang et al., 2005) and software reuse (Mohagheghi and Conradi, 2007; Vitharana et al., 2003)
Participating in OSS communities	No clearly related research area within software engineering/information systems. However, Ågerfalk and Fitzgerald (2008) relate their research with offshoring and outsourcing.
Providing OSS products	No clearly related research area, however Ågerfalk and Fitzgerald (2008) relate their research with offshoring and outsourcing.
Using OSS development practices	Software process improvement (Aaen et al., 2001; Dybå, 2005), distributed development (Persson et al., 2005), and global (Spinellis, 2006) and agile (Warsta and Abrahamsson, 2003) software development

Chapter 3

Context and Research Design

This chapter briefly presents the context of this thesis: the ITEA COSI project and the Norwegian software industry. Then it introduces the thesis' research design, which is a combination of several studies concerning how organizations adopt and leverage OSS. Finally, we give an introduction to the research methods used to address the thesis' research questions: surveys, case studies, and systematic literature reviews. These methods are common to software engineering research, and the introduction is therefore very brief.

3.1 Research Context

The research presented in this thesis has been performed in the context of the ITEA COSI project and the Norwegian software-intensive industry. This section will give a brief background for these two contexts.

3.1.1 The COSI Project

Co-development using inner and Open source in Software Intensive products (COSI) was a three year European research project (2006 to 2008). COSI was part of the Information Technology for European Advancement (ITEA) 2 programme and it had both industrial and academic participants from Spain, the Netherlands, Sweden, Finland, and Norway.

Project Goals

The overall goal of the COSI project was to understand how software-intensive industry may benefit from OSS and from distributed collaborative software development. More precisely, the project aimed to achieve the goals (G1-G4) listed below (van der Linden, 2006). While these goals were dominated by the project's large industrial partners focusing on embedded systems, the more operational goal of the project was to improve each individual partner's exploitation of OSS. The goals of the project were:

- **G1** Provide instruments, which will determine how and when to perform software engineering in heterogeneous distributed concurrent collaborations.
- **G2** Introduce the advantages of open source methodology in systems development.

G3 Improve the understanding and cooperation between the open source world and the industry.

G4 Improve the capabilities of the system producers to strategically use the shift of software towards commodity.

Project Organization

The project was organized in five work packages (WPs) focusing on business and organization issues (WP1), **development processes** (WP2), models for requirements, architecture, and design (WP3), dissemination (WP4), and project management (WP5).

NTNU and the other Norwegian partners were almost exclusively involved in WP2. The work in this work package focused on processes for providing OSS products, integrating (and maintaining) OSS components into new or existing systems, learning from the development practices used in OSS communities, and participating in OSS communities.

The Norwegian Sub-Project

The Norwegian part of the project was supported by the Research Council of Norway. It consisted of a project manager from ICT Norway and three industrial partners: eZ Systems, Keymind, and Linpro. Oslo Stock Exchange, IT Farm, Tell.U, and Friprog were later involved in the project as well. NTNU participated as an academic partner with Reidar Conradi, Carl-Fredrik Sørensen, Thomas Østerlie, Sven Ziemer, and Øyvind Hauge.

eZ Systems, established 1999, has since their start-up had success providing their own OSS content management system, eZ Publish. eZ Systems has around 60 employees spread across their offices in Norway, Denmark, France, USA, Japan, and Germany. In addition to eZ Publish, eZ Systems provides premium services, product responsibility, and other related software products. Around their products they have a community consisting of customers and contributors from all over the world.

Keymind Computing AS, formed in 1998, is a small consulting company focused on developing, maintaining and supporting IT solutions. When the COSI project started they had six employees, located in three different offices. Keymind uses OSS components and OSS tools in the development of their solutions. Through the COSI project they also released an OSS product named Keywatch (Eide, 2007).

Linpro has since their beginning in 1995 focused on services around the Linux platform and other OSS products. When the COSI project started Linpro had about 50 employees, but it has since then merged with Swedish Redpill. Redpill Linpro has today about 180 employees. Redpill Linpro integrates OSS into their customers' products, provides their own OSS products, participates in the development of other OSS products, and provides a wide spectrum of services based on OSS solutions.

3.1.2 The Norwegian Software Industry

The main part of the Norwegian software sector[1] consists of software houses and software consultancy companies. The 8 750 software companies, which in 2007 constituted the software sector, employed 31 000 employees, and according to SSB (2009) had a turnover of NOK 45 billion (about €5 billion). The majority of these companies are small, and only 1100 of them had five or more employees. Most of the companies focus on consultancy services and provide products mainly to one customer. The percentage of companies developing software products for a resale market is significantly lower, and they employ about 20% of the employees in the software sector.

The Telenor Group focuses on offering mobile subscriptions, fixed phone lines, cable TV, and other similar services. Even though it is not part of the "core" of the software industry, it needs an IT and software infrastructure. In Norway alone, they have an IT department of about 380 employees, contract external consultants at a regular basis, and outsource services to their partners. The Telenor IT employees develop, maintain, and support the software infrastructure which is necessary for Telenor's products and services.

3.2 Applied Research Methods

This section describes the research methods and some of the tools for data collection and data analysis used in this thesis. In addition, it will also present some of the rationale behind using these tools and methods.

3.2.1 An Empirical Approach

Section 2.1.1 identified a general need for more empirical research within software engineering. Fitzgerald and Feller (2001) describe a similar situation for research on OSS. In this work in particular, we have chosen an empirical approach to studying the adoption of OSS for mainly two reasons:

- To tidy up the misconceptions about what OSS adoption really means to organizations, and to allow practitioners to see the true possibilities of OSS.
- To understand practitioners' real practices and problems, and thereby to be able to focus research on issues which really matters.

In addition to the two reasons mentioned above, the ITEA COSI project's research design expected an empirical approach to (1) identify a baseline for how the industrial partners leveraged OSS, and (2) conduct process improvement work in collaboration with them.

3.2.2 Choice of Research Methods

In their classification of software engineering research methods, Glass et al. (2002) list as many as 22 empirical and non-empirical research methods. Each of these methods

[1]The part of the economic activity classified under NACE 72.2, see NACE (2009).

or combinations of them may be applied to almost any research problem. Researchers are thus left with a large number of possibilities. However, as we discussed in Section 2.1.1, software engineering researchers tend to limit themselves to a few options like experiments, case studies, surveys, and data analysis (Höfer and Tichy, 2007). Finding appropriate methods for the research reported here was influenced by three factors.

First, the uncertainty about what OSS adoption entails and the lack of empirical research on adoption of OSS motivated an initial **explorative** and **descriptive** approach. This initial phase of exploration (Phase 1) consisted of the three first studies reported in this thesis (see Figure 3.1). Phase 2, which consisted of two new studies, was used to complete the framework presented in Section 4.3. In Phase 3, we studied the integrator role, and in particular the selection of OSS components.

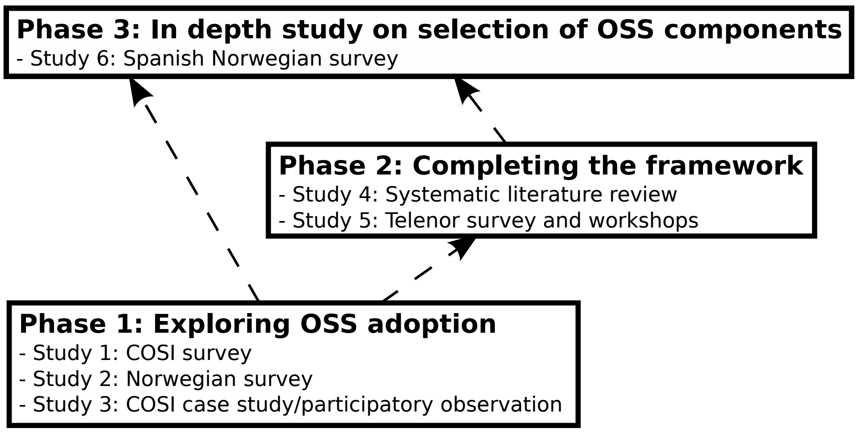

Phase 3: In depth study on selection of OSS components
- Study 6: Spanish Norwegian survey
Phase 2: Completing the framework
- Study 4: Systematic literature review
- Study 5: Telenor survey and workshops
Phase 1: Exploring OSS adoption
- Study 1: COSI survey
- Study 2: Norwegian survey
- Study 3: COSI case study/participatory observation

Figure 3.1: Relation between studies and the three phases of this research

Second, the COSI project through its research design expected participation from the academic partners through our involvement in the execution of several industry **case studies**. This participation may be considered **participatory observation**. The access to a number of geographically distributed and significantly different companies, all of which leveraged OSS, motivated the use of **survey** research.

Third, we wanted to conduct valid and reproducible studies. Even though we have focused on the survey method, we wanted to triangulate and gather data through different methods and from various settings.

In this research we have used different tools for collecting both **quantitative** and **qualitative** data. The quantitative data has been used to understand the scope of OSS adoption and to understand trends. Qualitative data has been used to understand some of the many nuances which we observe when studying organizations. The two main tools for data collection used in this research have been **questionnaires** and **semi-structured interviews**. However, we have also held workshops, participated in project meetings, visited and observed some of the industrial partners on several occasions, and participated in conferences and other events together with them.

3.2.3 Survey Research

Survey research is a method for understanding the opinions and activities of a population at large (Babbie, 1990). The purpose of conducting a survey can be either descriptive, explorative, or explanatory, and Fink (2002, p. 1) defines a survey as:

> a system for collecting information from or about people to describe, compare, or explain their knowledge, attitudes, and behavior.

Surveys are typically conducted by drawing a (representative) sample from a population, and based on the results from this sample, one tries to generalize the results to the whole population. Samples may be drawn through both probability (e.g. random, systematic, stratified random, and multistage) and non-probability (e.g. quota, dimensional, convenience, purposive, and snowball) sampling techniques (Robson, 2002). Surveys can be used to collect both qualitative and quantitative data, typically through questionnaires or interviews (Wohlin et al., 2003). Questionnaires are typically designed to contain mostly closed questions, but open, more explorative questions are also used.

3.2.4 Case Study Research

A case study is a way of studying real projects in a particular context through detailed monitoring and observation of the project's activities (Wohlin et al., 2003). Case study research aims to gather rich evidence from real life contexts (Kitchenham et al., 1995). Yin (2003, p. 13) defines a case study as:

> an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident.

Case studies can, according to Easterbrook et al. (2008), be both exploratory and confirmatory. In the industrial context of the COSI project we had little or no control over the context and the events taking place in this context. Case studies are then a suitable approach to studying and exploring the phenomenon taking place (Yin, 2003). Evidence is typically gathered from a large number of sources like documentation, archival records, interviews, direct and participant observation, post-mortem analysis, and (physical) artifacts.

3.2.5 Systematic Literature Reviews

A systematic literature review is a rigorous and transparent meta- or secondary-study which reviews a set of (empirical) primary studies. Systematic literature reviews can, according to Turner et al. (2008), be conducted to either identify research trends and categorize research papers, or to answer a specific research question. Kitchenham (2007, p. vi) defines a systematic literature review as:

a form of secondary study that uses a well-defined methodology to identify, analyse and interpret all available evidence related to a specific research question in a way that is unbiased and (to a degree) repeatable.

Systematic reviews are characterized by their use of a predefined *review protocol* and a planned, defined, and documented *search strategy* with explicit *inclusion* and *exclusion* criteria (Budgen and Brereton, 2006). The rigor of the publications in this set of primary studies is assessed according to a predefined set of criteria. In the end, the researcher should be able to write a synopsis based on the final sample of (high quality) publications.

3.3 Research Process

This section provides an overview of the research conducted in this thesis. This work is a combination of several related studies centered around how software-intensive organizations leverage OSS adoption. In total, six studies (S1-S6) have been conducted as part of this work. These studies were, as mentioned, conducted in three phases. The next section describes the three phases, the six studies, and the background of each of these studies.

3.3.1 Phase 1: Exploration of how Organizations Adopt OSS

The first phase involved an initial exploration of the opportunities that organizations have when approaching OSS. This resulted in two papers which discuss how organizations can leverage OSS (**P1** and **P3**, also see further discussion in Chapter 4). Moreover, **P2** discusses the level of adoption of OSS in the software sector. In addition to this initial exploration of how organizations approach OSS, we gave a little bit more attention to organizations which integrate OSS components into their software (**P4**), and to one case from the COSI project where a company was providing their own OSS products (**P6**).

Study 1: COSI Survey

The first phase of the COSI project's research design consisted of creating a baseline description of how the industrial partners adopted OSS. Based on a literature review and conversations with several of the project partners, we defined four roles for how the industrial partners adopted OSS: *OSS provider, ISS participant, OSS participant, and OSS integrator*. For each of these roles we wanted to address the following questions:

- Why do industrial actors undertake the four OSS roles?
- What are the advantages and challenges related to undertaking them?
- Which software development practices are used in these roles and how does the adoption of OSS influence these practices?

Based on these questions, we created an interview guide which was used in in-depth **semi-structured interviews** with employees in the Norwegian COSI partners. We interviewed two developers in Keymind, one developer in Linpro, and one developer and the CEO at eZ Systems. Each interview lasted more than three hours, interrupted by breaks.

The interview guide and the results from the interviews were used as a basis for a **web-based questionnaire**. The questionnaire had one part for each of the four roles, and each part covered the questions listed above with several metrics in the questionnaire.

We distributed the web-questionnaire to all of the industrial partners in the COSI project and encouraged them to ask at least one of their employees to respond. Selection of the individual respondents was, because of the composition of the project, left up to the companies. We received in total 24 responses for the four roles: 3 OSS providers, 6 ISS participants, 6 OSS participants, and 9 OSS integrators.

Study 2: Norwegian Survey

The first COSI study was followed up with a new survey which, through the ITEA office, was distributed to all of the project members in all ongoing ITEA projects. Due to restrictions related to access to the project members' e-mail addresses, we had limited control over the population and sampling, and we had no way of sending reminders. Due to low response rates we decided to run an extended version of this survey with a Norwegian sample. The Norwegian survey consisted of two parts.

The first part focused on integration and selection of OSS components. Results from this part are reported in **P4**. It combined the interviews from the COSI survey with a number of new **semi-structured interviews** with Norwegian software developers, several of which were conducted through a Master's thesis by Gerea (2007). Developers from a total of 16 companies were interviewed, some of them more than once. Almost all of the interviews were recorded and transcribed. The goal of the interviews was to understand the challenges OSS integrators faced, and the practices they used when selecting and integrating OSS components into their software solutions.

The second part of the survey covered software-intensive organizations that (1) provided their own OSS products, and (2) integrated OSS components and possibly participated in OSS communities. This part of the survey was conducted through an **e-mail survey** and a **web-based questionnaire**, and it was reported in **P2**. The objective of this survey was to (1) assess the extent of OSS adoption in the software industry, and (2) survey the practices that integrators and providers of OSS used in software development. The e-mail survey consisted of a sample of more than 1000 Norwegian software-intensive companies. We contacted these by email asking four simple questions. We received more than 700 valid responses to our questions concerning the respondents' adoption of OSS. 569 of these companies were involved in software development, and 266 (47%) of them adopted OSS components.

Thereafter, we invited 204 of the 266 organizations that said they had adopted OSS components to participate in a web-based questionnaire. The questionnaire had two parts focusing on (1) providing OSS products and (2) integrating OSS components into a software product and possibly participating in one or more OSS communities. 95 of these companies completed our questionnaire, and left us with 74 useful responses. This part of the survey gave us a lot of descriptive statistics about how the respondents adopted OSS and the practices they used when doing so. Even though much of this material has not been published, it served as valuable input for further studies, and is important for our understanding of OSS adoption.

Study 3: COSI Case Study

The research design of the COSI project consisted of the five phases listed below. Each of the industrial partners were to identify one particular challenge they faced in relation to their adoption of OSS. Based on this challenge, they conducted two iterations with case study and improvement activities.

1. State of the art/baseline of the company's adoption of OSS and related processes
2. 1st case study iteration
3. Improvement of their OSS related processes
4. 2nd case study iteration
5. Improvement of their OSS related processes and validation of improvement

While the COSI project had its defined goals, each individual partner had their own improvement goals within the framework of the project. In accordance with the project's research design, the academic partners aided the industrial partners in improvement activities and helped them in reporting[2] the outcome of these activities. The research we did was thus an ongoing activity for the whole duration of the project.

NTNU participated in a number of activities in the project. All of these activities increased our understanding of the challenges the industrial partners met and the practices they used to deal with them. In addition to 3-4 yearly project meetings, we visited each of the Norwegian industrial partners several times and participated in meetings, workshops, seminars, and conferences together with them. To further increase our understanding of their challenges, we conducted several interviews as well as the survey mentioned above (Study 1). Finally, several Master's students did their theses in collaboration with the industrial partners, e.g. Eide (2007), Gerea (2007), and Schanke (2007).

Results from this combined effort are reported in **P3** and **P6** where we look in particular at the eZ Systems case and some of their experiences. Results from the COSI project are also reported in **P1** and **P4**.

3.3.2 Phase 2: Completing the Framework

To complete the picture of how organizations leverage OSS, we conducted a systematic literature review and another case study or extended survey in a Norwegian company. The iterative process related to conducting the systematic review and creating the framework (see Section 4.3) was a synergic process. On the one hand, conducting the literature

[2]Public deliverables are available from the project's web site: `http://www.itea-cosi.org/`

review helped in developing the framework. On the other hand, developing the framework and having a good understanding of how organizations leverage OSS was crucial to conduct the literature review and to organize the literature.

The companies in the COSI project and the companies in the surveys focused on software development. Even though Telenor IT develops software as well, their focus in our collaboration was not directly software development, but rather deploying OSS products and maintaining them as part of their software infrastructure. However, this software infrastructure was used in their software development.

Study 4: Systematic Literature Review

Based on our previous studies and the lack of clarity about what OSS adoption really is, we conducted a systematic literature review of the published research on OSS in organizations. This study is reported in **P8**. The goal of this study was to answer the following questions:

- How do organizations adopt OSS?
- What has been the focus of empirical research on OSS adoption?
- What are the characteristics and limitations of this research?

The review was conducted in accordance with the guidelines by Kitchenham (2007) for systematic literature reviews. We reviewed papers published in 24 journals and 7 conference and workshops proceedings from 1998 to 2008. From a population of close to 25 000 publications, we got more than 1500 hits on the keyword "open source". These publications were reviewed in several iterations and we ended up with a sample of 112 empirical papers with evidence on how organizations adopt OSS.

Study 5: Telenor IT Survey

To study another way of adopting OSS, we performed a survey with Telenor IT Norway. Telenor is a large Norwegian telecom/mobile operator which was considering to increase their adoption of OSS operating systems, databases, application servers etc. Much of the work was conducted through a Master's thesis by Skarpenes and Velle (2009). Some of the results related to risks and risk mitigation are reported in **P7**. The aim of the work was the following:

- How and to what extent is Telenor IT adopting OSS today?
- What are their employees' attitudes towards an increased adoption of OSS products?
- What are the benefits and drawbacks (risks) of such an increased adoption?
- Which measures can and should be put in place for succeeding (reducing the risks) with an increased adoption of OSS products?

In this work we conducted **semi-structured interviews**, a **questionnaire** with responses from more than 80 employees, and two **workshops** with members of Telenor's Open Source 2010 project group and other employees of Telenor IT. One of the workshops was organized as an interactive KJ session (Birk et al., 2002), where the participants were engaged through (1) writing down their concerns on post-it notes and placing them on a board, and (2) reorganizing these notes into groups of related issues. The interviews, questionnaire, and workshops reflected the goals listed above.

3.3.3 Phase 3: Going in Depth on Selection of OSS Components

In creating the framework and elaborating how software-intensive organizations adopt OSS, we maintained a bird's eye view of OSS in organizations. In this view there are several opportunities to go further in depth in a variety of issues (see **P8**). In the light of earlier work (Torchiano and Morisio, 2004; Li et al., 2006a, 2008; Ayala, 2008) and (**P4**), we saw a particular opportunity to look closer at the challenges related to integrating and selecting OSS components.

Study 6: Spanish/Norwegian Survey

The Norwegian survey was followed up with an extended Norwegian/Spanish collaborative effort. Preliminary results from this work are reported in **P5**. In total, the study consisted of 23 **semi-structured interviews** with a convenience sample of Spanish and Norwegian software developers. Even though the sample was a convenience one, we were able to get a good representation of a variety of different companies. All but two of the interviews were recorded and transcribed. However, only the first eight were included in **P5**. Results from all the interviews will appear in a new paper by Ayala, Hauge, Franch, Conradi, and Li. The interviews aimed at answering the following two questions:

- How do software developers select OSS components?
- Which resources do these developers use when selecting OSS components?

3.4 Evaluation and Validity

Research should be conducted in such a way that its results are believable, or in other words valid. Yin (2003) recognizes that the validity may be increased through triangulation by using different data sources, different researchers, different theories, and different research methods. The research reported here is conducted in collaboration with other researchers. We have collected data from several different settings as an ongoing activity for the last four to five years. While we have had a focus on survey research, we have also used different methods for gathering data. Nevertheless, this work is not without limitations. These limitations are discussed in each of the papers, but we will also provide a brief validity discussion in Section 6.6. Wohlin et al. (2000) listed the following four types of validity:

- **Internal validity** concerns the degree to which the results can be trusted, based on the study context. Internal validity is discussed in Section 6.6.
- **External validy** concerns the degree to which the results can be generalized to other populations and settings. External validity is also discussed in Section 6.6.

- **Construct validity** concerns the relationship between theory and observation. This work has mainly been explorative and descriptive, rather than theory testing, and construct validity is therefore not that relevant. However, construct validity also concerns using the right tools and metrics for gathering the data. This will be discussed in Section 6.6.
- **Conclusion validity** concerns whether or not the right conclusions are drawn based on the collected data. The conclusion validity is in, experiments and qualitative analysis, concerned with ensuring (significant) statistical relationships between the treatment and outcome, or the independent and the dependent variables. Even though we have gathered qualitative data (**P1**, **P2**, and **P7**), these have mainly been used to explore and describe rather than to test (causal) relationships. Statistical conclusion validity is therefore not that relevant for this thesis.

Chapter 4

Results Part 1: Organizational Adoption of OSS

This chapter presents three of the thesis' contributions (C1-C3). First, we illustrate empirically how several organizations adopt OSS (C1). These organizations adopt OSS quite differently, and they are therefore gaining different benefits and facing different challenges. Second, we present the results from a systematic literature review on OSS adoption (C2). Third, to help organizations in (1) seeing the real opportunities in OSS adoption and (2) understanding the challenges they may face, we have developed a classification framework containing six ways in which organizations may adopt OSS (C3).

These three contributions constitute a platform for future research on OSS adoption by establishing a solid empirical foundation and providing increased precision to researchers and practitioners who talk about OSS adoption.

4.1 C1: Descriptions of Actual Adoption of OSS

We observe that **different organizations adopt OSS in distinctly different ways**. We made this observation already in **P1** where we presented four roles for how organizations adopt OSS. The roles were: *OSS integrator*, *OSS participant*, *OSS provider*, and *Inner Source Software (ISS) participant*. An ISS participant is an organization which uses OSS tools and development practices within the organization or a consortium of organizations (van der Linden, 2006). **P1** furthermore discussed some of the motivations these four roles have for adopting OSS, and some of the challenges they face (see Table 4.2). The differences between how organizations adopt OSS are further illustrated by **P6** and **P7**:

- In **P6** we reported from our collaboration with the **OSS provider**, eZ Systems. The paper presents a brief timeline from the "accidental" release of an OSS product, through the evolution of a successful and healthy OSS company. We focused on the development of eZ System's main product from two independent software products to a three-layered system architecture. With this three-layered architecture, eZ is able to attract quite significant contributions to the plug-in layer on the top and the component library on the bottom, while maintaining quite strict control over the business-critical middle layer. We discuss some of the possible benefits and challenges of having a community and provide a few lessons learned from the case, focusing on the importance of allowing the adoption of OSS (or business model) to evolve.

- **P7** reports a study conducted together with Telenor IT. The paper focuses on the advantages Telenor IT sees in increasing their **deployment of OSS products**, the risks related to such an adoption, and steps for mitigating these risks. Telenor IT is primarily adopting OSS to increase its freedom from their proprietary vendors and thereby reducing their expenses on licenses and support. Telenor IT is mainly concerned about 24/7 support for the products they adopt, since around-the-clock operations is key to their business. Ensuring support and placing the responsibility for operations is therefore an important step to reduce risks for Telenor IT. By contrasting these findings with the literature, we illustrate that the companies have different motivations for adopting OSS. Moreover, by combining findings from the Telenor study with the literature (see Table 4 in **P7**), we begin bridging the gap between literature and practice. We furthermore aim to make OSS adopters aware of some of issues to consider when adopting OSS, and some of the steps they may take to deal with these issues.

We see that these companies adopt OSS differently because they want to gain different benefits from OSS. Telenor IT Norway wants to reduce its costs on proprietary software licenses and support agreements, while eZ Systems wants to create a sustainable business around their own OSS products. eZ Systems benefit from reduced expenses on marketing, simplified sales, and community contributions. Moreover, the organizations described in **P4** and **P5 integrate OSS components** into their software systems and gain benefits from software reuse, reduced license fees, and the availability of OSS communities. Quite often they also benefit from the availability of a lot of information (see Chapter 5).

Finally, in **P3** we discussed three ways of OSS-based software development; *developing with OSS products*, *developing OSS products*, and *developing with OSS tools and practices*. We argue that OSS is used in different ways by each of the companies participating in the COSI project and that the way they adopt OSS is shaped by the opportunities they see in OSS. We exemplify this by showing some of the many practices the COSI companies use when adopting OSS in a certain way (see Table 1 in **P3**).

We moreover see that **the adoption of OSS in software intensive organizations is significant**. In **P2**, we found that from a sample of 569 Norwegian companies developing software, 47% were integrating OSS into their products. These products served all main business sectors and covered a spectrum of different functionalities. However, in our sample we saw that the respondents had a small overweight of customers within the public sector, and primarily delivered web-based and enterprise solutions. Consultancy companies used OSS components more frequently than software houses, and large companies adopted OSS components somewhat more frequently than smaller ones. In addition, we found that about 16% participated in OSS communities and 5% provided their own OSS products. However, in the web-survey, 30 out of 66 respondents said that they contributed to one or more OSS communities. This participation was in most cases limited to forum

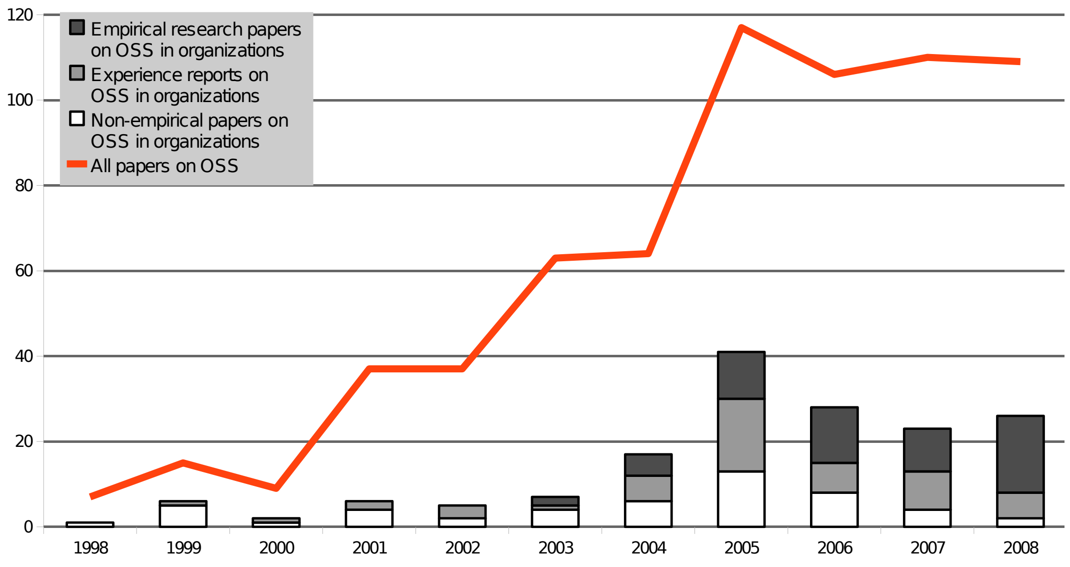


Figure 4.1: The number of papers on OSS and OSS in organizations

activity, bug reports, bug fixes, and so on.

Finally, we saw that a significant part of the respondents in the web-survey relied quite extensively on OSS. However, the extent to which their business was centered around OSS, varied. This is exemplified by **P6** and **P7**. While OSS is a (small) part of Telenor IT's business, eZ Systems has established its whole business around a few OSS products. We saw the same in **P2**. While, 43% of the respondents said that less than 20% of their turnover came from OSS related activities, 13% responded that more than 80% of their turnover came from such activity.

4.2 C2: A Systematic Literature Review on Adoption of OSS

Another important contribution from this thesis is a systematic review of the literature on OSS adoption (**P8**). In this review we systematically evaluated and classified empirical research on OSS adoption and created an overview of this research. This overview forms a solid foundation for this thesis and hopefully also for the work of others to come.

From a population of close to 25000 papers we identified 674 publications which we found relevant to OSS, and 112 publications containing empirical evidence on how organizations adopt OSS. From Figure 4.1, we see that there was an increase in the number of publications focusing on OSS adoption around 2004. However, empirical research on OSS adoption was still limited when the work with this thesis started (fall 2005).

We found that the published empirical evidence on OSS in organizations covered a wide spectrum of topics, had little overlap, and was dominated by experience reports. Almost all of the publications came from Europe or the US, and most of them could benefit from increased rigor in their descriptions of method, research questions, findings etc. There was moreover an emphasis on studies which presented evidence from only one context.

To provide an overview of this empirical research in the literature, we classified the 112 empirical publications from the literature review using our classification framework. From this classification we found that a large number of papers do not focus on the specific ways of adopting OSS (Figure 4.2). There was instead an overweight of papers focusing on general issues like motivation for adopting OSS (Bonaccorsi and Rossi, 2006; Morgan and Finnegan, 2007; Ven and Verelst, 2008), the extent to which OSS is adopted (Nikula and Jantunen, 2005; Lundell et al., 2006), and so on. There is also some research on the adoption of OpenOffice.org in the public sector (Ven et al., 2006; Rossi et al., 2006; Dobusch, 2008). However, the majority of the publications focus on a diverse set of issues.

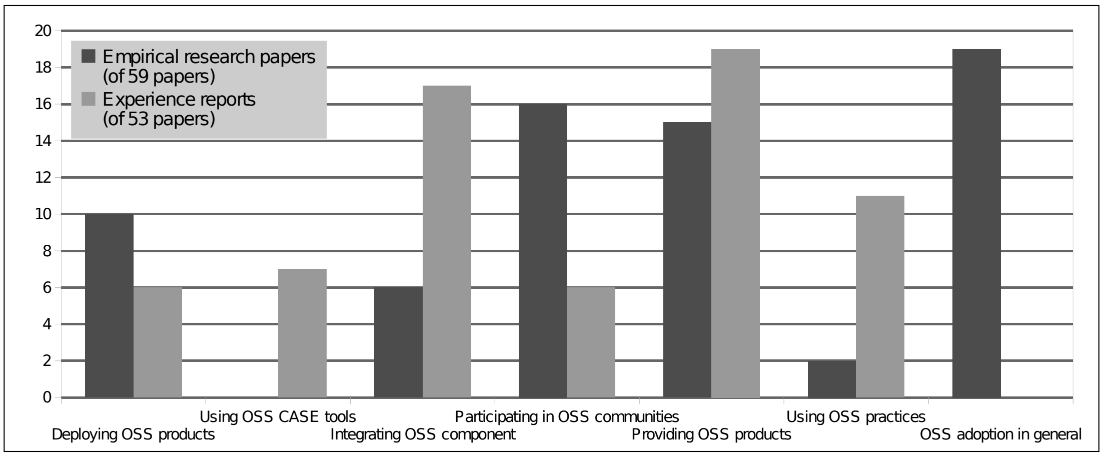


Figure 4.2: Overview of the number of empirical publications on OSS

Even though a clear common goal was missing in the research on OSS adoption, it supported our empirical observation that organizations have different motivations for adopting OSS, and that they adopt OSS in distinctly different ways.

4.3 C3: A Framework for Organizational Adoption of OSS

By *ways of adopting OSS*, we think of ways in which software-intensive organizations can benefit from OSS products, the communities surrounding many of these products, and the collaborative development practices often found in these communities. We briefly present the framework in Table 4.1, before we describe the development of the framework.

4.3.1 Benefits and Challenges Related to Adopting OSS

The different ways of adopting OSS are distinguished by the practical steps organizations take when leveraging OSS. However, the ways of adopting OSS are also distinguished

Table 4.1: Organizational adoption of OSS

Way of adopting OSS	**Description**
Deploying OSS products	Deploy OSS products or tools (e.g. OpenOffice.org, Linux, or JBoss) in their operation environment as end users.
Using OSS CASE tools	Using OSS CASE tools (e.g. Eclipse, Subversion, or GCC) in their software development.
Integrating OSS components	Integrate OSS components (e.g. Hibernate, Google Web Toolkit, or Plone) into their own or their clients' software systems. The components may also be extended or modified.
Participating in the development of OSS products	Participate (through e.g. code contributions, forum activity, or financial support) in the development of OSS products controlled by another organization or community (e.g. Linux, Eclipse, or OpenOffice.org).
Providing OSS products	Develop, maintain, and provide their own OSS products, and relate to the communities around these products. MySQL, Qt Software, and JBoss are examples of such companies.
Using OSS development practices	Use (collaborative) development practices (e.g. code sharing, peer review, and user contributions) often associated with OSS communities to support (distributed) software development inside an organization or a consortium of organizations.

by their advantages and their challenges. These differences make the framework valuable as it may help an organization to understand the real benefits of adopting OSS, while preparing to deal with the right challenges.

Table 4.2: Potential benefits and drawbacks of adopting OSS

Way of adopting OSS	Potential benefits	Potential challenges
Deploying OSS products	● Access to a very large number of software products without licensing fees ● Increased independence from proprietary vendors	● Aligning the use of OSS products with existing technology, skills, and resources ● Ensuring sufficient support and expertise
Using OSS CASE tools	● Access to a large number of professional development tools which are widely used	● The tools might influence the development process and the resulting product
Integrating OSS components	● Access to reusable software components which are developed, tested, and maintained by someone else ● Access to source code, information, and assistance from a community	● Maintaining software systems consisting of (modified) components from several providers over which they have little or no control ● Navigating through and evaluating a large number of products and vast amounts of heterogeneous information
Participating in the development of OSS products	● Sharing development and maintenance costs ● Influencing the direction of the product and ensuring its future	● Deciding what to contribute without giving away critical intellectual property ● Gaining the necessary status to get contributions accepted ● Avoiding excessive use of resources
Providing OSS products	● Increased diffusion of product, and simplified marketing and sales ● Value-adding community contributions (bug reports, bug fixes, extensions, etc.)	● Attracting and sustaining a community ● Aligning the interests of a large number of stakeholders ● Incorporating contributions from the community
Using OSS development practices	● Improved development practices, particularly related to distributed collaborative software development	● Changes to existing practices

In Table 4.2 we provide an overview of what we believe are the most important benefits and challenges related to adopting OSS in a certain way. This overview is based on our work in the COSI project (see **P1**, **P3**, and **P6**), the Telenor case (**P7**), and the literature review (**P8**). This does not however imply that adopting OSS in a certain way guarantees anyone the benefits of OSS. Rather these benefits are the most prominent possibilities which the individual actor may unlock. Moreover, an organization adopting OSS may also face other challenges than the ones listed below.

4.3.2 Relationships between the Ways of Adopting OSS

It must be emphasized that there are interdependencies between the approaches in the classification framework and the possibility that an organization may adopt OSS in several ways at the same time. Campbell-Kelly and Garcia-Swartz (2009) show that organizations evolve how they adopt OSS over time. This further complicates the interdependencies between the different ways of adopting OSS. The categories in our framework are therefore not mutually exclusive. There are, in particular, a few categories which are closely related.

First, organizations which participate in the development of an OSS product are most likely integrating this product into one of their own systems (Jaaksi, 2007; Mannaert and Ven, 2005). In **P2**, we found that at least 30% of the companies which integrate OSS into their systems, also participate in one or more OSS community. Second, organizations adopting OSS development practices are also frequently using OSS CASE tools to facilitate the adoption of these practices (Lindman et al., 2008; Wesselius, 2008). In the case described in **P6**, we saw how a provider of OSS products relied on the integration of OSS products developed by someone else. A consequence of this reliance was that they also spent a quite significant amount of resources participating in the development of PHP.

Next, the difference between a few of the categories is one of degrees rather than orthogonality. Grand et al. (2004) make similar observations, in that the dedication to OSS is a matter of allocating different levels of resources. There are, for instance, overlapping areas between deploying OSS products and integrating OSS products into a system (Adams et al., 2005b,a), and between participating in the development of an OSS product and providing an OSS product (Ågerfalk and Fitzgerald, 2008; Dahlander et al., 2008). While **P7** focused on the deployment of OSS (infrastructure) products, Telenor IT was also integrating OSS components into some of their products. Different organizations may also adopt the same OSS product quite differently. In one case, several organizations simply deployed Linux on their severs (Ven and Verelst, 2006), while others extended it, integrated it into their products, and participated in the development of it (Henkel, 2006).

There are also internal differences within each of the ways of adopting OSS. Even though different organizations provide OSS products, they have different motivations, resources, and success. Where Bleek et al. (2005) describe a public project which is struggling, **P6** describes an organization which has successfully established a sustainable business around providing OSS products. Li et al. (2006c) show that the reuse of OSS components is adapted to the organizations' individual development processes. In **P2** we show that the level of participation for most organizations is limited. However, other organizations may participate quite significantly (Henkel, 2006; Robles et al., 2007).

4.3.3 The Development of the Framework

The framework has undergone a certain evolution. Figure 4.3 illustrates this evolution and the relationship between **P1**, **P3**, and **P8**. Already in **P1** we presented four roles for how organizations adopt OSS. We discussed the roles; *OSS integrator*, *OSS participant*, *OSS provider*, and *Inner Source Software (ISS) participant*. Then, in **P3** we discussed three ways of OSS-based software development; *developing with OSS products*, *developing OSS products*, and *developing with OSS tools and practices*. Finally, in **P8** we developed the full framework, consisting of six ways of adopting OSS. Results from this paper are already presented in Section 4.2 and in the description of the framework, in Table 4.1.

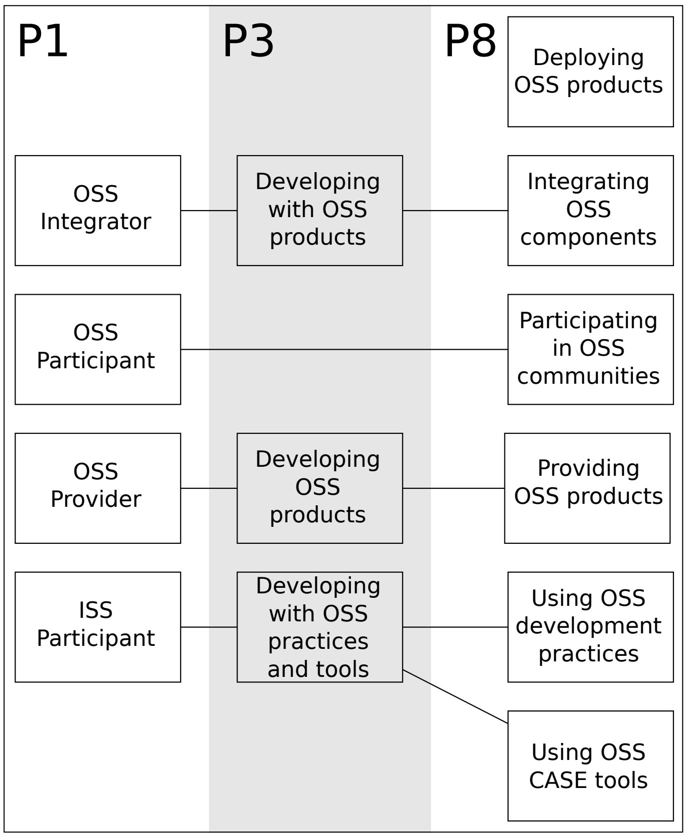

P1
P3
P8
Deploying OSS products
OSS Integrator
Developing with OSS products
Integrating OSS components
OSS Participant
Participating in OSS communities
OSS Provider
Developing OSS products
Providing OSS products
ISS Participant
Developing with OSS practices and tools
Using OSS development practices
Using OSS CASE tools

Figure 4.3: The development of the framework for organizational adoption of OSS

Chapter 5

Results Part 2: Selection of OSS Components

Limited understanding of actual selection practice, has led researchers to make unrealistic assumptions about how to improve practices. While numerous quality models, selection methods, and evaluation schemes have been proposed, they see very limited use. Empirically grounded insight into actual practices is therefore essential to understanding how to inform these practices.

Based on surveys of the selection practices used in a number of software-intensive organizations, we identify and describe several practices for identifying and evaluating OSS components. The **new insight into actual selection practices** offered by the empirical evidence in **P4**, **P5**, and from the web-survey in Study 2, constitutes the fourth contribution of this thesis (C4). We find that the use of formalized methods is very limited and that developers:

- *Identify* components through using previous experience, monitoring OSS related resources, getting recommendations, and unstructured web searches.
- *Evaluate* components based on previous experience, the track record of the component, informal evaluation of community resources, and prototyping.

With this insight as a background, we have developed a **model for situated software selection** (C5). This model offers a new dimension to the research on software selection. Most of this research has criticized practice for being ad hoc and unreliable, and it has tried to formalize selection through quality models, selection methods, and evaluation schemes (see Section 2.3). Our situated selection model shows that the many constraints inherent to the situation the selection is conducted in, significantly influence the outcome of the selection process. These constraints furthermore contribute to preventing the adoption of formalized methods. Even though formalized approaches are rare, we argue that current practices are not ad hoc and unreliable. Developers benefit from their experience and the experience of others, and thereby reduce the uncertainty and costs related to evaluating, learning, and relying on new technology. Based on these observations we suggest directions for future research.

5.1 C4: Descriptions of Actual Selection Practices

5.1.1 The use of Formalized Selection Methods

The use of formalized (or normative) selection methods was almost not existing, and almost none of the respondents in our surveys mentioned using any of the methods proposed by the literature (see Section 2.3). When asked, "*To what extent did your local business unit perform the following activities when evaluating OSS components to this product?*" the respondents in the web-survey (Study 2) gave the following items an average rating[1] of 1.73 and 1.48 (see Table 5.2):

- Used documented checklists to evaluate the components
- Used a selection process which is well documented in the company

When asked about their typical selection process, one interviewee said that "*there is no defined method in that sense*" [P5.ES1]. The norm was instead to utilize an informal and developer dependent selection of components, using the practices for identifying and evaluating OSS components described below.

However, some of the respondents said that they had their own informal, lightweight processes for selection, particularly for central and important components. In addition, a few companies had standardized architectures, consisting of a set of components, that were maintained by an architecture team. In some cases, these teams conducted the selection on behalf of the individual projects. Hence, some companies had processes for selecting components, but these were generally not influenced by the research literature.

Nevertheless, one organization in the second part of Study 6[2] did in fact try to follow the Open Source Maturity Model (OSMM). While both the respondents from that organization agreed that it had been useful to use this method, they did not apply it on a daily basis. "*We could not do it as it implies a lot of time to find all the information, so for non critical components we mostly base our decisions only on recommendations from our development team*" [ES12].

5.1.2 Identification of OSS Components

Perhaps the most prominent source of components, and information about these components, is the individual developer's **previous experience** with the components. As one respondent said: "*It's usually a matter of using people's experience and knowledge about OSS components*" [P4.Beta]. Previous knowledge was also rated relatively high in the web-survey in Study 2 (see Table 5.1[3]). This knowledge comes first of all from experience with specific components through, for instance, previous projects. In addition, many

[1]Using a five point Likert scale with the following labels: None at all (1), Small (2), Some (3), Large (4), and Very large (5)

[2]Results from this part of the survey will appear in a paper by Ayala, Hauge, Franch, Conradi, and Li.

[3]Using a five point Likert scale with the following labels: None at all (1), Small (2), Some (3), Large (4), and Very large (5)

developers spend time monitoring both the OSS community and specific domains. “*We follow a lot of forums, news groups and stuff like that to monitor the areas of interest*” [P4.Upsilon].

The **monitoring** was frequently mentioned as being done through the use of web portals, news sites, mailing lists, forums, RSS-feeds, and blogs. One respondent said that he read “*different private blogs where one basically picks up trends*” [P5.NO2]. These resources were typically related to the technological platform(s) (e.g. Java or .net), but some were also related to either technical (e.g. content management systems or web applications) or business (e.g. banking or energy) domains. The resources reported the experiences of other developers who had used certain components, together with news about recent releases, and so on. Through this monitoring, the developers were frequently aware of the components they would need, or at least where to look for them. The monitoring can thus be seen as an early investment for later identification of components.

Many of these web resources contained **recommendations** of specific components for specific use. “*We select components people are talking about by reading articles on certain web sites*” [P4.Delta]. The developers we interviewed were also using these resources to request advice and recommendations (typically forums and mailing lists). However, these web resources were not the only place where developers sought advice. Colleagues and other people in the developers’ social network were often consulted when they were looking for components, and some respondents said they looked at similar (OSS) products to see which components they contained.

Finally, when the developers were unaware of any components, and could not find any useful advice through their familiar channels, they opted for unstructured **web searches**, typically through Google. “*When we do not have a clear idea of the kind of components that may cover the functionality we are looking for, we directly go to Google*” [P5.ES2]. Sources like SourceForge and other language or domain specific repositories were also used. “*We google what we need or we go to some repository*” [P4.Theta]. However, repositories like SourceForge were more prominent in the web-survey than in the interviews (see Table 5.1). Google was by far the service most frequently mentioned by the respondents. This could be explained by the fact that some respondents found it a bit cumbersome to navigate through SourceForge and similar sources.

5.1.3 Evaluation of OSS Components

Previous experience was again the most important factor. If a developer had a positive experience with a suitable component, it was often reused right away without closer evaluation. “*If someone has experience [with a component], we normally select this one*” [P5.ES2]. Another respondent explained this further saying that, “*if you have used the tool or the component before . . . you know it works*” [P4.Gamma]. Our respondents in both the interviews and the web-survey also relied on the **experiences of other people** whom they trusted (see Table 5.2). They moreover put a lot of emphasis on such recommendations. Equally, a component was rapidly rejected if the developer had or heard

Table 5.1: Practices used to identify OSS components (n=66)

Practice	Mean	STD
Searched OSS portals (SourceForge.net, tigris.org, apache.org, eclipse.org, etc.)	3.67	0.95
Used search engines (Google, Msn search, etc.) to search for individual OSS components	3.59	1.05
Selected OSS components based on previous experience	3.51	1.16
Used search engines (Google, Msn search, etc.) to search for comparisons of several OSS components	3.18	0.94
Asked friends and colleagues etc. whether they know any OSS candidate components	3.05	1.10
Requested advice at forums and mailing lists	2.43	1.13
Reviewed books and magazines	2.17	1.04
Used a company internal "knowledge base"	1.90	1.17
Used an (external) component broker to find the component	1.18	0.70

negative experiences concerning it.

Not only the experiences of colleagues were consulted, but also those of other developers who reported their experiences on the Internet. "*Another thing which we almost always do is to read opinions . . . and examine a bit the experience other people have*" [P5.ES3]. They do this because, "*if we can say, by looking at the Internet or the references that . . . this is a component that is used in other places . . . we do not need to have . . . a very serious evaluation*" [P4.Tau]. These experiences may consist of (problem) reports or discussions in forums, comparisons of several components in a blog post, reviews or tutorials in domain specific portals, or similar. We see that all of these sources are rich, textual experience reports.

Moreover, a famous (OSS) product's usage of a certain component was considered to be a testimonial to the component's usefulness and quality. That the component had a **track record** was in fact considered to be quite important by several of the respondents. However, "*sometimes it is difficult to formulate an opinion from information contained in the Internet because some of the opinions are extremely contradictory*" [P5.ES2]. Often, to deal with contradictory information, a variety of sources was consulted.

Many of the communities behind OSS products offer open access to a large number of information resources like web sites, issue trackers, documentation, forums, mailing lists, and of course the product's source code. These **community resources** are used to get an impression of the product, potential problems with the component, how active the community is, its plans for the future, and so on. However, we found that respondents primarily made such assessments in an unstructured manner, based on gut feelings instead of rigorous evaluations following extensive lists of evaluation criteria.

To be able to evaluate whether or not a new component was good enough for the intended use, most respondents said they did some prototyping or test integration with it. "*If we do*

Table 5.2: Practices used to evaluate OSS components (n=65)

Practice	**Mean**	**STD**
Performed testing and/or prototyping with the components	3.74	1.15
Looked for references and/or other experiences with the components	3.46	0.99
Reviewed documentation for the components	3.35	1.04
Assessed the activity within the community around the components	3.05	1.27
Estimated how much effort would be used on selection and integration of the component and included this estimate into the project plan	2.95	1.19
Performed architecture reviews of the components	2.64	1.15
Defined a list of requirements for the components before the selection started	2.63	1.27
Used shortlists (to identify several components with similar functional-ity)	2.42	1.07
Documented the choice of the components and the rationale behind this choice	2.32	1.03
Performed code reviews of the components	2.00	0.98
Used documented checklists to evaluate the components	1.73	1.08
Used a selection process which was well documented in the company	1.48	0.86

not know the component beforehand, we just try to make a prototype" [P4.Beta]. Through this **prototyping** they gained experience with the component, got a feel for how easy it was to integrate it with other components, and assessed the component's documentation like tutorials, API descriptions etc. Prototyping was also rated highly in the web-survey (see Table 5.2). We believe this is because it is so easy to download an OSS component and test it, and because getting first-hand experience with a component gives the developer rich experience with its capabilities.

5.2 C5: Situated Selection of Components

Based on the empirical evidence from Studies 2 and 6, we have developed a model for *situated selection*. By "situated" (Suchman, 1987) we mean that the selection process is always situated in a particular context with a particular team of developers[4] (see Figure 5.1). This offers a new dimension to the understanding of selection and the use of formalized selection methods.

The selection of OSS components is, like many other decisions in life, influenced by a number of factors besides the objective requirements. When buying a car for instance, one might have a preference for a certain brand, might get recommendations from a friend, and might already trust a certain dealer. The goal of the model in Figure 5.1 is to illustrate the complexity of component selection. Moreover, it shows that this selection is constrained and heavily influenced by a number of factors besides the client's requirements, and the many evaluation criteria suggested in the literature (see Section 2.3).

[4]By developers we also include architects, designers, testers, and so on.

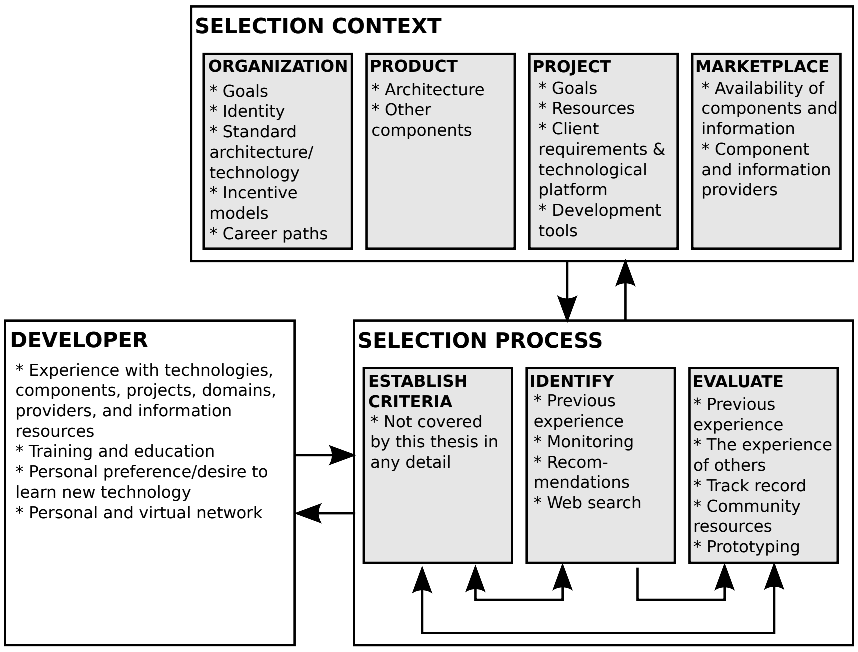

SELECTION CONTEXT
ORGANIZATION
* Goals
* Identity
* Standard architecture/ technology
* Incentive models
* Career paths
PRODUCT
* Architecture
* Other components
PROJECT
* Goals
* Resources
* Client requirements & technological platform
* Development tools
MARKETPLACE
* Availability of components and information
* Component and information providers
DEVELOPER
* Experience with technologies, components, projects, domains, providers, and information resources
* Training and education
* Personal preference/desire to learn new technology
* Personal and virtual network
SELECTION PROCESS
ESTABLISH CRITERIA
* Not covered by this thesis in any detail
IDENTIFY
* Previous experience
* Monitoring
* Recom-mendations
* Web search
EVALUATE
* Previous experience
* The experience of others
* Track record
* Community resources
* Prototyping

Figure 5.1: Situated selection of OSS components

5.2.1 The Selection Process

The purpose of component selection is to find a component that provides certain functionality. In **P4** we found that developers often select the first component which is good enough ("first fit"), rather than struggling to find the best component ("best fit")[5]. While several components could be compared to each other at the same time to find the best one, the respondents often identified and evaluated one component at a time. If the first component was assessed to be good enough, it was selected. If it did not satisfy the developer's requirements, he identified another component and evaluated this one. This is confirmed by **P5**.

Rather than exploring the unknown, developers tend to stick with what they have found and what they know of (as long as it is good enough). One respondent said that "*we prefer to use a component we already know, rather than assuming the risk of using a new one, even when the new component could perform better*" [P5.ES4].

Even though we have separated *identification* of components from *evaluation* of components in both the papers and in Figure 5.1, the relationship between these two activities is very close. When a developer looks for components he is constantly evaluating what he finds, such as project web sites, recommendations from colleagues, or forum posts. Moreover, when evaluating a component he may identify other related components through, for instance, experience reports from other developers. Hence, there is an ongoing interaction between identification and evaluation.

[5]Mistree and Allen (1997) use the terms satisfy (first fit) and optimize (best fit) to make a similar distinction.

This thesis has not focused on the establishment of evaluation criteria and requirements engineering. However, we have included this activity in Figure 5.1, because we saw that the learning taking place during the identification and evaluation of a component was significant. If the component was rejected, the developer would not only have a better understanding of the problem, but also of possible solutions to it. Hence, he would have new and more precise criteria for the evaluation of the actual component and of similar components. While the developer bases his evaluations on the initial requirements from the client, many of the evaluation criteria are in fact not identified until the identification and evaluation of the components actually starts.

A three way relationship exists between the (1) requirements/evaluation criteria, (2) the selection context and the developer, and (3) the available components. Based on the initial requirements, the selection context, and the developer's previous experience, the developer starts looking for components in the search space he is familiar with. This search space constrains and to some extent determines the outcome of his search. The available components and the evaluation of these extends the developer's experience and (most likely) causes him to update and change his initial requirements and evaluation criteria.

5.2.2 The Selection Context

The selection context consists of the organization and the project developing the product, the product itself, and the component marketplace. All these elements set boundaries for the outcome of the selection.

Some organizations use standard architectures, have close relationships with certain providers, or have decided to be, for instance, a 'Java company". Moreover, the product and/or the client may already have an (IT) architecture consisting of other products and components. Any new components must be compatible with the existing ones. The client may also have close relationships to a vendor or support provider, or have employees with certain training. "*Often ... the client wants this [particular] technology*" [P5.ES1]. Finally, the marketplace may not offer a large number of similar components. At least in some of the domains, certain combinations of OSS components were more or less de facto standards, or at least very commonly used together. For instance, Java web applications often consisted of Spring, Hibernate, several Apache common libs, and so on.

5.2.3 The Developer

The developer(s) conducting the selection was obviously one of the strongest factors influencing its outcome. Their experience with technologies, domains, component providers, information resources, and specific components influenced not only where they would look, but also what they would look for, and how (closely) they would evaluate a component. "*There are a lot of portals about OSS and technologies, but I tend to use the ones I usually follow and trust*" [P5.ES3]. The developers' interest in technology and desire to learn new technologies also influenced their decisions. "*Even though the team has ex-*

perience with one line of technology, it may want to learn another" [P5.ES1]. Finally, the developers were influenced by advice from their network of both personal and on-line contacts.

Chapter 6

Discussions and Evaluation of the Research

In this chapter we evaluate and discuss the research presented in Chapters 4 and 5. First, we discuss the results in relation to the research literature. Second, we discuss implications for research and practice. Third, we evaluate the fulfilment of the research questions (see also Table 7.1). Fourth, we evaluate how the thesis has contributed towards completing the COSI project's goals. Fifth, we present a brief discussion of issues related to the validity of the thesis' results. Finally, we briefly discuss the scope of the thesis.

6.1 Results vs. Existing Literature

6.1.1 Adoption of OSS

In Chapter 2 we highlighted several shortcomings in the OSS literature. First, several OSS researchers and advocates have described OSS as a homogeneous phenomenon and as something different than *traditional* software and software development. Second, researchers have discussed *adoption of OSS* without really clarifying what they mean. Third, there has been uncertainty about what OSS adoption actually entails.

This thesis extends, and to some degree contradicts, previous work through empirically illustrating that **organizations adopt OSS in distinctly different ways**, and by providing a classification framework describing six such ways. By doing this we show that **OSS is a multi-faceted phenomenon** which organizations actually embrace quite differently, and we increase the understanding of what OSS adoption actually entails.

Kitchenham et al. (1995) recommend that researchers describe the context of the studied organizations. Since organizations adopt OSS in different ways, this recommendation becomes even more important. Our framework may be used for this purpose, as researchers may use it to (1) describe how the organizations they study adopt OSS, and (2) position and align their work. We have used the framework to support three activities. First, we have used it to organize the literature on OSS adoption (see **P8** and Section 4.2). Second, we have used the framework to relate OSS research to other relevant software engineering research (see Section 2.4.3). Third, we have used the framework to suggest directions for future research (see Section 6.2.1).

We support, Grand et al. (2004) and Lundell et al. (2006) when they state that many organizations show significant commitment to OSS, and the observation of Bonaccorsi et al. (2006) that most organizations combine OSS with commercial (and often proprietary) products and services. We moreover agree with Fitzgerald (2006) who claims that OSS is evolving into a more commercially viable form. For instance, Henkel (2006) and Robles et al. (2007) illustrate this by showing that many OSS products are to a large extent developed by paid developers.

As a consequence of the commitment many organizations show to OSS, we claim that the strict distinction many OSS and software engineering researchers make (between traditional software (development) and OSS) is futile. OSS should rather be treated as a part of software engineering. We furthermore find profound similarities between research on OSS adoption and established fields within software engineering and information systems research. Thus, we relate OSS research to these areas (see Section 2.4.3).

Unlike existing literature reviews (Feller et al., 2006; Scacchi et al., 2006; von Krogh and von Hippel, 2006; Østerlie and Jaccheri, 2007; Stol and Babar, 2009), which give little attention to organizations, our review focuses on the adoption of OSS in organizations. In addition, we introduce the *systematic* literature review to the OSS research arena together with Stol and Babar (2009).

While there are publications which focus on how organizations approach OSS through business models (Hecker, 1999; Fitzgerald, 2006) and allocation of resources (Grand et al., 2004), few focus on the practical implications of using OSS for software-intensive organizations. This thesis, on the other hand, aims to explain how software-intensive organizations adopt OSS and some of the implications of this adoption.

6.1.2 Selection of Software Components

Our findings relating to the practices developers use when selecting components are similar to previous research on COTS (Tran and Liu, 1997; Kunda and Brooks, 2000; Torchiano and Morisio, 2004; Li et al., 2005, 2006a) and OSS (Norris, 2004; Chen et al., 2008). We may conclude two things from this observation:

- Since the practices used to select OSS and COTS are similar one could argue that the two types of components are not that different in use. OSS is not something very different from COTS and traditional software. This observation is supported by e.g. Li et al. (2006b) and Ajila and Wu (2007).
- The selection practice has not changed that much over the last ten years. Selection is still conducted as an informal process, based on the individual developer's experience, unstructured web searches, and prototyping. Research on selection, and in particular formalized selection methods, has not seen successful adoption, and has not been able to influence practice to any significant degree.

In the eighties and nineties, software research was criticized for having a bias towards proposing a string of new methods (see Section 2.1.1). These were not validated and they had limited impact on practice. The same can be said about much of the research on both COTS and OSS selection. Researchers have proposed a number of selection methods and evaluation schemes, but few of these have been empirically validated and applied in practice. Based on the limited influence research has had on practice, several researchers have suggested various explanations as to why these efforts have failed (see Section 2.3.2). However, not all of these explanations are satisfactory. We should therefore ask ourselves: (1) why is this so, and (2) are there other ways to aid practitioners in software selection?

In the dominating understanding of software selection, selection has either been *ad hoc* or *formalized*. We believe that our model for situated selection adds a new dimension to the discussions around software selection (see Figure 6.1). Rather than being either ad hoc or formalized, selection practices are fitted to the situation the selection is conducted in. The new dimension added by the situated model could contribute to explaining why formalized approaches have had limited impact on practice, and it should influence the direction of research on software selection.

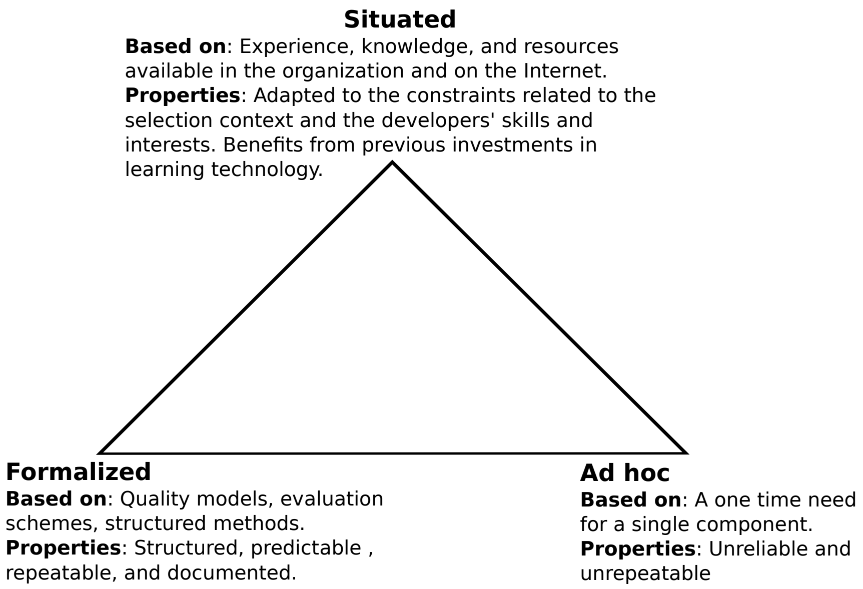

Situated
Based on: Experience, knowledge, and resources available in the organization and on the Internet.
Properties: Adapted to the constraints related to the selection context and the developers' skills and interests. Benefits from previous investments in learning technology.
Formalized
Based on: Quality models, evaluation schemes, structured methods.
Properties: Structured, predictable , repeatable, and documented.
Ad hoc
Based on: A one time need for a single component.
Properties: Unreliable and unrepeatable

Figure 6.1: A three-dimensional view of software selection

We agree to a large extent with the work of Fitzgerald (1996) on methods for software development, and we believe that a few of the constraints of the situation the selection is conducted in are the main reasons for why formalized selection methods have seen limited adoption.

The evaluation criteraia are frequently not defined up front: Lack of, unclear, or overlapping evaluation criteria have been claimed to be shortcomings of the proposed methods for OSS selection (see Section 2.3.2). However, if we are to develop one evaluation scheme containing every useful criteria (reflecting functional, non-functional, non-technical, and other types of requirements), it will become a complex ontology. Defining and agreeing on these criteria is at best very hard (Bowker and Star, 2000). Such schemes will also be very complex to use for developers, regardless of being customizable or not.

In practice, we see that it is difficult for developers to define, select, and weigh all the criteria for evaluating a component until they actually starts identifying and evaluating their alternatives. We support the view of Land et al. (2009) that the actual evaluation criteria used to select components are often vague and not defined before the selection is under way. It is rather "*the candidate [components that] decide the evaluation criteria for you*" [P5.ES1]. We saw that the learning taking place during the identification and evaluation of a component was an important input to the evaluation of the component, and of other components.

A few context specific constraints are far more important than general evaluation criteria: Several of the evaluation schemes for OSS contain a large number of evaluation criteria. Most of these are general and may be used in several contexts. A component may get a high score in such a formalized evaluation. However, we find that, most often, one or a few specific constraints decide the outcome of the selection. Instead of using a number of general evaluation criteria, developers make their decisions based on a few important constraints specific to their own context (see Figure 5.1). This is supported by, for instance, Bhuta and Boehm (2005, p. 140), who found that cost constraints set by the client "eliminated most single-solution, end-to-end COTS products". It is also supported by Land et al. (2008), who found that it was quite common to select one technology or component (a keystone) that would significantly constrain the selection of the remaining components.

In addition, both Li et al. (2006a) and Land et al. (2008) found that the designation of which evaluation criteria are the most important ones depends on the domain, the organization, the particular system, and the importance of the component in question. Different projects used different processes and criteria in their selection of components. This is also acknowledged by Goh et al. (2008, p. 88), who review web portals and state that "each of the four portals has unique strengths and weaknesses, and each may be the best fit under different circumstances and needs".

Developers rely on experience (reports) rather than metrics: While much of the literature has focused on metrics, developers often rely on the rich (text) experiences from other developers, and their own hands-on experiences with the components. An experience report from a trusted source is in many circumstances far more valuable than the (weighted) numbers any metric or evaluation matrix could produce. Li et al. (2006a) made similar observations in that developers relied on newsgroups to evaluate COTS and were therefore more likely to select components with rich forums around them.

Relying on experience saves time and reduces uncertainty: Li et al. (2006a) describe how developers need time to learn and evaluate new technology. By relying on their own or others' experiences with the component, the developers are able to cut the time needed to evaluate components. This may involve everything from relying on the testament of a few people, to, in extreme cases, the testament of thousands or even millions of users. A more formalized approach would require a more thorough evaluation of several components.

The most valuable experience is their own. By reusing a component they know, they reduce the uncertainty related to adopting a new component. Another technology may offer a "better" solution to the problem. However, the reuse of familiar technology enables the developer to benefit from his experience and helps him to avoid the uncertainty related to adopting a new component. The developers may also avoid further diversifying the company's technological portfolio.

6.2 Implications for Future Research

6.2.1 Adoption of OSS

OSS and software engineering researchers should, because of the similarities mentioned above, align their efforts and study common research problems, instead of alienating OSS and treating it as something new and unique (see Section 2.1.2). OSS researchers should in particular borrow more support from related areas within software engineering and information systems research (see Section 2.4.3).

Research on adoption of OSS has so far also been limited and fragmented (particularly up to the point where this work was initiated). OSS researchers should therefore align their effort and focus on a few issues. We agree with e.g. Basili et al. (1986) in that these issues should be of high relevance to practitioners. To this end we:

- Provide empirically grounded descriptions of how organizations adopt OSS (C1).
- Have systematically collected, assessed, and summarized existing research (C2).
- Offer a framework for organizational adoption of OSS, that gives researchers a more precise vocabulary when talking about OSS adoption (C3).

In addition, we provide a set of topics which could direct further research (see Table 6.1). By doing so we hope to create a platform that other researchers can build on in their own research.

6.2.2 Selection of Software Components

Almost 15 years ago, Fitzgerald (1996) suggested changing the focus of research away from formalized methodologies software development. We believe research on general selection methods, evaluation criteria, and quality models could benefit from a similar change of focus. First, we suggest that research should acknowledge developers' use of experience and recommendations, both from people they know and from the Internet. Second, research should reflect the constraints inherent to the situation the selection is conducted in, not just suggest general product-specific evaluation criteria. There are, as pointed out by Land et al. (2008), several concerns that need to be satisfied, not just technical ones. We furthermore agree with Glass (2004) in that research should provide

Table 6.1: Topics for future research

Way of adopting OSS	**Possible topics for future research**
Deploying OSS products	• What are long-term costs and consequences of deploying and keeping OSS products operational?
Using OSS CASE tools	• What kinds of tools are needed for collaborative software development across organizational and community borders? • How do organizations collaborate using such software development tools?
Integrating OSS components	• How may software developers most efficiently navigate through and select OSS components? • How may software developers benefit from OSS communities and the (unstructured) resources available over the Internet? • How can software developers maintain and secure the sustainability of software systems consisting of components from a variety of providers?
Participating in OSS communities	• When, how, and with what should an organization participate in the development of OSS products controlled by someone outside of the organization? • How can we effectively contribute only parts of a product and at the same time retain other parts private?
Providing OSS products	• How are OSS providers able to attract and sustain a community? • What are the success criteria for incorporating contributions (requirements, code, bug reports/fixes etc.) from a community?
Using OSS development practices	• How can development practices from OSS communities be adopted within organizations? • How may organizations successfully collaborate through community or consortium based software development?

practitioners advice on how to use current methods rather than constantly suggesting new ones.

Despite research's excessive focus on suggesting new methods, there are a few interesting trends we believe are worth more attention in the future. Contradictory to many of the suggested selection methods, these trends may result in concrete tools which may be very useful to practitioners.

Metrics for automated quality evaluation (Samoladas et al., 2008). If successful, these metrics may be integrated directly into tools and one may avoid the problems of manually gathering necessary information. However, automated solutions are most likely unable to deal with neither all kinds of requirements and evaluation criteria (functional, non-functional, platform, and non-technical requirements) nor identifying the critical criteria.

Specialized Internet-based search for code fragments and software components (Hummel et al., 2008; Gallardo-Valencia and Sim, 2009). Specialized search tools may find more relevant information while avoiding some of the problems related to irrelevant hits from general search tools. A few tools already exist e.g. Google Code[1] and Koders[2], but these should be developed and explored further. We believe that rich experience and problem reports should be made accessible to simplify the evaluation of the components.

Using a **community-based approach** to populate and maintain reuse repositories (Ayala et al., 2007). This may split the cost of constructing a reuse repository and increase its value. Platforms like ohloh[3] and SourceForge[4] incorporate some of this functionality, but still focus on the developer of the components rather than the reuser.

6.3 Implications for Practice

6.3.1 Adoption of OSS

This thesis shows that there are several ways of leveraging OSS, and that each of these ways involves different challenges. Organizations may also adopt OSS in several different ways at the same time. Organizations should therefore not try to carbon copy the success of others, but rather identify the advantages of OSS adoption in their own context. They may find that OSS provides other opportunities than they expected. Hence, we agree with Melian and Mähring (2008) and Ven et al. (2008) in that organizations must analyse the opportunities and consequences of OSS adoption in their own context.

The framework provided above, and the success stories identified in **P8**, may aid organizations in understanding what "OSS adoption" means to their organization. The concrete advantages, challenges, and experiences reported in **P6** and **P7** may help organizations

[1]`http://code.google.com/`
[2]`http://www.koders.com/`
[3]`http://www.ohloh.net/`
[4]`http://www.sf.net/`

which either provide their own OSS products, or which deploy OSS products into their operation environment.

6.3.2 Selection of Software Components

Practitioners should, to a greater extent, acknowledge the value of employees with OSS based experiences. They should first of all allow, and encourage, their developers to spend time familiarizing themselves with OSS components, concepts, communities, Internet resources, and so on. This is a valuable investment which will reduce later efforts needed to identify and evaluate OSS components. Second, developers should share their experiences with the components they use openly on the Internet. As these experiences are a valuable resource during the identification and evaluation of OSS components, everyone should collectively contribute to this pool of resources. By investing a little bit in contributing to the community, one may encourage others to do the same, and hopefully reduce one's own efforts in the future.

6.4 Results vs. the Research Questions

To answer *how* organizations adopt OSS (RQ1), we have provided a classification framework for organizational OSS adoption (C3). This framework contains six ways of adopting OSS, each offering different opportunities and challenges. The framework is based on empirical evidence (C1) from the COSI project (**P1** and **P3**) and from two Norwegian organizations, eZ Systems (**P6**) and Telenor IT Norway (**P7**). The framework is furthermore supported by evidence from our systematic literature review (**P8**) (C2).

The *extent to which organizations adopt OSS* (RQ1) is partly answered by the empirical evidence (C1) provided by this thesis, and in particular by **P2**. Based on this paper and related literature (Ghosh, 2002; Glynn et al., 2005; Nikula and Jantunen, 2005), we see that the adoption of OSS in software-intensive organizations is *significant*. The awareness that the level of OSS adoption is significant, and the knowledge that there are many organizations which successfully leverage OSS, provides motivation for increased adoption of OSS and further research on issues related to this adoption. However, further exploration of the actual level of adoption was not done. Knowing the exact level of OSS adoption among software-intensive organizations will most likely not contribute to improving their practice.

In **P8** we present a systematic review of the OSS literature published between 1998 and 2008 (C3). This review addresses the second research question (RQ2). We find that the number of publications has been limited but is increasing, and that these publications have focused on a large number of fragmented topics. Much of the research has furthermore had a somewhat introverted, focus has and treated OSS as something distinctly different from software engineering. The OSS research community may increasingly aid practitioners through focusing its research on a few topics rather than diversifying it. These topics should be identified in collaboration with practitioners. In Table 6.1, we have provided an overview of what we believe could be interesting topics for future research.

Furthermore, OSS research should also borrow more support from related areas within software engineering and information systems research (see Table 2.3).

The results presented in Chapter 5 address the third research question (RQ3), by presenting empirical grounded descriptions of actual selection practices. While focusing on the practices developers use, the previous sections also describe several of the resources developers use when identifying, evaluating, and selecting OSS components. Through these descriptions we provide a baseline for understanding and improving both the practices and resources developers use when selecting OSS components.

6.5 Results vs. the Goals of the COSI Project

Due to the participation of a few large multi-national companies, the COSI project initially focused on distributed development, or heterogeneous distributed concurrent collaborations. This focus was also reflected in the project's goals (see Section 3.1.1). The industrial partners from Norway were not in this position, and did not have the same focus as some of the project's larger partners. The project was furthermore divided into three work packages, while we were mainly involved with one them.

For these reasons, this thesis does not aim to fulfill all of the project's goals. This thesis has mainly focused on aiding the project in reaching its third goal (G3). Through G3, COSI intended to "*improve the understanding and cooperation between the open source world and the industry*". This thesis has helped the project in reaching this goal through providing new knowledge about how organizations adopt OSS, and thereby increasing the understanding of how industry may benefit from OSS. However, the research on selection of software components was conducted outside the scope of the COSI project.

6.6 Validity

While, the individual papers contain discussions of validity, this section discusses and summarizes some of the most important issues. Some of these issues are general to all the research conducted in this thesis, while others are mainly concerned with either the (1) adoption of OSS or (2) selection of OSS components. Conclusion validity will not be discussed since this thesis does not include statistical testing.

6.6.1 Construct Validity

In this thesis, we have used several questionnaires (**P1**, **P2**, and **P7**) and interview guides (**P1**, **P2**, **P7**, **P4** and **P5**). These interview guides and questionnaires are available in Appendix B, and they may be reused in other (replication) studies.

The interview guides and questionnaires were first of all inspired by the literature, but also by earlier work within the group e.g. (Conradi and Li, 2005; Li et al., 2008). Hence, the focus and choice of metrics were in line with the literature. Both the interview guides and the questionnaires were developed over several iterations as a collaborative effort by several researchers. They were also pretested internally by colleagues, and most often also by a small number of respondents. The interview guides were furthermore semi-structured and allowed a dialogue between the respondent and the interviewer(s). This was useful for avoiding misunderstandings and for exploring new and interesting topics. This dialogue would not have been possible with questionnaires, and some of the respondents could have interpreted a few of the questions differently.

With an even more careful design of the questionnaires, we could have increased the number of metrics for each of the variables in the questionnaires. This would most likely have forced us to reduce the number of variables included in the questionnaires, but it would have enabled us to statistically test the relationship between some of the variables. However, the questionnaires were mainly used for exploring how and to what extent the various organizations adopted OSS. Testing of relationships between different practices is deferred to future research.

6.6.2 Internal validity

One of the main challenges related to the internal validity of this work is the interpretation we, as researchers, had to do with the data we collected. However, for most of the work we were a team of researchers who worked with the data over time. We furthermore gathered data from different sources, and taped and transcribed almost all of the interviews we conducted. This simplified the analysis of the interviews and allowed other researchers to take part in this work.

Issues related to the Adoption of OSS: Even though the data from the COSI project and from the collaboration with Telenor has been collected over an extended period of time, we were not able to get the same level of data richness as ethnographic or observational case studies. Nevertheless, we are confident that we have presented an accurate picture of how these organizations have adopted OSS. In addition to our own empirical work, we gathered an extensive amount of empirical evidence from the literature and based our development of the classification framework on both.

Issues related to the Selection of Software Components: In **P4** and **P5** we preferred width over depth, and interviewed one person from several organizations rather than several persons from just a few organizations. Interviewing more people from the same organization could have give an even better understanding of the specific organization's practices. However, our goal was to evaluate the practices used in the industry, rather than in just one company.

The interviews were conducted by different researchers, and in the case of **P4** slightly different interview guides. However, these guides and the interview guide used in **P5**, were developed through a collaborative effort over time. The interview guides also helped counter the fact that people have a tendency to talk about their practices "in general". Through the interview guides, we tried to make them focus on a specific project and the selection of specific components. When doing so, we have to be aware that the practices in other projects or for other components may have been different. Therefore, in Study 6, we also asked if the practices from the specific case were representative for how they usually selected components.

6.6.3 External Validity

All of the companies involved in this research were European, with an overweight of Norwegian and Spanish companies. Dybå (2005) points out that there may be organizational differences between European and, for instance, American companies. The majority of Norwegian software-companies are relatively small and most of them have flat organizational structures.

Issues related to the Adoption of OSS: While we were able to draw a randomized sample in **P2**, there may have been differences between respondents and non-respondents. These differences may be an issue as OSS enthusiasts are probably more likely to participate in the survey. However, we did a follow up of non-responses without finding any significant differences.

In addition, the companies involved in the COSI project had a strong interest in OSS. Even though they represented different types of companies, this strong interest in OSS should be kept in mind when attempting to extrapolate the results from studies within the project (**P1**, **P3**, **P6**) to other contexts. Based on collaboration with other organizations, we do not see any reason why the OSS adoption in the COSI project should be different from how these organizations adopted OSS. However, the COSI partners' adoption of OSS may have been more extensive.

OSS is furthermore a multifaceted phenomenon which spans several research fields (see Section 2.1.2). In our systematic literature review (**P8**), we focused on the intersection between OSS, software engineering, and to a minor extent information systems and management literature. We should therefore be careful when generalizing the results from this review to OSS research in other areas.

Issues related to the Selection of Software Components: While we were able to draw a randomized sample in **P2** (Norwegian companies), the samples **P4** (Norwegian companies) and **P5** (Norwegian and Spanish companies) were based on convenience sampling. This included both the companies and the individual respondents included in the surveys. However, we were able to get a varied sample where most kinds of organizations were represented.

Nevertheless, the majority of these companies were small or medium sized, and we have not extensively explored potential differences between organizations of different sizes (**P2**, **P4**, and **P5**). The majority of the projects/products included in **P4** and **P5** were relatively small and focused on business and web applications. Large projects and projects with real-time or life-critical requirements may behave differently, and formalized methods for software selection may see more significant adoption in such contexts.

6.7 The Scope of the Thesis

In retrospect it would have been beneficial to define a more narrow focus for the thesis at an earlier stage. However, the state of the research on OSS adoption back in 2005, did not permit this (see Section 2.2.3). Instead, the thesis contributes to building a foundation for further research on OSS adoption (C1-C3). Moreover, when this foundation was established we decided to focus on the selection of OSS components (C4-C5).

In (**P8**), we criticize OSS research for not borrowing enough support from related literature. The scope of this thesis has been limited to OSS and software engineering (see Section 2.4.3). We acknowledge that the part on adoption of OSS could have benefited from having a wider scope, and from borrowing more support from research on diffusion and adoption of (information) technology (Fichman, 1992; Rogers, 2003). However, after establishing the platform for future research (described above), we decided to build on this platform and focus our work on a topic closer to the core of software engineering, software selection.

We also acknowledge that the part on software selection could also have benefited from having a wider scope, and from borrowing more support from (in particular) theories for decision making such as social aspects (Munda, 2004) and uncertainty (Begg et al., 2003). Like the previous part of this thesis, this part may also have benefited from research on diffusion and adoption of (information) technology. So, all these directions must be explored through future work.

When studying software selection, we have mainly looked at the identification and evaluation of components. There are indeed other phases of CBSE that are highly relevant to software selection. However, these had to be left out of this work. Future work may look closer at the actual decision being taken after the evaluation is completed, requirements engineering and requirements negotiation with respect to the functionality offered by the available components, and the actual integration and maintenance of the selected components.

Chapter 7

Conclusions and Future Work

This thesis has presented an empirical view on the adoption of OSS in software-intensive organizations. The overall goal of the thesis has been to increase the understanding of how software-intensive organizations may adopt OSS. Through several industrial surveys and studies conducted in collaboration with industrial partners, we have explored and described several aspects of OSS adoption. We have in particular contributed to building a theory around what it means to adopt OSS, and how software-intensive organizations actually select OSS components.

7.1 Conclusions

The research presented in this thesis has focused on two topics. First, it has tried to sort out some of the unclarity and misconceptions around what it means for an organization to *adopt OSS*. Second, it has attempted to aid researchers and practitioners in reducing some of the risk and complexity related to the selection of OSS components. From these two topics three research questions (RQ1-RQ3) were defined. Table 7.1 summarizes how this thesis has responded to these research questions, and it relates the contributions (C1-C5) and the papers (P1-P8) tied to these research questions.

7.1.1 Adoption of OSS

Based on several industrial studies, we have provided an empirical description of how a number of organizations adopted OSS in distinctly different ways (C1). Each of these ways offered unique benefits and involved different challenges. Organizations should acknowledge this variety and should not blindly follow the success of others. OSS is not a one-size-fits-all solution. Instead, organizations must analyze their opportunities and shape their adoption of OSS to their own organization.

We have also conducted an extensive review of the literature on OSS adoption (C2). The literature review showed that existing empirical research on OSS adoption has been relatively limited and fragmented. However, our work, together with more recent work by other researchers has contributed to filling this gap and building a solid foundation for future work.

Table 7.1: Relation between research questions, contributions, and papers

Research Question	Response	Reported in
RQ1 How and to what extent are software-intensive organizations currently adopting OSS?	Organizations adopt OSS in significantly different ways. This thesis offers a classification framework describing six such ways (C3). This framework is based on:	P1, P3, and P8
	- Descriptions of how a number of organizations actually adopt OSS (C1).	P1-P7
	- A systematic literature review (C2).	P8
	The adoption of OSS in software-intensive organizations is considerable.	P2
RQ2 What is the current status of research on OSS adoption in organizations and how may this research benefit practitioners?	The status of the OSS adoption research is given in our literature review (C2). The amount of empirical evidence has until recently been limited and the literature is relatively fragmented and troubled with a lack of quality studies. To increase the value of research for practitioners, we offer directions for future research.	P8 and Section 6.2.1
RQ3 Which strategies and resources do software developers use to identify, evaluate, and select OSS components?	Practitioners most often use informal selection practices, heavily depending on their previous experience. We provide empirically founded descriptions of these practices (C4).	P4, P5
	Moreover, we put these practices in to context by describing a model for situated software selection (C5).	Section 5.2

The differences between the ways of adopting OSS should be reflected by researchers to a far greater extent than they are today. To this end, we have provided a classification framework for organizational adoption of OSS (C3). Our framework gives researchers and practitioners a more precise vocabulary when discussing OSS adoption. Practitioners may also use it to explore their own adoption of OSS. The framework identifies that organizations adopt OSS in the following six ways:

- Deploying OSS products
- Using OSS CASE tools
- Integrating OSS components into their systems
- Participating in the development of OSS products controlled by someone else
- Providing their own OSS products and relating to their surrounding communities
- Using OSS development practices in their own software development

7.1.2 Selection of OSS Components

Based on the platform described above, we have drawn our attention towards integration of OSS components, and in particular the selection of such components. Research on software selection has suggested a continuous stream of formalized methods and evaluation schemes, without any particular influence on practice. The first step towards improving practice, is to describe the practices currently in use.

Our empirical findings (C4) describe how software developers identify, evaluate, and finally decide upon OSS components. We have found that practice is heavily influenced by previous experience (primarily the developer's or his colleagues', but also by experience found across the Internet), informal monitoring of Internet sites and communities, unstructured Internet searches, and prototyping. Formalized methods and evaluation schemes see very limited adoption.

Based on these results, we have provided a model for situated selection (C5). This model offers a new dimension to software selection and to the traditional view that selection is either ad hoc or formalized. The model presents some of the complexity and constraints involved in the selection of OSS components. Moreover, it illustrates how important the developer and his work context are to the selection process. The process cannot be separated from the context and the developer(s) executing it.

We have furthermore used the model to present an understanding of why formalized methods for software selection have failed to influence practice. This is because selection is a process where the understanding of both the requirements and the candidate components is evolving, as the developer explores and evaluates his alternatives. Moreover, a few context specific criteria or constraints are often far more important than the many evaluation criteria suggested by research. Finally, practitioners rely on rich (textual) experience and prototyping rather than quantifiable metrics. To rely on experience partly from inside the team and partly from various Internet sources, saves time and reduces uncertainty.

7.2 Future Work

Through the research presented above, we have created a basis for future research on both OSS adoption and software selection. We recommend that OSS researchers (1) focus their attention on a limited number of issues, (2) identify these issues together with practitioners, and (3) align their work with research on software engineering and information systems. There are several ways of extending the work on OSS adoption, and we have suggested several possibilities in Section 6.2.1. We hope that these topics may serve as input to the ongoing discussions on OSS adoption.

However, we decided to focus our work on integration of OSS components, and in particular selection of such components. Based on this work, we recommend that research on software selection should benefit from putting more emphasis on actual practice and less on suggesting new methods and evaluation schemes. Furthermore, we have provided some considerations in Section 6.2.2 on general research on software selection. In addition to these general issues, there are a few issues which may continue and extend the work in this thesis in particular.

We have mainly conducted a single interview with one, or in some cases two, developers from each organization. Future studies of the practices developers actually use when selecting OSS components could benefit from increased depth and closer collaboration with the developers. Moreover, our samples in these studies had an overweight of projects developing web and business applications. They did not contain projects that developed critical applications, and the practices used in such projects may be more formalized. Future research may also look into this.

The most interesting issue which may add the most value to practitioners, is perhaps the use of (simple) search tools for aggregating rich experience reports for software components or a combination of such components. This is not something we have explored, but we see this as a very interesting topic, perhaps under the umbrella by Gallardo-Valencia and Sim (2009), named "Internet-Scale Code Search".

In addition, we believe that systems consisting of combinations of components from several (OSS) providers, upon which the integrator has no direct control, could be an important challenge for the future (see also Boehm (2006b)). This is, component providers, integrators, customers, and users that constitute large and evolving networks of distributed and interrelated stakeholders. Selecting the right components, integrating them, maintaining them, and influencing their future in such a network will be a tough challenge which needs scrutiny. Finally, in such networks, how do integrators solve the mismatch between the offering in the marketplace and the customer's requirements? Are customers involved in requirements renegotiation based on this mismatch, does the integrator modify the component or develop addware, or does he reject all available components and start from scratch?

Glossary

Adoption of OSS	With adoption of OSS we consider software-intensive organizations at any of the five stages of the adoption process (Rogers, 2003). This includes both organizations that plan (knowledge, persuasion, and decision) to adopt OSS and organizations that have already included OSS as part of their software development (implementation and confirmation).
Component-Based Software Engineering	The process of defining, implementing and integrating, or composing loosely coupled independent components into systems (Sommerville, 2007, p. 440).
Copyleft	Copyleft or reciprocity requires that derivative works that are based on a piece of software with certain software licenses (e.g. GPL) use the same license (Rosen, 2005).
Formalized method	A formally-defined, brand-named or published methodology, of which there are many examples in the literature (Fitzgerald, 1996, p. 4).
Inner Source Software	Inner Source Software is the use of OSS development tools and development practices within an organization or a consortium of organizations (van der Linden, 2006).
NACE	Nomenclature statistique des activités économiques dans la Communauté européenne (French for Statistical Classification of Economic Activities in the European Community). The NACE codes are used to classify economic activity (companies) according to their main business (NACE, 2009).

Open source software	Open source software (OSS) is a multifaceted phenomenon consisting of a wide spectrum of software products provided by heterogeneous communities using a variety of software development and maintenance practices.
Open Source Software product	A piece of software released with a software license approved by either the Open Source Initiative (OSI) or the Free Software Foundation (FSF).
OSS community	An OSS community consists of the users and developers surrounding an OSS product. These community members are often, based on their level of involvement, placed into the following groups: core-developers, co-developers, active users, and passive users (Crowston and Howison, 2006).
Situated	That an action is situated means that it depends heavily on its circumstances (Suchman, 1987, p. 50).
Software	Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system (IEEE 1990).
Software component	A unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties (Szyperski et al., 2002, p. 41).
Software engineering	The practical application of scientific knowledge in the design and construction of computer programs, and the associated documentation required to develop, operate, and maintain them (Boehm, 1976, p. 1226).
Software engineering research	Software engineering research concerns (1) the development of new, or modification of existing, technologies (process models, methods, techniques, tools or languages) to support SE activities, and (2) the evaluation and comparison of the effect of using such technology in the often very complex interaction of individuals, teams, projects and organisations, and various types of task and software system (Sjøberg et al., 2007, p. 358) [sic].

Software-intensive organization	An organization is here defined as any public or private institution, company, or similar entity that develops, maintains, or makes heavy use of software.
Software reuse	The use of existing software or software knowledge to construct new software (Frakes and Kang, 2005, p. 529).
System	An entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena. These other systems are the environment of the given system. The system boundary is the common frontier between the system and its environment (IEEE, 1990).

GLOSSARY

References

Ivan Aaen, Jasper Arent, Lars Mathiassen, and Ojelanki Ngwenyama. A Conceptual Map of Software Process Improvement. *Scandinavian Journal of Information Systems*, 13: 123–146, 2001. ISSN 0905-0167.

Alain Abran, James W. Moore, Pierre Bourque, and Robert Dupuis, editors. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2004. ISBN 0-7695-2330-7.

Paul Adams, Cornelia Boldyreff, David Nutter, and Stephen Rank. Adaptive Reuse of Libre Software Systems for Supporting On-line Collaboration. In Feller et al. (2005a), pages 1–4. ISBN 1-59593-127-9. doi: 10.1145/1082983.1083259.

Paul Adams, David Nutter, Stephen Rank, and Cornelia Boldyreff. Using Open Source Tools to Support Collaboration within CALIBRE. In Scotto and Succi (2005), pages 61–65.

Pär. J. Ågerfalk and Brian Fitzgerald. Outsourcing to an Unknown Workforce: Exploring Opensourcing As a Global Sourcing Strategy. *MIS Quarterly*, 32(2):385–409, June 2008. ISSN 02767783.

Samuel A. Ajila and Di Wu. Empirical study of the effects of open source adoption on software development economics. *Journal of Systems and Software*, 80(9):1517–1529, 2007. ISSN 0164-1212. doi: 10.1016/j.jss.2007.01.011.

Orlando Alfonzo, Kenyer Domínguez, Lornel Rivas, Maria Pérez, Luis Mendoza, and Maryoly Ortega. Quality Measurement Model for Analysis and Design Tools Based on FLOSS. In Farookh Khadeer Hussain and Elizabeth Chang, editors, *Proceedings of the 19th Australian Conference on Software Engineering (ASWEC'08), March 26th-28th, Perth, Australia*, pages 258–268. IEEE Computer Society, 2008. ISBN 978-0-7695-3100-7. doi: 10.1109/ASWEC.2008.4483214.

Carina Alves and Anthony Finkelstein. Investigating Conflicts in COTS Decision-Making. *International Journal of Software Engineering and Knowledge Engineering*, 13(5):1–21, 2003.

Claudio A. Ardagna, Ernesto Damiani, and Fulvio Frati. FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments. In Feller et al. (2007), pages 3–16. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_1.

Claudia P. Ayala. *Systematic Construction of Goal-Oriented COTS Taxonomies*. PhD thesis, Technical University of Catalunya (UPC), 2008.

Claudia P. Ayala, Carl-Fredrik Sørensen, Reidar Conradi, Xavier Franch, and Jingyue Li. Open Source Collaboration for Fostering Off-The-Shelf Components Selection. In Feller et al. (2007), pages 17–30. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_2.

Claudia P. Ayala, Øyvind Hauge, Reidar Conradi, Xavier Franch, Jingyue Li, and Ketil Sandanger Velle. Challenges of the Open Source Component Marketplace in the Industry. In Boldyreff et al. (2009), pages 213–224. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_19.

Earl R. Babbie. *Survey Research Methods*. Wadsworth Publishing, 2nd edition, 1990. ISBN 978-0534126728.

Stacy Avery Baird. The Heterogeneous World of Proprietary and Open-Source Software. In Tomasz Janowski and Theresa A. Pardo, editors, *Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance (ICEGOV '08), December 1st-4th, Cairo, Egypt*, volume 351 of *ACM International Conference Proceeding Series*, pages 232–238. ACM, 2008. ISBN 978-1-60558-386-0. doi: 10.1145/1509096.1509143.

Victor R. Basili. The Role of Experimentation in Software Engineering: Past, Current, and Future. In Tom Maibaum, Dieter Rombach, and Marvin V. Zelkowitz, editors, *Proceedings of the 18th International Conference on Software Engineering (ICSE '96), March 25th-29th, Berlin, Germany*, pages 442–449. IEEE Computer Society, 1996. ISBN 0-8186-7246-3. doi: 10.1109/ICSE.1996.10002.

Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, 12(7):733–743, July 1986.

Steve H. Begg, Reidar B. Bratvold, and John M. Campbell. Decision-Making under Uncertainty. In *Proceedings of the 7th International Symposium on Reservoir Simulation Symposium on Reservoir Simulation, June 23rd-27th, Baden-Baden, Germany*, 2003.

Manuel F. Bertoa, José M. Troya, and Antonio Vallecillo. A Survey on the Quality Information Provided by Software Component Vendors. In Fernando Brito e Abreu, Mario Piattini, Geert Poels, and Houari A. Sahraoui, editors, *Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'03), July 21st-25th, Darmstad, Germany*, pages 25–30, 2003.

Jesal Bhuta and Barry W. Boehm. A Method for Compatible COTS Component Selection. In Franch and Port (2005). ISBN 3-540-24548-0. doi: 10.1007/b105900.

Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane. Postmortem: Never Leave a Project without It. *IEEE Software*, 19(3):43–45, 2002. ISSN 0740-7459. doi: 10.1109/ms.2002.1003452.

Dominik Birkmeier and Sven Overhage. On Component Identification Approaches - Classification, State of the Art, and Comparison. In Grace A. Lewis, Iman Poernomo, and Christine Hofmeister, editors, *Proceedings of the 12th International Symposium on Component-Based Software Engineering (CBSE 2009), June 24th-26th, East Stroudsburg, USA*, volume 5582/2009 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009. ISBN 978-3-642-02413-9. doi: 10.1007/978-3-642-02414-6_1.

Wolf-Gideon Bleek, Matthias Finck, and Bernd Pape. Towards an Open Source Development Process? Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model. In Scotto and Succi (2005), pages 37–43.

Barry W. Boehm. Software Engineering. *IEEE Transaction on Computers*, C-25(12): 1226–1241, 1976.

Barry W. Boehm. A View of 20th and 21st Century Software Engineering. In Osterweil et al. (2006), pages 12–29. ISBN 1-59593-375-1. doi: 10.1145/1134285.1134288.

Barry W. Boehm. Some future trends and implications for systems and software engineering processes. *Systems Engineering*, 9(1):1–19, 2006b. ISSN 1098-1241. doi: 10.1002/sys.v9:1.

Barry W. Boehm and C. Abts. COTS integration: Plug and Pray? *Computer*, 32(1): 135–138, Jan 1999. ISSN 0018-9162. doi: 10.1109/2.738311.

Cornelia Boldyreff, David Nutter, and Stephen Rank. Communication and Conflict Issues in Coollaborative Software Research Projects. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, *Collaboration, Conflict and Control Proceedings of the 4th Workshop on Open Source Software Engineering (WOSSE 2004), May 25th, Edinburgh, Scotland*, pages 14–17, 2004.

Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors. *Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3rd-6th, Skövde, Sweden*, volume 299/2009 of *IFIP Advances in Information and Communication Technology*, 2009. Springer. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2.

Andrea Bonaccorsi and Christina Rossi. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology, and Policy*, 18(4):40–64, dec 2006. doi: 10.1007/s12130-006-1003-9.

Andrea Bonaccorsi, Silvia Giannangeli, and Christina Rossi. Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management Science*, 52(7):1085–1098, 2006. ISSN 0025-1909. doi: 10.1287/mnsc.1060.0547.

Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Christina Rossi. Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research. In Andrea Capiluppi and Gregorio Robles, editors, *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007), May 21th, Minneapolis, USA*, pages 1–5, Minneapolis, US, 2007. IEEE Computer Society. ISBN 0-7695-2961-5. doi: 10.1109/floss.2007.3.

Geoffrey C. Bowker and Susan Leigh Star. *Sorting Things Out: Classification and Its Consequences*. The MIT Press, 2000. ISBN 978-0262522953.

Pearl Brereton and David Budgen. Component-Based Systems: a Classification of Issues. *Computer*, 33(11):54–62, Nov 2000. ISSN 0018-9162. doi: 10.1109/2.881695.

Alan W. Brown and Grady Booch. Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors. In Cristina Gacek, editor, *Software Reuse: Methods, Techniques, and Tools Proceedings of the 7th International Conference Software Reuse: Methods, Techniques, and Tools (ICSR-7), April 15-19, Austin, USA*, volume 2319/2002 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2002. ISBN 978-3-540-43483-2. doi: 10.1007/3-540-46020-9_9.

Alan W. Brown and Kurt C. Wallnau. The current state of CBSE. *IEEE Software*, 15(5): 37–46, Sep/Oct 1998. ISSN 0740-7459. doi: 10.1109/52.714622.

David Budgen and Pearl Brereton. Performing Systematic Literature Reviews in Software Engineering. In Osterweil et al. (2006), pages 1051–1052. ISBN 1-59593-375-1. doi: 10.1145/1134285.1134500.

Michele Cabano, Cesare Monti, and Giulio Piancastelli. Context-Dependent Evaluation Methodology for Open Source Software. In Feller et al. (2007), pages 301–306. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_32.

Martin Campbell-Kelly and Daniel D. Garcia-Swartz. Pragmatism, not ideology: Historical perspectives on IBM's adoption of open-source software. *Information Economics and Policy*, 21(3):229 – 244, 2009. ISSN 0167-6245. doi: 10.1016/j.infoecopol.2009.03.006.

Andrea Capiluppi, Patricia Lago, and Maurizio Morisio. Characteristics of Open Source Projects. In Gerardo Canfora, Mark van den Brand, and Tibor Gyimóthy, editors, *Proceedings of the Seventh European Conference on Software Maintenance and Reengineering (CSMR '03), March 26th-28th, Benevento, Italy*, pages 317–327. IEEE Computer Society, 2003a. ISBN 0-7695-1902-4. doi: 10.1109/CSMR.2003.1192440.

Andrea Capiluppi, Patricia Lago, and Maurizio Morisio. Evidences in the evolution of os projects through changelog analyses. In Feller et al. (2003).

Weibing Chen, Jingyue Li, Jianqiang Ma, Reidar Conradi, Junzhong Ji, and Chunnian Liu. An Empirical Study on Software Development with Open Source Components in the Chinese Software Industry. *Software Process: Improvement and Practice*, 13(1): 89–100, 2008. ISSN 1077-4866. doi: 10.1002/spip.v13:1.

Henry W. Chesbrough. The era of open innovation. *MIT Sloan Management Review*, 44 (3):35, 2003. ISSN 15329194.

Reidar Conradi and Jingyue Li. Observations on Versioning of Off-The-Shelf Components in Industrial Projects. In Jim Whitehead, editor, *Proceedings of the 12th International Workshop on Software Configuration Management (SCM '05), September 5th-6th, Lisbon, Portugal*, pages 33–42. ACM, 2005. ISBN 1-59593-310-7. doi: 10.1145/1109128.1109131.

Ivica Crnkovic. Component-based Software Engineering - New Challenges in Software Development. *Software Focus*, 2(4):127–133, 2001. doi: 10.1002/swf.45.

Kevin Crowston and James Howison. Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology, and Policy*, 18(4):65–85, dec 2006. doi: 10.1007/s12130-006-1004-8.

Kevin Crowston, Qing Li, Kangning Wei, U. Yeliz Eseryel, and James Howison. Self-organization of teams for free/libre open source software development. *Information and Software Technology*, 49(6):564 – 575, 2007. ISSN 0950-5849. doi: 10.1016/j.infsof.2007.02.004. Qualitative Software Engineering Research.

David Cruz, Thomas Wieland, and Alexander Ziegler. Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis. *Software Process: Improvement and Practice*, 11(2):107–122, 2006. doi: 10.1002/spip.257.

Daclin. ITEA Report on Open Source Software. Technical report, Information Technology for European Advancement (ITEA), January 2004. URL `http://www.itea2.org/itea_report_on_oss`.

Linus Dahlander and Mats G. Magnusson. Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms. *Research Policy*, 34 (4):481–493, 2005. doi: 10.1016/j.respol.2005.02.003.

Linus Dahlander and Mats G. Magnusson. How do Firms Make Use of Open Source Communities? *Long Range Planning*, 41(6):629 – 649, 2008. ISSN 0024-6301. doi: 10.1016/j.lrp.2008.09.003.

Linus Dahlander, Lars Frederiksen, and Francesco Rullani. Online Communities and Open Innovation: Governance and Symbolic Value Creation . *Industry & Innovation*, 15(2):115–123, April 2008. doi: 10.1080/13662710801970076.

Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, and Marco Scotto, editors. *Proceedings of the 2nd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2006) - Open Source Systems, June 8th-10th, Como, Italy*, volume 203/2006 of *IFIP Advances in Information and Communication Technology*, 2006. Springer. ISBN 978-0-387-34225-2. doi: 10.1007/0-387-34226-5.

Paul B. de Laat. Copyright or copyleft?: An analysis of property regimes for software development. *Research Policy*, 34(10):1511 – 1532, 2005. ISSN 0048-7333. doi: 10.1016/j.respol.2005.07.003.

Vieri del Bianco, Luigi Lavazza, Sandro Morasca, and Davide Taibi. Quality of Open Source Software: The QualiPSo Trustworthiness Model. In Boldyreff et al. (2009), pages 199–212. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_18.

Jean-Christophe Deprez and Simon Alexandre. Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS. In Andreas Jedlitschka and Outi Salo, editors, *Product-Focused Software Process Improvement Proceedings of the 9th International Conference on Product-Focused Software Process Improvement (PROFES 2008), June 23rd-25th, Monte Porzio Catone, Italy*, volume 5089/2008 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 2008. ISBN 978-3-540-69564-6. doi: 10.1007/978-3-540-69566-0_17.

Mariella Di Giacomo. MySQL: Lessons Learned on a Digital Library. *IEEE Software*, 22(3):10–13, 2005a. ISSN 0740-7459. doi: 10.1109/ms.2005.71.

Piergiorgio Di Giacomo. COTS and Open Source Software Components: Are They Really Different on the Battlefield? In Franch and Port (2005), pages 301–310. ISBN 3-540-24548-0. doi: 10.1007/b105900.

Chris DiBona, Sam Ockman, and Mark Stone, editors. *Open Sources: Voices from the Open Source Revolution*. O'Reilly, 1999. ISBN 1-56592-582-3.

Edsger W. Dijkstra. The Humble Programmer. *Communications of the ACM*, 15(10): 859–866, 1972. ISSN 0001-0782. doi: 10.1145/355604.361591.

Trung T. Dinh-Trong and James M. Bieman. The FreeBSD Project: A Replication Case Study of Open Source Development. *IEEE Transactions on Software Engineering*, 31 (6):481–494, June 2005. ISSN 0098-5589. doi: 10.1109/tse.2005.73.

Leonhard Dobusch. Migration Discourse Structures: Escaping Microsoft's Desktop Path. In Russo et al. (2008), pages 223–235. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_18.

Tore Dybå. An Empirical Investigation of the Key Factors for Success in Software Process Improvement. *IEEE Transactions on Software Engineering*, 31(5):410–424, May 2005. ISSN 0098-5589. doi: 10.1109/tse.2005.53.

Tore Dybå, Barbara A. Kitchenham, and Magne Jørgensen. Evidence-based software engineering for practitioners. *Software, IEEE*, 22(1):58–65, Jan.-Feb. 2005. ISSN 0740-7459. doi: 10.1109/ms.2005.6.

Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting Empirical Methods for Software Engineering Research. In Shull et al. (2008), pages 285–311. ISBN 978-1-84800-043-8. doi: 10.1007/978-1-84800-044-5_11.

Tor Erik Eide. Study of the Release Process of Open Source Software - Case Study. Master's thesis, Norwegian University of Science and Technology NTNU, 2007.

Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison Wesley, 2002. ISBN 0-201-73496-6. ISBN :0-201-73496-6.

Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors. *Taking Stock of the Bazaar: 3rd Workshop on Open Source Software Engineering (WOSSE 2003), May 3rd, Portland, USA*, 2003.

Joseph Feller, Brian Fitzgerald, Scott A. Hissam, Karim R. Lakhani, and Walt Scacchi, editors. *Open Source Application Spaces: Proceedings of the Fifth Workshop on Open Source Software Engineering (WOSSE 2005), May 17th, St. Louis, USA*, 2005a. ACM. ISBN 1-59593-127-9.

Joseph Feller, Brian Fitzgerald, Karim R. Lakhani, and Scott A. Hissam, editors. *Perspectives on Free and Open Source Software*. The MIT Press, Cambridge, Massachusetts, 2005b. ISBN 0-262-06246-1.

Joseph Feller, Patrick Finnegan, David Kelly, and Maurice MacNamara. Developing Open Source Software: A Community-Based Analysis of Research. In Eileen M. Trauth, Debra Howcroft, Tom Butler, Brian Fitzgerald, and Janice I. DeGross, editors, *Social Inclusion: Societal and Organizational Implications for Information Systems FIP TC8 WG 8.2 International Working Conference, July 12th-15th, Limerick, Ireland*, volume 208 of *IFIP International Federation for Information Processing*, pages 261–278. Springer, 2006. ISBN 978-0-387-34587-1. doi: 10.1007/0-387-34588-4_18.

Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors. *Proceedings of the 3rd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2007) - Open Source Development, Adoption and Innovation, June 11th-14th, Limerick, Ireland*, volume 234/2007 of *IFIP Advances in Information and Communication Technology*, 2007. Springer. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7.

Norman Fenton. How Effective Are Software Engineering Methods? *Journal of Systems and Software*, 22(2):141–146, 1993. ISSN 0164-1212. doi: 10.1016/0164-1212(93) 90092-c.

Robert G. Fichman. Information Technology Diffusion: A Review of Empirical Research. In Janice I. DeGross, Jack D. Becker, and Joyce J. Elam, editors, *Proceedings of the Thirteenth International Conference on Information Systems (ICIS '92), December 13th-16th, Dallas, USA*, pages 195–206, Minneapolis, MN, USA, 1992. University of Minnesota.

Arlene G. Fink. *The Survey Handbook*. Sage Publications, 2nd edition, 2002. ISBN 978-0761925804.

Brian Fitzgerald. Formalized systems development methodologies: A critical perspective. *Information Systems Journal*, 6(1):3–23, January 1996. ISSN 1350-1917.

Brian Fitzgerald. An empirical investigation into the adoption of systems development methodologies. *Information & Management*, 34(6):317–328, 1998. ISSN 0378-7206. doi: 10.1016/s0378-7206(98)00072-x.

Brian Fitzgerald. Has Open Source Software a Future? In Feller et al. (2005b), pages 93–106. ISBN 0-262-06246-1.

Brian Fitzgerald. The Transformation of Open Source Software. *MIS Quarterly*, 30(3): 587–598, 2006.

Brian Fitzgerald. Open Source Software Adoption: Anatomy of Success and Failure. *International Journal of Open Source Software & Processes*, 1(1):1–23, 2009. ISSN 1942-3926.

Brian Fitzgerald and Joseph Feller. Guest Editorial Open source software: investigating the software engineering, psychosocial and economic issues. *Information Systems Journal*, 11(4):273–276, 2001. doi: 10.1111/j.1365-2575.2001.00109.x.

William B. Frakes and Kyo Kang. Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering*, 31(7):529–536, 2005. ISSN 0098-5589. doi: 10.1109/tse.2005.85.

Xavier Franch and Daniel N. Port, editors. *Proceedings of the 4th International Conference on Component-Based Software Systems (ICCBSS 2005), February 7th-11th, Bilbao, Spain*, volume 3412/2005 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-24548-0. doi: 10.1007/b105900.

Alfonso Fuggetta. A Classification of CASE Technology. *Computer*, 26(12):25–38, 1993. ISSN 0018-9162. doi: 10.1109/2.247645.

Alfonso Fuggetta. Open source software–an evaluation. *Journal of Systems and Software*, 66(1):77 – 90, 2003. ISSN 0164-1212. doi: 10.1016/s0164-1212(02)00065-1.

Cristina Gacek and Budi Arief. The Many Meanings of Open Source. *IEEE Software*, 21 (1):34–40, 2004. ISSN 0740-7459. doi: 10.1109/ms.2004.1259206.

Rosalva E. Gallardo-Valencia and Susan Elliott Sim. Internet-Scale Code Search. In Sushil Bajracharya, Adrian Kuhn, and Yunwen Ye, editors, *Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation (SUITE '09), May 16th, Vancouver, Canada*, pages 49–52. IEEE Computer Society, 2009. ISBN 978-1-4244-3740-5. doi: 10.1109/suite.2009.5070022.

Marinela Gerea. Selection of Open Source Components - A Qualitative Survey in Norwegian IT Industry. Master's thesis, Norwegian University of Science and Technology NTNU, 2007.

Rishab Aiyer Ghosh. Free libre and open source software: Survey and study. Technical report, International Institute of Infonomics, University of Maastricht, 2002. URL `http://www.infonomics.nl/FLOSS/report/`.

Rishab Aiyer Ghosh. Study on the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communication Technologies (ICT) Sector in the EU. Technical report, UNU-MERIT, 2006. URL `http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf`.

Robert L. Glass. The Software-Research Crisis. *IEEE Software*, 11(6):42–47, 1994. ISSN 0740-7459. doi: 10.1109/52.329400.

Robert L. Glass. Matching Methodology to Problem Domain. *Communications of the ACM*, 47(5):19–21, 2004. ISSN 0001-0782. doi: 10.1145/986213.986228.

Robert L. Glass, Iris Vessey, and Venkataraman Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8):491–506, 2002. ISSN 0950-5849. doi: 10.1016/s0950-5849(02)00049-6.

Eugene Glynn, Brian Fitzgerald, and Chris Exton. Commercial Adoption of Open Source Software: An Empirical Study. In June Verner and Guilherme Horta Travassos, editors, *Proceedings of International Symposium on Empirical Software Engineering (ISESE 2005), November 17th-18th, Noosa Heads, Australia*, pages 225–234. IEEE Computer Society, 2005. doi: 10.1109/ISESE.2005.1541831.

Dion Hoe-Lian Goh, Alton Chua, Davina Anqi Khoo, Emily Boon-Hui Khoo, Eric Bok-Tong Mak, and Maple Wen-Min Ng. A checklist for evaluating open source digital library software. *Online Information Review*, 30(4):360, 2006. ISSN 14684527.

Dion Hoe-Lian Goh, Alton Chua, See-Yong Yee, Kia-Ngoh Poh, and How-Yeu Ng. Evaluating open source portals. *Journal of Librarianship and Information Science*, 40(2): 81–92, 2008. doi: 10.1177/0961000608089344.

Bernard Golden. *Succeeding with Open Source*. Addison-Wesley Professional, 2004. ISBN 978-0321268532.

Sigi Goode. Something for nothing: management rejection of open source software in Australia's top firms. *Information & Management*, 42(5):669–681, 2005. ISSN 0378-7206. doi: 10.1016/j.im.2004.01.011.

Ian Gorton, Anna Liu, and Paul Brebner. Rigorous Evaluation of COTS Middleware Technology. *Computer*, 36(3):50–55, 2003. ISSN 0018-9162. doi: 10.1109/mc.2003.1185217.

Burton Grad. A Personal Recollection: IBM's Unbundling of Software and Services. *IEEE Annals of the History of Computing*, 24(1):64–71, Jan-Mar 2002. ISSN 1058-6180. doi: 10.1109/85.988583.

Simon Grand, Georg von Krogh, Dorothy Leonard, and Walter Swap. Resource allocation beyond firm boundaries: A multi-level model for Open Source innovation. *Long Range Planning*, 37(6):591–610, December 2004. doi: 10.1016/j.lrp.2004.09.006.

Jim Hamerly, Tom Paquin, and Susan Walton. Freeing the Source: The Story of Mozilla. In DiBona et al. (1999), pages 197–206. ISBN 1-56592-582-3.

Bo Hansen, Jeremy Rose, and Gitte Tjørnehøj. Prescription, description, reflection: the shape of the software process improvement field. *International Journal of Information Management*, 24(6):457 – 472, 2004. ISSN 0268-4012. doi: 10.1016/j.ijinfomgt.2004.08.007.

Øyvind Hauge and Sven Ziemer. Providing Commercial Open Source Software: Lessons Learned. In Boldyreff et al. (2009), pages 70–82. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_8.

Øyvind Hauge, Claudia P. Ayala, and Reidar Conradi. Open Source Software in Organizations - A Systematic Literature Review. *TO APPEAR*.

Øyvind Hauge, Carl-Fredrik Sørensen, and Andreas Røsdal. Surveying Industrial Roles in Open Source Software Development. In Feller et al. (2007), pages 259–264. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_25.

Øyvind Hauge, Carl-Fredrik Sørensen, and Reidar Conradi. Adoption of Open Source in the Software Industry. In Russo et al. (2008), pages 211–222. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_17.

Øyvind Hauge, Thomas Østerlie, Carl-Fredrik Sørensen, and Marinela Gerea. An Empirical Study on Selection of Open Source Software - Preliminary Results. In Andrea Capiluppi and Gregorio Robles, editors, *Proceedings of the ICSE 2009 Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS 2009), May 18th, Vancouver, Canada*, pages 42–47. IEEE Computer Society, 2009. ISBN 978-1-4244-3720-7. doi: 10.1109/FLOSS.2009.5071359.

Øyvind Hauge, Daniela S. Cruzes, Reidar Conradi, Ketil Sandanger Velle, and Tron André Skarpenes. Risks and Risk Mitigation in Open Source Software Adoption: Bridging the Gap between Literature and Practice. In *Proceedings of the 6th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2010), May30th-June 2nd, Notre Dame, USA*, IFIP International Federation for Information Processing, 2010.

Frank Hecker. Setting Up Shop: The Business of Open-Source Software. *IEEE Software*, 16(1):45–51, 1999. ISSN 0740-7459. doi: 10.1109/52.744568.

Joachim Henkel. Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, 35(7):953 – 969, 2006. ISSN 0048-7333. doi: 10.1016/j.respol.2006.04.010.

Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159 – 1177, 2003. ISSN 0048-7333. doi: 10.1016/s0048-7333(03)00047-7. Open Source Software Development.

Andreas Höfer and Walter Tichy. Status of Empirical Research in Software Engineering. In Victor R. Basili, Dieter Rombach, Kurt Schneider, Barbara A. Kitchenham, Dietmar Pfahl, and Richard W. Selby, editors, *Proceedings of the International Workshop on Empirical Software Engineering Issues. Critical Assessment and Future Directions, June 26th-30th, Dagstuhl Castle, Germany*, volume 4336/2007 of *Lecture Notes in Computer Science*, pages 10–19. Springer, 2007. ISBN 978-3-540-71300-5. doi: 10.1007/978-3-540-71301-2_3.

Oliver Hummel, Werner Janjic, and Colin Atkinson. Code conjurer: Pulling reusable software out of thin air. *IEEE Software*, 25(5):45–52, 2008. ISSN 0740-7459. doi: 10.1109/ms.2008.110.

IEEE. IEEE Standard 610.12-1990: Standard for Glossary of Software Computer Engineering Terminology. Technical report, IEEE, 1990.

Ari Jaaksi. Experiences on Product Development with Open Source Software. In Feller et al. (2007), pages 85–96. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_7.

Juha Järvensivu and Tommi Mikkonen. Forging A Community - Not: Experiences On Establishing An Open Source Project. In Russo et al. (2008), pages 15–27. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_2.

Björn Johansson and Frantisek Sudzina. Choosing Open Source ERP Systems: What Reasons Are There For Doing So? In Boldyreff et al. (2009), pages 143–155. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_14.

Magne Jørgensen and Kjetil Moløkken-Østvold. How Large Are Software Cost Overruns? Critical Comments on the Standish Group's CHAOS Reports. *Information and Software Technology*, 48(4):297–301, April 2006. doi: 10.1016/j.infsof.2005.07.002.

Elena Karahanna, Detmar W. Straub, and Norman L. Chervany. Information technology adoption across time: A cross-sectional comparison of pre-adoption and post-adoption beliefs. *MIS Quarterly*, 23(2):183–213, 1999. ISSN 02767783.

Even-André Karlsson, editor. *Software Reuse: a Holistic Approach*. John Wiley & Sons, Inc., New York, NY, USA, 1995. ISBN 0-471-95819-0.

Barbara A. Kitchenham. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical report, Software Engineering Group, School of Computer Science and Mathematics, Keele University, and Department of Computer Science, University of Durham, 2007. EBSE Technical Report, EBSE-2007-01.

Barbara A. Kitchenham, Lesley M. Pickard, and Shari Lawrence Pfleeger. Case Studies for Method and Tool Evaluation. *IEEE Software*, 12(4):52–62, July 1995. ISSN 0740-7459. doi: 10.1109/52.391832.

Barbara A. Kitchenham, Tore Dybå, and Magne Jørgensen. Evidence-based software engineering. In Anthony Finkelstein, Jacky Estublier, and David Rosenblum, editors, *Pro-*

ceedings of the 26th International Conference on Software Engineering (ICSE 2004), May 23th-28th, Edinburgh, Scotland, pages 273–281. IEEE Computer Society, May 2004.

Sandeep Krishnamurthy. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday*, 7(6), 2002.

Sandeep Krishnamurthy. An Analysis of Open Source Business Models. In Feller et al. (2005b), pages 279–296. ISBN 0-262-06246-1.

Philippe Kruchten. Putting the "Engineering" into "Software Engineering". In Paul Strooper, editor, *Proceedings of the 2004 Australian Software Engineering Conference, 13-16 April, Melbourne, Australia*, pages 2–8. IEEE Computer Society, 2004. doi: 10.1109/aswec.2004.1290452.

Douglas Kunda and Laurence Brooks. Identifying and Classifying Processes (traditional and soft factors) that Support COTS Component Selection: a Case Study. *European Journal of Information Systems*, 9(4):226–234, December 2000.

Karim R. Lakhani and Eric von Hippel. How open source software works: 'free' user-to-user assistance. *Research Policy*, 32(6):923 – 943, 2003. ISSN 0048-7333. doi: 10.1016/s0048-7333(02)00095-1.

Rikard Land, Lauren Blankers, Michel Chaudron, and Ivica Crnkovic. COTS Selection Best Practices in Literature and in Industry. In Hong Mei, editor, *Proceedings of the 10th International Conference on Software Reuse (ICSR 2008), May 25th-29th, Beijing, China*, volume 5030/2008 of *Lecture Notes in Computer Science*, pages 100–111. Springer, July 2008. ISBN 978-3-540-68062-8. doi: 10.1007/978-3-540-68073-4_9.

Rikard Land, Daniel Sundmark, Frank Lüders, Iva Krasteva, and Adnan Causevic. Reuse with Software Components - A Survey of Industrial State of Practice. In Stephen H. Edwards and Gregory Kulczycki, editors, *Proceedings of the 11th International Conference on Software Reuse (ICSR 2009) - Formal Foundations of Reuse and Domain Engineering, September 27th-30th, Falls Church, USA*, volume 5791/2009 of *Lecture Notes in Computer Science*, pages 150–159, 2009. ISBN 978-3-642-04210-2. doi: 10.1007/978-3-642-04211-9_15.

Luigi Lavazza. Beyond Total Cost of Ownership: Applying Balanced Scorecards to Open-Source Software. In Sergiu Dascalu, Petre Dini, Sandro Morasca, Tadashi Ohta, and Andre Oboler, editors, *Proceedings of the International Conference on Software Engineering Advances (ICSEA 2007), August 25th-31st, Cap Esterel, France*, pages 74–74, 2007. doi: 10.1109/icsea.2007.19.

Grace A. Lewis and Edwin J. Morris. From System Requirements to COTS Evaluation Criteria. In Rick Kazman and Dan Port, editors, *Proceedings of the Third International Conference on COTS-Based Software Systems, (ICCBSS 2004), February 1st-4th, Redondo Beach, USA*, volume Volume 2959/2004 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2004. doi: 10.1007/b96987.

Jingyue Li. *Process Improvement and Risk Management in Off-The-Shelf Component-Based Development*. PhD thesis, Norwegian University of Science and Technology NTNU, 2006.

Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Christian Bunse, Umair Khan, Marco Torchiano, and Maurizio Morisio. Validation of New Theses on Off-the-Shelf Component Based Development. In Filippo Lanubile and Carolyn B. Seaman, editors, *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS '05), September 19th-22nd, Como, Italy*, page 26. IEEE Computer Society, 2005. ISBN 0-7695-2371-4. doi: 10.1109/metrics.2005.53.

Jingyue Li, Finn Olav Bjørnson, Reidar Conradi, and Vigdis By Kampenes. An Empirical Study of Variations in COTS-Based Software Development Processes in the Norwegian IT Industry. *Empirical Software Engineering*, 11(3):433–461, 2006a. ISSN 1382-3256. doi: 10.1007/s10664-006-9005-5.

Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Christian Bunse, Marco Torchiano, and Maurizio Morisio. An Empirical Study on Decision Making in Off-The-Shelf Component-Based Development. In Osterweil et al. (2006), pages 897–900. ISBN 1-59593-375-1. doi: 10.1145/1134285.1134446.

Jingyue Li, Marco Torchiano, Reidar Conradi, Odd Petter N. Slyngstad, and Christian Bunse. A State-of-the-Practice Survey of Off-the-Shelf Component-Based Development Processes. In Maurizio Morisio, editor, *Proceedings of the 9th International Conference on Software Reuse (ICSR'06), June 12th-15th, Torino, Italy*, volume Volume 4039/2006 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2006c. ISBN 3-540-34606-6. doi: 10.1007/11763864_2.

Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Marco Torchiano, Maurizio Morisio, and Christian Bunse. A state-of-the-practice survey of risk management in development with off-the-shelf software components. *IEEE Transactions on Software Engineering*, 34(2):271–286, March 2008. ISSN 0098-5589. doi: 10.1109/tse.2008.14.

Jingyue Li, Reidar Conradi, Christian Bunse, Marco Torchiano, Odd Petter N. Slyngstad, and Maurizio Morisio. Development with Off-The-Shelf Components: 10 Facts. *IEEE Software*, 26(2):80–87, 2009. ISSN 0740-7459. doi: 10.1109/MS.2009.33.

Juho Lindman, Matti Rossi, and Pentti Marttiin. Applying Open Source Development Practices Inside a Company. In Russo et al. (2008), pages 381–387. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_36.

Björn Lundell, Brian Lings, and Edvin Lindqvist. Perceptions and Uptake of Open Source in Swedish Organisations. In Damiani et al. (2006), pages 155–163. ISBN 978-0-387-34225-2. doi: 10.1007/0-387-34226-5.

Sajjad Mahmood, Richard Lai, and Y. S. Kim. Survey of component-based software development. *IET Software*, 1(2):57–66, 2007. doi: 10.1049/iet-sen:20060045.

Annick Majchrowski and Jean-Christophe Deprez. An Operational Approach for Selecting Open Source Components in a Software Development Project. In Rory V. O'Connor, Nathan Baddoo, Kari Smolander, and Richard Messnarz, editors, *Proceedings of the 15th European Conference on Software Process Improvement (EuroSPI 2008), September 3rd-5th, Dublin, Ireland*, volume 16 of *Communications in Computer and Information Science*, pages 176–188. Springer, 2008. ISBN 978-3-540-85934-5. doi: 10.1007/978-3-540-85936-9_16.

Pekka Maki-Asiala and Mari Matinlassi. Quality Assurance of Open Source Components: Integrator Point of View. In Carl K. Chang, Aditya Mathur, and Johnny Wong, editors, *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC '06), September 17th-21st, Chicago, USA*, volume 2, pages 189–194. IEEE Comuter Society, 2006. ISBN 0-7695-2655-1. doi: 10.1109/compsac.2006.153.

Herwig Mannaert and Kris Ven. The Use of Open Source Software Platforms by Independent Software Vendors: Issues and Opportunities. In Feller et al. (2005a), pages 35–38. ISBN 1-59593-127-9. doi: 10.1145/1083258.1083266.

Douglas McIlroy. Mass Produced Software Components. In Naur and Randell (1969), pages 138–151.

Catharina Melian and Magnus Mähring. Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard. In Russo et al. (2008), pages 93–104. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_8.

Janne Merilinna and Mari Matinlassi. State of the Art and Practice of OpenSource Component Integration. In Ivica Crnkovic and Paul Grünbacher, editors, *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), August 29th - September 1th, Dubrovnik, Croatia*, pages 170–177. IEEE Computer Society, 2006. ISBN 0-7695-2594-6. doi: 10.1109/euromicro.2006.61.

Hafedh Mili, Fatma Mili, and Ali Mili. Reusing Software: Issues and Research Directions. *Software Engineering, IEEE Transactions on*, 21(6):528–562, Jun 1995. ISSN 0098-5589. doi: 10.1109/32.391379.

Farrokh Mistree and Janet K. Allen. Optimization in Decision-Based Design. In *Proccedings of Decision-Based Design Workshop, April, Orlando, USA*, 1997.

Audris Mockus and James D. Herbsleb. Why Not Improve Coordination in Distributed Software Development by Stealing Good Ideas from Open Source? In Joseph Feller, Brian Fitzgerald, Frank Hecker, Scott A. Hissam, and Karim R. Lakhani, editors, *Meeting challenges and surviving success: the 2nd Workshop on Open Source Software Engineering (WOSSE 2002), May 25th, Orlando, USA*, 2002.

Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002. ISSN 1049-331X. doi: 10.1145/567793.567795.

Parastoo Mohagheghi. *The Impact of Software Reuse and Incremental Development on the Quality of Large Systems*. PhD thesis, Norwegian University of Science and Technology NTNU, July 2004. ISBN 82-471-6408-6 ISSN 1503-8181.

Parastoo Mohagheghi and Reidar Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, 12 (5):471–516, 2007. ISSN 1382-3256. doi: 10.1007/s10664-007-9040-x.

Abdallah Mohamed, Güenther Ruhe, and Armin Eberlein. COTS Selection: Past, Present, and Future. In John Leaney, Jerzy W. Rozenblit, and Jianfeng Peng, editors, *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS '07), March 26th-29th, Tucson, USA*, pages 103–114. IEEE Computer Society, March 2007. ISBN 0-7695-2772-8. doi: 10.1109/ecbs.2007.28.

Lorraine Morgan and Patrick Finnegan. How Perceptions of Open Source Software Influence Adoption: An Exploratory Study. In Hubert Österle, Joachim Schelp, and Robert Winter, editors, *Proceedings of the Fifteenth European Conference on Information Systems (ECIS 2007), June 7th-9th, St. Gallen, Switzerland*, pages 973–984. University of St. Gallen, 2007.

Maurizio Morisio, Colin Tully, and Michel Ezran. Diversity in Reuse Processes. *IEEE Software*, 17(4):56–63, 2000. ISSN 0740-7459. doi: 10.1109/52.854069.

Maurizio Morisio, Carolyn B. Seaman, Victor R. Basili, A. T. Parra, Steve E. Kraft, and Steven E. Condon. COTS-based software development: Processes and open issues. *Journal of Systems and Software*, 61(3):189 – 199, 2002. ISSN 0164-1212. doi: doi: 10.1016/s0164-1212(01)00147-9.

Giuseppe Munda. Social multi-criteria evaluation: Methodological foundations and operational consequences. *European Journal of Operational Research*, 158(3):662–677, 2004. ISSN 0377-2217. doi: 10.1016/s0377-2217(03)00369-2.

NACE. Nomenclature statistique des activités économiques dans la Communauté européenne, 2009. URL `http://ec.europa.eu/environment/emas/pdf/general/nacecodes_en.pdf`. Accessed 2009-07-14.

Peter Naur and Brian Randell, editors. *Software Engineering, Report on a conference sponsored by the NATO SCIENCE COMMITTEE, October 7th-11th, Garmisch, Germany*. Scientific Affairs Division NATO, Brussels, Belgium, 1969.

Cornelius Ncube and John Dean. The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components. In John Dean and Andrée Gravel, editors, *Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS 2002), February 4th-6th, Orlando, USA*, volume 2255/2002 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2002. ISBN 978-3-540-43100-8. doi: 10.1007/3-540-45588-4_17.

Uolevi Nikula and Sami Jantunen. Quantifying the Interest in Open Source System: Case South-East Finland. In Scotto and Succi (2005), pages 192–195.

John Noll. What Constitutes Open Source? A Study of the Vista Electronic Medical Record Software. In Boldyreff et al. (2009), pages 310–319. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_27.

Jeffrey S. Norris. Mission-critical Development with Open Source Software: Lessons Learned. *IEEE Software*, 21(1):42–49, 2004. doi: 10.1109/MS.2004.1259211.

Michael Ochs, Dietmar Pfahl, Gunther Chrobok-Diening, and Beate Nothhelfer-Kolb. A Method for Efficient Measurement-based COTS Assessment and Selection - Method Description and Evaluation Results. In *Proceedings of the Seventh International Software Metrics Symposium (METRICS 2001), April 4th-6th, London, England*, pages 285–296. IEEE Computer Society, 2001. doi: 10.1109/metric.2001.915536.

OpenBRR, 2005. Business Readiness Rating for Open Source. Technical Report BRR 2005 - RFC 1, www.openbrr.org, 2005. URL `http://www.openbrr.org/wiki/images/d/da/BRR_whitepaper_2005RFC1.pdf`.

Thomas Østerlie and Letizia. Jaccheri. A Critical Review of Software Engineering Research on Open Source Software Development. In Wrycza Stanislaw, editor, *Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, June 5th, Gdansk, Poland*, pages 12–20. Gdansk University Press, 2007. ISBN 978-83-7326-447-2.

Thomas Østerlie and Alf Inge Wang. Debugging Integrated Systems: An Ethnographic Study of Debugging Practice. In Ladan Tahvildari and Gerardo Canfora, editors, *Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM'2007), October 2nd-5th, Paris, France*, pages 305–314. IEEE Computer Society Press, 2007. ISBN 978-1-4244-1256-3. doi: 10.1109/ICSM.2007.4362643.

Leon J. Osterweil. A Future for Software Engineering? In Lionel C. Briand and Alexander L. Wolf, editors, *Proceedings of the 29th International Conference on Software Engineering archive (ICSE) - Future of Software Engineering, May 20th-26th, Minneapolis, USA*, pages 1–11, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2829-5. doi: 10.1109/fose.2007.1.

Leon J. Osterweil, Dieter Rombach, and Mary Lou Soffa, editors. *Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), May 20th-28th, Shanghai, China*, 2006. ACM Press. ISBN 1-59593-375-1.

Bülent Özel, Uros Jovanovic, Beyza Oba, and Manon van Leeuwen. Perceptions on F/OSS Adoption. In Feller et al. (2007), pages 319–324. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_35.

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972. ISSN 0001-0782. doi: 10.1145/361598.361623.

James W. Paulson, Giancarlo Succi, and Armin Eberlein. An Empirical Study of Open-Source and Closed-Source Software Products. *IEEE Transactions on Software Engineering*, 30(4):246–256, April 2004. ISSN 0098-5589. doi: 10.1109/TSE.2004.1274044.

Bruce Perens. The Open Source Definition. In DiBona et al. (1999). ISBN 1-56592-582-3.

Anna Persson, Brian Lings, Björn Lundell, Anders Mattsson, and Ulf Ärlig. Communication, Coordination and Control in Distributed Development: an OSS Case Study. In Scotto and Succi (2005), pages 88–92.

Colin Potts. Software-Engineering Research Revisited. *IEEE Software*, 10(5):19–28, Sep 1993. ISSN 0740-7459. doi: 10.1109/52.232392.

Pascal Ravesteyn and Gilbert Silvius. Willingness to Cooperate Within the Open Source Software Domain. In Russo et al. (2008), pages 367–373. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_34.

Eric S. Raymond. A Brief History of Hackerdom. In DiBona et al. (1999). ISBN 1-56592-582-3.

Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 2001. ISBN 0596001088.

Eric S. Raymond. Up from Alchemy. *IEEE Software*, 21(1):88, 90, 2004. ISSN 0740-7459. doi: 10.1109/ms.2004.1259228.

Donald J. Reifer, Victor R. Basili, Barry W. Boehm, and Betsy Clark. Eight Lessons Learned during COTS-Based Systems Maintenance. *IEEE Software*, 20(5):94–96, 2003. ISSN 0740-7459. doi: 10.1109/ms.2003.1231161.

Gregorio Robles, Santiago Dueñas, and Jesús M. González-Barahona. Corporate involvement of libre software: Study of presence in debian code over time. In Feller et al. (2007), pages 121–132. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_10.

Colin Robson. *Real World Research*. Blackwell Publishing, 2nd edition, 2002. ISBN 978-0-631-21305-5.

Everett M. Rogers. *Diffusion of Innovations*. Free Press, New York, USA, 5th edition, 2003. ISBN 0-7432-2209-1.

Larwrence Rosen. *Open Source Licensing: Software Freedom and Intellectual Property Law*. Prentice Hall PTR, 2005. ISBN 0-13-148787-6.

Bruno Rossi, Barbara Russo, and Giancarlo Succi. A study on the introduction of Open Source Software in the Public Administration. In Damiani et al. (2006), pages 165–171. ISBN 978-0-387-34225-2. doi: 10.1007/0-387-34226-5_16.

Marcus A. Rothenberger, Kevin J. Dooley, Uday R. Kulkarni, and Nader Nada. Strategies for software reuse: a principal component analysis of reuse practices. *Software Engineering, IEEE Transactions on*, 29(9):825–837, Sept. 2003. ISSN 0098-5589. doi: 10.1109/tse.2003.1232287.

Güenther Ruhe. Intelligent Support for Selection of COTS Products. In Akmal B. Chaudhri, Mario Jeckle, Erhard Rahm, and Rainer Unland, editors, *Web- and Database-Related Workshops (NODe 2002), October 7th-10th, Erfurt, Germany*, volume 2593/2009 of *Lecture Notes in Computer Science*, pages 34–45. Springer, 2002. ISBN 978-3-540-00745-6. doi: 10.1007/3-540-36560-5_3.

Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors. *Proceedings of the 4th IFIP Working Group 2.13 International Conferences on Open Source Software (OSS2008) - Open Source Development Communities and Quality, September 7th-10th, Milano, Italy*, volume 275/2008 of *IFIP Advances in Information and Communication Technology*, 2008. Springer. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1.

Ioannis Samoladas, Georgios Gousios, Diomidis Spinellis, and Ioannis Stamelos. The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation . In Russo et al. (2008), pages 237–248. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_19.

Walt Scacchi. Free and Open Source Development Practices in the Game Community. *IEEE Software*, 21(1):59–66, 2004. ISSN 0740-7459. doi: 10.1109/ms.2004.1259221.

Walt Scacchi, Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani. Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice*, 11(2):95–105, 2006. doi: 10.1002/spip.255.

Per Kristian Schanke. Going open: Building the platform to reach out. Master's thesis, Norwegian University of Science and Technology NTNU, 2007.

Marco Scotto and Giancarlo Succi, editors. *Proceedings of The First International Conference on Open Source Systems (OSS2005), July 11th-15th, Genova, Italy*, 2005.

Judith Segal, Antony Grinyer, and Helen Sharp. The type of evidence produced by empirical software engineers. *SIGSOFT Software Engineering Notes*, 30(4):1–4, 2005. ISSN 0163-5948. doi: 10.1145/1082983.1083176.

Raphaël Semeteys, Oliver Pilot, Laurent Baudrillard, Gonéri Le Bouder, and Wolfgang Pinkhardt. Method for Qualification and Selection of Open Source software (QSOS) version 1.6. Technical report, Atos Origin, April 2006. URL `http://www.qsos.org/download/qsos-1.6-en.pdf`.

Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors. *Guide to Advanced Empirical Software Engineering*. Springer, 2008. ISBN 978-1-84800-043-8. doi: 10.1007/978-1-84800-044-5.

Dag I. K. Sjøberg, Tore Dybå, and Magne Jørgensen. The Future of Empirical Methods in Software Engineering Research. In Lionel C. Briand and Alexander L. Wolf, editors, *Proceedings of Future of Software Engineering (FOSE '07), May 23rd-25th, Minneapolis, USA*, pages 358–378, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2829-5. doi: 10.1109/fose.2007.30.

Tron André Skarpenes and Ketil Sandanger Velle. Open Source Software at Telenor IS. Master's thesis, Norwegian University of Science and Technology, 2009.

Ian Sommerville. *Software Engineering*. Addison Wesley, 8th edition, 2007. ISBN 978-0321313799.

Diomidis Spinellis. Global software development in the FreeBSD project. In Philippe Kruchten, Deependra Moitra, Wolfgang Strigel, and Christof Ebert, editors, *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner, May 23rd, Shanghai, China*, pages 73–79. ACM Press, 2006. doi: 10.1145/1138506.1138524.

SSB. Statistics Norway - Databehandlingsvirksomhet. Strukturstatistikk, 2007, 2009. URL `http://www.ssb.no/stdata/`. Accessed 2009-07-13.

Richard M. Stallman. The GNU Operating System and the Free Software Movement. In DiBona et al. (1999). ISBN 1-56592-582-3.

Richard M. Stallman and Lawrence Lessig. *Free Software Free Society: selected essays of Richard M. Stallman*. Free Software Foundation, June 2002. ISBN 978-1882114986.

Wounter Stam. When does community participation enhance the performance of open source software companies? *Research Policy*, 38(8):1288–1299, October 2009. doi: 10.1016/j.respol.2009.06.004.

Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou, and G. L. Bleris. Code quality analysis in open source software development. *Information Systems Journal*, 12(1):43–60, 2002.

Klaas-Jan Stol and Muhammed Ali Babar. Reporting Empirical Research in Open Source Software: The State of Practice. In Boldyreff et al. (2009), pages 156–169. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_15.

Lucy A. Suchman. *Plans and Situated Actions : The problem of human-machine communication*. Cambridge University Press, 1987. ISBN 0-521-33137-4.

Clemens Szyperski, Dominik Gruntz, and Stephan Murer. *Component Software, Beyond Object Oriented Programming*. Addison-Wesley, 2nd. edition, 2002. ISBN 0-201-74572-0.

Davide Taibi, Luigi Lavazza, and Sandro Morasca. OpenBQR: a framework for the assessment of OSS. In Feller et al. (2007), pages 173–186. ISBN 978-0-387-72485-0. doi: 10.1007/978-0-387-72486-7_14.

Davide Taibi, Vieri del Bianco, Davide Dalle Carbonare, Luigi Lavazza, and Sandro Morasca. Towards The Evaluation of OSS Trustworthiness: Lessons Learned From The Observation of Relevant OSS Projects. In Russo et al. (2008). ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_37.

Walter F. Tichy, Nico Habermann, and Lutz Prechelt. Summary of the Dagstuhl workshop on future directions in software engineering: February 17–21, 1992, SchloßDagstuhl. *SIGSOFT Software Engineering Notes*, 18(1):35–48, 1993. ISSN 0163-5948. doi: 10.1145/157397.157399.

Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28 (1):9 – 18, 1995. ISSN 0164-1212. doi: 10.1016/0164-1212(94)00111-y.

Marco Torchiano and Maurizio Morisio. Overlooked Aspects of COTS-Based Development. *IEEE Software*, 21(2):88–93, 2004. ISSN 0740-7459. doi: 10.1109/ms.2004.1270770.

Vu Tran and Dar-Biau Liu. A procurement-centric Model for Engineering Component-based Software Systems. In Ez Nahouraii, editor, *Proceedings of the Fifth International Symposium on Assessment of Software Tools and Technologies, June 2nd-5th, Pittsburgh, USA*, pages 70–79. IEEE Computer Society, June 1997. doi: 10.1109/ast.1997.599913.

Mark Turner, Barbara A. Kitchenham, David Budgen, and Pearl Brereton. Lessons learnt Undertaking a Large-scale Systematic Literature Review. In Guiseppe Visaggio, Maria Teresa Baldassarre, Stephen Linkman, and Mark Turner, editors, *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE 2008), June 26th-27th, Bari, Italy*. British Computer Society, 2008.

Medha Umarji, Susan Elliott Sim, and Crista Lopes. Archetypal Internet-Scale Source Code Searching. In Russo et al. (2008), pages 257–263. ISBN 978-0-387-09683-4. doi: 10.1007/978-0-387-09684-1_21.

Frank van der Linden. Full Project Proposal COSI Co-development using inner & Open source in Software Intensive products. Technical report, ITEA, 2006.

Frank van der Linden, Björn Lundell, and Pentti Marttiin. Commodification of Industrial Software: A Case for Open Source. *IEEE Software*, 26(4):77–83, July-Aug. 2009. ISSN 0740-7459. doi: 10.1109/ms.2009.88.

Kris Ven and Jan Verelst. The Organizational Adoption of Open Source Server Software by Belgian Organizations. In Damiani et al. (2006), pages 111–122. ISBN 978-0-387-34225-2. doi: 10.1007/0-387-34226-5_11.

Kris Ven and Jan Verelst. The Impact of Ideology on the Organizational Adoption of Open Source Software. *Journal of Database Management*, 19(2):58–72, April 2008.

Kris Ven, Dieter Van Nuffel, and Jan Verelst. The Introduction of OpenOffice.org in the Brussels Public Administration. In Damiani et al. (2006), pages 123–134. ISBN 978-0-387-34225-2. doi: 10.1007/0-387-34226-5_12.

Kris Ven, Jan Verelst, and Herwig Mannaert. Should You Adopt Open Source Software? *IEEE Software*, 25(3):54–59, 2008. ISSN 0740-7459. doi: 10.1109/ms.2008.73.

Viswanath Venkatesh, Michael G. Morris, Gordon B. Davis, and Fred D. Davis. User Acceptance of Information Technology: Toward a Unified View. *Mis Quarterly*, 27(3): 425–478, September 2003.

Iris Vessey, Venkataraman Ramesh, and Robert L. Glass. Research in Information Systems: An Empirical Study of Diversity in the Discipline and Its Journals. *Journal of Management Information Systems*, 19(2):129–174, 2002. ISSN 0724-1222.

Padmal Vitharana. Risks and Challenges of Component-Based Software Development. *Communications of the ACM*, 46(8):67–72, 2003. doi: 10.1145/859670.859671.

Padmal Vitharana, Fatemah "Mariam" Zahedi, and Hemant Jain. Design, Retrieval, and Assembly in Component-based Software Development. *Communications of the ACM*, 46(11):97–102, 2003. ISSN 0001-0782. doi: 10.1145/948383.948387.

Georg von Krogh and Eric von Hippel. Special issue on open source software development. *Research Policy*, 32(7):1149 – 1157, 2003. ISSN 0048-7333. doi: 10.1016/s0048-7333(03)00054-4.

Georg von Krogh and Eric von Hippel. The Promise of Research on Open Source Software. *Management Science*, 52(7):975–983, July 2006. ISSN 0025-1909. doi: 10.1287/mnsc.1060.0560.

Huaiqing Wang and Chen Wang. Open Source Software Adoption: A Status Report. *IEEE Software*, 18(2):90–95, 2001. ISSN 0740-7459. doi: 10.1109/52.914753.

Juhani Warsta and Pekka Abrahamsson. Is open source software development essentially an agile method? In Feller et al. (2003).

Steven Weber. *The Success of Open Source*. Harvard University Press Camebridge, 2004. ISBN 0-674-01292-5.

Jacco Wesselius. The Bazaar inside the Cathedral: Business Models for Internal Markets. *IEEE Software*, 25(3):60–66, 2008. ISSN 0740-7459. doi: 10.1109/ms.2008.79.

Mike N. Wicks and Richard G. Dewar. A new research agenda for tool integration. *Journal of Systems and Software*, 80(9):1569–1585, September 2007. ISSN 0164-1212. doi: 10.1016/j.jss.2007.03.089.

Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, Boston / Dorrecht / London, 2000. ISBN 0-7923-8682-5.

Claes Wohlin, Martin Höst, and Kennet Henningsson. Empirical Research Methods in Software Engineering. In Reidar Conradi and Alf Inge Wang, editors, *Empirical*

Methods and Studies in Software Engineering, volume 2765/2003 of *Lecture Notes in Computer Science*, pages 7–23. Springer, 2003. ISBN 978-3-540-40672-3. doi: 10.1007/b11962.

Ye Yang, Jesal Bhuta, Barry W. Boehm, and Daniel N. Port. Value-Based Processes for COTS-Based Applications. *IEEE Software*, 22(4):54–62, August 2005. ISSN 0740-7459. doi: 10.1109/MS.2005.112.

Robert K. Yin. *Case Study Research Design and Methods*. Applied Social Research Methods. Sage Publications, 3rd edition, 2003. ISBN 0-7619-2553-8.

Liguo Yu. Understanding component co-evolution with a study on Linux. *Empirical Software Engineering*, 12(2):123–141, apr 2007. doi: 10.1007/s10664-006-9000-x.

Marvin V. Zelkowitz. An update to experimental models for validating computer technology. *Journal of Systems and Software*, 82(3):373–376, March 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2008.06.040.

Marvin V. Zelkowitz and Dolores R. Wallace. Experimental Models for Validating Technologies. *IEEE Computer*, 31(5):23–31, 1998. ISSN 0018-9162.

Sven Ziemer, Øyvind Hauge, Thomas Østerlie, and Juho Lindman. Understanding Open Source in an Industrial Context. In Albert Dipanda, Richard Chbeir, and Kokou Yetongnon, editors, *Proceedings of the 4th IEEE International Conference on Signal-Image Technology & Internet-Based Systems (SITIS'2008), November 30th-December 3rd, Bali, Indonesia*, pages 539–546. IEEE Computer Society, 2008. ISBN 978-0-7695-3493-0. doi: 10.1109/SITIS.2008.99.

Appendix A

Selected Papers

PAPER 1

Øyvind Hauge, Carl-Fredrik Sørensen, and Andreas Røsdal. *Surveying Industrial Roles in Open Source Software Development.* In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors, Proceedings of the 3rd IFIP WG 2.13 International Conference on Open Source Software (OSS2007) - Open Source Development, Adoption and Innovation, June 11-14, Limerick, Ireland, volume 234/2007 of IFIP, pages 259-264, 2007. 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)

PAPER 2

Øyvind Hauge, Carl-Fredrik Sørensen, and Reidar Conradi. *Adoption of Open Source in the Software Industry.* In Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors, Proceedings of the 4th IFIP WG 2.13 International Conferences on Open Source Software (OSS2008) - Open Source Development Communities and Quality, September 7-10, Milano, Italy, volume 275/2008 of IFIP, pages 211-222, 2008. Springer.

Adoption of Open Source in the Software Industry

Øyvind Hauge, Carl-Fredrik Sørensen, and Reidar Conradi

Norwegian University of Science and Technology
{oyvind.hauge|carl.fredrik.sorensen|reidar.conradi}@idi.ntnu.no

Abstract: Is Open Source Software (OSS) undergoing a transformation to a more commercially viable form? We have performed a survey to investigate the adoption of OSS in the Norwegian software industry. The survey was based on an extensive screening of software companies, with more than 700 responses. The survey results support the transformation predicted by Fitzgerald [4]. Close to 50% of the software industry integrate OSS components into vertical solutions serving all major business sectors. In addition, more than 30% of the 95 respondents in our survey have more than 40% of their income from OSS related services or software. The extensive adoption of OSS in the software industry may be a precursor of the OSS adoption in other business sectors.

Introduction

Open source software (OSS) is predicted to transform "into a more mainstream and commercially viable form" [4], where companies play an increasingly more important role. A premise for this transformation is increased commercial participation in the development of OSS products and increased use of OSS in vertical domains. However, only a few surveys provide empirical findings which support this assumed transformation, and most of these focus on the use of desktop tools and horizontal infrastructures like the LAMP stack. Is really OSS undergoing a transformation?

To answer this question we have performed a large scale survey in the Norwegian software industry. Our analysis shows that close to 50% of the software industry integrate OSS components into vertical solutions targeting customers from all major business sectors. In addition, more than 30% of the respondents in our survey have more than 40% of their income from OSS related services or software.

Our results show that the adoption of OSS in the Norwegian software industry is significant. The industry's contribution to the OSS community is however limited. Nevertheless, it is reason to believe that OSS is actually undergoing a transformation into a more commercially viable form. The use of OSS in the software industry may eventually influence the rest of the market when software companies integrate OSS into their products. However, a lack of software companies adopting OSS may hamper the adoption of OSS in other sectors [15].

Related Work

Estimating the market share of OSS is a comprehensive task. Nevertheless, several attempts have been made e.g. [5, 6, 18]. Common to most of these is their focus on a few products like the LAMP stack and end-user applications like mail or office tools. One example is the Netcraft[1] survey of web servers on the Internet. While several consultancy companies have attempted to estimate the adoption of OSS, we rather focus on research published through academic channels.

Without providing any numbers, Glynn et al. conclude that OSS has had significant penetration in the software/consultancy and service/communication sector, but that it is more limited in the government/public sector [7]. Studies from the UK [17], Finland [15], and the U.S. [16] report only limited OSS adoption in the public sector with Linux as the only exception. Linux was used by more than 50% of the respondents in both the study from Finland and the U.S. Together with the other elements of the LAMP stack, Linux is quite frequently used in other sectors as well [6]. However, this adoption varies from country to country, on company size, and between sectors. For a mixed sample from industry and public sector, the use or planned use is reported to be as low as 17.7% in Sweden and as high as 43.7% in Germany [6]. Furthermore, numbers vary between about 10% and 75% for different strata [6]. A survey on Australia's top companies reports that 26% of the respondents used a varied spectrum of OSS products [8]. With the exception of Linux, Apache HTTP Server and perhaps a few others, most surveys report that less than 30% of the respondents have adopted OSS. Yet, little is known about the extent of the internal adoption of OSS in these companies.

In studies focusing on the software sector, 44% of the companies in a Finnish sample use OSS in their business [13] and in a study on Off-the-Shelf development, 44 or 38.3% of the 115 projects use OSS components [11]. Without being able to provide any numbers representative for the whole population, an Italian study found that software companies using OSS commonly adapt or build on top of these OSS products [1].

The transformation predicted by Fitzgerald involves company participation in the development of OSS products [4]. Companies are already known to be contributing by allowing employees spend their time at work participating in OSS projects [10]. Companies are among others involved in 97 of the 300 most active SourceForge projects [2]. However, this is most likely not representative for all of

[1] http://news.netcraft.com/

SourceForge's more than 170 000 projects. A Swedish survey also found that several companies actively contribute to OSS projects [12].

We see that companies and organizations have adopted OSS and that they are involved in the development of OSS. There are however only a limited number of empirical findings which show the extent of this adoption and the demography of these companies. This paper will provide results which quantifies the adoption of OSS components in the Norwegian software industry.

Survey Method

The purpose of the study was to investigate to what extent the Norwegian software industry approaches OSS development. As an expansion of [9], we carried out a nationwide survey to investigate this matter.

Population: The Norwegian Software Industry

Legal entities in Norway are registered in The Norwegian Central Coordinating Register for Legal Entities[2] (CCRLE) with a Nomenclature Generale des Activites Economiques dans L`Union Europee (NACE) code. Based on 2005 data from CCRLE and other registers, Statistics Norway3[3] (SSB) reports that about 70 000 employees, or 4.7 % of all employees in Norway, are employed in the whole ICT sector [14]. In addition, the sector has a turnover of about €22 billion [14].

Table 1 The Norwegian 72.xx sector based on data from CCRLE 2007.

Sub sector	NACE	Entities
Computer and related activities	72.00	26105
Hardware consultancy	72.10	251
Software consultancy and supply	72.20	21559
- Publishing of software (software houses: resale)	72.21	1295
- Other software consultancy and supply (single sale)	72.22	20264
Data processing	72.30	489
Database activities	72.40	2916
Maintenance & repair: office, accounting and comp. machinery	72.50	733
Other computer related activities	72.60	163

The ICT sector in Norway includes telecommunication (64.20), ICT manufacture industry (32.xx), ICT wholesale and retail trade (51.8x), and the soft- and

[2] http://www.brreg.no/

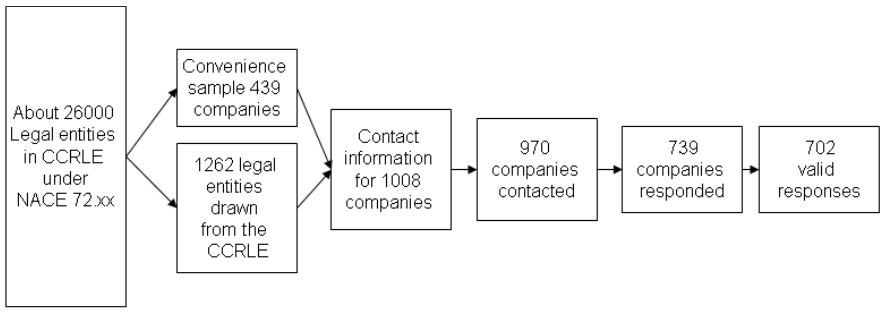
[3] The Norwegian counterpart to the U.S. Census Bureau http://www.ssb.no/

hardware sector (72.xx). Based on CCRLE data from 2007, we found that approximately 26 000 legal entities and 38 500 employees constitute the soft- and hardware sector, see Table 1. This gives an average company size of about 1.5 employees. According to SSB only about 13 000 of these legal entities are active companies. More than 70% of these have less than one full time employee and about 1300 have five or more employees [14]. We will in this paper focus on the software sector (72.2x).

The Sampling Process

Data from CCRLE helped us constructing a close-to representative sample of software companies [3], with a focus on software (72.21) and consultancy (72.22) companies with more than five employees. However, the sample also included companies from the other 72.xx sub-sectors and companies with fewer than five employees. The purpose of the sampling process illustrated in Fig. 1 was twofold. First, estimate the share of companies integrating OSS components into their products. Second, create a sample for our survey consisting of companies using OSS components.

Fig. 1 The sampling process.


About 26000 Legal entities in CCRLE under NACE 72.xx
Convenience sample 439 companies
1262 legal entities drawn from the CCRLE
Contact information for 1008 companies
970 companies contacted
739 companies responded
702 valid responses

Step 1: The sample was constructed based on a convenience sample of 439 companies and a stratified random sample of 1262 legal entities from CCRLE. The convenience sample was based on stratified random samples two from earlier studies and supplemented with companies from our knowledge and companies appearing in the media. The strata were defined according to the business organization form, the 72.xx sub-sectors, and the number of employees.

Step 2: Then, the two lists were merged. 300 duplicates entries were removed during this merger. Several companies occurred in both samples and some companies were registered with more than one legal entity, typically larger companies. We used data from CCRLE and the Internet to find web-sites and email addresses for the companies. Another 395 or 31.3 % of the 1262 randomly selected legal entities were removed from the list because no contact information could be found. Knowing that only about 50% of the companies in the sector were active, this was not a surprise. The vast majority of these companies were small and most likely inactive companies. The final list contained contact information for 1008 companies from the Norwegian software industry.

Step 3: The screening process was carried out by sending the companies a brief request on email containing the questions stated below. About 200 of the companies from the convenience sample were contacted in March 2007 and the rest in June/July. One reminder was sent by email in September. The 200 companies contacted in March were only asked the first three questions while the remaining companies were asked all four questions.

1. How many employees do you have in Norway?
2. Are you doing software development in Norway?
3. Do you use open source components in your products or services (other than Linux, Apache HTTP Server, Eclipse, PhP/Perl, MySQL etc.)?
4. Do you participate in or run any open source projects?

38 of the 1008 email addresses did not work, leaving 970 companies. 201 of these companies came from the convenience sample, 555 from the stratified random sample, and 236 companies were included in both samples. 739 companies replied which give us a response rate of 73.3%. 32 companies responded that their company was inactive or about to be dissolved, one company did not want to participate, and another four duplicate legal entities were found, leaving 702 or 69.6% valid responses. The response rates were similar across most of the different strata (size and sector). The names of the respondents and their email addresses were stored together with the other contact information.

Step 4: 569 or 81.1% of the 702 companies in our screening process confirmed that they perform software development. These companies make the basis for further analysis. The percentage of companies involved in software development is similar across different size and business types. However, when looking at sectors, the percentage varies from 73.6% (72.40 Database activities) to 90.2% (72.20 Publishing of software).

The Survey Process

Close to 50% of the 569 companies constituting our sample integrate OSS components into their products. 204 of these companies were invited to participate in a web survey. The survey contained three parts focusing on (1) development of a commercial OSS product, (2) integration of OSS components into a software product, and (3) demographic information. The respondents should answer based on their experiences with the development of a typical software product containing OSS components. This product was selected by respondents and we had no control over this selection. To learn more about the companies and to increase the response rates, every second company ordered by size was contacted by phone. The companies were asked if they could participate and were sent an email with instructions if they accepted our invitation. The other half was invited to participate through email. One reminder was sent by email about a month later. 12 or 5.9% of the 204 companies could not or did not want to participate. Nevertheless, 95 of the 204 companies or 46.6% completed the survey. Of these 95, 21 were only involved in software development without directly developing software products, for instance consultancy companies providing developers to external customers. This left 74 or 36.3% valid responses for the two main parts.

Results

This section presents results from both the screening process and from the survey.

Selection and Integration of OSS Components

Out of the 569 companies constituting our sample, 266 or 46.9% integrate OSS components into their software solutions. This use goes beyond merely using OSS operating systems, databases, infrastructure, development tools, and programming languages. The companies actually find, evaluate, and integrate OSS components into their software solutions. The integration of OSS components happens less frequently in software houses. Only 34.1% of the companies registered in sector 72.21, use OSS components in their products, see Table 2.

Table 2 Adoption of OSS components distributed over sectors.

Sector	Sample Size	OSS Adoption
72.21 Publishing of software	129	34.1%
72.22 Other software consultancy and supply	328	51.5%
72.30 Data processing	18	38.9%
72.40 Database activities	39	53.8%
Other	55	47.3%

From Table 3 we see that large companies integrate OSS components more often into their products than smaller companies. 56.9% of the companies with more than 100 employees use OSS and 50.0% of the companies with 25 to 99 employees integrate OSS components into their products compared to around 43% of the companies with between 2 and 24 employees. Companies with one or less than one full time employee, use OSS components somewhat more frequently.

66 companies completed the second part of the survey based on their experiences from the development of a software product containing OSS component. The products delivered by these companies serve all main business sectors with a small emphasis on the public and health sector. The functionality of these products was directed mainly towards web/portals and enterprise solutions. The respon-

dents classified 40.9% of the products as domain specific and 36.4% as differentiating end-user products.

Table 3 Adoption of OSS components distributed over the number of employees.

Number of employees	Sample Size	OSS Adoption
0 to 1	33	48.5%
2 to 4	61	42.6%
5 to 9	80	43.8%
10 to 24	189	43.9%
25 to 99	146	50.0%
More than 100	58	56.9%

Even though OSS components can reduce the development effort substantially, they are in most cases integrated as part of a larger solution. In 72.7% of the products, OSS components provide less than 40% of the functionality of the end product. The number of OSS components is also kept low. 68.2% of the products contain less than six OSS components and 83.3% contain less than eleven OSS components. The effort spent developing these products during the last year, range from less than one (1) person-month to between 101 and 500 person-months.

OSS Related Activities

During the last year, 75.8% of the companies have developed between one and three software products containing OSS components. In one extreme case, one company had developed more than 50 products containing OSS components. In Part 3 of the survey, we requested the respondent to estimate how much of the company's income is generated by OSS related services or software development, see Table 4. Even though 41 of the 95 respondents answered less than 20% and 22 answered "don't know", 29 or 30.5% answered that more than 40% of their income comes from OSS related services or software.

Table 4 Income from OSS related services and software development.

Income from OSS	Number of companies
NA/Don't know	22
0%	8
1-20%	33
21-40%	3
41-60%	7
61-80%	9
81-99%	6
100 %	7

Participation in OSS Projects

In total 368 of the 569 companies developing software responded to the fourth screening question. 60 or 16.3% of the respondents said that employees in their company participated in OSS projects. This participation was in some cases part of their job and in other more a hobby. Another 18 or 4.9% of the 368 companies said they have their own OSS project. However, through the researchers' previous experience with some of these companies we would say that the OSS projects are only an important part of the business for a few of them.

30 of the 66 companies completing Part 2 of the survey answered that they interacted with or participated in OSS projects during the development of their product. This interaction and participation was in all but three cases not organized through the company but rather left up to the individual developer.

Discussion

The use of OSS in Norwegian software industry is significant. Close to 50% of the companies developing software have integrated OSS components into one or more of their products and more than 30% of the respondents to our survey get over 40% of their income from OSS related services or software. The use of OSS in software houses (72.21) living of the sales of software licenses is somewhat lower than in other software sectors. This could be caused by reciprocal OSS licenses (e.g. GPL) which requires derivate products to be released under the same license, thus removing the software houses profits from sales of licenses. Another conceivable explanation is that companies focusing on development of their own software products are involved in the development of fewer products per year than consulting companies serving several different customers.

The products developed by the respondents served all major business sectors and 77.3% of them were classified as domain specific or differentiating end-user products. Thus, we can conclude that OSS is used in vertical products targeting all business sectors.

We observed increasing OSS use in relation to increasing company size. However, Lundell et al. observed that companies with more than 250 employees seemed more conservative towards OSS adoption [12], Bonaccorsi et al. found that size does not favor OSS adoption [1], and Ghosh et al. found variations in the adoption of OSS over company size, countries, and sectors [6]. The relation between size and OSS adoption needs to be investigated in future research. However, we see two possible explanations for this increased use of OSS. First, small companies commonly focus on a limited number of customers and specialize on a small set of different technologies. Several such companies replied that they were not using OSS because they focused only on one not-OSS-compatible technology. While large companies, often serve many customers using several different technologies, including OSS. Second, large companies hire more people. Because they hire more people it is more likely that they employ people with prior experiences with OSS.

Comparing the results from this survey with other results is a bit complicated. First, the number of related studies is fairly limited. Second, while we focused on integration of OSS components, most studies include all kinds of OSS products. Third, other studies focus on other sectors than the software sector. The 46.9% adoption of OSS components in the software sector is therefore lower than some of the extreme results in [5, 6]. If our survey had included all kinds of OSS, we suspect the percentage of OSS users to be significantly higher. On the other hand, our results are in line with the results from [13] where 44% of the software companies used OSS in their business.

Generalization to other countries is made somewhat more difficult because of the variations found in other studies [6]. There are also factors which influence the adoption of OSS in the Norwegian software sector. While the effects of these factors must be further examined, we believe the size of the companies, influence the OSS adoption. Most software companies in Norway are small or medium sized. Many have relatively few and mostly domestic customers, and they have also chosen to focus on a limited set of technologies. In addition, the Norwegian government has the last years increased its focus on open standards and OSS through public reports and the establishment of a national centre of expertise of OSS. Furthermore, there seems to be a push in Norway towards increased use of agile methods. While using such methods it is important to get something up and running as fast as possible. This and the fact that the cost of personnel in Norway is quite high may encourage higher reuse of code and components, including OSS.

Industrial participation in OSS projects seems limited and managed on an individual level. Only 16% of the software companies confirm that they do participate in the development of one or more external OSS products. This number has however some uncertainty. First, 200 of the companies in the screening process were not asked whether they participated in any OSS projects. Second, there is some confusion about what participation is. Some companies answered "we do not participate but we report bugs and share occasional bug fixes" while others answered "we do participate with some bug reports and bug fixes". However, we interpreted both these statements as participation. Third, participation in OSS projects is in most cases managed on a personal level. Knowing what all other employees are doing is difficult if not impossible for the respondents in our screening process and the number of companies participating in OSS projects could therefore be higher.

The sample has an intentional bias towards companies with more than five employees. This bias was reinforced by the fact that we were unable to find contact information for several small and probably inactive companies. The majority of companies without a web site are most likely inactive since nowadays the Web is considered the most important communication channel.To aid the sampling we benefited from CCRLE and the NACE sector codes. While this classification was of great help, the software industry is an industry with rapid and frequent changes.

As a consequence of these changes, central registers such as CCREL are not up-to-date at all times. For example, only about 90% of the companies under 72.20 "Publishing of software" actually develop software.

Response rates of 73.3% for the screening process and 36.3% valid responses in the main survey is decent compared to many other studies but low response rates is one of the challenges with survey research. Even though there is room for improvement, we have been able to get responses from a large and close to representative sample of the Norwegian software industry. The research design is well documented and replicating the survey in another setting should be easy, though labor-intensive.

Conclusion

Results from our study show limited company involvement in the development of OSS products but widespread use of OSS components. By integrating OSS into vertical products serving all major business sectors, the software industry will contribute to wider adoption of OSS. The software industry has clearly started to adopt OSS products and contribute to the transformation of OSS into a commercially viable form. However, this transformation is far from completed.

The results presented here are currently followed up in several ways. In parallel to this survey, we have also approached OSS adoption through qualitative studies. Data from both the survey and these qualitative studies is currently being analyzed. Findings from these analyses will provide a basis for further research. This survey focuses on the Norwegian software sector, which is dominated by small and medium sized companies. The adoption of OSS may be different in other sectors and in other countries. We are therefore looking at the possibilities of conducting similar surveys in both other sectors and European countries. To understand the trend of OSS adoption, we are also considering a replication of the survey in Norway. Furthermore, the survey identified a few companies developing their own OSS products. We plan to follow up on these companies to try to understand if they manage to attract and sustain communities and how they interact with these communities.

References

1. Andrea Bonaccorsi, Silvia Giannangeli, and Cristina Rossi. Entry Strategies under Competing Standards: Hybrid Business Models in the Open Source Software Industry. Management Science, 52(7):1085-1098, July 2006.
2. Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Cristina Rossi. Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research. In Proceedings of the First International Workshop on Emerging Trends in FLOSS

Research and Development FLOSS'07, page 13, Minneapolis, US, 2007. IEEE Computer Society.

3. Reidar Conradi, Jingyue Li, Odd Petter N. Slyngstad, Vigdis By Kampenes, Christian Bunse, Maurizio Morisio, and Marco Torchiano. Reflections on Conducting an International Survey of Software Engineering. In June Verner and Guilherme H. Travassos, editors, Proceedings on International Symposium on Empirical Software Engineering ISESE'05, pages 214-223, Brisbane, Australia, 2005.
4. Brian Fitzgerald. The Transformation of Open Source Software. MIS Quarterly, 30(3), 2006.
5. Rishab Aiyer Ghosh. Study on the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communication Technologies (ICT) Sector in the EU. Technical report, UNU-MERIT, 2006.
6. Rishab Aiyer Ghosh, Gregorio Robles, and Ruediger Glott. Free Libre and Open Source Software: Survey and Study. Technical report, International Institute of Infonomics, University of Maastricht, 2002.
7. Eugene Glynn, Brian Fitzgerald, and Chris Exton. Commercial Adoption of Open Source Software: An Empirical Study. In Proceedings of International Conference on Empirical Software Engineering, pages 225-234, Noosa Heads, Australia, 2005.
8. Sigi Goode. Something for Nothing: Management Rejection of Open Source Software in Australia's Top Firms. Information & Management, 42(5):669-681, 2005.
9. Øyvind Hauge, Carl-Fredrik Sørensen, and Andreas Røsdal. Surveying Industrial Roles in Open Source Software Development. In Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors, Proceedings on the Third International Conference on Open Source Systems, pages 259-264, Limerick, Ireland, 2007. Springer.
10. Karim R. Lakhani and Robert G. Wolf. Why Hackers Do What They Do: Understanding Motivations and Effort in Free/Open Source Software Projects. In Joseph Feller, Brian Fitzgerals, Scott A. Hissam, and Karim R. Lakhani, editors, Perspectives on Free and Open Source Software, pages 3-23. MIT Press, 2005.
11. Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Christian Bunse, Umair Khan, Marco Torchiano, and Maurizio Morisio. An Empirical Study on Off-the-Shelf Component Usage in Industrial Projects. In Frank Bomarius and Seija Komi-Sirvio, editors, Proceedings of the 6th International Conference on Product Focused Software Process Improvement PROFES'2005, pages 54-68. Springer, 2005.
12. Björn Lundell, Brian Lings, and Edvin Lindqvist. Perceptions and Uptake of Open Source in Swedish Organisations. In Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, Marco Scotto, and Giancarlo Succi, editors, Proceedings of The Second International Conference on Open Source Systems, pages 155-163, Como, Italy, 2006. Springer.
13. Uolevi Nikula and Sami Jantunen. Quantifying the Interest in Open Source System: Case South-East Finland. In Marco Scotto and Giancarlo Succi, editors, OSS 2005: Proceedings of the First International Conference on Open Source Systems, 11-15 Juli 2005, Genova, Italy, pages 192-195, 2005.
14. SSB. StatBank Norway, 2007. http://statbank.ssb.no/statistikkbanken/, accessed 2007-08-01.
15. Mikko Välimäki, Ville Oksanen, and Juha Laine. An Empirical Look at the Problems of Open Source Adoption in Finnish Municipalities. In Proceedings of the 7th International Conference on Electronic Commerce ICEC'05, pages 514-520, Xi'an, China, 2005. ACM.
16. Shahron van Rooij. Open Source software in US higher education: Reality or illusion? Education and Information Technologies, 12(4):191-209, December 2007.
17. Teresa Waring and Philip Maddocks. Open Source Software implementation in the UK public sector: Evidence from the field and implications for the future. International Journal of Information Management, 25(5):411-428, October 2005.
18. David A. Wheeler. Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!. http://www.dwheeler.com/oss_fs_why.html, accessed 2007-12-09.

PAPER 3

Sven Ziemer, **Øyvind Hauge**, Thomas Østerlie, and Juho Lindman. *Understanding Open Source in an Industrial Context.* In Albert Dipanda, Richard Chbeir, and Kokou Yetongnon, editors, Proceedings of the 4th IEEE International Conference on Signal-Image Technology&Internet-Based Systems (SITIS 2008), November 30-December 3, Bali, Indonesia, pages 539-546, 2008. IEEE Computer Society.

Understanding open source in an industrial context

Sven Ziemer, Øyvind Hauge and Thomas Østerlie
Norwegian University of Technology and Science
Sem Sælandsvei 7–9, NO-7491 Trondheim, Norway
{sven.ziemer, oyvind.hauge, thomas.osterlie}@idi.ntnu.no

Juho Lindman
Helsinki School of Economics
Runeberginkatu 22-24, FI-00101 Helsinki , Finland
juho.lindman@hse.fi

Abstract

This paper discusses the meaning of open source in an industrial context. Building a grounded theory from an industry-driven R&D project, our analysis shows that open source in an industrial context is multifaceted. We find that the meaning of open source must be established in the context of the individual organization.

Keywords: Software development with open source, Adoption of open source

1 Introduction

Over a decade has passed since the fledgling Linux industry coined the term open source to overcome resistance to industry adoption of free software [24]. Throughout this period, open source has been advocated as a viable alternative form of software development that will revolutionize the software industry [2]. A recent addition to the ongoing discussion on the impact of open source contends that open source has undergone a transformation to a more mainstream commercially viable form where vendors provide their products as open source [9].

While we agree that open source has undergone such a transformation, the nature and form of this shift is more of an open question. Indeed, we argue that open source in an industrial context is more multifaceted than merely providing open source products. This paper offers a grounded theory based on open source adoption in a European industrial R&D project. With basis in this, we seek to contribute towards developing a more nuanced understanding of what open source in an industrial context means. We find that 'what open source in an industrial setting is' can only be established in the context of individual organizations. For open source to remain a viable alternative in an industrial context, the meaning of 'open source' needs to remain fluid and multifaceted. As such, efforts to provide definite characterizations of the phenomenon remains at best fruitless and at worst counterproductive to the goal of industrial adoption of open source.

However, it is out collective experience that the fluidity of the open source term is a major stumbling block for many actors in the software industry when adopting open source. It remains unclear to them what open source actually means to their organization. Without the ability to match the fluidity of open source with the activities of their organizations, no viable form can be found. As diffusion of open source at the industry level require that individual organizations adopt open source industrially, problems of matching the fluidity of the term may hamper the industrial adoption of open source [28]. By offering a grounded theory with three categories of open source in an industrial context, this paper offers a guide for practitioners to match the fluidity of open source with their organization's activities.

To this end, the paper is organized as follows. Section 2 gives an overview over related work on industrial adoption of open source. Section 3 presents the research setting and method, while Section 4 presents an analysis of the data and our findings. Finally, concluding remarks are given in section 5.

2 Related work

OSS has been advocated as a viable alternative form of software development that which revolutionize the software industry [2]. With this premise, a critical topic for research is therefore to understand how OSS changes the way the software develop and provide software. Even though there is a trend towards higher industrial involvement in OSS development [12], most research has focused on the big, successful OSS projects such as Apache, FreeBSD, Linux, and Mozilla [23]. Two recent literature reviews on *OSS development* [26] and *OSS research* [30] illustrates this by barely mentioning OSS in a commercial context.

Defining the OSS phenomenon has been a key preoccupation of the OSS research literature. These definitions have been targeted at a bilateral audience consisting of the OSS communities and commercial companies. First, to shape the hacker identity [29] and second to create a credible option for companies [25]. Although many observe that OSS "is not a precise term" [11, page 35] and that "[t]here is a great variation in perceptions of the OS phenomenon" [18, page 157], definitions and characterizations of OSS are still predominantly of community-driven OSS development [23]. This, despite the fact that the existing literature on OSS in a commercial context illustrates that there is a spectrum of commercial approaches to OSS [8].

Depending on their organizational opportunities, a company may try to implement one of several business models [13, 15]. Two generic business models are the "support seller", where a company sells services on top of OSS, and the "loss-leader", where a company uses OSS to drive the use of industrial software product by promoting OSS [13]. How companies actually implement these business models vary and Bonaccorsi et al. found a "significant heterogeneity in ... the degree of openness to OS" [4, page 1085].

This heterogeneity is reflected in the literature by a variety of specific topics, like adoption [10], re-use [19], integration [21], maintenance [33], and evaluation [6]. Several of these activities are also studied within the software engineering field for instance [17]. In addition to adopting OSS products companies have started to show interest in the tools and development practices used in OSS communities. By looking at successful OSS projects, companies can adopt these tools and practices into their own organization [7, 20].

Despite company involvement in about one third of the top 300 SourceForge projects [5], company contributions to OSS projects have so far gained minor attention. Company contributions to OSS projects will primarily come through individual developers [16] or from more substantial company commitment [14]. Companies not only contribute to the development of community-driven OSS, but many well-known OSS products originate in commercial companies as well.

Companies may provide a variety of different OSS products. For instance programming languages, operating systems, and integrated development environments [32]. Providing such products is not trivial [3], and therein lie the challenges in maximizing the benefits from an OSS development model through for example code contributions [31].

In the following, we will study and analyze what *industrial open source* entails for companies participating in an European industry-driven R&D project.

3 Research method

Research setting – This paper reports from research performed in COSI[1], a European industrial research and development project. The project's goal is to increase awareness of industrial usage of distributed collaborative software and OSS. COSI is organized as a consortium of 13 industrial and academic partners from 5 countries.

The COSI project runs for 3 years, from November 1 2005 until October 31 2008. It is is organized into work packages (WPs) covering: community and business models, development processes, and software architecture. Each work package has a five-phased research design as outlined in Figure 1. The industrial partners execute two case executions, related to improvement work in their own organization. The overall COSI research design is that of a multi-case study; each industry partners improvement work is considered a single case study. A short overview of the phases are given below:

- *State of the art:* The partners document their state of the practice of interest, while the academic partners work on the state of the art of a related area. Based on this, the companies select their case for the first case execution.
- *1st case execution:* Depending on the company context, this case execution is improving the company practices on the selected area or preparing an improvement in the second case execution.
- *Improvement:* Using the results of the first case execution, the goals for the second case execution are identified and defined.
- *2nd case execution:* Activities to realize the improvement goals of the improvement phase are planned, executed and observed/measured.
- *Validation:* The case executions are validated; depending on each partners use of this design, both case executions or only a single case execution are validated.

The authors' role – All of the authors participate in the COSI project as representatives of academic partners. As representatives of the project partners, we function as sub-project managers of one sub-project within the WP on development processes, and two sub-projects within the WP

[1] `http://www.itea-cosi.org/`

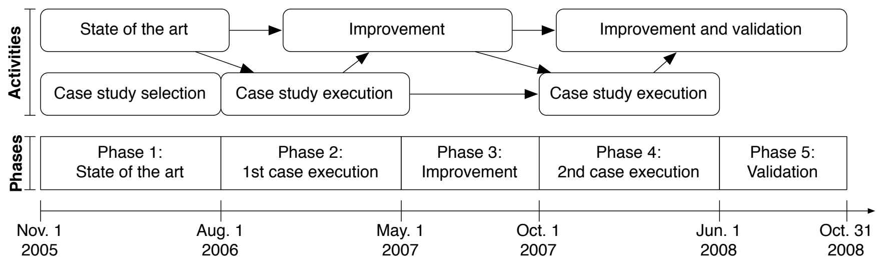


Figure 1. COSI research design

on community and business models. As sub-project managers, we have been closely involved with planning for and reporting from the industry partners' case studies.

We participate in the COSI project as part of our individual research agendas as open source researchers within respectively software engineering and information systems research. While we participate in working towards the common goals of the COSI project as project partners, the project is also as one of several research settings where we are doing research on open source. In this paper, we therefore report from the COSI project as open source researchers.

Materials – The results presented in this paper are based on an analysis of the industrial partners' activities during the three first phases of the COSI R&D project. The analysis is based on several data sources'.

- Two project deliverables: Each documenting the first iteration of the case studies and planning for the second iteration from the WP on communities and business models and from the WP on development processes [34, 22].

- We have performed 21 interviews with developers and managers from 5 industrial partners. We have taped and transcribed some of the interviews, while making notes from others.

- Three of the authors have performed 2 workshops with one industrial partner to facilitate planning for the second iteration of case studies. In addition, one of the authors had a postmortem session with another industrial partner. In both cases, notes were made.

As project participants, we have also spent time talking to the industrial partners during five project meetings held in 2006 and 2007 to learn more about their organizations and case studies.

Data analysis – The analysis focused on exploring how the industrial partners develop with open source. Through an iterative process of open and axial coding, we have built a grounded theory [27] with three categories of 'developing with open source in an industrial context' among the COSI partners, along with a set of sub-categories. These are presented in Table 3.

4 Developing with OSS in an industrial context

Two of the authors grouped and summarized the COSI partner reports, describing their activities in the project. Together, all reports [23, 34] illustrate the variety among the industrial partners' adoption of open source in industrial software development. While all COSI partners believe that they 'are doing open source software development', they use open source in different ways and in different contexts, aiming at different partner specific goals and experiencing that there exist different success criteria and enablers. As a consequence of the COSI research design, the partners are focusing on how to improve their own use of open source and describe their goals for the improvements activities. The purpose of this analysis is therefore to empirically illustrate how OSS is used in different ways by the COSI partner companies, and how the way the COSI companies develop software with open source is shaped by the *opportunities* the organization sees in adopting open source in software development. In the context of these companies, we seek to illustrate how OSS in industrial software development is a multifaceted phenomenon shaped by factors internal to companies and the environment in which they are situated. As such, we identify three broad categories of developing with OSS among these companies:

- Developing with OSS tools and practices

- Developing with OSS products

- Developing OSS products

Category	Sub-category	Data from COSI
Developing with OSS practices and tools	Inner source	• Code sharing to cooperate on joint code across departmental boundaries • Adoption of SourceForge and OSS project structure within a multi-national company to overcome reuse and redevelopment issues across projects in different parts of the organization
	Taking advantage of OSS tool standardization	• Providing plug-ins for company's commercial process modeling and improvement product on top of Eclipse
	Distributed development	• Use of SVN to facilitate software development across different networks within a global company • Use of SVN within a consultancy to facilitate work towards a source code repository while working in sites with no Internet access
Developing with OSS products	Selection and evaluation	• Developing schemas for comparing different components for use in particular projects • Schema for evaluating the risk of adopting ”infrastructure” components
	Integration	• Using OSS as off-the-shelf components to build own products • Base products upon OSS applications (database, web server, etc.) • Wrap OSS components before integration
	Outsourcing	• Company reports that it is providing a software product based upon an existing content management system in order to outsource its own development activities to community members well-versed in the product's code, for a free or as consultants
Developing OSS product	Distribution of effort	• Recruit community to help develop product as it is too expensive to develop own product from scratch • Recruit members of community to contribute to expanding existing product
	Standardization	• Establish product as de facto standard • Standardized tests for product evaluation
	Inter-company cooperation	• Releasing product as OSS in order to make use of sourceforge.net as a platform for inter-company cooperations • Product released as OSS to avoid legal issues on ownership between companies cooperating on a joint product
	Differentiating	• Using the ”open source” label to differentiate product in a saturated market

Table 1. Categories of development with OSS in the COSI project

There are variations among the companies, and we give some examples within each of the three main categories. These categories, along with examples from the COSI project are summarized in Table 3. The remainder of this section is dedicated a more thorough presentation of each of the three categories with examples from the COSI project. These examples reflect the companies own use of open source. The relation between the categories is discussed later in this paper.

4.1 Developing with OSS tools and practice

This category covers the use of OSS development tools and OSS practices [20] in the development activities of the COSI partner companies. Examples of tools that are used by the COSI partners are editors, compilers, build environments and issue trackers. Some partners have also introduced software practices that are commonly associated with successful OSS communities. The software developed by partners in this category does not have to be an OSS product.

Beyond the issue of tool choice for software development and/or use of OSS practices, we found the COSI partners addressing several issues within this category.

Code Sharing – Two of the COSI partner companies applied tools and practices to improve their development practices on code sharing [7]. In one case, the company – developing a software product line – is introducing code sharing between different development teams, where one team is developing core components and the remaining teams are integrating these into different products. In doing so, the company sees two opportunities, making this approach worthwhile: increased *knowledge sharing* between the development teams and shorter *time-to-market*. In a second case, a company adopts SourceForge and OSS project structures, to provide an infrastructure for *code sharing*. However, in this case the focus is on *reusing previously developed software* in new projects.

OSS tool standardization – The use of OSS *development tools* is widespread and several of the COSI partners use OSS development tools by default. In addition to software development, OSS tools can be used as part of *consulting services*. One company in the project has built their own framework for process modeling on top of Eclipse. Together with Eclipse, this tool is used in the company's consulting services.

Distributed development – Another company in the COSI project experienced a shift towards *geographically distributed software development* internally in the company. To support this development, a transition towards an OSS configuration management tool with Internet support was started. A different case is the support for *nomadic software development* that has been introduced by another COSI partner company. In this case the company wants to support their developers in case they are without network access, and make it possible to make changes to local copies of the code repository.

4.2 Developing with OSS products

Developing with OSS products encompasses development activities where OSS products are adopted and integrated into a software system. An OSS product can – among others – part of the technical infrastructure of a software system, or a component offering some service. Typical activities for this category are selecting OSS products [6], maintaining or customizing OSS products [33] as well as integration issues [21]. The examples from the COSI R&D project give insight into how these practices are shaped by company context and opportunities each company pursues in adapting these practices.

Selection – Several COSI partner companies are addressing issues in their selection practices. The variety of issues illustrate how different the partners handle OSS product selection based on their environment. These issues involve *full lifetime management* – managing all stages involved in using an OSS product for its entire lifetime, including licenses and the companies available competence with a product – , *risk evaluation* [17] – evaluating the risk related to selecting an OSS product, especially "infrastructure" components –, and *balancing responsibility* – by balancing the responsibility for the selection of an OSS product between organization and individual; this aims at preserving the individuals enthusiasm and initiatives, and reducing the risk for bad selections through organization control of the selection.

Integration – There are two concerns among the COSI partner companies that have an influence on their integration of OSS products. First, one partner chose to *wrap OSS products* before integration. In order to have a stable version of their own software this partner decided to make it independent of the actual OSS products and make it possible to replace the OSS products. Second, another partner designs an *architecture of OSS products* that their own product is using.

Outsourcing – By providing an OSS content management system (CMS) one of the companies in the project is offering an extendable platform for its users and allow them to extend the platform with plug-ins. By catering for this development, the company outsources the development of plug-ins to the platform users, which in turn use the OSS platform as a building block in their software. These plug-ins are important for the products success and much of the customized functionality the users want is found in these plug-ins. Some of these plug-ins have also been incorporated into the final CMS. The company has thereby succeeded in outsourcing some of the development effort to the community while the users benefit from the plug-ins developed by others.

4.3 Developing OSS products

The last category covers the development of OSS products, i.e. releasing the software using an open source license and making it available for download [9]. Users of this OSS product are free to use and change the software as long as the license agreement is not violated. Other typical activities within this category are promotion of an OSS product and the establishment of an associated community.

Distribution of effort – Among the opportunities that are pursued by partners developing OSS product is distribution of effort. This can be achieved by establishing a community of volunteers, and motivating it to submit ideas, bug fixes and code to the product. In one case, a company establish a community to *recruit partners* that have an interest into the product and are willingly to take on their share of effort and responsibility to develop a product. Developing the product alone would be to expensive for the company. In another case, a company was starting a community of volunteers to receive contributions to *expand the product* with new functionality. In addition the community is regarded as a *recruitment opportunity* to hire new staff.

Standardization – One project partner has in one of its products implemented a much used standard for handling, storing, and transmitting information. To be able to test implementations of such a standard it is necessary to have a validation tool which checks the conformity of an implementation. The one who controls the validation tool controls, how the standard should be implemented. To establish this implementation as a de facto standard and to ensure that other implementations are compatible with their implementation the project partner has developed and released a validation tool. This tool has been released as open source to increase the diffusion of the validation tool and to strengthen the position of their implementation as the de facto standard.

Inter-company cooperation – Inter-company collaboration on a software product owned by one of the companies in the collaboration can be difficult. The participants in such collaboration may not want to give the ownership of their software to another company and, in fear of lawsuits it may be difficult to accept code from another company. To avoid challenges related to intellectual property and legal issues, one company released their product under an open source license and created an open source community with the other partners in the collaboration. This made it easier to get contributions from other companies while avoiding the legal conflicts related to accepting them.

Differentiating – In a saturated marked with many competitors, a software provider may *use the open source label to differentiate itself and its products* from the competitors and their products. There are several possible advantages of having a product licensed as an OSS product. Some examples are; community contributions may increase product quality, the product is available without any license fees, and the fact that OSS products give the customers vendor freedom and enable them to get support in case the provider goes out of business. These and other advantages can be used to make the company and their product stand out.

5 Conclusions

The above analysis of development with open source in the COSI R&D project shows that while there is a shift towards a more commercially viable form of OSS among the partner companies. However, the nature and form of this shift is not given. How they adopt OSS in their software development is shaped by factors internal to the companies, as well as the environment they are part of.

The implication of this is as follows. Rather than seeing companies' adoption of OSS in software development taking a singular form it needs to retain its multifaceted form in order for OSS to remain a commercially viable alternative. Companies must adapt open source to their particular organizational context and environments. However, providing three categories for developing with open source in an industrial context, can contribute to make practitioners more aware of the multifaceted nature of the phenomena. This awareness may help them to understand how they better can utilize open source in their organization.

Our analysis identifies three distinct categories for developing with open source in an industry context, based on our experience from the COSI project. The different categories show how OSS is used in different ways and for different purposes. The way OSS used by the companies is thus also shaped by the opportunities that are perceived and pursued. Other classifications of how OSS is used in an industrial context exist and highlight other aspects than the one highlighted by this paper. One such example uses the following three categories: using existing OSS, contributing to existing OSS projects, and releasing proprietary software under an OSS license [1].

The categories identified by this work are not exclusive in the way that a company only uses OSS in one way. Based on our experience from the COSI project, we find it likely that companies that develop with OSS products also will use OSS tools and practices, but not necessarily vice versa. The same holds for companies developing OSS products, where it is most likely that they also develop with OSS products, tools and practices. While having performed the first two steps of grounded theory, we have not performed the third step, selective coding [27]. The purpose of selective coding is to refine the developed theory. Doing selective coding to establish the relationship between our categories will be future work.

Our sample is a convenience sample of the 13 companies in the COSI project. The theory we build is grounded in and limited to this sample. There is no saturation of subcategories and new aspects of using open source in an industrial context will emerge both from our sample and when we expand our sample of companies in our future work.

However, our results support for our claim that open source is not a singular phenomena and that companies have to match the fluidity of the term with the activities of their organization to utilize open source in their organizations.

References

[1] O. Alexy and J. Henkel. Promoting the penguin: Who is advocating open source software in commercial settings? In G. T. Solomon, editor, *Proceedings of the Sixty-Sixth Annual Meeting of the Academy of Management (CD)*, 2007.

[2] B. Behlendorf. Open Source as a Business Strategy. In C. DiBona, S. Ockman, and M. Stone, editors, *Open Sources: Voices from the Open Source Revolution*, pages 149–170. O'Reilly & Associates, Sebastapol, CA, 1999.

[3] 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. In M. Scotto and G. Succi, editors, *Proceedings of the First International Conference on Open Source Systems (OSS'2005)*, pages 37–43, 2005.

[4] A. Bonaccorsi, S. Giannangeli, and C. Rossi. Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management Science*, 52(7):1085–1098, July 2006.

[5] A. Bonaccorsi, D. Lorenzi, M. Merito, and C. Rossi. Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research. In *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*, 2007.

[6] D. Cruz, T. Wieland, and A. Ziegler. Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis. *Software Process: Improvement and Practice*, 11(2):107–122, March-April 2006.

[7] J. Dinkelacker, P. K. Garg, R. Miller, and D. Nelson. Progressive Open Source. In *Proceedings of the 24rd International Conference on Software Engineering (ICSE'2002)*, 2002.

[8] M. Fink. *The Business and Economics of Linux and Open Source*. Prentice Hall, New Jersey, 2002.

[9] B. Fitzgerald. The Transformation of Open Source Software. *MIS Quarterly*, 30(2):587–598, 2006.

[10] B. Fitzgerald and T. Kenny. Developing an Information Systems Infrastructure with Open Source Software. *IEEE Software*, 21(1):50–55, January-February 2004.

[11] C. Gacek and B. Arief. The Many Meanings of Open Source. *IEEE Software*, 21(1):34–40, January-February 2004.

[12] Ø. Hauge, C.-F. Sørensen, and R. Conradi. Adoption of Open Source in the Software Industry. In B. Russo, E. Damiani, S. A. Hissam, B. Lundell, and G. Succi, editors, *Open Source Development Communities and Quality Working Group 2.3 on Open Source Software*, volume 275 of *IFIP International Federation for Information Processing*, pages 211–222. Springer, 2008.

[13] F. Hecker. Setting Up Shop: The Business of Open-Source Software. *IEEE Software*, 16(1):45–51, January-February 1999.

[14] J. Henkel. Selective Revealing in Open Innovation Processes: The Case of Embedded Linux. *Research Policy*, 35(7):953–969, September 2006.

[15] S. Krishnamurthy. An Analysis of Open Source Business Models. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 279–296. MIT Press, Cambridge, Massachusetts, 2005.

[16] K. R. Lakhani and R. G. Wolf. Why hackers do what they do: Underdstanding motivation and effort in free/open source software projects. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, editors, *Perspectives on Free and Open Source Software*. MIT Press, 2005.

[17] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. P. N. Slyngstad, and M. Morisio. A State-of-the-Practice Survey on Risk Management in Development with Off-The-Shelf Software Component. *IEEE Transaction on Software Engineering*, 34(2):271–286, February 2008.

[18] B. Lundell, B. Lings, and E. Lindqvist. Perceptions and Uptake of Open Source in Swedish Organisations. In E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, and G. Succi, editors, *Open Source Systems, IFIP Working Group 2.13 Foundation on Open Source Software*, volume 2003, pages 155–163. Springer, 2006.

[19] T. Madanmohan and R. De'. Open Source Reuse in Commercial Firms. *IEEE Software*, 21(6):62–69, November-December 2004.

[20] K. Martin and B. Hoffman. An Open Source Approach to Developing Software in a Small Organization. *IEEE Software*, 24(1):46–53, January/February 2007.

[21] Z. Obrenovic and D. Gaševic. Open Source Software: All You Do Is Put It Together. *IEEE Software*, 24(5):86–95, September-October 2007.

[22] T. Østerlie. Improved community models. Deliverable D1.2.2. To be made available for download from `http:`

//www.itea-cosi.org/modules/wikimod/index.php?page=ProjectDeliverables, October 2007.

[23] T. Østerlie and L. Jaccheri. A Critical Review of Software Engineering Research on Open Source Software Development. In *Proceeding of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, Gdansk, Poland, June 5, 2007*, 2007.

[24] B. Perens. The Open Source Definition. In C. DiBona, S. Ockman, and M. Stone, editors, *Open Sources: Voices from the Open Source Revolution*. O'Reilly & Associates, Sebastapol, CA, 1999.

[25] E. Raymond. *The Cathedral & The Bazar - Musings On Linux And Open Source By An Accidental Revolutionary*. O'Reilly, revised edition edition, 2001.

[26] W. Scacchi. Free/Open Source Software Development: Recent Research Results and Methods. In M. V. Zelkowitz, editor, *Advances in Computers*, volume 69, pages 243–269. Academic Press, 2007.

[27] A. C. Strauss and J. M. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory (Second Edition (6 Nov 1998))*. Sage Publications, Inc, 1998.

[28] E. B. Swanson and N. C. Ramiler. The Organizing Vision in Information Systems Innovation. *Organization Science*, 8(5):458–474, September - October 1997.

[29] A. M. Szczepanske, M. Bergquist, and J. Ljungberg. High Noon at OS Corral: Duels and Shoot-Outs in Open Source Discourse. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, editors, *Perspectives on Free and Open Source Software*, pages 431–446. MIT Press, Cambridge, Massachusetts, 2005.

[30] G. von Krogh and E. von Hippel. The Promise of Research on Open Source Software. *Management Science*, 52(7):975–983, July 2006.

[31] A. I. Wasserman and E. Capra. Evaluating Software Engineering Processes in Commercial and Community Open Source Projects. In *Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*, page 5, Washington, DC, USA, 2007. IEEE Computer Society.

[32] J. West and S. O'Mahony. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, page 196c, 2005.

[33] L. Yu. Indirectly predicting the maintenance effort of open-source software: Research Articles. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(5):311–332, 2006.

[34] S. Ziemer and T. Østerlie. Improved heterogeneous process models, Deliverable D2.1.3. To be made available for download from http://www.itea-cosi.org/modules/wikimod/index.php?page=ProjectDeliverables, October 2007.

PAPER 4

Øyvind Hauge, Thomas Østerlie, Carl-Fredrik Sørensen, and Marinela Gerea. *An Empirical Study on Selection of Open Source Software - Preliminary Results*. In Andrea Capiluppi and Gregorio Robles, editors, Proceedings of the ICSE 2009 Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS 2009), May 18th, Vancouver, Canada, pages 42-47, 2009. IEEE Computer Society.

An Empirical Study on Selection of Open Source Software - Preliminary Results

Øyvind Hauge, Thomas Østerlie, Carl-Fredrik Sørensen, and Marinela Gerea
Norwegian University of Technology and Science
{oyvind.hauge, thomas.osterlie, carl.fredrik.sorensen, marinela.gerea}@idi.ntnu.no

Abstract

Growing attention on component-based development has inspired the development of several normative methods for selection of software components. Despite these efforts, empirical studies show only minor adoption of such methods. To understand how research can contribute to improving the selection of components we interviewed developers from 16 Norwegian software companies which integrate Open Source Software (OSS) components into their systems. We find that the selection of OSS components has a situational nature where project specific properties significantly constrain the selection's outcome, and that developers employ a 'first fit' rather than 'best fit' approach when selecting OSS components. This could explain the limited adoption of normative selection approaches and general evaluation schemas. Moreover, it motivates a shift from developing such methods and schemas towards understanding the situational nature of software selection.

1 Introduction

With an increased attention on component-based development the past decades, companies have widely adopted open source software (OSS). With the view that using the right software is critical to project success [31], software engineering researchers have focused their attention on developing normative methods for selecting OSS components [1, 20, 24, 27]. While successful applications of such methods have been reported, research shows that component selection in practice is ad hoc and developer dependent [16, 19, 22, 28]. Beyond sweeping statements about the prevalence of ad hoc and developer dependent selection methods, little is known about what software developers actually do when selecting OSS components.

To better understand if and in what ways researchers may contribute to improving how developers select OSS components in practice, we have conducted interviews with developers in 16 Norwegian software companies integrating OSS into their products. Based on these interviews we find that:

- Project specific constraints are much more decisive in the selection of OSS components than the general evaluation criteria suggested by existing evaluation schema.
- Software developers employ the principle of 'first fit' as the principle of evaluation, whereas existing research on evaluation and selection methods employs 'best fit'. Rather than identifying a set of components to evaluate, software developers evaluate individual OSS components sequentially. Knowledge gained in rejecting one component is fed back as new evaluation criteria in the evaluation of the next.

As such, we contribute to existing research on OSS component evaluation in practice with an understanding of the situational and contingent nature of OSS selection. Our contribution motivates a shift in research from developing generalized schemas for OSS component evaluation, towards an appreciation of the situational and contingent nature of software evaluation.

2 Related Literature

2.1 Normative Selection Methods

Several initiatives have proposed a variety of normative approaches suggesting how selection of components-off-the-shelf (COTS) *should be done*. Most of these methods focus on identifying the requirements to the components, defining evaluation criteria based on these requirements, and comparing candidate components using weighted evaluation matrices. An overview of the eighteen most significant COTS selection methods aggregates these approaches into a five-step general COTS selection process [20]. This process has a few properties which are worth noticing; the requirements are expected to be defined and weighted up front, it is a 'best fit' formal or mathematical competition between several likely candidate components, and the identification of these components is basically ignored.

The availability of the Internet as a marketplace for components and the wide adoption of OSS have introduced new challenges for selection of software components. SourceForge, other general and domain specific software repositories, different software foundations and individual OSS providers offer an abundance of OSS components. Getting an overview of these resources is a challenging task. Moreover, OSS component are not usually backed by marketing campaigns, and they are not pushed towards software developers by a provider. This makes identifying and evaluating OSS components even more challenging. As a reaction to these challenges and as a continuance of the research done on COTS, several methods, frameworks and evaluation schema for OSS have been put forward like the Open Business Readiness Rating (OpenBRR), Open Source Maturity Model, Qualification and Selection of Open Source (QSOS) and for instance [4, 6, 7, 17, 27]. These normative approaches still emphasize defining and weighting a set of requirements before comparing two or more components using formal evaluation schemas consisting of extensive lists of reusable evaluation criteria.

Only a limited[1] number of empirical studies on selection of OSS components have been performed [16]. However, both studies on COTS and OSS conclude that even though there are successful applications of normative selection methods for both COTS and OSS, such methods are rarely applied in practice [16, 19, 22, 28]. Moreover, these studies conclude that practitioners use ad hoc, manual, and developer dependent methods for selection of components. These ad hoc methods rarely take advantage of pre-defined selection processes and formal mathematical evaluations.

The literature presents various possible explanations for this, none of which are particularly satisfying. First, the literature mentions problems with the methods like missing operational descriptions on how to use them [17], overlapping or missing evaluation criteria [7], missing match between requirements and evaluation criteria [14], and missing context sensitivity [7]. Second, it mentions that practitioners think it is impractical to perform complete evaluations in terms of time and cost [10]. Third, it mentions issues related to the situation in which the selection is performed. These are issues like missing information needed to satisfy complex evaluation criteria [3], availability of only one or a few candidate components, influence of a strong relationship with one provider, and selection of components during all stages of a project rather than just a specific component-selection stage [16]. Cabano et al. acknowledge the need for methods which are sensitive to the context in which the selection is performed but address this with an approach similar to existing methods [4].

2.2 Identification and Evaluation of OSS

The way developers identify components and develop evaluation criteria influences the outcome of a selection process. Empirical studies on selection of OSS are as mentioned generally missing and it is therefore a bit unclear what the manual selection methods comprise. Conferences, literature reviews, training, and communication with vendors are mentioned as common ways of identifying commercial components [13, 29]. This is relevant for OSS but much has changed with the availability of the Internet as a marketplace for components. Consequently, Internet searches are described as one of the most important methods for identifying OSS components [5, 22]. These searches are primarily executed through search engines but also through project hosting sites like SourceForge, code specific search engines like Google Code and to some extent social tagging sites like delicious [30]. Familiarity and previous experience is next to Internet searches mentioned as an important source of components [2, 15].

The evaluation of OSS components and the development of evaluation criteria have also attracted limited attention in empirical studies. However, basic developer dependent rules of thumbs like assessing the vitality of community, listen to the experience of others, and search for information in mailing lists, forums and so on, are observed [19]. Respondents in another study said that wide adoption of an OSS component could be a substitute for run-time tests [18].

3 Method

The setting of this study is the Norwegian software industry which consists of about 13000 active companies. Most of these are small, and only 1300 of them have more than five employees [26]. In 2005, this industry had about 36 500 employees and a turnover of more than €6 billion [26]. OSS is widely used and close to 50% of the Norwegian software industry integrate OSS components into their software products [12].

The interviews included here were conducted in the context of two Masters Theses focusing on selection of OSS components [9] and the use of OSS in the software industry [11]. The interviewees were selected from consultancy companies, software houses, or internal software development departments in large organizations. Table 1 gives an overview of the respondents' employers. However, Greek letters are used to anonymize the company names. The first thesis contains 45 minutes long interviews with one developer from all of the 16 companies in Table 1 but Alpha and Lambda. The second thesis contains two hour long interviews with two developers in the following companies Alpha, Epsilon, and Kappa, and with one developer in Lambda. The short interviews were mainly conducted by phone while the longer ones were done face to face.

All but one of the interviews were recorded and transcribed. The data was extensively analyzed through listening to the recordings, reading through the transcriptions and

[1]We ignore one frequently cited paper due to the paper's controversy related to copyright infringement.

No. of employees	**Consultancy company**	**Software house**	**Internal development**
< 10	Kappa	Gamma, Theta	
10 to 24	Eta	Delta	
25 to 49	Upsilon		Iota
50 to 99	Beta, Lambda	Epsilon	Sigma
> 100	Alpha, Rho, Tau, Xi	Zeta	

Table 1. The Respondents' Employers

field notes, and coding the material according to topics.

4 Results

4.1 The Use of Normative Methods

The use of normative selection methods as defined by the research community like OpenBRR, QSOS and so on was totally absent in our sample. The development was ad hoc and informal developer dependent selection was the norm throughout all of the companies. As one developer respondent: *"I have not read about any formal processes. It [selection] is done in an ad hoc manner"* [Iota]. The informal selection was primarily based on previous experience, monitoring of the OSS community, Internet searches and recommendations from people in the developers' social and on-line networks.

We did, however, observe some use of company specific processes related to important components within some of the larger companies in the sample. These companies had formalized a few activities and evaluation criteria which should be followed and one developer said that *"We have a formal process in [Xi] when integrating big components because we are developing software which can used in other parts of [Xi]"* [Xi]. A few companies had created their own selection processes but these were in general quite informal and as put by another developer *"We have a process for selection of OSS components but it is not a formal one with a specific name"* [Upsilon].

4.2 Identification of OSS Components

The experience the project members have with components is perhaps the most important, decisive and commonly used source of OSS components. If a project member has experience with a suitable component, they often leapfrog the whole selection process and starts using the component right away. *"It's usually a matter of using people's experience and knowledge about OSS components"* [Beta].

Monitoring the OSS community is another way of pro-actively identifying components and many developers maintain an awareness of available components by keeping an eye on the OSS community. *"We follow a lot of forums, news groups and stuff like that to monitor the areas of interest"* [Upsilon]. In addition to forums and news groups they subscribe to mailing lists and news letters, and read OSS related news sites. The awareness created by monitoring the OSS community is useful when they need a component. The monitoring is often done as part of the developer's own interest but also as part of the company strategy. *"A framework team actually does this job [selection of components]. They are Java developers that are up to date in the Java community and know what is available"* [Tau].

Unstructured searches, primarily on the Internet, is probably the most common way of finding components if no one in the project has any experience with or knowledge of any components, one developer said that *"We google what we need or we go to some repository"* [Theta]. Components may be identified through search engines like Google, general software repositories like SourceForge and language or domain specific repositories and sites like CPAN, TheServerSide and so on. Developers use such searches particularly when they have specific problems which need a particular kind of functionality.

Recommendations from a developer's social network is also an important source for identification of OSS components. Moreover, recommendations may also be found through the Internet as developers frequently read references and experience reports from developers with similar needs or particular experience with a component. One respondent said that *"We select components people are talking about, by reading articles on certain web sites"* [Beta]. It is also common to identify components through seeing them in use somewhere else, for instance in another commercial or OSS product. *"A major project like JBOSS may be using for instance a database like Hypersonic ... so we know that it is a good recommendation"* [Delta]. A developer may discover such components through monitoring the OSS community or by investigating which components specific software systems consist of.

4.3 Evaluation of OSS Components

The open nature of OSS communities allows simple evaluation of an OSS component's web site, documentation, license, release frequency, number of bugs, mailing lists, forums and so on. This can be done in a relatively short time and it educates the developer on the component and its capabilities but also on the problem domain in general.

Despite the fact that information on new components is easily available, previous experience is also the most prominent source of confidence in a component. It is also the first place developers look when they wants to evaluate a component. As one developer respondent, *"If you have used the tool or the component before ... you know it works"* [Gamma]. If developers have positive experience with a component they will easily select it again. No extensive selection process is needed, no training is necessary and no new technology has to be adopted. *"If we have a component we try to stick to this one until we decide that this is not good enough anymore"* [Tau].

The experience other people have with a specific type of components gives clear indications of the capabilities of the available components. Developers are therefore actively searching for and reviewing feedback from people in similar situations. The experience of other people can help identifying components and contributing to building confidence in a specific component. These references are often found on the Internet through general search engines, the communities of the specific OSS components, domain or technology specific sites, and technology blogs. By reading and listening to the experiences others have it is often quite easy to decide whether to reject a component, continue evaluating it, or even start using it. *"If we can say, by looking at the Internet or the references that ...this is a component that is used in other places ... we do not need to have ... a very serious evaluation"* [Tau]. However, most companies do not want to be the first one to try a new component. One developer said that *"If there is very little information about this component if we cannot find it used anywhere else ... then we are quite skeptical that this is good for us in the long run"* [Tau]. Companies prefer respected components with a proven track record and they select components which have been successfully applied in other commercial or OSS projects. *"It is important for us that we find components that have a reputation"* [Kappa].

The easiest way to gain experience with a component is of course to download it and create a small prototype. Prototyping is therefore the most common way to test and get acquainted with new components. *"If we do not know the component beforehand we just try to make a prototype"* [Beta]. A test integration can expose the component and enable the developer to assess whether the component provides the necessary functionality and performs as expected.

5 Discussion

5.1 The Situational Nature of Selection

The previous section describes how software developers identify OSS components by using their experience, monitoring the OSS community, reviewing recommendations, and performing Internet searches. It also describes how developers evaluate OSS components through using their experience, reviewing information available on the Internet, and developing prototypes. Based on these observations, we highlight two aspects of OSS selection which have a fundamental influence on the adoption of both normative selection methods and general evaluation schemas used in formal comparison of components; (1) *the situated nature of OSS selection* and (2) how developers employ *'first fit' rather than 'best fit' selection* of OSS components.

The situational nature [25] of OSS selection is evident as selection of OSS components is always performed in a situation consiting of a developer and a customer, their strategies, technologies, infrastructures and more than anything their employees. These elements create a situation in which the selection is performed and they significantly constraints the outcome of the selection. Component selection was in most cases left up to the individual developers who searched for and evaluated components using their distinct experience, skills, and preferences. These developers work within companies which often focus on one or a few technologies, and with customers which have a certain infrastructure and personnel skilled to manage this infrastructure. Furthermore, there may be project specific properties like an existing architecture, budget constraints, and constraints on which OSS licenses which could be used. All these properties, which are specific to the situation in which the selection is performed, significantly restrict the solution space and thereby the number of possible OSS components.

The 'first fit' selection is seen when developers start searching for suitable components based on their experience, and social and online networks. When identifying an OSS component which could solve the problem it is very easy for them to download and test it through a prototype. If the component solves the developers' current problem they would normally not see any need to look for other components. The developers then select the first component which solves the problem in a satisfactory way making the selection a 'first fit' rather than a 'best fit'. If the first component does not solve the problem they sequentially broaden the search to identify new components. At the same time as the identification and evaluation of new components continues, the developers educate themselves on the problem they are trying to solve. Knowledge gained from the identification and evaluation of one component is fed into the development of evaluation criteria for the next component.

These two observations have implications for the use of normative selection methods and general evaluation schemas, we believe the situated nature of OSS selection and the 'first fit' rather than 'best fit' selection explains why such selection methods and evaluation schemas see only limited adoption. First, the properties specific to the situation of the individual development projects are far more important than the evaluation criteria proposed in general OSS evaluation schemas. The fact that a few project specific constraints are so much more decisive in the selection of

OSS components, makes using general pre-defined evaluation schemas impractical at best. The need for greater sensitivity to the situation where the selection is taking place is also observed by others but not necessarily reflected in proposed evaluation methods [7]. Second, the identification and evaluation of components is actually done sequentially in short iterations. The criteria used to evaluate OSS components are gradually developed during the identification and evaluation of components, rather than before the selection starts. This implies that it is difficult to use normative methods because many of these methods suggest that the final evaluation criteria should be defined and weighted before the selection starts.

Similar observations about both the iterative nature of development of evaluation criteria [21] and the use of ad hoc selection processes [15, 19] have been made by others. It is therefore quite surprising that the research community has reflected so little over the influence this has on software selection and the use of normative selection methods and general evaluation schemas.

5.2 Limitations and Future Work

This research has mainly focused on small companies and the selection of components which should fit within a technological framework. Software companies may use other approaches when evaluating and selecting the fundamental technological platform(s) on which they want to base their business. While the selection of fundamental technology is important, companies select the components which should be integrated into this technology much more frequently. To aid software companies it is therefore valuable to understand the choices they frequently make and the rational behind these decisions.

We observed some signs of process formality related to important components in some of the companies. None of the software systems which were discussed in the interviews were life or mission critical systems. We could expect that the criticality of the requirements would increase in such situations and thus also the rigorousness of the process.

The interviews included in this study have been performed in different contexts with slightly different purposes and it is therefore difficult to discuss the validity of these results. Nevertheless, the observations reported here are in line with other empirical studies on the selection of OSS and COTS [5, 15, 16, 19, 30]. Even though this study focused on the selection of OSS the results may also be valid for COTS selection. However, this is input for future studies.

To verify the empirical results and to investigate some of the issues discussed above, we are performing further studies on how software developer identify and select OSS components, the resources they use while doing this, and the availability of relevant information in the Internet [2].

While the respondents reported only minor problems with selection of OSS components, ad hoc selection has drawbacks. These drawback should be further explored in light of for instance decision making theory to better understand the trade-offs in ad hoc selection.

6 Conclusion

Normative selection methods and general evaluation schemas may be valuable to increasing practitioners' awareness of issues related to OSS components and their communities. However, neither have seen significant adoption. We suggest that this is caused by two key characteristics of how developers identify, evaluate and select components. First, component selection is always taking place in a situation where constraints specific to this sitation are much more important than the criteria proposed in general evaluation schemas. Second, software developers employ the principle of 'first fit' rather than 'best fit' which is proposed by most normative selection methods. Software developers evaluate individual OSS components sequentially rather than first identifying a set of components before evaluating them. New or refined evaluation criteria are added as knowledge gained in evaluating one component, is fed back into the evaluation.

Researchers should therefore focus on understanding the situated nature of the OSS selection and the development of evaluation criteria which are sensitive to the situation, rather than continue the development of normative selection methods and general evaluation schemas meant to fit any situation. To succeed at this it is important to understand how practitioners actually select OSS components.

OSS providers could make information about their components like feature lists, future plans, known issues and dependencies, documentation, tutorials, release cycle and so on easily available to help software developers in the selection of OSS components. Furthermore, OSS providers should encourage their users to share their experience, and facilitate this sharing. Simple measures like this would simplify the evaluation of their software and it would most likely contribute to increasing the adoption of OSS.

References

[1] C. A. Ardagna, E. Damiani, and F. Frati. FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments. In Feller et al. [8], pages 3–16.

[2] C. P. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, and K. S. Velle. Challenges of the Open Source Component Marketplace in the Industry. In *FORTHCOMING OSS2009 Proceedings of the Fifth International Conference on Open Source Systems*. Springer, 2009.

[3] M. F. Bertoa, J. M. Troya, and A. Vallecillo. A Survey on the Quality Information Provided by Software Component

Vendors. In *QAOOSE'03 Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 25–30, 2003.

[4] M. Cabano, C. Monti, and G. Piancastelli. Context-Dependent Evaluation Methodology for Open Source Software. In Feller et al. [8], pages 301–306.

[5] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, and C. Liu. An Empirical Study on Software Development with Open Source Components in the Chinese Software Industry. *Software Process: Improvement and Practice*, 13(1):89–100, 2008.

[6] D. Cruz, T. Wieland, and A. Ziegler. Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis. *Software Process: Improvement and Practice*, 11(2):107–122, 2006.

[7] J.-C. Deprez and S. Alexandre. Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS. In *PROFES'2008 Proceedings of the 9th International Conference on Product-Focused Software Process Improvement*, volume 5089/2008 of *Lecture Notes in Computer Science*, pages 189–203. Springer, June 2008.

[8] J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti, editors. *Open Source Development, Adoption and Innovation IFIP Working Group 2.13 on Open Source Software, June 11-14, 2007, Limerick, Ireland*, volume 234 of *IFIP International Federation for Information Processing*. Springer, 2007.

[9] M. Gerea. Selection of Open Source Components - A Qualitative Survey in Norwegian IT Industry. Master's thesis, Norwegian University of Science and Technology NTNU, 2007.

[10] I. Gorton, A. Liu, and P. Brebner. Rigorous Evaluation of COTS Middleware Technology. *Computer*, 36(3):50–55, 2003.

[11] Ø. Hauge. Open Source Software in Software Intensive Industry - A Survey. Master's thesis, Norwegian University of Science and Technology NTNU, 2007.

[12] Ø. Hauge, C.-F. Sørensen, and R. Conradi. Adoption of Open Source in the Software Industry. In Russo et al. [23], pages 211–222.

[13] D. Kunda and L. Brooks. Identifying and Classifying Processes (traditional and soft factors) that Support COTS Component Selection: a Case Study. *European Journal of Information Systems*, 9(4):226–234, Dec. 2000.

[14] G. A. Lewis and E. J. Morris. From System Requirements to COTS Evaluation Criteria. In *ICCBSS 2004 Proceedings of the Third International Conference on COTS-Based Software Systems*, volume Volume 2959/2004 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2004.

[15] J. Li, F. O. Bjørnson, R. Conradi, and V. B. Kampenes. An Empirical Study of Variations in COTS-based Software Development Processes in Norwegian IT Industry. In *METRICS '04: Proceedings of the Software Metrics, 10th International Symposium*, pages 72–83, Washington, DC, USA, 2004. IEEE Computer Society.

[16] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. P. N. Slyngstad, and M. Morisio. Development with off-the-shelf components: 10 facts. *IEEE Software*, 26(2):2–9, 2009.

[17] A. Majchrowski and J.-C. Deprez. An Operational Approach for Selecting Open Source Components in a Software Development Project. In *EuroSPI'2008 Proceedings of the 15th European Conference on Software Process Improvement*, volume 16 of *Communications in Computer and Information Science*, pages 176–188. Springer, Sept. 2008.

[18] P. Maki-Asiala and M. Matinlassi. Quality Assurance of Open Source Components: Integrator Point of View. In *COMPSAC '06 Proceedings of the 30th Annual International Computer Software and Applications Conference*, volume 2, pages 189–194, Los Alamitos, CA 90720-1314, 2006. IEEE Comuter Society.

[19] J. Merilinna and M. Matinlassi. State of the Art and Practice of OpenSource Component Integration. In *EUROMICRO'06 Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 170–177, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[20] A. Mohamed, G. Ruhe, and A. Eberlein. COTS Selection: Past, Present, and Future. In *ECBS '07 Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 103–114. IEEE Computer Society, Mar. 2007.

[21] C. Ncube and N. A. M. Maiden. Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm. In *Development Paradigm. International Workshop on Component-Based Software Engineering*, pages 1–12, 1999.

[22] J. S. Norris. Mission-critical Development with Open Source Software: Lessons Learned. *IEEE Software*, 21(1):42–49, 2004.

[23] B. Russo, E. Damiani, S. A. Hissam, B. Lundell, and G. Succi, editors. *Open Source Development Communities and Quality IFIP Working Group 2.13 on Open Source Software September 7-10, 2008, Milano, Italy*, volume 275 of *IFIP International Federation for Information Processing*. Springer, 2008.

[24] I. Samoladas, G. Gousios, D. Spinellis, and I. Stamelos. The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation . In Russo et al. [23], pages 237–248.

[25] L. A. Schuman. *Plans and Situated Actions : The problem of human-machine communication*. Cambridge University Press, 1987.

[26] SSB. StatBank Norway, 2007. Online, `http://statbank.ssb.no/statistikkbanken/`, accessed 2007-08-01.

[27] D. Taibi, L. Lavazza, and S. Morasca. OpenBQR: a framework for the assessment of OSS. In Feller et al. [8], pages 173–186.

[28] M. Torchiano and M. Morisio. Overlooked Aspects of COTS-Based Development. *IEEE Software*, 21(2):88–93, 2004.

[29] V. Tran and D.-B. Liu. A procurement-centric Model for Engineering Component-based Software Systems. In *Proceedings of the Fifth International Symposium on Assessment of Software Tools and Technologies*, pages 70–79, June 1997.

[30] M. Umarji, S. E. Sim, and C. Lopes. Archetypal Internet-Scale Source Code Searching. In Russo et al. [23], pages 257–263.

[31] H. Wang and C. Wang. Open Source Software Adoption: A Status Report. *IEEE Software*, 18(2):90–95, 2001.

PAPER 5

Claudia P. Ayala, **Øyvind Hauge**, Reidar Conradi, Xavier Franch, Jingyue Li, and Ketil Sandanger Velle. *Challenges of the Open Source Component Marketplace in the Industry.* In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, Proceedings of the 5th IFIP WG 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3-6, Skövde, Sweden, volume 299/2009 of IFIP, pages 213-224, 2009. Springer

Challenges of the Open Source Component Marketplace in the Industry

Claudia Ayala[1,2,*], Øyvind Hauge[1], Reidar Conradi[1], Xavier Franch[2], Jingyue Li[1] and Ketil Sandanger Velle[1]

[1]Norwegian University of Science and Technology (NTNU)
{oyvind.hauge, reidar.conradi, jingyue.li, ketilsan}@idi.ntnu.no
[2] Technical University of Catalunya (UPC)
{cayala, franch}@lsi.upc.edu

Abstract The reuse of Open Source Software components available on the Internet is playing a major role in the development of Component Based Software Systems. Nevertheless, the special nature of the OSS marketplace has taken the "classical" concept of software reuse based on centralized repositories to a completely different arena based on massive reuse over Internet. In this paper we provide an overview of the actual state of the OSS marketplace, and report preliminary findings about how companies interact with this marketplace to reuse OSS components. Such data was gathered from interviews in software companies in Spain and Norway. Based on these results we identify some challenges aimed to improve the industrial reuse of OSS components.

1. Introduction

Nowadays, Component-Based Software Development (CBSD) is considered the standard way of developing software systems [3]. The main motivation behind this is reusability as "reusing components avoid reinventing the wheel". This allows companies a faster technology adoption and innovation whilst reducing costs and time-to-market [18]. In particular, the existence of the Open Source Software (OSS) marketplace, consisting of tens of thousands of OSS components which are developed and freely available over the Internet, has greatly influenced the software reuse practices and the overall economy behind [8, 11].

In this huge diversity, one of the most influential activities in CBSD is the selection of components [7, 26]. Successful reuse of OSS components highly depends on being able to navigate in the OSS marketplace to identify and evaluate which component(s) may (best) fit the requirements [25]. In the industrial practice, the selection of OSS components is considered a highly risky activity as

*Her work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

companies are confronted with incredibly large amounts of diverse, partial, and ephemeral, information about OSS components. This information could be tacit and it is not always trustable. Therefore, it is not easy for companies to fully exploit the benefits of reusing OSS components [18].

To support the industry in this crucial task, we need to understand the industrial needs for selecting components and the state of the OSS marketplace as the place where components are found. To do so, we investigated: (1) the elements that constitute the OSS marketplace, and the current state of such infrastructure from the point of view of industrial reusers. (2) How reusers interact with the OSS marketplace to select OSS components. First, we assessed a variety of Internet resources and identified their main characteristics and utility. Second, we performed semi-structured interviews in companies in Spain and Norway. Based on these investigations we identify challenges aimed to improve the industrial reuse of OSS components.

2. State-of-the-Art

Systematic software reuse is an engineering strategy proposed to increase productivity and software quality, and lead to economic benefits [9]. It is based on the premise that for reuse to be effective, a proper infrastructure enabling reusers to find and understand the components that best fit their needs should exist [21]. Although systematic software reuse has been an active research area for more than a decade [9], the special nature of OSS has taken the original concept of systematic reuse based on centralized repositories into a completely different arena based on massive reuse over Internet. Therefore, the need of new approaches for effectively finding and understanding components has been widely recognized [2, 9].

Although research and practitioners have proposed a diverse set of methods and evaluation guidelines for supporting components selection (e.g., [16, 17, 19]), these proposals have not been widely adopted in the industrial practice [15]. The literature presents various possible explanations for this: that the proposed methods are failing to deal with identification of components and information for evaluating them [4, 6], and that it is impractical to perform complete evaluations in terms of time and cost [12].

The Internet as the infrastructure of the OSS marketplace constitute as we see an important part of both identification and evaluation of OSS components [25]. However, we know little about this OSS marketplace and how practitioners search, evaluate and choose components from this marketplace.

In order to envisage effective solutions for enabling successful reuse of OSS components, further empirical studies are needed to better understand how researchers may contribute to developing the marketplace and improving how reusers select OSS components. The work presented in this paper is trying to contribute to this fact.

3. Elements of the OSS Marketplace

The OSS reuse environment greatly differs to the “classical” reuse environment based on centralized repositories [21]. In this section, we describe the high-level elements that constitute the OSS marketplace in order to understand the new required needs for improving OSS components reusability.

Although the word marketplace may have different connotations [5, 14, 24], in the context of this paper, as OSS marketplace we refer to the self-organizing virtual place on the Internet that includes the exchange interactions between reusers and providers of OSS components as well as the actions of other actors that facilitate or promote such transactions.

The OSS marketplace implies diverse elements, relationships and interactions over internet: *Providers* offer OSS components through their own websites. *Reusers* use a *search mechanism* or *Intermediary* services to find and select components, whilst *Promoters* foster the OSS movement.

Although it is not our intention to further describe the marketplace interactions, as it implies a broad line of research, we briefly describe its main elements and provide some examples of the actors that actually cover these elements.

- **Reuser:** Refers to developers that search in the marketplace for components that may cover certain functionality. Such component(s) are intended to be integrated in a software system. The work described in this paper focus on the problems reusers face in their OSS selection practices.
- **Provider:** Refers to OSS communities or companies which develop and release OSS components. Currently there are thousands of OSS communities and therefore thousands of potential component providers. Examples are moodle.org, linux.org, eclipse.org, FreMed.org, Openmrs.org etc.
- **Search Mechanism:** Refers to the mechanism that allows navigation through the marketplace. General-purpose tools to navigate through the Internet as *Google* exist. But some specialized tools as *Google Code Search* or *Kooders* have been designed for indexing various open source repositories and to allow more focused component searches on the web.
- **Promoter:** Refers to individuals and organizations which main aim is to foster the OSS movement. Examples are the Open Source Technology Group (OSTG), Free Software Foundation (FSF), Apache Foundation, and personal blogs with useful resources. Practical research efforts from academia and/or industry can be also found, an example is the CeBASE repository that provides a "lessons learned" database.
- **Intermediary:** Refers to profit or non-profit organizations or individuals that index and/or distribute OSS components or other related products and services. Examples are companies selling support around certain components or domains as Forrester or Gartner; and *General-oriented or Domain specific portals* as SourceForge or TheServerSide respectively.

4. The Study

The aim of the study is to establish an empirical foundation of the challenges of the OSS marketplace when dealing with the needs of industrial *Reusers*. The study consisted of two parts: a) investigation of the actual state of the marketplace, b) investigation of how *Reusers* interact with the OSS marketplace to select OSS components.

4.1 Investigation of the OSS Marketplace

To better understand the state of the marketplace and the kind of resources it offers for supporting components selection, we further assessed more than 60 related sites and search mechanisms on the Internet. The elements of the marketplace introduced in Section 3 were identified throughout this analysis. The studied sites were identified from the answers of the respondents of the study described in Section 4.2, previous studies e.g. [10, 13, 15], research team's experience, and web investigation. Of course, we do not claim that we have reviewed all existing portals or search tools of the marketplace, indeed by the nature of the marketplace it is not realistic to think that this can be done. However, we think that the ones we have assessed are representative of the marketplace elements and their actual offerings. The focus of the assessment was on the factors affecting the selection of OSS components. This set of resources was analyzed between March and November 2008. Main results are briefly summarized in the context of Section 5.

4.2 Investigation of Reusers Interaction with the Marketplace

We performed an explorative study in Small and Medium Enterprises (SME). The study consisted of semi-structured interviews with people involved in CBSD using OSS components (i.e., *Reusers*). We asked about how they identify and evaluate OSS components, which resources they use and the problems they face with such resources.

With a basis in earlier studies e.g. [10, 13, 15], we developed and tested an interview guide. It focuses on the experience of industrial *Reusers* with a finished project in which one or more OSS components were used. The interview guide contained one part each about identification and evaluation, the use of internet and social resources in OSS selection, and finally demographic information. The interviews were performed in 5 companies from Spain (ES1-ES5) and 3 from Norway (NO1-NO3). See Table 1 for an overview of the respondents, their companies and the projects. We considered having data from different countries valuable to strengthen the external validity of the results.

Table 1. Some details of the study.

Id	Company Scope	No. Employees	Project	Experience with CBSD
ES1	Web monitoring SW	10	Web-statistics, Ruby, 1 person/month, part of larger system	2 years, MSc
ES2	HW sales, add-on SW development	150-200	CMS/e-commerce, PHP, 4 person/months	6 years, MSc
ES3	SW development/ consultancy	4	Web application, Java, 8 person/months	10 years, MSc
ES4	Organizational IT department	15	Web application, PHP 24 person/month	8 years, MSc
ES5	SW development/ consultancy	70-100	e-Business application 10 person/month	11 years, MSc
NO1	SW development/ consultancy	20	Web search, Java, 2 person/months, part of larger system	11 years, MSc
NO2	SW development/ consultancy	200	Web application, Java, 6 person/years	4 years, MSc
NO3	SW house	12	Linguistic SW, .Net, 10 person/years	7 years, BSc

Each interview took around one hour, and was performed face to face by one or two researcher in the native language of the interviewees (Spanish or Norwegian). To establish the interviews' context and limitations, we began each interview by stating our motivation. We rigorously avoided suggesting any Internet resources during the interviews. Moreover, in cases when the interviewee did not remember the URL or location of a specific resource, they were asked to send us such information by e-mail. The resources mentioned by the interviewees were evaluated as described in Section 4.1. The semi-structured nature of the interview, allows to further inquiry in relevant areas and to get useful qualitative data. Each interview was recorded and transcribed. To perform the data analysis, the research team listened through the recorded interviews or read through the transcriptions. Then, as all authors speak English, we were able to analyze and discuss the obtained data in several meetings.

The following section relates the results obtained from these studies.

5. Resources Used to Select OSS Components in the Industry

In this section, we report our assessment of the existing resources in the marketplace and how the interviewees used these resources to select OSS components.

Scenarios are used to describe our findings, followed by discussions of some of the problems the *Reusers* face when using resources in the marketplace.

It is important to remark that the objective of the scenarios is to show how *Reusers* used existing resources and not to explain the process they follow in detail. The scenarios may therefore be overlapping.

5.1 Searching

The search process departs from the need to find a component that may cover certain functionality in the final system. Further assessment of components is performed in the context of the subsequent evaluation and decision activities.

5.1.1 Existing Resources for Performing OSS Searching

To enable navigation throughout the OSS marketplace, some *Search mechanisms* and a variety of portals issued by *Promoters* and/or *Intermediaries* exist (see Section 3). One of the main goals of these portals is to offer categorizations aimed to guide their users to find information, services and components themselves. From our assessment of several of these portals, we observed that understanding and use of the portal content is not an easy task, especially if the domain is absolutely unknown.

Furthermore, according to their topics, these portals range from general-oriented as SourceForge to domain-specific portals. Domains can be understood at different levels. For instance, domain technologies as presented in TheServerSide which is related to the Java technology or more specific ones as the health care domain as Openmrs.org. On the other hand, the collaborative and "open source" philosophy has also enabled the formation and explosion of open and collaborative portals. These are aimed to discuss and exchange experiences around specific domains. Examples are CMSMatrix and WikiMatrix in the content management system and wiki management system domains respectively.

To describe how interviewees used these resources, we identify three different situations described in terms of the following scenarios.

5.1.2 Scenario S1- *No Search is Required*

Reusers are aware of a component that may fit the functionality they are looking for, or someone (a colleague or member of the project team) has used and recommended the components. As one developers said *"if someone has experience [with a component] we normally select this one"* (ES2). *Reusers* do in other words quite often select components based on previous experience, even without looking for other candidates.

5.1.2.1 Problems Related to Scenario S1

Although no problems with the use of resources were found in the context of this scenario (i.e., it does not imply any search), it highlights the influence previous experiences have on the outcome of the whole selection process.

5.1.3 Scenario S2 – *Regular Monitoring of the Marketplace*

Experienced *Reusers* tend to be familiar with the domains they work with, and they typically monitor the marketplace to inform themselves of technologies and trends (even before they have a specific need). As a result, the component searching process is influenced by this familiarity, and it turns out to be a gradual process rather than a momentarily one.

In this context, when *Reusers* are looking for components, they review the portals they already know to see what components are being used by the community. *"We know or have bookmarks of several portals we usually review to be aware of the components and technologies that are being used by the community. In particular, the most representative portal for our work is TheServerSide"* (ES3). Portals are however not the only kind of resources used in this monitoring and *Reusers* benefit from a range of information sources. One respondent said that they read *"different private blogs where one basically picks up trends"*(NO2), another preferred information from printed source. "*We buy a lot of magazines and typically O´Really Books"* (NO3).

In addition, *Reusers* will tend to come back to the ones he knows and trust. "*There are a lot of portals about OSS and technologies, but I tend to use the ones I usually follow and trust"* (ES3). Reading a variety of news sites and portals could be time consuming and one respondent said that *"I am more depending on the RSS feeds which I subscribe to. I do not actively read as much on the page [as before]"* (NO1).

5.1.3.1 Problems Related to Scenario S2

Reusers mentioned to be concerned about trust and contradictory information. *"Sometimes it is difficult to formulate an opinion from information contained in internet because some of the opinions are extremely contradictory, so it is not easy to decide if a component could be a candidate"* (ES2).

In general, *Reusers* considered that having more comprehensive search functionality and more flexible classification systems in portals could be valuable. Since search was used a lot it could simplify the identification of components.

5.1.4 Scenario S3 – *Open Search*

When *Reusers* do not have strong experience in a domain, they usually do not know where to find components that may cover the functionality they are looking for. In these cases, they mostly mentioned to use general search engines (Google was mentioned the most). *"When we do not have a clear idea of the kind of components that may cover the functionality we are looking for, we directly go to Google"* (ES2).

From our assessment of the marketplace we observed that some specialized search engines as Google Code Search or Koders exist, but none of the interviewees mentioned to use them. *"No, we do not know any specialized tool to find components, we always use Google"* (ES1).

Regarding the use of *intermediary* portals hosting hundreds of OSS components, a reuser said*"I know SourceForge, but in portals like this it is really difficult to navigate and find relevant components. It is better to find components in a specialized portal (i.e., domain-specific) and then go to SourceForge for download it"* (ES1)

5.1.4.1 Problems Related to Scenario S3

Reusers recognize that using general search engines as Google, the number of returned hits is frequently large and many of these hits are irrelevant. *"We know we will have a lot of irrelevant hits but anyway it is easier to arrive to the component and its community by Google"* (ES5). In addition, the accuracy of the search relies heavily on how well they are able to identify the exact terms to describe the functionality *"It is more than anything Google search tactics"* (ES1).

5.2 Evaluation and Decision

The evaluation and decision process implies the investigation of the features of the candidate components in order to choose the one(s) that (best) fit the system requirements.

5.2.1 Existing Resources for Performing OSS Evaluation and Decision

Our assessment of the existing resources to perform such tasks leads us to confirm that each existing portal describes components following its own classification and description model. There are in other words no standard for describing components. Thus, very different kinds of information are included: from technical documentation, newsletters, and articles to information coming from social collaborative features, as wikis, chats, blogs, and forums.

The following scenarios describe how *Reusers* use existing resources when they come to perform the component evaluation and decision activities.

5.2.2 Scenario E1- *Experience-Based Evaluation*

Reusers usually do not have enough time to evaluate components as much as they would like. As a result, they tend to use components they already know. *"We usually cannot evaluate a component as much as we would like because it requires time we usually do not have, so we tend to use only components we know or some colleagues have recommended"* (ES4).

In some cases they did not even considered other candidates that could fit better to the required task. *"We prefer to use a component we already know than assuming the risks of using a new one, even when the new component could perform better"* (ES4).

5.2.2.1 Problems Related to Scenario E1

Reusers recognize that the evaluation and decision process is not performed properly and it is mainly biased for personal experiences that narrow the exploitation of reuse.

We observed that the reuse of components is mainly seen in the form of fine-grained or commodity components that do not imply so much risk if they should be replaced.

5.2.3 Scenario E2 – *Searching for Information for Evaluating Components*

Although the formal documentation of the OSS component obtained from the *Provider* was considered important at the technical level, experience either personal or of others plays a crucial role when evaluating and deciding which component to use, as stated in Scenario E1.

If the *Reuser* does not have personal experience with a particular component he usually consults his personal network (i.e., colleagues) or uses Google to search for more information. *"Another things which we almost always do is to read opinions ... and examine a bit the experiences other people have."* (ES3). These searches are usually performed by using the name of the component as a keyword, in order to find other portals, forums or blogs which can provide more information. The most valuable information is: experience reports of successful implementations, possible problems with the use of the component and solutions to these problems. If required, *Reusers* may for instance pose questions in forums or on mailing lists as well.

5.2.3.1 Problems Related to Scenario E2

Existing portals are facing serious problems for making information available in a suitable way [6]. Some reusers mentioned to be unhappy with the completeness and quality of the documentation. One developer complained over this quality saying *"There are a lot [of OSS projects] which are not well documented"* (NO3). Improving this documentation could simplify the evaluation of the components. *"What would have made it [OSS selection] a lot easier was if more of those offering OSS libraries put a bit more work into the documentation because this is often insufficient"* (NO1). However, others mentioned to be quite happy with the documentation of other components. "*We have been using Hybernate for a long time and we are quite happy with the extensive documentation and community support we have"* (ES3). Assessing the interviewees answers and the characteristics of their projects we realized that the level of maturity of the community where the component come from, seem to have a significant impact on the quality of the documentation. Mature communities tend to provide better documentation.

The general perception of existing collaborative mechanisms for sharing knowledge as forums and wikis was really appreciated, but reusers mentioned to have problems with dealing with the subjective nature of opinions and unstructured information. *"There are a lot of subjective meanings out there"* (NO1). Indeed, existing portals do not offer advanced features that could help reasoning about and structuring scattered and subjective information. Furthermore, personalization features are also rare, and as basic as configuration of the appearance, payment personalization and RSS advertisement.

6. Discussion of Results

Although the benefits of having OSS components available are well-known, their successful reuse implies many challenges. Based on the empirical observations from the studies presented in the previous sections, we discuss the main issues that should be faced to improve OSS components selection.

- *The rapid changes and growing nature of the OSS marketplace.* New components and technologies are continuously offered, but also new and improved versions of existing components are frequently released. One straightforward example of the difficulty to deal with the changing nature of the marketplace can be seen in the evolution of mail servers products, which at present could also provide instant messaging or even video-conferencing facilities. So, to effectively classify components in order that they can be easily found by reusers is not trivial [2]. In this context, existing resources for enabling finding of components are facing some problems. Although industrial reusers do not really bother about searching, they are aware that the narrow character of their current searching processes and the influence of previous experiences hamper the fully exploitation of OSS reuse.

- *The lack of standards for describing OSS and huge of information diversity.* The lack of standards for describing OSS and the huge amounts of diverse, partial and subjective information about components makes it tough for reusers: first, to trust on the information and second, to perform and informed evaluation and decision. This contributes to the fact that decisions are mostly based on experience and limited knowledge of available components. Hence, the need for decision support mechanisms has been recognized [19]. Several research proposals recognize this problem and have proposed solutions that range from developing a general ontology [6, 23], to the use of semantic web technologies [1] and description logics. However, the real applicability of most of these proposals have resulted scarce [15, 19]. At this respect, personalization and recommendation functionality in portals are perceived as desirable by industrial reusers (e.g. users that selected this component also selected this…) as this could help them to get relevant information.

- *The influence of Experience on the selection process.* As our results confirm, experience either personal or of others play a critical role on the selection of OSS components. Although, the OSS spirit encourages the free and collaborative production and sharing of knowledge, there is a demanding need to effectively deal with the inherent subjectivity of the information. Reputation mechanisms as used in other business domains as ebay.com could be really valuable to deal with the subjectivity of diverse opinions.

Table 2 summarizes and relates the identified practical problems of *Reusers* when selecting OSS components and the challenges related to the elements of the marketplace.

Table 2. Summary of Reuser Problems and Associated Challenges

	Reuser Problem	**Marketplace Element**	**Challenge**
Searching	Difficulty to navigate through diversity of portals	Search Mechanism	Advanced and configurable search engines.
	Difficulty to find Components	Intermediary	More flexible classification schemas able to efficiently represent and deal with volatility and growing size of the marketplace.
Evaluation & Decision	Poor documentation quality	Provider	To improve components documentation
	Difficulty to find relevant information and deal with its subjectivity	Provider	More sophisticated knowledge portals with decision-making support
		All	Need of integral efforts for improvements based on *Reusers* needs.

7. Threads of Validity

The results presented here are preliminary and we are about to conduct further studies, including more interviews to verify our results and get more observations.

The companies in this study were selected by convenience and we had only limited control over the selection of the projects. However, the results presented here come from companies in several domains and of different size. The projects are also of different size and types, and the interviewees have different backgrounds, see Table 1. One limitation is that most interviewed companies were developing web applications. The web applications domain is normally not represented by critical systems and, it is a domain in which there are plenty of OSS components. Both facts have an impact on how components are selected. We could expect somewhat different results in critical systems and in domains where OSS components are rare.

Furthermore, the interview guide used was prepared based on previous experience with similar studies [10, 13, 15] and it was pretested both in Spain and in Norway. To make answering as easy as possible for the interviewees, the interviews were performed in their respective native tongues. Two members of the research team are native Spanish speakers and two are native Norwegian speakers, one of which also speaks Spanish. To avoid misunderstandings and to simplify the analysis and discussions of the interviews, we taped, transcribed and translated the interviews to English as all authors speak English.

8. Conclusions and Future Work

The study and the results presented in this paper are discussed from the perspective of industrial reusers of OSS components with a particular focus on their needs when performing OSS selection. The systematic software reuse theory is used as a background for the discussions of the needs. Traditional view of reuse as a centralized reuse databases is challenged by the Internet as a massive marketplace for OSS components.

The relevance of this study is twofold: First, it may serve as a solid basis for understanding the real needs of OSS industrial reusers when selecting components, and therefore to properly address research and industrial efforts from several arenas. Second, the challenges identified here could help to the OSS marketplace elements to contribute to maturing the marketplace. A more mature marketplace would allow reusers to fully exploit the benefits of OSS components and thereby contribute to increasing the adoption of OSS in the industry.

As future work, our goal is to further explore the industrial OSS selection by performing more interviews in both Norway and Spain. We expect to collect data that could help to describe how reusers and to understand how reusers can maximize the benefit of the OSS marketplace.

9. References

1. Ankolekar A, Herbsleb JD, Sycara K (2003) “Addressing Challenges to Open Source Collaboration With the Semantic Web”. In Feller J, Fitzgerald B, Hissam SA, Lakhani KR (Editors), Taking Stock of the Bazaar: 3rd Workshop on Open Source Software Engineering, pp 9-14.
2. Ayala CP (2008) Systematic Construction of Goal-Oriented COTS Taxonomies. PhD Thesis. Technical University of Catalunya.
3. Basili VR, Boehm BW (2001) COTS-based Systems Top 10 List. Computer, 34(5):91-95.
4. Bertoa M, Troya JM, Vallecillo A (2003) A Survey on the Quality Information Provided by Software Component Vendors. In QAOOSE'03 Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, pp. 25-30.
5. Brereton P, Linkman S, Thomas N, Bøegh J, De Panfilis S (2002) Software Components - Enabling a Mass Market. In STEP'2002: Proceedings of the 10th International

Workshop on Software Technology and Engineering Practice, pp. 169-176. IEEE Computer Society.

6. Cechich A, Requile-Romanczuk A, Aguirre J, Luzuriaga JM (2006) Trends on COTS Component Identification. In ICCBSS'2006: Proceedings of the Fifth International Conference on Commercial-Off-The-Shelf (COTS)-Based Software Systems, pp. 90-99. IEEE Computer Society.
7. Clark J, Clarke C, De Panfilis S, Granatella G, Predonzani P, Sillitti A, Succi G, Vernazza T (2004) Selecting components in large COTS repositories. Journal of Systems and Software, 73(2):323-331.
8. Fitzgerald B (2006) The Transformation of Open Source Software. MIS Quarterly, 30(3): 587-598.
9. Frakes WB, Kang K (2005) Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering, 31(7):529-536, 2005.
10. Gerea M (2007) Selection of Open Source Components - A Qualitative Survey in Norwegian IT Industry. Master's thesis, Norwegian University of Science and Technology.
11. Ghosh RA (2006) Study On the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communication Technologies (ICT) Sector in the EU. Technical report, UNU-MERIT.
12. Gorton I, Liu A, Brebner P (2003) Rigorous Evaluation of COTS Middleware Technology. Computer, 36(3):50-55.
13. Hauge Ø, Sørensen C-F, Conradi R (2008) Adoption of Open Source in the Software Industry. In Russo et al. [22], pp. 211-222.
14. Knudsen D (2003) B2B-Marketplace Value Creation, Conceptual Predictions and Empirical Findings. In NOFOMA'2003 Proceedings of the 15th Annual Conference for Nordic Researchers in Logistics, pp 318-331.
15. Li J, Conradi R, Bunse C, Torchiano M, Slyngstad OPN, Morisio M (2009) Development with Off-The-Shelf Components: 10 Facts. IEEE Software, March-April 2009.
16. Majchrowski M, Deprez J-C (2008) An Operational Approach for Selecting Open Source Components in a Software Development Project. In EuroSPI'2008 Proceedings of the 15th European Conference on Software Process Improvement. Springer - Volume 16 of Communications in Computer and Information Science, pp 176-188.
17. Merilinna J, 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, pp. 170-177. IEEE Computer Society.
18. Mohagheghi P, Conradi R (2008) An Empirical Investigation of Software Reuse Benefits in a Large Telecom Product. ACM Transactions of Software Engineering and Methodology, Vol. 17, No. 3, Article 13, 30 pages.
19. Mohamed A, Ruhe G, Eberlein A (2007) COTS Selection: Past, Present, and Future. In Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, pages 103-114. IEEE Computer Society.
20. Norris JS (2004) Mission-Critical Development with Open Source Software: Lessons Learned. IEEE Software, 21(1):42-49.
21. Prieto-Díaz R, Freeman P (1987) Classifying Software for Reusability. IEEE Software. 4(1):6-16.
22. Russo B, Damiani E, Hissam SA, Lundell B, Succi C (2008) (Editors) Open Source Development Communities and Quality Working Group 2.3 on Open Source Software. IFIP International Federation for Information Processing. Springer.
23. Simmons G, Dillon T (2006) Towards An Ontology for Open Source Software Development. In Damiani E, Fitzgerald B, Scacchi W, Scotto M (Editors), Open Source Systems IFIP Working Group 2.13 Foundation on Open Source Software, pp 65-75. Springer.

24. Ulkuniemi P, Seppanen V (2004) COTS Component Acquisition in An Emerging Market. IEEE Software, 21(6):76-82, 2004.
25. Umarji M, Elliott-Sim S, Lopes C (2208) Archetypal Internet-Scale Source Code Searching. In Russo et al. [22], pp. 257-263.
26. Vitharana P, Zahedi F, Jain H (2003) Design, Retrieval, and Assembly in Component-based Software Development. Communications of the ACM, 46(11):97-102.

PAPER 6

Øyvind Hauge and Sven Ziemer. *Providing Commercial Open Source Software: Lessons Learned.* In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3-6, Skövde, Sweden, volume 299/2009 of IFIP, pages 70-82, 2009. Springer.

Providing Commercial Open Source Software: Lessons Learned

Øyvind Hauge and Sven Ziemer

Norwegian University of Science and Technology

{oyvind.hauge|sven.ziemer}@idi.ntnu.no

Abstract Even though companies like Sun, IBM, MySQL and others have released several commercial Open Source Software (OSS) products, little evidence exist of how to successfully launch such products and establish a living community around them. This paper presents a case study from a small software company succeeding at establishing a business model and a vivid community around their own OSS products. Based on this case study, the paper presents lessons learned which could help other OSS providers.

1 Introduction

Open Source Software (OSS) development has become a serious source of revenue for the software industry [10, 13]. Large companies like Apple, IBM, Sun and others have released significant amounts of their software as OSS. Going open source can however be a significant change for a commercial organization [5]. Small and medium enterprises (SME) do not have the same resources as large companies to adapt to these changes. Yet, companies like JBOSS, MySQL and Qt Software have successfully established businesses around their own OSS products. Even though these OSS providers have been quite successful, the research literature contains only limited empirical evidence on the challenges and benefits which face a commercial OSS provider [27]. We define a commercial OSS as an OSS product being released by for-profit organizations like MySQL [7], Philips Healthcare [18], JBoss [25], and IBM, Apple and Sun [26]. While these large well known OSS providers have received some attention, small companies providing their own OSS products are overlooked. This is unfortunate since SMEs with less than 250 employees constitute almost 70% of the sector for computer and related activities in the European Union [9].

In this paper we present the story of a small Norwegian software company that has built their business around their OSS products. We analyze the case and compare the findings from this case with what has been reported in the literature.

Based on this discussion we also present some lessons learned that may help other companies in their establishment of a viable business model around their own OSS products.

2 Related Works

Companies and organizations providing OSS have attracted some attention in the literature as for instance [1, 7, 16, 25, 26]. Nevertheless, research on commercial OSS providers is generally missing [27]. Here we discuss three important topics from this literature; business models, communities and software licenses.

2.1 Business Models and Related Issues

The ways companies approach OSS development are diverse [28] and several business models are described in the literature [10, 14, 17, 20]. Four such models are the value-adding service enabler, market creation, leveraging community development and leveraging the OSS brand [10]. Two of these models are particularly interesting for OSS providers (1) using an OSS product to create a market for other services and products and (2) getting contributions from the OSS community [26]. An OSS provider may also use OSS branding to promote its products. While service enabling is more appropriate for companies extending existing OSS communities rather than OSS providers seeking to create their own. Companies may also use OSS products to reach other strategic goals besides directly making money on them. The DICOM validation tool was released as OSS primarily to establish a de facto standard to save rather than to make money [18]. Moreover, Sun established the Java platform to limit Microsoft's control over industry standards [26].

Companies like MySQL and JBOSS do on the other hand build their business around their OSS products [7, 25]. Profiting from the OSS product and its community is for these companies particularly important. Thus creating or identifying a demand for one's products and services is one of the most important risks facing an OSS provider [25]. Roxen tried to make an OSS competitor to the Apache HTTP Server but was forced to change focus due to the strong position of Apache and a lack of demand for their own OSS solution [7]. To be able to create or identify a need for ones products, a commercial OSS provider must understand its customers and their domains. They should therefore hire developers with domain knowledge [27] and use their own software [15] to better understand its strengths and weaknesses.

Making adjustments to the business model and adapting to opportunities and challenges, is key for an OSS provider. When Firefox started to get popular, a wave of viruses and security issues came across the Internet and created a need for a new browser. Firefox was there to fill that need [1]. JBOSS has also been able to adapt to changing opportunities and customer needs [25]. First, the community requested training and documentation. Second, customers demanded advice on building Java applications on top of JBOSS. Third, customers wanted support. Fourth, customers all over the world needed local expertise. JBOSS has evolved its business model by providing training, documentation, consulting services, support and finally an international partner program [25].

2.2 Community

Succeeding at attracting a community is one of the most difficult challenges related to releasing a commercial OSS product [7, 18]. Just releasing the source code is clearly not enough [1]. Considerable investment and several support functions may be needed to successfully release an OSS product [15, 27]. First, practical measures must be taken to prepare a product for release. The source code should be documented and written in a comprehensible manner so it can be understood by users and developers, and the product should be packaged and distributed in easy installable packages [2, 19].

Next, it is necessary to create a common infrastructure on which the company and the community can collaborate. The provider has to set up tools for easy communication and sharing of code, knowledge, experiences and problems [2]. In one project, the participants failed to agree on a configuration management strategy and a set of tools for version control [6]. This made development difficult and contributed to the failure of the project [6].

Another prerequisite for releasing an OSS product is a stable team of core developers which can secure the continuity of the project [15]. This core team should provide the necessary structure to keep the project moving forward [27]. The provider must have resources which can support the product's community including responding to questions and bug reports, fixing problems, take care of contributions and so on [2, 15]. Even though companies may release a product to get contributions from the community [10] most end up implementing almost all the code themselves [24]. A reason for this could be that it proves difficult to rely on the community performing mundane tasks like maintenance, support and so on [15]. Next, in many cases the company wants control over the product to be able to guarantee the quality of it to its customers. Furthermore, the company's employees work with the product the whole time and they are therefore the ones with the most extensive knowledge of it.

To run a community it must be included in the ways of the company, the community members must feel able to contribute to and influence the product, and the provider must respect the norms and values of an OSS community [7, 27]. The OSS norms and values must also be spread to the community, in particular other companies, as the idea of not sharing with other companies is still rooted in the culture of many companies [2].

To include the community, the provider must apply a governance model which is appropriate for the needs of all the stakeholders involved in the community [27]. Too much focus on only a group of stakeholders could be harmful in the long run [15]. Consequently, the provider must be open to new community members and make it as simple as possible to participate in the community [4, 15]. Open communication and transparency should help community members understanding the provider and ongoing activities. OSS projects should furthermore have well documented goals, roles and responsibilities [4]. When opening up the development around Mozilla, the development crew had to release more information and to use public information channels to include the community members [1]. In another project, the project team wanted to deliver a mature product to the OSS community and decided to develop it internally before releasing a mature version [6]. This was a big mistake as communication with the community was very scarce during the development. External users were because of the lack of communication and a product, not particularly interested when the product was released.

To encourage community contributions, the provider should also consider letting go of some control [1]. Too strict control over the product and the community may be counterproductive [18]. If necessary, payment or gifts could be considered to encourage certain behavior or to get contributions [7].

2.3 Software Licensing

Commercial OSS providers must apply a license which is fruitful for both the company and the community [7, 27]. The license must enable the company to make money on either the product or related services and it should enable the growth of a vivid community. A license which the users are unhappy with can severely limit the adoption of a product and it may provoke strong reactions from the community [12].

An OSS provider has a few choices when it comes to selecting a license, as he may develop new licenses or reuse existing ones. Creating new licenses is discouraged [11] since potential users will be unfamiliar with the new license, and since it would require significant resources to create a license of high quality. By reusing existing and well known licenses it is more likely that potential users are familiar with the license, that it is tested, and that it is of good quality.

When reusing existing licenses the OSS provider basically has three choices [8]. First, the OSS provider may use a license like GPL which requires all derivate products to be released under the same license. This may enable him to release the product under a proprietary license as well and thereby create an income from a dual licensing scheme [11]. However, a dual licensing scheme requires that the provider own intellectual property rights for the whole code base. Second, the OSS provider may select a license like MPL which requires direct changes to the original code base to be licensed with the same license, and thereby ensuring that bug fixes and similar changes done by others will be available. Third, the OSS provider may use a license like the new BSD license which sets no restrictions on the choice of license on derivate works, and thereby encourage adoption in any kinds of products.

3 Method

This paper reports on research performed in the COSI project. COSI stands for "Co-development using inner & Open source in Software Intensive products" and is a European industrial research and development project. The project ran for three years, from November 2005 until October 2008 and was organized as a consortium of 13 industrial and academic partners from five countries. The project's goal was to increase awareness of industrial usage of distributed collaborative software and OSS. The research design of the COSI project consisted of five phases, including two case executions, where the companies were working on selected issues identified by the project's plan. During the case executions the companies documented their practices, identified problematic issues and improved these practices.

The authors worked with the five Norwegian companies in the project, supporting and guiding their activities in the project. In addition, we collected data relevant for OSS research. In the case of eZ, the activities were focused on understanding and improving the community management practice, and both case executions addressed this issue.

This research has applied two methods for data collection in this approach: the qualitative research interview and post-mortem analysis (PMA). In addition, we had access to the project deliverables from eZ and had also several informal meetings with the company at COSI workshop meetings, community conferences and other occasions.

Eleven interviews have been conducted with four persons from the development group from eZ at several occasions, distributed over the three years the project lasted. The interviews have been unstructured [21] and have been focused on both on the current community management practice and the history of eZ's main product eZ Publish (hereafter Publish). Notes were taken from all interviews and sent to the interviewees for review.

The authors organized two PMA [3] sessions with most of the developers in the development team. Both sessions focused on how the community management process could be changed in order to increase the number of community contributions to Publish. During these sessions we described the current community management practice and identified both positive and negative issues with this practice. In addition root-cause analyses for some of the negative issues were conducted.

This paper presents the story of a SME that has successfully developed an OSS product and attracted a large community that contributes substantially to the ongoing development of the product. The authors had access to eZ for more than three years. During this time an understanding of how eZ was able to make these achievements was built up based on the conversations with the employees and the authors' reflection. As mentioned above there is little literature on how SMEs develop OSS products, what business models they choose and how they create and take advantage of a community to develop an OSS products. This paper shares lessons learned from such a company and contributes thus to a broader understanding of how SMEs can release OSS products and used the products to attract a community of users and developers.

In analyzing the data and identifying potential lessons learned we found that there are two ways of understanding of eZ's achievements. The first way of understanding is the one of the interviewees, who presented the development of Publish as a series of strategically planned activities. The second way of understanding is from the authors, who see the development of Publish not as a strategic planned activity but rather driven by the skill to identify new opportunities and to make rapid decisions to realize the opportunities. It is the authors' view that both understandings are equally valuable and needed to attract and take advantage of a community.

4 The eZ Systems Case

eZ Systems is a Norwegian software provider founded in 1999. Today they have around 60 employees spread over offices in Norway, Denmark, Germany, France and North America. eZ has almost since its origin focused on providing a PHP based OSS Content Management System (CMS), eZ Publish. The company has a large customer base from all over the world and the CMS has been downloaded more than 2.5 million times from their web site, as of February 2009.

4.1 The Early Days 1999-2001

In the beginning, eZ focused on developing applications for stock brokers but delivered at the same time consultant services to local businesses. These services included network and systems administration, and application and web development. The increasing popularity of the Internet gave them several customers who wanted web sites. Many of these sites contained similar functionality and eZ soon started reusing code from one site to another. This reusable code was quickly bundled into two packages, Publish (article management) and Trade (shop management) and released under the GPL, see Fig. 1. The employees' support for the OSS ideology made releasing the packages as OSS, natural.

The company continues developing stock market applications. Meanwhile, the CMS attracts attention in the OSS community and requests for consulting services related to Publish are coming in. In parallel, they start selling the OSS philosophy to local businesses. The philosophy is simple, if eZ disappears or if the customer is unhappy with eZ's work, he has access to the source code and he may hire someone else. Publish is an attractive product and as a consequence of growing interest from both customers and the OSS community, Publish gradually requires more and more attention. This growing interest forces them to focus on either the stock market applications or Publish. Even though it is a bold move including significant risks, the final decision is to discontinue the stock market application and focus 100% on Publish. The employees have a strong desire for OSS, they really want to create a viable business model based on OSS, and releasing an OSS product sounds fun.

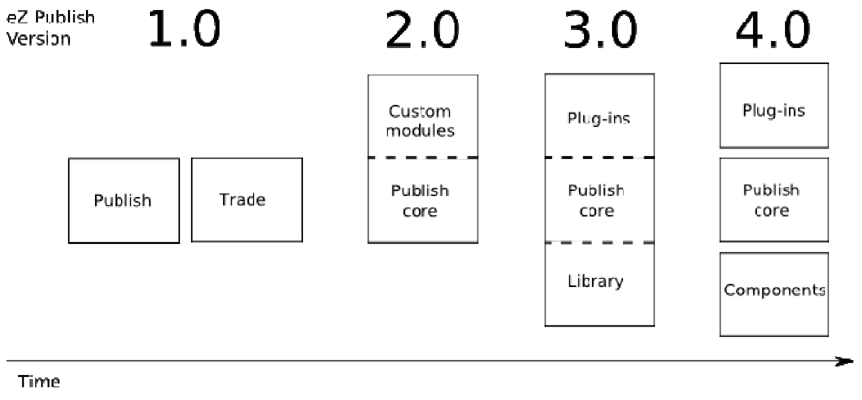


Fig. 1. The development of the Publish architecture

4.2 The Middle Ages 2001-2005

After deciding to focus on the development of the CMS, eZ starts developing Publish 2.0. This version is module based with the intention of enabling custom modules extending the core functionality, see Fig. 1. However, the possibility to extend existing modules without changing the kernel is very limited, if existing. Even though there are some problems with the modular architecture, the system provides interesting functionality, and it therefore attracts a rather large community of OSS users.

The development of the third version starts in 2003 and the PHP 4 based 3.0 version is released in March the next year. The focus of this version is increasing the modularity of Publish, allowing Plug-ins and simplifying the configuration of the system. A simple two layer architecture consisting of a library and the application itself is attempted in addition to the plug-ins, see Fig. 1. However, the two layers are soon too dependent of each other, making it eventually impossible to use the library without installing the application. Even though eZ is unable to keep the two layers separated the plug-in architecture is a success in the sense that it enables the users of Publish to extend it with their own functionality.

4.3 Components and Publish 4.0 2005-Today

Due to dependency problems in Publish it is decided to make a new independent library, giving birth to eZ Components (hereafter Components), see Fig. 1. The library is built separately from the CMS and the development process is opened up to the community. The idea is to create a library which could be used for a wide variety of PHP applications. The library should also be included into Publish when it reached a mature state. This is done iteratively to straighten out eventual problems one at a time. The Library is furthermore a way of refactoring the code in Publish, gradually introducing PHP 5 to the CMS and ensuring support for Windows, Unix and Linux. Late 2007, the forth major version of Publish is released. Through refactoring of Publish and by incorporating Components into the CRM, it gains PHP 5 support. Components furthermore enables those making plug-ins for Publish to make use of the functionality it provides and thereby achieving synergies between the two communities. The division of the system into independent parts enables the growth of three communities around Components, Publish and the plug-ins, see Fig. 2.

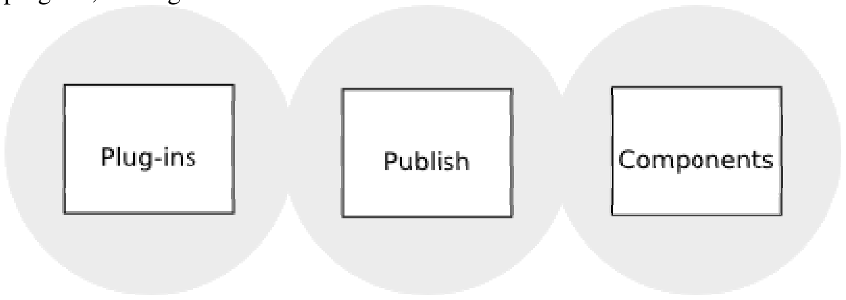


Fig. 2 The parts of eZ Publish and their surrounding communities.

5 Analysis of the eZ Case and Comparison with Findings from the Literature

In the previous section we gave an historical overview of how a small Norwegian software company has successfully launched an OSS product and attracted a large ecosystem of users and developers. This ecosystem can, as illustrated in Fig. 2, be divided into three communities. In this section we will review the case, using the challenges identified in the literature.

5.1 Business Model and the Benefits of Communities

Having a large number of potential customers in the community around Publish creates a greater need for services like support, quality assurance, training, installation, and hosting. Furthermore, it makes selling these services easier and reduces the need for marketing. Users are made aware of Publish through the Internet and services are often sold through bottom-up adoption of the product. Advantages like reduced marketing efforts and shorter sales cycles are also observed elsewhere [19, 25].

The Plug-ins community has developed a large number of plug-ins which extend the functionality of Publish. These plug-ins increase the whole value of the product, enable community members to solve their specific problems, and help eZ to understand these problems. Furthermore, one might see the activity in the Plug-ins community as a way of outsourcing the development and maintenance of these plug-ins, and thereby reducing eZ's development efforts. The community members' investments in developing these plug-ins build a stronger connection between them and Publish and thereby increase their loyalty to it.

The Components community contributes code to a library eZ would have needed to develop regardless of these contributions. Next, the future of PHP is essential to eZ's products and Components, particularly if it becomes widely adopted, is a tool eZ can use to keep up with and influence the development of PHP. Adoption of the library will also contribute positively to increasing eZ's reputation, particularly in the OSS community.

Using the categorization of business models in [10] we see that the communities around eZ support different strategic goals. Publish is creating a market for the supplementary services eZ and their partners provide. More, through the two other communities, eZ gets contributions from the OSS community. OSS products can as we see be used to reach other strategic goals than directly increasing the income of a company [18, 27]. Components, the plug-ins and their communities illustrate this as they contribute to reducing eZ's development costs, increasing the value of Publish, and to monitoring and influencing the future of PHP. eZ are in other words using different strategies for each of the communities to support their over all business strategy.

eZ is furthermore able to construct a good understanding of the needs of their users through feedback, requirements and interaction with all three communities. Community developed plug-ins, recruitment of developers from the community and the use of their own product give eZ better understanding of the domain and thereby reduce their expenses on market research.

The business strategy of eZ has evolved from application development targeting a specific domain to providing services and support to the ecosystem around an OSS product. An evolution of the business model can also be seen in the JBOSS case [25]. Income from services and support are more predictable and consistent than from licenses and consulting, and less sensitive to economic turnaround [25]. This is being particularly true when having a large install base. It is therefore natural to evolve the business model as the customer base grows.

5.2 Community

Infrastructure eZ has been investing in a common infrastructure for the three communities. For the Plug-in community, eZ is hosting a portal for plug-ins, as well as organizing developer days at their annual Publish event. The infrastructure for Publish consists of forums, mailing lists, issue trackers, documentation and source code. For the Components community mailing lists and an open issue tracker are provided.

Providing this infrastructure is a rather small investment, even for a small company. In addition, eZ did not set up their community infrastructure before the product was released but did so over time, driven by the activity level and demand of the communities. The cost of establishing the infrastructure has thus been spread out over several years. This contrasts the findings of [15, 27], that both mention that considerable investment is needed to release an OSS product and to set up support functions. One possible explanation is that eZ never planned from the start to provide an OSS product.

Attracting and governing a community Attracting and governing a community is one of the most challenging aspects of releasing an OSS product [7, 18]. Today eZ has an ecosystem that consists of three communities, serving its two products. Together this ecosystem attracts users, volunteers and customers to use the products and to be part of the communities. eZ is attracting the communities by providing two interesting products that are downloaded and used by a large user base. eZ is further attracting member to their communities by accepting and hosting plug-ins to their Publish product, and by accepting contributions from both the Publish and Components community. eZ also communicates a positive attitude towards open source to the outside world and uses the open source label to differentiate itself from non-open source competitors.

Attracting a community starts with releasing an attractive product, that is of interest to a potential large user base. The most active community in eZ ecosystem is the Plug-in community. It started when users started developing their own functionality by using the plug-in mechanism in the architecture of Publish. These developers wanted to share their plug-ins with other Publish users, and reflected thus the same attitude to open source that made eZ release Publish as an open source product in the first place. The plug-in community is attractive to its members even when the members are not included in the way of the company. The inclusion in the way of the company is suggested to be a necessity to attract a community [7, 27]. eZ is including the members of the Publish and Component communities in varying degrees, but in none of the communities are the members fully included in the way of the company. The community members' motivation to contribute is thus not the inclusion in the way of a company but rather implementing functionality they are interested in themselves. The argument made here is not that it is not important to include community members into the ways of a company, but that the attraction of a community starts with a product that is appealing to a large number of users.

eZ is as of now not satisfied with the activity level in the Publish community and would like to increase it. This deals with how to govern a community, and with how to balance conflicting interests between the community and eZ customers. Since Publish is the strategic core product for eZ, control with the product and its future development is needed for strategic reasons. Exercising too much control, however, may result in that the community looses its attractiveness with its members [18].

5.3 Software Licensing

Publish and Components address different strategic goals. To avoid licensing problems, to attract a community and to reflect these strategic goals, eZ selected two different, but well established OSS licenses. The GNU Public License (GPL) allows the community to use Publish without paying any license fees. At the same time it gives eZ control over how Publish is used. GPL requires code sharing and prevents the use of the source code in proprietary products. Moreover, GPL enables eZ to dual license Publish and thereby getting some income from the license sales. eZ provides proprietary licenses for companies which (1) include Publish in their proprietary products, (2) build proprietary extensions on top of Publish and (3) use Publish as any other proprietary software. This last license is particularly useful for companies which not yet have legally approved the use of OSS licenses in their organization. However, to lower the threshold for adoption of Components, eZ released it under the New Berkeley Software Distribution (BSD) license which gives adopters quite unlimited freedoms.

6 Lessons learned

With eight years of experience, the eZ case identifies some lessons learned about how to release an open source product.

Allow your business model to evolve Providing an OSS product is not a trivial task, and the experience from the eZ case shows that providing an OSS product may take unexpected turns. Even though the use of OSS in the software industry is growing, OSS business models have yet to stabilize themselves. It is thus important to plan for a business model and to allow it to evolve with the opportunities and challenges presented by the product and its community. Core team needs experience from other OSS projects and communities Setting up an OSS community requires knowledge about how open source communities function. Having developers with experience from other open source communities is beneficial since they have fit hand experience with OSS values and practices. The Components community is a good example that this is helping to create an active community.

Balance control and bureaucracy related to community contributions Lack of control over community contributions directly to your product can reduce the quality of it and potentially introduce illegitimate source code into the product. Too strict control on the other hand may discourage contributions and community participation. It is therefore important to clearly specify where you are going with your product and what kind of contributions you want, and to make contributions and wanted behavior visible to other community members.

Be part of your own community In order to sustain a community of volunteers a community needs to be active and including. This can be achieved when the core development team is part of the community, and uses a common infrastructure to share information and to co-ordinate all activities. This creates the transparency that a community is expecting. The opposite of such a transparent community would be a community where the core team uses a parallel infrastructure to communicate and co-ordinate their activities internally.

Apply well known licenses which suit both you and your users Unnecessary strict licenses may limit the adoption of a product. Both OSS users and paying customers will most likely go elsewhere if their needs are not met by the software's license. To avoid intimidating the users, simple, well known licensing models should be chosen. Explain the OSS licenses, its permissions and restrictions. Launching a product as OSS could include a constant fear of license infringement. When the source code is available it is technically quite simple to misuse the source code. However, this has not been a problem for eZ and the very few incidents which have occurred have easily been solved.

7 Discussion and conclusions

Finally, some issues will be pointed out. First, investing in an infrastructure is not reserved only to open source providers. While this investment has been seen as something that is an extra investment for companies providing OSS, providers of commercial products need an infrastructure as well to stay in touch with their customers, and receive error reports and other feedback.

eZ Systems have established an ecosystem with three communities that are based on different business models and give different benefits in return. This strategy to create more than one community with an OSS product seems to enable eZ to take advantage of several of the benefits that are associated with having a community of users. This division helps attracting and directing contributions to two areas where it is more convenient to receive them while controlling the core product. At the same time as eZ wants to attract more contributions to Publish (the core), there is also a need to keep certain control with this product for commercial reasons. Resolving conflicts between community interests and commercial interests is a delicate balance.

This paper has presented the history of the two open source products provided by eZ and the three communities that constitute the ecosystem around these products. Based on eZ's experience, we have identified some lessons learned which could help other OSS providers. There is no single answer on how to succeed as an OSS provider. In case presented in this paper, however, there are some factors that contributed to the success of the provided OSS. This includes the evolvement of the business model, having an attractive product and adapting to community needs and opportunities.

Acknowledgments

The research has been conducted within the ITEA COSI project and is supported by the Research Council of Norway. We are grateful for the support from eZ Systems and to our colleagues in the COSI project, in particular Vidar Langseid, Thomas Østerlie, and Carl-Fredrik Sørensen.

References

1. Mitchell Baker. The Mozilla Project: Past and Future. In Chris DiBona, Danse Cooper, and Mark Stone, editors, open sources 2.0, pages 3-20. O'Reilly Media Inc, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2006.
2. Massimo Banzi, Guido Bruno, and Giovanni Caire. To What Extent Does It Pay to Approach Open Source Software for a Big Telco Player?. In Russo et al. [22],pages 307-315.
3. Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane. Postmortem: Never Leave a Project without It. IEEE Software, 19(3):43-45, 2002.
4. Wolf-Gideon Bleek and Matthias Finck. Ensuring Transparency - Migrating aClosed Software Development to an Open Source Software Project. In IRIS'28 Proceedings of the 28th Information Systems Research Seminar in Scandinavia, 2005. 6-9 August.
5. Wolf-Gideon Bleek, Matthias Finck, and Bernd Pape. Towards an Open SourceDevelopment Process ? Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model. In Scotto and Succi [23], pages 37-43.
6. Cornelia Boldyreff, David Nutter, and Stephen Rank. Communication and Conflict Issues in Collaborative Software Research Projects. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani, editors, Collaboration, Conflict and Control Proceedings of the 4th Workshop on Open Source Software Engineering, pages 14-17, 2004.
7. Linus Dahlander and Mats G. Magnusson. Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms. Research Policy, 34(4):481-493, 2005.
8. Paul B. de Laat. Copyright or copyleft?: An analysis of property regimes for software development. Research Policy, 34(10):1511-1532, 2005.
9. Eurostat: Number of persons employed by enterprise size-class in the EU-27, 2009. Online: http://ec.europa.eu/eurostat/, accessed 2009-02-12.
10. Brian Fitzgerald. The Transformation of Open Source Software. MIS Quarterly, 30(3):587-598, 2006.

11. Karl Fogel. Producing Open Source Software: How to Run a Successful Free Software Project. O'Reilly, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2005.
12. Jim Hamerly, Tom Paquin, and Susan Walton. Freeing the Source: The Story of Mozilla. In Chris DiBona, Sam Ockman, and Mark Stone, editors, Open Sources:Voices from the Open Source Revolution, pages 197-206. O'Reilly, 1999.
13. Øyvind Hauge, Carl-Fredrik Sørensen, and Reidar Conradi. Adoption of Open Source in the Software Industry. In Russo et al. [22], pages 211-222.
14. Richard E. Hawkins. The economics of open source software for a competitive firm. NETNOMICS, 6(2):103-117, August 2004.
15. Juha Järvensivu and Tommi Mikkonen. Forging A Community Not: Experiences On Establishing An Open Source Project . In Russo et al. [22], pages 15-27.
16. Chris Jensen and Walt Scacchi. Collaboration, Leadership, Control, and Conflict Negotiation and the Netbeans.org Open Source Software Development Community. In HICSS'2005 Proceedings of the 38th Annual Hawaii International Conference on System Sciences, page 196b. IEEE Computer Society, 2005.
17. Sandeep Krishnamurthy. An Analysis of Open Source Business Models. In Joseph Feller, Brian Fitzgerald, Karim R. Lakhani, and Scott A. Hissam, editors, Perspectives on Free and Open Source Software, pages 279-296. The MIT Press, Cambridge, Massachusetts, 2005.
18. Juho Lindman and Topi Uitto. Case study of company's relationship with open source community in open source software development. In IRIS'31 Proceedings of the 31st Information Systems Research Seminar in Scandinavia, pages 1-22, 2008.
19. Alberto Onetti and Fabrizio Capobianco. Open Source and Business Model Innovation. The Funambol Case. In Scotto and Succi [23], pages 224-227.
20. Eric Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly, Sebastapol, CA, 2001.
21. Colin Robson. Real World Research. Blackwell Publishing, 2nd edition, 2002.
22. Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors. Open Source Development Communities and Quality IFIP Working Group 2.13 on Open Source Software September 7-10, 2008, Milano, Italy, volume 275 of IFIP International Federation for Information Processing. Springer, 2008.
23. Marco Scotto and Giancarlo Succi, editors. OSS'2005 Proceedings of The First International Conference on Open Source Systems, 2005.
24. Anthony I.Wasserman and Eugenio Capra. Evaluating Software Engineering Processes in Commercial and Community Open Source Projects. In Andrea Capiluppi and Gregorio Robles, editors, FLOSS '07 First International Workshop on Emerging Trends in FLOSS Research and Development, page 1, Washington, DC, USA, May 2007. IEEE Computer Society.
25. Richard T. Watson, Donald Wynn, and Marie-Claude Boudreau. JBoss: The Evolution of Professional Open Source Software. MIS Quarterly Executive, 4(3):329-341, September 2005.
26. Joel West. How open is open enough?: Melding proprietary and open source platform strategies. Research Policy, 32(7):1259 - 1285, 2003.
27. Joel West and Siobhán O'Mahony. Contrasting Community Building in Sponsored and Community Founded Open Source Projects. In HICSS'2005 Proceedings of the 38th Annual Hawaii International Conference on System Sciences, page 196c. IEEE Computer Society, 2005.
28. Sven Ziemer, Øyvind Hauge, Thomas Østerlie, and Juho Lindman. Understanding Open Source in an Industrial Context. In SITIS'2008 Proceedings of the 4th IEEE International Conference on Signal-Image Technology & Internet-Based Systems, pages 539-546. IEEE Computer Society, 2008.

PAPER 7

Øyvind Hauge, Daniela Soares Cruzes, Reidar Conradi, Ketil Sandanger Velle, and Tron Ándre Skarpenes. *Risks and Risk Mitigation in Open Source Software Adoption: Bridging the Gap between Literature and Practice.* In Pär J. Ågerfalk, John Noll, and Cornelia Boldyreff, Proceedings of the 6th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2010), May 30th-June 2nd, Notre Dame, USA, volume 319/2010 of IFIP Advances in Information and Communication Technology, pages 105-118, 2010. Springer.

Risks and Risk Mitigation in Open Source Software Adoption: Bridging the Gap between Literature and Practice

Abstract. The possible benefits of open source software (OSS) have led organizations all over the world into adopting a variety of OSS products. However, the risks related to such an adoption, and how to reduce these risks, are not well understood. Based on data from interviews, a questionnaire, and workshops, this paper reports ongoing work in a multi-national telecommunications company. The paper has three main contributions. First, it extends identifies and discusses several risks related to OSS adoption. Second, it identifies steps for reducing several of these risks. Third, it shows how research can be used to increase the visibility of, and involve the employees in, ongoing OSS efforts.

1 Introduction

The promise of reduced costs, increased flexibility, and freedom from proprietary vendors has convinced organizations worldwide into deploying open source software (OSS) products in their production environments and integrating OSS components into their software systems [15,16,19,20]. While a couple of studies have looked at the possible benefits and drawbacks of such OSS adoption [2,24,35], few have discussed steps for dealing with related risks.

Our primary goal is to identify relevant risks and risk mitigation steps for organizations that adopt OSS. The secondary goal of the study presented here is to explore the opportunities for increasing organizations' adoption of OSS. This includes identifying potential benefits of an increased OSS adoption. The main research questions investigated in this study are:

RQ1. What are the perceived benefits of an increased adoption of OSS products?
RQ2. What are the perceived risks of such an adoption?
RQ3. What steps may an organization take to reduce the risks related to such an adoption?

The study presented in this paper was partially conducted at Telenor, a large international telecommunications company. Telenor's Norwegian IT division has already adopted some OSS products, but it is currently looking into increasing its adoption. However, to avoid the possible pitfalls of OSS adoption, Telenor IT wants to identify (1) the benefits s and risks which are relevant to their context and (2) how to deal with potential risks. To support Telenor in finding the answers to these questions, we have conducted a study consisting of semi-structured interviews, a questionnaire with 86 responses, and three workshops.

This paper is structured as follows: Section 2 gives an overview of related work. Section 3 presents the interviews, questionnaire, and workshops conducted in this study. Section 4 presents the results from this work and forms a basis for the discussions in Section 5. Section 6 discusses limitations of this paper while Section 7 concludes the paper and presents input for future work.

2 Related Literature

OSS can be adopted in different ways. In [18] we show that OSS can be adopted through deploying OSS products, using OSS CASE tools, integrating OSS components, participating in the development of OSS products, providing OSS products, or through using OSS development practices. Grand et al. [18] present a four level model for resource allocation to OSS. In a company perspective, the four levels are (1) company as a user of OSS software, (2) OSS software as complementary asset, (3) OSS software as a design choice, and (4) OSS compatible business mode. This paper focuses on the **deployment of OSS products** (like operating systems, database servers, application servers etc) within in a company at **level 1 or 2** in Grand et al.'s model for resource allocation.

2.1 Possible benefits of OSS

The literature discusses several possible benefits (**B**) of OSS adoption:

Cost cuts (**B1**): OSS has been claimed to enable costs cuts through for instance reduced license fees, hardware requirement, scaling costs, etc. [2,14,24,33].

Independence from proprietary vendors (**B2**): The adopter of OSS may also get increased freedom from vendor lock-in and increased influence on providers of both proprietary and OSS products [2,5,6,23,24].

Simplified procurement and license management (B3): The majority of OSS products tend to come with only a few different licenses and without licensing fees. This may simplify the procurement of the software and the licensing of derivative products [23,28].

Software reuse (**B4**): Through adopting software products that are developed, tested, and used by others, we may gain the benefits of software reuse. This includes extra/new functionality, increased R&D and innovation, improved quality (e.g. reliability, security, performance, defect density) and increased productivity [2,5,6,14,25,34]. OSS may also contribute to increased standardization [1,21] or to establishing de-facto standards if no standards exist [23].

High availability (**B5**): OSS products are most often easily available together with source code and trustworthy information about the products' true state [22,23,35].

Community support (**B6**): This openness may lead to increased collaboration between community members [2,24]. The community might also provide free maintenance and upgrades of the software together with user support [5,24,33]. .

2.2 Potential risks of OSS adoption

There are also risks (**R**) related to adopting OSS but not all organizations consider them, as there are organizations adopting OSS without performing any cost/benefit analysis [35]. There are no papers that explicit focus on potential risks of OSS adoption, but the literature mentions several possible drawbacks of OSS Adoption.

Hidden costs (**R1**): Adoption of OSS products is not without costs: It may be time-consuming to evaluate them [31]. Adoption may involve user training and configuration [24,31]. We might need to spend resources on community participation [23]. Many organizations would need premium professional support [14,35].

Lack of products (**R2**): While there are many OSS products available, there may still be a lack of products with specific functionality [6,24]. The quality of these products can also be questionable [14,35]. OSS products may furthermore suffer from limited standardization and compatibility with document formats or with versions of other software products [2,24,25].

Lack of providers, expertise, and support (**R3**): Despite the significant adoption of OSS, there may still be a lack of expertise and support for specific products [2,24,31,35]. The lack of professional providers may also introduce unclear liability and uncertainty about the longevity of OSS project as OSS projects may lack roadmaps and documentation [2,24]. Holck et al. hypothesized that this lack of traditional vendor-customer relationship could stop the adoption of OSS [22].

Customization needs (**R4**): It may be necessary to customize the OSS products to fit them into the context in which they are going to be used [1]. When changing an OSS product we may get a maintenance responsibility [36] as these changes must be updated when more recent versions of the software are adopted. When these situations arise, the adopter must decide to follow the new releases or ensure backward compatibility with his own changes [23].

Licensing issues (**R5**): The variety of OSS licenses available is confusing, as there is a lack of guidance on how to interpret them [31]. When adopting OSS and in particular when integrating OSS into derivative software systems, it may be challenging to combine code under an OSS licenses with proprietary licenses and APIs [23].

2.3 Risk mitigation in OSS adoption

As there are few publications discussing risk mitigation (**RM**), this section describes results from the literature on success criteria and enabling versus inhibiting factors of OSS adoption. Some of these issues may contribute to reduce the risks related to OSS adoption.

Employee attitude, awareness, and skills (**RM1**): A positive employee attitude towards OSS and the OSS ideology can enforce the adoption of it [4,5,16,25,34]. The adoption of OSS should also be made visible, so that end users have an awareness of the technology adoption [6]. Finally, if employees have the necessary skills, training, and experience with OSS, the probability of a successful adoption will increase [6,16,32].

Management support (**RM2**): Management support is also important for OSS adoption [6,14,16]. Management should provide resources for driving the adoption, and a clear plan for analysis, testing, and pilot projects [6, 13, 16].

Access to support (**RM3**): The quality of the OSS components [29] is an important factor for a successful adoption, and for many adopters it is also necessary to have access to professional support [6,13].

Success stories (**RM4**): It is furthermore an advantage if the products have been successfully adopted by other companies [16]. Lack of such success stories or lack of other users could easily complicate the OSS adoption [2,4].

No lock-in (**RM5**): An organization may have a hard time adopting OSS if they are locked in by industry-wide purchasing agreements and standards for IT or already have a coherent stable IT infrastructure based on proprietary or legacy technology [6,16]. The costs related to moving away from such a lock-in situation can significantly impede the adoption of OSS [4,17,35].

3 Context and Research Method

Telenor is currently among the ten largest mobile operators in the world, with over 164 million mobile subscriptions and revenues of more than €12 billion in 2008. Telenor Norway IT (hereafter Telenor IT) is the information technology and software support division under Telenor's Norwegian branch. It has about 380 employees that together with several external partners develop, maintain, and support more than 500 IT systems. Open Source 2010 is an internal project aiming at exploring the opportunities of increased adoption of OSS products like databases, operating systems, and application servers.

Telenor IT's motivation for conducting this study was threefold. First, Telenor IT wanted to increase the awareness of OSS and its Open Source 2010 project within the organization. Second, it wanted to get feedback from and involve the employees in the project's work. Third, it wanted to weaken proprietary providers' grip on Telenor reducing expenses on licensing and support.

This study consisted of four main steps. As part of a **systematic literature review** on OSS in organizations, we reviewed the literature for evidence on the perceived benefits and drawbacks of OSS [20]. Some of the output from this review was used as an input for Section 2. Next, we conducted **semi-structured interviews** with four employees from different parts of Telenor IT. The interviews were carried out through 30 minutes face-to-face sessions. These sessions were recorded and the recordings were later transcribed. Based on these interviews, we developed a **questionnaire**. The questionnaire was pre-tested by colleagues at the university and nine employees from different parts of Telenor IT. The questionnaire was written in Norwegian, and the final version contained 42 open and closed questions. For the closed questions we used 5-point Likert scales. Several of the 42 questions concerned the three research questions. However, we will focus mainly on the questions below (Q1-6):

Q1. Which advantages and disadvantages do you see with the use of OSS in Telenor IT? (See Table 1);

Q2. For which reasons do you think Telenor IT should select OSS instead of proprietary products and vise versa? (See Table 2);
Q3. Why should Telenor IT increase its use of OSS? (Open, textual input);
Q4. Which risks do you see with increased use of OSS in Telenor IT? (Open question);
Q5. If the use of OSS in Telenor IT should be increased, what should do Telenor do to facilitate this? (See Table 3);
Q6. Where would an increased use of OSS be appropriate? (Open question).

The questionnaire was conducted with a sample of 140 employees from Telenor IT that were handpicked by our local contact. This sampling technique was used to get a representative sample of employees from all relevant parts of the organization while avoiding employees who were not involved in development and/or support of Telenor's software systems. In total 86 respondents completed the survey, giving a response rate of over 60%. The analysis of the data consisted of descriptive statistics, statistical tests and comparison/grouping of about 200 comments from the open questions.

After the analysis, we held three **workshops**. First, we presented the results from the questionnaire to several employees from various parts of the organization. Second, three project members and three employees with experience from different operating environments participated in a discussion around (1) benefits, (2) risks, and (3) approaches related to increasing the organization's adoption of OSS. These three sessions were performed as "KJ sessions" [3], where each of the workshop participants used post-it notes to write down their concerns and put these notes on a white board. Then the participants re-arranged related notes into groups as a collaborative effort. These groups of related issues were then discussed. 152 post-it notes were collected and grouped into 11, 10, and 7 groups for the three sessions. Finally, we presented these results during a third dissemination workshop, open to all employees at Telenor IT.

4 Results

Results presented in this section were grouped according to the research questions stated in the introduction.

RQ1. Based on the interviews, questions (Q1, Q2, Q3), and the workshop, we have identified the main perceived benefits (**BT**) of OSS adoption;
RQ2. Based on the interviews, question (Q4), and the workshop, we have identified several potential risks (**RT**) related to adoption of OSS;
RQ3. Mainly through the workshop and the interviews, but also questions (Q5, Q6), we have identified steps for (1) facilitating the adoption of OSS and (2) steps for mitigating (**RMT**) some of the risks related to it.

In the following, we summarize the main findings related to the research questions, while keeping a focus on the results most relevant to Telenor IT.

4.1 RQ1: Potential benefits of OSS

Reduced costs (BT1): Cost reduction is the most cited advantage of OSS adoption. Table 1 shows that the respondents to the questionnaire agreed (Q1.1). Several respondents stressed the value of reducing the expenses on support agreements and claimed that OSS could contribute to this. One respondent suggested that they could simplify the administration of (proprietary) software licenses. Moreover, Table 2 shows that the respondents expected both development and maintenance costs to be lower with OSS (Q2.1 and Q2.2). Finally, if Telenor could standardize on one OSS platform, the IT department could increase its productivity and reduce costs from running on a more homogeneous and cheaper hardware platform.

Table 1. Potential advantages and disadvantages with OSS in Telenor IT (Q1)

ID	Statement	Mean	STD
Q1.1	Reduced licenses costs	4.56	0.86
Q1.2	Independence from providers	4.48	0.88
Q1.3	Ability to apply pressure on providers	4.41	0.93
Q1.4	Motivational factor for the employees	4.16	0.99
Q1.5	Access to read and modify source code	4.10	1.01
Q1.6	Confidence and experiences with provider	3.26	1.18
Q1.7	Existing contracts with providers	3.19	1.28

Independence from proprietary vendors and ability to apply pressure on providers (BT2) was frequently discussed by interviewees, workshop participants and many of the responses to Q3 (see also Q1.2 and Q1.3). They highlighted in particular the ability to use OSS to apply pressure on their vendors in order to make them lower their license and support fees. As one respondent wrote "[when using OSS, one] *may chose to pay for support if you actually need it (often one does not need it)*" (Q3).

Attractive and future-oriented technology as a motivational factor for the employees (BT3): Several popular technologies are offered as OSS, and the interviewees mentioned that using OSS could improve the Telenor brand (Q2.4), be a source of motivation for current employees (Q1.4), and be a way to attract skilled employees. The ability to work with new and open technology was also perceived as being fun by the workshop participants. In fact, quite a lot of attention was drawn to this issue. OSS technology was also considered to be the future for several areas. For instance, one workshop participant wrote that "*OSS is future oriented and it enables access to competency*". A respondent in the questionnaire wrote that "*OSS is becoming the industry standard in many areas*" (Q3).

Ease of use through access to information and the source code (BT4): The respondents also suggested that OSS technology was easier to use because of its openness and the high availability of the software, its source code, and related information (see also Q1.5, Q2.3, and Q2.5). One workshop participant wrote that because of this availability "*[it] is easier to make prototypes and to evaluate the software*". A respondent in the questionnaire wrote: "*it is better to modify what is meant to be modified rather than buying a final package and doing extra development around it [the package]*" (Q3). The workshop participants furthermore believed that the flexibility and openness of OSS could give them better and more innovative solutions. Easy access to new technology and a lot of development tools, together with the technical support, documentation, and other resources, could further reduce the effort needed to develop and maintain their systems. Having relatively open access to the communities that develop OSS products was seen as a great advantage not only to get support but also to influence the development of the products they would use. One responded: "*OSS products are quite often having active communities with dedicated users who are more than willing to help*" (Q3). OSS communities were in these aspects considered to be more accessible than proprietary vendors.

Table 2. Reasons for selecting OSS versus proprietary software (Q2)

ID	Statement	Mean	STD
Q2.1	Reduced maintenance costs.	4.15	1.15
Q2.2	Reduced development costs.	4.05	1.13
Q2.3	Possibility to run pilot-tests (alpha/beta tests) before release.	3.94	1.2
Q2.4	Improve Telenor's brand and reputation.	3.76	1.17
Q2.5	Adaptability to existing systems.	3.68	1.32
Q2.6	Development time.	3.64	1.13
Q2.7	Influence on provider (add new or changed functionality).	3.64	1.43
Q2.8	Availability of external expertise and experience.	3.48	1.35
Q2.9	Availability of support during development.	3.32	1.33
Q2.10	Available information (manuals etc.).	3.24	1.43
Q2.11	Functional requirements (adequate functionality)	3.19	1.17
Q2.12	Non-functional requirements (quality, reliability, security, scalability, performance, usability etc.	2.95	1.26
Q2.13	Availability of support in production	2.87	1.38

4.2 RQ2: Potential risks and drawbacks

Lack of support and expertise (RT1): The lack of a professional provider is not necessarily a problem. However, the lack of support and expert advice, in particular for complex problems, was considered as one of the major challenges with OSS. One of the interviewees feared that they would need to increase their internal resources quite dramatically. Telenor requires professional support 24/7. However, providers of such support are not necessarily available for all OSS products. One workshop participant pointed this out and wrote that "*[there are] few/no international support organizations (for instance when you need 24/7 operation)*". Moreover, since the diffusion of OSS products is not always as large as their proprietary equivalents, the workshop participants feared that it could be difficult to get hold of both expert consultants and highly skilled employees.

Hard to select the right OSS product (RT2): The respondents expressed an uncertainty related to whether there existed OSS equivalents for some of the largest and most advanced systems they had. The respondents moreover feared that existing OSS products were immature and would miss key functions. One respondent wrote that "*there are in some cases no OSS products, or no OSS products which are good enough, for solving certain problems*" (Q4). The products may also lack support from a viable community and they may therefore have an uncertain future. Adopting such immature or unsupported products can introduce significant costs further down the road, and it was therefore considered important to find the right products.

Change and hidden costs (RT3): OSS products would in most cases be acquired and maintained somewhat differently than proprietary products. Most OSS products are available over the Internet and do not have the same number of providers pushing and supporting the products. These changes may improve the way the organization works but any change introduces challenges, uncertainty, and at least some costs. A workshop participant wrote that "*[Telenor] has to find and relate to new partners*", something which would include both change and cost. The respondents were uncertain whether the cost savings from reduced licensing and support fees would outweigh the cost related to switching technology and changing the way they worked, as some of them described the total cost of adopting OSS as "*foggy*". One respondent wrote that "*replacing familiar technology*" (Q4) could be a potential risk. Replacing existing technology would also make some of the expertise they currently possess less valuable.

Unclear liability and responsibility (RT4): As of today Telenor's partners have relatively clearly defined responsibilities. Changing these relationships was considered an important challenge. One responded that it could lead to "*unclear distribution of roles between provider - customer [Telenor]*" (Q4). Most OSS products lack a clear (professional) vendor and the respondents feared ending up in situations with unclear liability, where they were unable to influence the provider, and where they would not get sufficient support. One respondent wrote "*[we have] no provider to make responsible in situations with critical errors*" (Q4). Such situations could put a significant strain on Telenors's internal resources.

Uncontrolled adoption and modification (RT5): Changes, or potential anarchy, related to the acquisition of software was discussed to some length in the workshop. This is because (1) there are a lot of easily available OSS products (in many different versions), (2) there is a lot of hype around many of these products, and (3) they are very easy to modify. Some participants feared that this could lead to uncontrolled and constant adoption and modification of new OSS products, giving them a diverse and uncontrolled and expensive to maintain a software portfolio. One workshop participant wrote that he feared that "*one [Telenor employees] selects products because they are OSS, not because they solve our problems*". A respondent in the questionnaire feared what he called "*product anarchy*" meaning that the selected a lot of products without really making sure that they were the right ones.

4.3 RQ3: Mitigating the risks related to OSS adoption

Place responsibility, dedicate resources, and ensure support (RMT1): To make sure that Telenor IT's employees have access to necessary resources to develop, support, and operate OSS based software systems, it was considered very important to place the responsibility for the products Telenor adopts between internal resources and external partners. This was highlighted by several participants in our study. One of them wrote that "*[Telenor must] coordinate with development, internal operations, and external [service] providers*". This could involve increasing the internal resources or allocating employees to not only support of OSS solutions, but also to developing new solutions and monitoring the OSS community. It may also involve finding and dealing with new partners, or driving existing partners into adopting new technology. The participants in the study expressed particular concerns about ensuring support for the really difficult problems.

Start pilot projects (RMT2): The respondents agreed that it was important not to rush the adoption of OSS, but promoted instead a cautious, stepwise approach to OSS. According to them, Telenor IT had to gain experience with one project at the time through identifying projects where OSS would be give real benefit. These pilot projects could then be used to illustrate the potential and true benefits of OSS within the organization. The respondents acknowledged that pilot projects were important not only to illustrate the potential of OSS products, but also to have a more moderate learning curve and limit the consequences of problems. One workshop participant wrote that "*[Telenor should] incrementally introduce OSS and consider new/revise measures based on our own experience*".

Increase awareness and make the OSS initiative visible (RMT3): The first thing which could be done, is making the organization's current and planned use of OSS visible to, not only its employees and management, but also its partners (see Table 3). In the workshop one participant wrote that "*[Telenor IT must] make the concrete advantages visible"*. By identifying successful cases of OSS adoption and making these visible, they may create a positive attitude towards OSS and show that it is a viable option for the future. Moreover, it was considered important to explain why Telenor IT is planning to increase its adoption of OSS.

Table 3. Possible steps for increasing the adoption of OSS (Q3)

ID	Statement	Mean	STD
Q3.1	Start one/several pilot projects to show possible effects of OSS	4.54	0.85
Q3.2	Make the OSS initiative visible for all employees	4.48	0.63
Q3.3	Make visible the OSS already present in the organization	4.44	0.85
Q3.4	Top management commitment to the OSS initiative	4.42	0.95
Q3.5	Make someone responsible for monitoring selected OSS domains	4.15	0.93
Q3.6	Improve both internal and external knowledge management (e.g. with a Wiki, message boards, mailing lists, blogs or similar)	4.14	0.94
Q3.7	Hire new employees with OSS experience	3.96	0.99
Q3.8	Hire external consultants with updated expertise	3.06	1.19
Q3.9	Restructure the business model of Telenor IT	2.74	1.12

Include OSS in strategies supported by top management (RMT4): Finally, the adoption of OSS should not be left up to chance and the individual employees' taste. A workshop participant wrote that *"[Telenor] should not allow the system or project select freely [it should rather] be part of a strategic technological decision"*. To ensure that the OSS adoption was planned, it should be part of a strategy where (top) management, developers, operations, and support were involved in the decision making process. It was furthermore considered important to assess the benefits versus the costs in each specific case. Management support was considered important because Telenor IT currently used OSS mainly in development but not so much in production. Moving OSS to production environments would require wide support.

5 Risks and Risk Mitigation Strategies

Our empirical results confirm many of the findings from the literature review presented in Section 2. Through the literature review and our study we have identified several risks related to the adoption of OSS products. Table 4 shows an aggregation of the results from this study and from the literature, in a first step towards a risk mitigation approach in OSS adoption. Most of these are already presented in Section 2 or 4. The table is divided in three main columns. The first column lists the main risks identified in the study. The second column describes possible steps for mitigating these risks. This is once more based in our study and the papers that implicitly or explicitly discuss these steps. The third column describes other possible steps that were only identified in the literature. Still, we think they are worth mentioning.

Besides the results shown in Table 4, we identified some general steps for reducing the risks related to adoption of OSS products such as: (1) increasing the employees' skills (hire new or train existing) (**RM1**), (2) increasing the employees' attitude towards, and awareness of, current adoption of OSS and ongoing OSS initiatives (**RM1**), (3) ensuring top management commitment to the OSS initiative (**RM2**), and (4) avoiding going from a proprietary to an OSS lock-in (**RM5**).

The literature also mentions licensing and customization of the OSS products as potential risks related to OSS adoption. These risks were not discussed in the table or in our results. First, Telenor IT's Open Source 2010 project did not consider licensing issues to be a problem, particularly since Telenor is not going to distribute its software. Issues related to releasing the source code were therefore not relevant. However, it was suggested to seek legal advice to approve a set of OSS licenses, and adopt only products with these licenses. Second, customization needs was not given much attention. One possible explanation could be that Telenor IT focused on software like operating systems, database servers, and application servers. These products constitute a “software infrastructure” and are mainly configured and deployed. Customization problems is perhaps more relevant for other kinds of software products like software components and desktop tools.

6 Limitations of this study

The sampling for the questionnaire was conducted by our contact person at Telenor. This may pose a possible threat to the validity of our results, since the sample and respondents may have more experience with OSS than the rest of the organization. However, our contact has long experience from the company, we got a high response rate, and the respondents reflect the organization at large. Moreover, when we presented the results at the workshops the audience was allowed to participate, and we did not get any feedback indicating that the results were wrong.

There are many different OSS products available, and these products do not share the same properties. The same holds for proprietary products. Asking about benefits, risks, and steps for reducing risks related to an increased OSS adoption is therefore a bit problematic. We must have in mind that the answers reflect the individual respondent’s perception of OSS and proprietary products. To get more precise data one would need to compare individual OSS products against specific proprietary products.

Table 4. Possible risks and steps for reducing these risks

Potential risks of adopting OSS products	**Possible risk reduction steps**	
	From the Telenor case	**From the literature**
OSS products may lack (professional) support. There may be limited access to expertise, and situations involving unclear liability and division of responsibility may occur. (**R3+RT1+RT4**)	- Place responsibility at an early stage (**RMT1**) - Make sure that your service and support providers support OSS products (find new ones or ask existing ones to extend their service offering) (**RMT1**) - Increase and dedicate internal resources to OSS (**RMT1**) - Increase employee skills (hire new or train existing) (**RM1+RMT1**)	- Encourage local "OSS champions" [16]
Hidden costs related to adopting OSS, replacing existing technology, and changing current processes. (**R1+RT3**)	- Conduct risk assessments - Execute pilot studies and a planned stepwise adoption (**RM2+RMT2**) - Adopt (only) products which show a clear added-value and have a proven track record (**RM4&5**)	- Evaluate the total costs of ownership of OSS products in your own context [35]
Hard to select the right product due to (1) lack of products or products with matching functionality and/or quality, and (2) the amount of products and information available. (**R2+RT2**)	- Adopt only mature products which give clear benefits (**RMT4**) - Dedicate personnel to monitoring the OSS community and selecting OSS products (**RMT1**)	Research suggests several methods for selecting OSS products like for instance [7,9,30]
Uncontrolled adoption and modification, due to the high availability of OSS products, their low purchase price, and the access to these products' source code. (**RT5**)	- Have a plan/strategy behind adopting the various OSS products (**RMT4**) - Adopt (only) products which show a clear added-value and have a proven track record (**RM4&5**) - Standardize on a limited set of technologies/products (**RMT4**) - Begin with a few products (e.g. operating systems, databases, and server applications) - Have specific requirements in call for tenders/requirements specifications that the products should run on OSS platforms - Keep track of the adopted software - Create guidelines for adoption - Dedicate personnel with responsibilities for OSS adoption (e.g. review and monitoring) (**RMT1**) - Conduct risk assessments - Involve management, development, operation, support, (and external partners). (**RMT1**)	- Define a strategy for maintenance and modifications [35] - Set up a central software repository for adopted products [10]

7 Conclusion and Future Work

Based on an extensive literature review and a study from a telecom company, we have identified several risks related to the deployment of OSS products, and several possible steps for reducing these risks. The main results of the studies conducted at Telenor are described in this paper, and the link between our results from a company and the literature is established in Section 5. There are limitations associated with the findings from this paper. Nevertheless, we believe the results of this study are a first step towards focusing the research, on risks of OSS adoption, on more measurable approaches for such evaluation. Finally, our study focuses on bridging the gap between OSS research and practice by focusing on topics highly relevant to practitioners. The study is furthermore an example of how researchers and practitioners may benefit from closer collaboration.

As future work we intend to follow the process of adoption of OSS at this company to further investigate and measure the real effect of the adoption of OSS. A particular focus will be directed towards the relationship between the Telenor IT's internal development and support, and their partners.

We also acknowledge that many of the risks and mitigation steps described in this paper are similar to the ones described in the literature of adoption/diffusion of general information technology e.g. [12, 26]. This research could also lend research on OSS adoption valuable support (see e.g. [13]). We intend to do more research in order to investigate these issues, so we can focus the OSS research on the issues that are mostly related to the OSS adoption, and not part of the general issues related to general adoption/diffusion of information technology.

References

1. Paul Adams, Cornelia Boldyreff, David Nutter, and Stephen Rank. Adaptive Reuse of Libre Software Systems for Supporting On-line Collaboration. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, Karim R. Lakhani, and Walt Scacchi, editors, Open Source Application Spaces: Proceedings of the Fifth Workshop on Open Source Software Engineering (WOSSE 2005), pages 1–4. ACM, 2005.
2. Pär. J.Ågerfalk, Andrea Deverell, Brian Fitzgerald, and Lorraine Morgan. Assessing the Role of Open Source Software in the European Secondary Software Sector: A Voice from Industry. In Scotto and Succi [26], pages 82–87.
3. Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane. Postmortem: Never Leave a Project without It. IEEE Software, 19(3):43–45, 2002.
4. Andrea Bonaccorsi, Silvia Giannangeli, and Christina Rossi. Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. Management Science, 52(7):1085–1098, 2006.
5. Andrea Bonaccorsi and Christina Rossi. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. Knowledge, Technology, and Policy, 18(4):40–64, dec 2006.
6. Daniel Brink, Llewelyn Roos, James Weller, and Jean-Paul Van Belle. Critical Success Factors for Migrating to OSS-on-the-Desktop: Common Themes across Three South African Case. In Damiani et al. [8], pages 287–293.

7. David Cruz, Thomas Wieland, and Alexander Ziegler. Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis. Software Process: Improvement and Practice, 11(2):107-122, 2006.
8. Ernesto Damiani, Brian Fitzgerald, Walt Scacchi, and Marco Scotto, editors. Proceedings of the 2nd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2006) - Open Source Systems, June 8-10, Como, Italy, volume 203/2006 of IFIP International Federation for Information Processing. Springer, 2006.
9. Vieri del Bianco, Luigi Lavazza, Sandro Morasca, and Davide Taibi. Quality of Open Source Software: The QualiPSo Trustworthiness Model. In Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman, editors, Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3-6, Skövde, Sweden, volume 299/2009 of IFIP International Federation for Information Processing, pages 199-212. Springer, 2009.
10. Jamie Dinkelacker, Pankaj K. Garg, Rob Miller, and Dean Nelson. Progressive Open Source. In Will Tracz, Jeff Magee, and Michal Young, editors, Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), May 19th-25th, Orlando, Florida, pages 177–184. ACM, 2002.
11. Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti, editors. Proceedings of the 3rd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2007) - Open Source Development, Adoption and Innovation, June 11th-14th, Limerick, Ireland, volume 234/2007 of IFIP International Federation for Information Processing. Springer, 2007.
12. Robert G. Fichman. Information Technology Diffusion: A Review of Empirical Research. In Janice I. DeGross, Jack D. Becker, and Joyce J. Elam, editors, Proceedings of the Thirteenth International Conference on Information Systems (ICIS '92), December 13th-16th, Dallas, USA, pages 195-206, Minneapolis, MN, USA, 1992. University of Minnesota.
13. Brian Fitzgerald. Open Source Software Adoption: Anatomy of Success and Failure. International Journal of Open Source Software & Processes, 1(1):1–23, 2009.
14. Brian Fitzgerald and Tony Kenny. Developing an Information Systems Infrastructure with Open Source Software. IEEE Software, 21(1):50–55, 2004.
15. Rishab Aiyer Ghosh. Study on the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communication Technologies (ICT) Sector in the EU. Technical report, UNU-MERIT, 2006.
16. Eugene Glynn, Brian Fitzgerald, and Chris Exton. Commercial Adoption of Open Source Software: An Empirical Study. In June Verner and Guilherme Horta Travassos, editors, Proceedings of International Symposium on Empirical Software Engineering (ISESE 2005), November 17th-18th, Noosa Heads, Australia, pages 225–234. IEEE Computer Society, 2005.
17. Sigi Goode. Something for nothing: management rejection of open source software in Australia's top firms. Information & Management, 42(5):669–681, 2005.
18. Simon Grand, Georg von Krogh, Dorothy Leonard and Walter Swap. Resource allocation beyond firm boundaries: A multi-level model for Open Source innovation. Long Range Planning, 37(6): 591-610, 2004.
19. Øyvind Hauge, Claudia P. Ayala, and Reidar Conradi. Open Source Software in Organizations - A Systematic Literature Review. TO APPEAR.

20. Øyvind Hauge, Carl-Fredrik Sørensen, and Reidar Conradi. Adoption of Open Source in the Software Industry. Barbara Russo, Ernesto Damiani, Scott A. Hissam, Björn Lundell, and Giancarlo Succi, editors. Proceedings of the 4th IFIP Working Group 2.13 International Conferences on Open Source Software (OSS2008) - Open Source Development Communities and Quality, September 7th-10th, Milano, Italy, volume 275/2008 of IFIP International Federation for Information Processing, pages 211–222. Springer, 2008.
21. Øyvind Hauge, Carl-Fredrik Sørensen, and Andreas Røsdal. Surveying Industrial Roles in Open Source Software Development. In Feller et al. [11], pages 259–264.
22. Jesper Holck, Michael Holm Larsen, and Mogens Kühn Pedersen. Managerial and Technical Barriers to the Adoption of Open Source Software. In Xavier Franch and Daniel N. Port, editors, Proceedings of the 4th International Conference on Component-Based Software Systems (ICCBSS 2005), February 7-11, Bilbao, Spain, volume 3412/2005 of LNCS, pages 289–300. Springer, 2005.
23. Ari Jaaksi. Experiences on Product Development with Open Source Software. In Feller et al. [11], pages 85–96.
24. Lorraine Morgan and Patrick Finnegan. Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms. In Feller et al. [11], pages 307–312.
25. Bülent Ozel, Uros Jovanovic, Beyza Oba, and Manon van Leeuwen. Perceptions on F/OSS Adoption. In Feller et al. [12], pages 319–324.
26. Everett M. Rogers. Diffusion of Innovations. Free Press, New York, USA, 5th edition, 2003.
27. Marco Scotto and Giancarlo Succi, editors. Proceedings of The First International Conference on Open Source Systems (OSS2005), July 11th-15th, Genova, Italy, 2005.
28. Nicolas Serrano, Sonia Calzada, Jose Maria Sarriegui, and Ismael Ciordia. From Proprietary to Open Source Tools in Information Systems Development. IEEE Software, 21(1):56– 58, 2004.
29. So Young Sohn and Min Seok Mok. A strategic analysis for successful open source software utilization based on a structural equation model. Journal of Systems and Software, 81(6):1014–1024, June 2008.
30. Davide Taibi, Luigi Lavazza, and Sandro Morasca. OpenBQR: a framework for the assessment of OSS. In Feller et al. [11], pages 173-186.
31. Francis Tiangco, Alison Stockwell, John Sapsford, and Austen Rainer. Open-source software in an occupational health application: the case of Heales Medical Ltd. In Scotto and Succi [26], pages 130–134.
32. Kris Ven, Dieter Van Nuffel, and Jan Verelst. The Introduction of OpenOffice.org in the Brussels Public Administration. In Damiani et al. [8], pages 123-134.
33. Kris Ven and Jan Verelst. The Organizational Adoption of Open Source Server Software by Belgian Organizations. In Damiani et al. [8], pages 111–122.
34. Kris Ven and Jan Verelst. The Impact of Ideology on the Organizational Adoption of Open Source Software. Journal of Database Management, 19(2):58–72, April 2008.
35. Kris Ven, Jan Verelst, and Herwig Mannaert. Should You Adopt Open Source Software? IEEE Software, 25(3):54–59, 2008.
36. Kris Ven and Herwig Mannaert. Challenges and strategies in the use of Open Source Software by Independent Software Vendors. Information and Software Technology, 50(9-10):991{1002, August 2008.

PAPER 8

P9 Øyvind Hauge, Claudia P. Ayala, and Reidar Conradi. *Open Source Software in Organizations - A Systematic Literature Review*. Submitted to Information and Software Technology (IST) on December 3rd 2009.

Adoption of Open Source Software in Software-Intensive Organizations - A Systematic Literature Review

Øyvind Hauge[*,1], Claudia Ayala[1,2], Reidar Conradi[1]

IDI, NTNU, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway

Abstract

Context: Open source software (OSS) is changing the way organizations develop, acquire, use, and commercialize software.
Objective: This paper seeks to identify how organizations adopt OSS, classify the literature according to these ways of adopting OSS, and with a focus on software development evaluate the research on adoption of OSS in organizations.
Method: Based on the systematic literature review method we reviewed publications from 24 journals and seven conference and workshop proceedings, published between 1998 and 2008. From a population of 24289 papers, we identified 112 papers which provide empirical evidence on how organizations actually adopt OSS.
Results: We show that adopting OSS involves more than simply using OSS products. We moreover provide a classification framework consisting of six distinctly different ways in which organizations adopt OSS. This framework is used to illustrate some of the opportunities and challenges organizations meet when approaching OSS, to show that OSS can be adopted successfully in different ways, and to organize and review existing research. We find that this research does not sufficiently describe the context of the organizations studied. It is furthermore fragmented and fails to benefit fully from related research fields. Finally, we present directions for future research.
Conclusion: Practitioners should embrace the many opportunities OSS of-

[*]Corresponding author. Tel: +47 73 59 07 31; Fax: +47 73 59 44 66
Email addresses: `oyvind.hauge@idi.ntnu.no` (Øyvind Hauge), `cayala@lsi.upc.edu` (Claudia Ayala), `reidar.conradi@idi.ntnu.no` (Reidar Conradi)
[1]Norwegian University of Science and Technology
[2]Technical University of Catalunya

fers, but consciously evaluate the consequences of adopting it in their own context. Practitioners may use our framework and the success stories provided by this literature review to understand how they can benefit from OSS. Researchers should align their work and perform more empirical research on topics which are important to organizations. Researchers may use our framework to position their work and to describe the context of the organization they are studying.

Key words:
open source software, organizations, software development, systematic literature review

1. Introduction

The open source software (OSS) phenomenon has over the last decade had a significant impact, not only on the software industry, but also on software-intensive organizations in both the public and private sector. The collaborative development model often associated with OSS communities has introduced a new software development model. This model has inspired software companies into evolving their existing development processes [60, 203] and collaborating both internally and across company borders [7]. Next, it is claimed that the existence of freely available software allows faster adoption of technology, increased innovation, and reduced costs and time-to-market [33, 139]. These potential advantages have influenced how organizations acquire software, and have led to a significant adoption of OSS products in several domains [89, 98, 140, 187]. Finally, OSS and its general lack of license fees contribute to shifting the software industry's traditional license-based business models towards service-based models [79]. Hence, OSS is significantly influencing the ways organizations develop, acquire, use, and commercialize software [69].

It is therefore vital to help organizations in meeting the challenges related to OSS, and to align our research efforts with their real needs. The topic for this study is therefore *adoption of OSS in software-intensive organizations, with a particular focus on software development.* To identify what we know about how organizations adopt OSS we have performed a systematic literature review following the guidelines proposed by Kitchenham [115]. With a focus on software development, this systematic literature review seeks to evaluate, synthesize, and present the empirical research results on OSS within organizations.

The targeted audience for this systematic review is primarily researchers in the OSS, software engineering, and information systems fields, wanting to study settings involving OSS and organizations. However, organizations and practitioners which adopt (or plan to adopt) OSS may also appreciate the review.

This systematic literature review contributes to the literature and ongoing research on OSS within the software engineering and information systems domain in three ways: (1) by reviewing and summarizing what we know about how organizations actually leverage OSS, (2) by providing a classification framework for how organizations adopt OSS, and (3) by offering directions for future research on OSS in organizations.

Researchers and practitioners may use these contributions to more clearly understand the practical challenges when adopting OSS, and properly align their efforts for facing them. Researchers may use this literature review to get an overview of current research, identify new research questions, and position and align their own work. More importantly, they may use our classification framework to describe and discuss the context of the organizations they study. Practitioners may use this framework and the success stories provided here to understand how they may leverage OSS in their own context, and to identify the practical challenges they might face when doing so.

The remainder of this paper is structured as follows: In Section 2 we provide a brief background to OSS and an overview of other reviews of the OSS literature. We develop a classification framework for how organizations adopt OSS. In addition, we relate OSS to relevant research areas and present the objectives of this study. In Section 3 we describe the systematic review process. In Section 4 we characterize the literature on adoption of OSS in organizations and use the classification framework to present key issues from this literature. In Section 5 we answer our research questions and discuss the results with their implications and limitations. Moreover, we provide directions for future research. In Section 6 we conclude the paper.

2. Background

In this section we give a brief background on the OSS phenomenon and summarize other reviews on OSS research. We present a classification framework for organizational adoption of OSS. Then, we relate research on OSS to relevant areas in software engineering and information systems research. Finally, we present the objectives for this literature review.

2.1. Open Source Software

Eric Raymond describes the development of OSS as a bazaar-like activity driven by volunteers, and claims that OSS is cheaper, has fewer defects, gets improvements faster, and is generally better than "other kinds" of software [154, 155]. Based on the Apache and the Mozilla projects, Mockus et al. [136] describe OSS development as controlled by groups of core developers and supported by large communities of contributors. They hypothesize that the OSS products have lower defect density than commercial software and that OSS development rapidly responds to user requests. Others, e.g. Crowston and Howison [44] and Scacchi [166], support this view and claim that the development in OSS communities is distinctly different from traditional software development.

This view of OSS and OSS development as being something radically different has triggered research on a variety of topics. These include OSS seen as a new innovation model [194], the motivations of OSS developers [102], OSS business models [27] and a wide spectrum of other research topics in computer science, management and organization science, social science, psychology, economics, and law [85].

Software engineering research has for instance studied self-organizing in OSS communities [45, 212], user-to-user support [119], knowledge management [173], and quality assurance [215]. Software engineering researchers have additionally used OSS products to study general software engineering problems like evolution [211], cloning [114], and the use of metrics to identify error prone classes [170].

However, the view that the development of OSS is something radically different from traditional software development is questioned by, for instance Fitzgerald [78] and Fugetta [84]. Østerlie and Jaccheri [146] offer a critique of how OSS development has been described as a homogeneous phenomenon in the software engineering research literature. The literature has not reflected the variety observed in the OSS phenomenon, but rather has focused on large, successful, and community-driven OSS projects. Moreover, Capiluppi et al. [40] provide evidence that the majority of OSS projects struggle to attract contributors, Noll [142] shows that OSS can also be developed inside commercial software development companies without any active communities, and Stamelos et al. [177] show that the quality of OSS software is not always as good as expected. Finally, Fitzgerald [79] argues that the OSS phenomenon has evolved into a more commercially viable form where volunteers and commercial organizations collaboratively contribute to evolving the phenomenon.

There are thus conflicting views on what the OSS phenomenon actually is and there is not even consensus on which label to use on the phenomenon. We acknowledge that there are (minor) differences between open source, free software, and free (libre) open source software (FOSS/FLOSS). However, this ongoing debate is beyond the scope of this paper. We will instead treat OSS, free software, and FLOSS as synonyms, and focus on software development and the parts of the phenomenon where commercial organizations are involved. We will in particular look at three aspects of it related to organizations: (1) the use of software products licensed with a license approved by the Open Source Initiative [145], (2) the interaction with the communities surrounding many OSS products, and (3) the use of the collaborative software development practices often associated with many of these communities.

2.2. Summary of Previous Reviews

Reviews of the literature on OSS are given by Feller et al. [71], Stol and Babar [180], Scacchi et al. [167], von Krogh and von Hippel [196], and the aforementioned paper by Østerlie and Jaccheri [146]. While all these reviews are on OSS, none of them focus on OSS in organizations. In fact, the adoption of OSS in organizations is hardly mentioned by any of the authors.

Feller et al. [71] aim to identify the kinds of OSS communities that have been studied, the kinds of research questions which have been asked, and the methods researchers have used to answer these questions. The paper mainly focuses on classifying and characterizing a set of 155 publications on OSS. Feller et al. find that the OSS research literature has large gaps, and that commercial organizations are underrepresented as subjects in the research on OSS.

Stol and Babar [180] reviewed 219 publications from the four first International Conferences on Open Source Systems. Like Feller et al. [71], Stol and Babar focus on assessing the quality of the 63 empirical studies and find that the literature needs to be improved. To this end, they offer a set of guidelines for improving the quality of studies on OSS. Moreover, they classify the empirical papers into: research on OSS communities (39.7% of the papers), OSS development and maintenance (20.6%), diffusion and adoption of OSS (28.6%), and characteristics of OSS (11.1%). Stol and Babar thereby confirm that OSS in organizations has attracted limited attention.

In an introduction to a special issue, Scacchi et al. [167] provide an overview of the research on the development processes found in OSS projects. Von Krogh and von Hippel [196] give an overview of some of the research on OSS and organize it into three categories: motivations of contributors, innovation processes, and competitive dynamics.

2.3. A Classification Framework for Organizational Adoption of OSS

To identify the challenges organizations face when approaching OSS and to classify the literature, we developed a classification framework consisting of six ways organizations adopt OSS. By *ways of adopting OSS*, we think of ways in which software-intensive organizations can benefit from OSS products, the communities surrounding many of these product, or the development practices often associated with the collaborative development of many such products. We limit our focus to *software-intensive organizations* which we define as private or public organizations extensively using or developing software. Moreover, we focus on ways of adopting OSS which influence an organization's software infrastructure or software development.

We briefly present this framework in Table 1. By using the empirical evidence identified in this review we will discuss some of the relations between the different categories in Section 5.1.

2.3.1. The Classification Framework

The classification framework in Table 1 contains two main areas in which organizations can benefit from OSS. First, *deploying OSS products* entails the use, and if necessary configuration, of a spectrum of software. These products range from infrastructure software (like operating systems, databases, and application servers), through server-based software applications, to desktop applications. Second, *using OSS in software development* can be broken down into five categories.

Using OSS CASE tools involves using tools like integrated development environments (IDEs), compilers, modeling tools, and so on. The use of OSS CASE tools is indeed an example of OSS deployment. Still, we decided to keep OSS CASE tools as a separate category for three reasons. First, the focus of this review is software development, where CASE tools are exten-

Table 1: Organizational adoption of OSS

Way of adopting OSS		**Example papers**
Deploying OSS products in their operation environment as end users (e.g. deploying OpenOffice.org, Linux, JBoss)		[62, 81, 190]
Software development	Using OSS CASE *tools* in software development (e.g. using Eclipse, Subversion, GCC)	[13, 131]
	Integrating OSS components into their own software systems (e.g. integrating or extending Hibernate, Google Web Toolkit, Plone)	[8, 43, 186]
	Participating in the development of OSS products controlled by another organization or community (e.g. contributing to Linux, Eclipse, OpenOffice.org)	[28, 101, 158]
	Providing their own OSS products and relating to a community around these products (e.g. providing MySQL, Qt, JBoss)	[7, 17, 202]
	Using software development *practices*, often associated with OSS communities, within a company or consortium of companies (e.g. using practices like code sharing, peer reviewing, user contributions)	[61, 135, 203]

sively used. Second, there are large numbers of OSS CASE tools available[3]. Third, there is already an established research field focusing on Computer Aided Software Engineering (CASE) tools.

The integration of OSS components involves including OSS components into other software products or systems. This integration may involve modifying, extending, or wrapping the OSS components. Even though both deployment and integration of OSS entail reusing OSS products, it is valuable to separate the two. Organizations which extend and possibly modify an OSS product increase their dependence on the product and face additional challenges related to maintenance. The difference between simply deploying an OSS product and integrating it into one of your software systems is one of degrees. For instance, building applications with tight integration of e.g. architectural frameworks and persistence layers poses significantly different challenges than simply deploying a desktop application.

By participating in the development of OSS, we mean the involvement of organizations in existing OSS communities, although without having decisive control over the OSS product or the community. Providing an OSS product, involves organizations like JBoss, MySQL, and Qt Software, which develop and release OSS products, control the development of these products, and relate to the community around them. The difference between the two categories is again one of degrees. However, the challenges tied to relating to a community of thousands of users around one of your own products

[3]For instance `http://www.tigris.org/` has more than 500 such tools

are different from those related to contributing a bug-fix to a product controlled by someone else. The division between providing and participating is also noticed by, for instance, Ågerfalk and Fitzgerald [7] and Dahlander and Magnusson [50].

There is no set of development practices which are universal to all OSS projects. Nevertheless, practices such as user participation, short release cycles, and peer code reviews have frequently been associated with OSS projects and are often labeled "OSS practices" [72, 166, 215]. Lately, several organizations have tried to learn from the development practices in successful OSS projects, through applying these practices within their own organization [60, 135, 203].

2.3.2. Related Classifications

The framework extends earlier work in three ways. First, we identify new ways of adopting OSS as compared to Hauge et al. [99] and Ziemer et al. [216]. In [99], we present the four roles of OSS integrator, OSS participant, OSS provider, and inner source software participant as possible ways of approaching OSS. In [216], we present cases from companies which adopt OSS through development with OSS practices and tools, development with OSS products, and development of OSS products. Second, we have a somewhat broader scope than Dahlander and Magnusson [49, 50], who focus on the relationships between organizations and OSS communities. Third, we focus on software development rather than resource allocation as in Grand et al. [94], or business models as in Hecker [100]. However, the levels in the four-level ladder for resource allocation presented by Grand et al. [94] are partly compatible with some of our categories. Table 2 relates our framework to relevant classifications in [94, 99, 216].

Table 2: The framework and its mapping to related classifications

Way of adopting OSS		**Mapping to other papers**
Deploying OSS products		Level 1 [94]
Software development	Using OSS CASE tools	Level 1 [94] and Development with OSS practices and tools [216]
	Integrating OSS components	Level 2 [94], OSS integrator [99], and Development with OSS products [216]
	Participating in OSS communities	Level 3 [94], OSS participant [99]
	Providing OSS products	Level 3/4 [94], OSS provider [99], and Development of OSS products [216]
	Using OSS development practices	Inner source software participant [99], and Development with OSS practices and tools [216]

2.4. Related Research Fields

In their effort to define OSS, Gacek and Arief [85] consider research fields like computer science, management and organization science, social science, psychology, economics, and finally law as relevant. To put OSS research into context we will relate it to relevant research areas. However, as this literature review focuses on software development, we will consider research areas only within software engineering and information systems. While we draw parallels between OSS and some related research areas in Table 3, it is not an extensive list.

Table 3: OSS research in relation to other research areas

Way of adopting OSS	**Related research areas**
Deploying OSS products	Introduction, deployment, diffusion, and acceptance of information systems (IS) and information technology [191, 192]
Using OSS CASE tools	Computer Aided Software Engineering (CASE) [83, 207]
Integrating OSS components	Component-Based Software Engineering (CBSE) [35, 122, 134, 208] and software reuse [137, 193]
Participating in OSS communities	No clearly related research area within SE/IS. However, Ågerfalk and Fitzgerald relate their research with offshoring and outsourcing [7]
Providing OSS products	
Using OSS development practices	Software process improvement [1, 65], distributed development [150], global software development [174], and agile development [199]

In this review we will consider three ways of using OSS products. OSS products may first of all be deployed as-is without any changes, used as CASE tools in software development, or integrated into other software systems. All these three ways of using OSS products are related to established research areas within software engineering and information systems research.

The introduction, diffusion, and acceptance of information systems and information technology have already been studied for a long time in the information systems field [191, 192]. As we mentioned above, there is an established research area on CASE tools [83, 207]. Finally, the integration of OSS components is closely related to research on, for instance, Component-Based Software Engineering (CBSE) [35, 122, 134, 208] and software reuse [137, 193].

A large part of the OSS phenomenon is centered on community interaction, either as a provider of an OSS product or as a participant in a community controlled by someone else. Within the software engineering and information systems research fields, we find no clearly related research areas. Nevertheless, Ågerfalk and Fitzgerald [7] relate their research on company intervention in OSS communities with offshoring and outsourcing.

Finally, Scacchi et al. [167] discuss the software development processes and practices used in OSS communities. Such work can be related to, for instance, software process improvement [1, 65]. Other researchers relate the development processes in OSS communities to distributed development [150], global software development [174], and agile methods [199].

2.5. Objectives of this Review

Our overall objective of summarizing what we know about how organizations adopt OSS has been broken down into three more concrete research questions:

RQ1 In what ways are software-intensive organizations adopting OSS?

RQ2 What has been the focus of the empirical research on adoption of OSS in organizations?

RQ3 What are the characteristics and limitations of current, empirical research on organizational adoption of OSS?

Even though we see that organizations approach OSS in different ways there are several publications discussing organizational adoption of OSS without clarifying how or what the involved organizations actually do related to OSS. Some papers discuss "F/OSS usage and adoption" in public administration [148], or companies which "have entered the open source field" [27] and "are active in the OSS domain" [153]. Understanding the practical implications of a specific way of adopting OSS is therefore difficult. So, by RQ1, we want to identify existing ways of leveraging OSS.

When identifying these ways of adopting OSS we will in particular focus on organizations which develop software and approaches to OSS related to software development. This focus is also valid for RQ2. Through RQ2, we seek to identify the focus of research on OSS in organizations, but with a particular focus on software development. Finally, by RQ3 we aim to characterize the research on OSS in organizations, assess its quality, and in particular identify its limitations.

3. Research Method

The evidence-based software engineering (EBSE) paradigm aims to integrate the current best evidence from research with practical experience [117]. Literature reviews, and in particular systematic literature reviews, have therefore become popular within the software engineering research field as a means of evaluating what we know in a specific area. For instance, researchers have reviewed published evidence on search-based testing [5], knowledge management [21], cost estimation [111], and several other topics [116].

To answer our research questions, we systematically assessed existing evidence related to the adoption of OSS in organizations, using Kitchenham's guidelines for systematic literature reviews [115]. The review process was split into several stages, each of which was performed individually by two researchers followed by an iterative process to reach consensus before facing a new stage. The following subsections describe the steps of the literature review.

3.1. Search and Classification Process

The search process combined searching digital libraries with manual evaluation and classification of the results. Figure 1 presents an overview of the review process and the number of publications included in each stage.

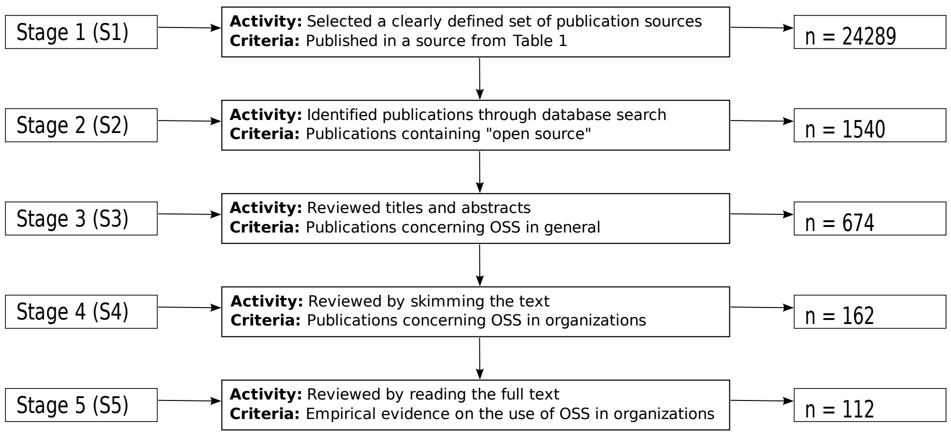

Stage 1 (S1)
Activity: Selected a clearly defined set of publication sources
Criteria: Published in a source from Table 1
n = 24289
Stage 2 (S2)
Activity: Identified publications through database search
Criteria: Publications containing "open source"
n = 1540
Stage 3 (S3)
Activity: Reviewed titles and abstracts
Criteria: Publications concerning OSS in general
n = 674
Stage 4 (S4)
Activity: Reviewed by skimming the text
Criteria: Publications concerning OSS in organizations
n = 162
Stage 5 (S5)
Activity: Reviewed by reading the full text
Criteria: Empirical evidence on the use of OSS in organizations
n = 112

Figure 1: Stages of the study selection process (Adapted from [66])

3.1.1. Stage 1 - Defining a Set of Publications

To have a clearly defined set of publications serving as a basis for this study, we selected a set of publication channels rather than openly searching available digital libraries. Relevant journals and conferences were taken from previous literature reviews on software engineering [66, 92, 113, 116, 171] and OSS [71, 146, 167, 196]. Table 4 gives an overview of the final sample of publication sources.

Appendices A and B contain more detailed overviews of where we collected the publications, the publication sources' total number of publications between 1998 and 2008, the number of publications included in each of the five stages (S1-S5), and so on.

Table 4: Publication sources

	Software Engineering	**OSS, Information Systems, and Management**
Journals/magazines	• ACM Transactions on Software Engineering and Methodology (TOSEM) • Communications of the ACM • Empirical Software Engineering (EMSE) • IEE Review • IEE Software Proceedings/IET Software • IEEE Computer • IEEE Software • IEEE Transactions on Software Engineering (TSE) • Information and Software Technology (IST) • Journal of Systems and Software (JSS) • Software Practice and Experience • Software Process: Improvement and Practice	• First Monday • Information Systems Journal (ISJ) • Journal of Database Management • Journal of Industrial Economics • Knowledge Technology and Policy • Long Range Planning • Management Science • MIS Quarterly • MIS Quarterly Executive • MIT Sloan Management Review • Organization Science • Research Policy
Conferences/workshops	• IEEE International Symposium on Empirical Software Engineering (ISESE) • IEEE International Symposium on Software Metrics (METRICS) • International Conference on Software Engineering (ICSE) • International Symposium on Empirical Software Engineering and Measurement (ESEM)	• ICSE Workshop on Open Source Software Engineering (WOSSE) • ICSE Workshop on Emerging Trends in FLOSS Research and Development • International Conference on Open Source Systems (OSS)

3.1.2. Stage 2 - Searching with Keywords in Digital Libraries

Publications on OSS from the specific journals and conference proceedings were identified by searching through a variety of digital libraries and manually reviewing several static web pages. Only papers published in English between 1998 and 2008 were considered.

To avoid overlooking relevant publications, we opted for a search strategy with high sensitivity [58]. This means that instead of using keywords like "commercial open source" and "open source in industry", we conducted all searches using the keywords "open source" (including quotation marks) and searched the digital libraries using all fields, including full text where available. For IEEE Xplore only metadata search was used, even though we discovered late in the review process that it was also possible to search the database using full text search (see the discussion in Section 5.5).

To verify that using the keywords "open source" included all relevant publications, we also tried searching using the keywords "free software". However, relevant papers also contained the keywords "open source".

The searches were conducted from November 2008 and finally revised January 2009. Bibliography for all the publications was stored in the external bibliography system[4].

3.1.3. Stage 3 - Reviewing Titles and Abstracts - Papers on OSS in General

To identify publications which in fact were about OSS and did not just contain the keywords "open source", we individually reviewed the 1540 papers from the previous stage based on their titles and abstracts, and if necessary by skimming the full text. Only papers on OSS topics like communities, software development, licensing, business models, adoption, use, and software engineering were included. Papers on other open, collaborative activities (open courseware, wikinomics, open access) were rejected. Additionally, we rejected introductions of panels, conferences, and special issues, book reviews, news flashes, and PhD symposium papers.

First, a total number of 763 publications were included by either of the two first authors. The agreement between the authors was very good (Kappa value of 0.83). A Kappa value between 0.81-1.0 is an almost perfect agreement and between 0.61-0.80 is a substantial agreement [120]. After two joint consensus iterations we discarded 89 of the 763 papers and ended up including 674 publications.

3.1.4. Stage 4 - Skimming the Text - Papers on OSS in Organizations

Next, to identify publications on adoption of OSS in organizations, we individually went through the output of the third stage and evaluated the papers' topics by reviewing the titles and abstracts, and by skimming the papers. Publications on adoption, use, and development of OSS in organizations were included, while those not related to OSS in software development or actual use of OSS products were rejected. This included approaches to the OSS phenomenon from economical or social sciences and papers on innovation theory, business models, etc. Moreover, papers proposing methods or (classification) frameworks without any empirical validation were rejected.

[4]Aigaion: `http://aigaion.nl/`

Initially 211 publications were included by either one of the two first authors. The agreement was again quite good (Kappa value of 0.68). After another joint iteration we rejected 49 papers and classified 162 publications as being relevant to OSS in organizations.

3.1.5. Stage 5 - Reading the Text - Papers with Empirical Evidence

Then, we classified the publications into three categories inspired by [138]; (R) empirical research papers where the authors present evidence from a research study having an explicit research question, (E) experience reports where the authors report experiences without having defined an explicit research question, and (N) non-empirical papers. The non-empirical category includes opinion papers and theoretical papers without explicit empirical evidence.

Of the 162 included papers, 59 were classified as empirical research papers (R), 53 as experience reports (E), and 50 as non-empirical papers (N). Non-empirical papers may increase the understanding of how organizations adopt OSS, but they are not providing any new evidence of how organizations actually do this. Hence, these papers were not included. Accordingly, the final stage of the review included 112 publications.

3.2. Quality Assessment

As the research field on OSS adoption is still immature, and since there are no other review papers on the same topic, we did not want to exclude papers because of their lack of rigor. The quality assessment was therefore performed only to evaluate the rigor of the presented research in each publication.

We assessed the 59 empirical research papers using the nine quality metrics (QM) presented below. This schema is inspired by other SLRs [66, 178], and the evaluation criteria used in different journals and conferences [138]. The schema was designed to contain only binary values (yes/no). The quality of the experience reports was not assessed as they did not contain any explicit research questions and most often no descriptions of method, findings and so on.

QM1: Does the paper have a description of the research method?
QM2: Does the paper describe an explicit research question/goal/purpose?
QM3: Does the paper describe motivation for the research question(s)?
QM4: Does the paper discuss limitations or validity?
QM5: Does the paper describe the context of the research?

QM6: Does the paper describe data collection?
QM7: Does the paper describe data analysis?
QM8: Does the paper describe sampling or selection of the study object(s)?
QM9: Does the paper present any data?

The quality assessment was performed by one of the two first authors and verified by the other. We simply verified whether or not the publications mentioned or discussed issues related to each of the quality metrics. Assessments of the extent to which a paper actually fulfilled each of the quality metrics, and assessments of the papers' relevance to practitioners were not performed. Any differences between the two first authors were solved through discussions until a consensus was established.

3.3. Data Extraction

In the data collection stage we extracted the following from each of the publications: main topic, research question or research goal, research method, study objects, a brief description of empirical evidence relevant to use of OSS in organizations, and the affiliation and home country of the first authors.

Following the recommendations by [36], one researcher extracted the data, while the other confirmed the extracted data. Similar to [67], we frequently used consensus meetings to extract data. These meetings were also used to identify the topics which are discussed in the next sections of this review.

4. Results

In this section we describe the results from the evaluations of the publications included in the final stages of this review. Based on these evaluations we present some characteristics of the literature on OSS in organizations. Then, we use the classification framework from Section 2.3 to give an overview of the topics discussed in the literature, and to identify the most important findings and challenges related to software development.

4.1. Characteristics of the Literature

Here we characterize the literature by presenting the number of publications per year, the contexts in which research has been performed, and the applied research methods. Finally, we present the results from the quality assessment. These overviews focus mainly on the 59 empirical research papers. More detailed data can be found in Appendix B.

4.1.1. OSS Related Activity in the Literature

Both the number of publications related to OSS in general (S3) and to OSS in organizations (S4) has increased significantly the last decade. However, this increase seems to have stabilized in the last years, with about 100 publications per year on OSS and between 20 to 30 publications on topics related to OSS in organizations (see Figure 2). While the results show a significant lack of empirical research papers on OSS in organizations until 2003, the number of such papers has increased since then.

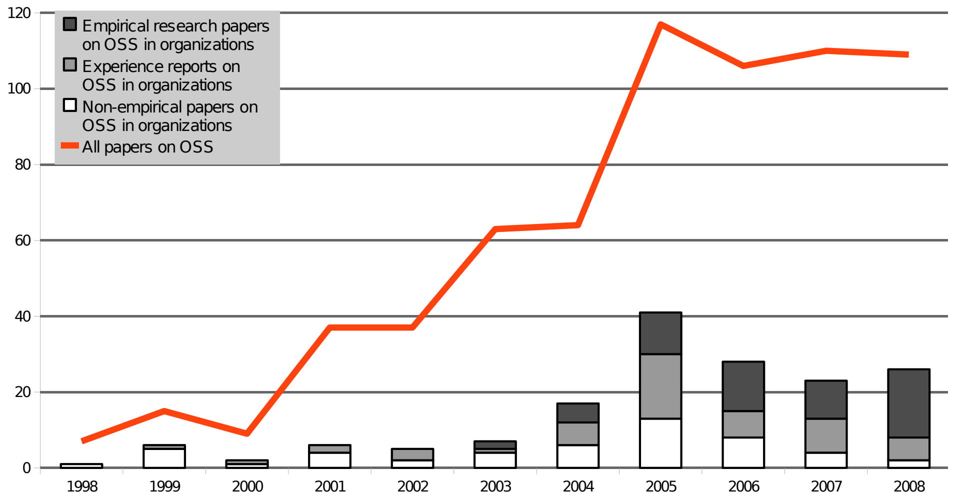


Figure 2: The number of papers on OSS and OSS in organizations

The distribution of the papers with respect to our classification framework is shown in Figure 3. The majority of the 112 empirical papers could be put into one of the six categories. However, 21 papers discussed issues relevant to two or more categories. Another 19 papers were not specific to any particular way of adopting OSS, but rather discussed topics related to adoption of OSS in general. Hence, it was necessary to add another category (OSS adoption in general) to classify all the publications. It is, however, important to note that Table 1 shows how an organization may adopt OSS, while Figure 3 presents a classification of the empirical papers on OSS in organizations.

4.1.2. The Contexts in which Research Is Done

We see that almost all the papers have a first author from Europe (49 empirical research papers (R) and 29 experience reports (E)) or North America

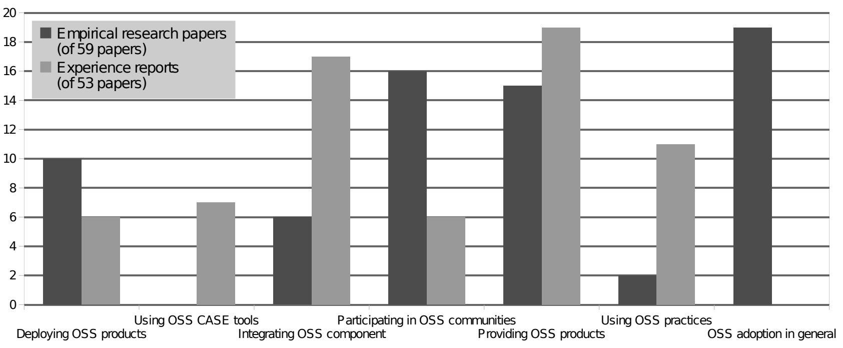


Figure 3: The number of papers in each category

(7 R and 22 E). Publications from the other continents are almost non-existing, as few publications occur from Australia (1 R and 0 E), Asia (2 R and 1 E), and Africa (1 R and 0 E). The USA (6 R and 22 E) and Italy (13 R and 8 E) were the two countries with by far the most publications.

From Figure 4 we see that the contexts where the research is performed come from both the private and public sector. However, the papers often focus on large communities such as GNOME and Debian GNU/Linux, portals like SourceForge [20, 28, 101, 158], large companies as Nokia, Philips Medical, and Hewlett Packard [107, 135, 203], and well known OSS companies like MySQL and JBoss [49, 202]. However, this bias was not as significant as expected.

A few other issues are worth mentioning. First, all of the eight empirical research papers from the public sector focus on deployment of OSS products. Besides [37], which has a mixed sample, no paper focuses on deploying OSS in the private sector. Second, 27 of the 59 empirical research papers report findings from samples of several organizations from the private sector. However, as few as eight papers report findings from one single private organization. Hence, most research papers dedicate relatively little space to describing the individual organizations.

4.1.3. The Research Methods

We classified 53 of the 112 empirical papers identified in this review as experience reports. Hence, the most common method of studying the OSS phenomenon in organizations is through experience reports. These experience reports lack explicit research questions, and most also lack a method description.

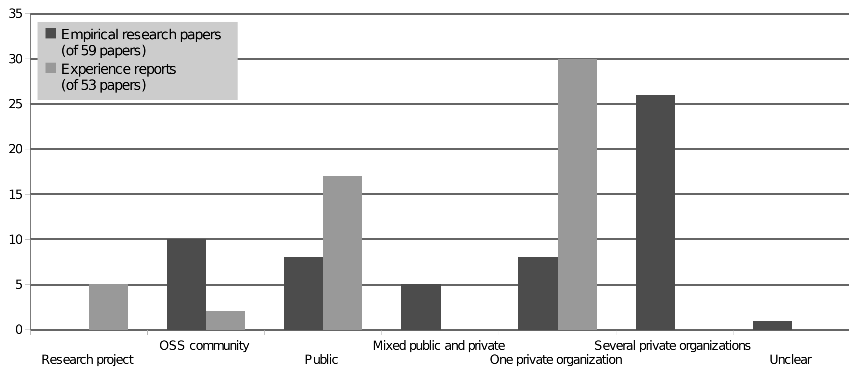


Figure 4: The context of the empirical research papers and the experience reports

Table 5 presents an overview of the research method reported in the 59 empirical research papers. Based on the classification provided by Glass et al. [92], we observed a focus on case studies and surveys.

Table 5: Research methods used

Research method	**Number of publications**
Case study	29
Survey	18
Data analysis	6
Experiment	2
Case study and survey	2
Field study	1
Grounded theory	1

Almost all of the studies were retrospective and gathered information about past events. This was often done through the use of interviews and questionnaires. Moreover, most of the studies collected information at one point in time. Of the 15 publications we classified as longitudinal, seven presented research in the form of mining and analysis of historical data.

4.1.4. Quality Assessment of the Research on OSS in Organizations

As described in Section 3.2, we developed nine quality assessment metrics. Each of the empirical research papers were evaluated to either cover or not to cover these metrics. This was indicated by assigning each of the nine metrics a binary value of either 1 (cover) or 0 (do not cover). Most of the 59 empirical research papers got a relatively high score in this quality assessment (see Table 6). The median score of this assessment was 8, the mode 9, and the mean value was 7.0.

Table 6: Quality assessment: Distribution of research papers

Quality assessment score	0	1	2	3	4	5	6	7	8	9	Total
Number of papers	0	0	2	5	1	3	8	8	15	17	59

The binary metrics unfortunately do not reflect the precise extent to which the papers deal with the issues covered by each of the quality metrics (see further discussion in Section 5.5). In fact, most papers do little more than mention such issues as research questions, limitations, data analysis, and so on. We furthermore found that many of the publications lack details about the research methods and findings. As a consequence, several papers have limitations when it comes to how they describe these issues. Moreover, many of the research papers are explorative and they are therefore lacking a precise focus and clear contributions.

4.2. Research on OSS in Organizations

In this section we present the focus of the research literature on OSS in organizations. This overview is presented according to the classification framework presented in Table 1. Given the review's focus on software development, we will emphasize the findings and challenges relevant to software development.

In the overview we will focus on the 59 empirical research papers, and will give a brief overview of all the included empirical research papers in Tables 7 to 12. However, we will in some cases include some of the experience reports as well. References to all the experience reports may be found in Appendix B.

4.2.1. General Topics on Adoption of OSS

As many as 19 research papers cover topics which are general to OSS adoption, rather than directly related to the individual ways in which organizations adopt OSS. These papers discuss mainly (1) the perceived benefits and drawbacks of OSS or the motivations for adopting it [33, 139], (2) the success factors for adoption of OSS [93, 172], and (3) the extent to which OSS is actually adopted [98, 141]. See Table 7 for an overview of the empirical research papers.

4.2.2. Deploying OSS Products

In total, 10 research papers and six experience reports focus on the deployment of OSS products. The majority of these are short papers (six pages

Table 7: Focus and findings from the empirical research papers on adoption in general

ID	Topic
[6]	Discusses different strengths/opportunities and threats/weaknesses to OSS in the secondary software industry. Concludes that the industry is aware of both the pros and cons.
[27]	Companies select hybrid (OSS and proprietary software) business strategies when approaching OSS.
[29]	Of 769 companies only 19 provide OSS (based) solutions alone and 236 provide both proprietary and OSS (based) ones.
[32, 33, 162]	Companies emphasize (pragmatic) economic and technological motivations for approaching OSS. However, intrinsic motivations may play a role as well.
[91]	Motivations for (reduced costs, freedom from vendors) and extent of adoption of OSS products (used by all respondents).
[93]	OSS adoption is complex. Develops a framework for adoption of OSS products consisting of an external environment, individual factors, and organizational and technological contexts.
[94]	Develops a four level model for resource allocation to OSS, showing varying dedication to OSS.
[99]	Illustrates four different ways of adopting OSS.
[98]	Shows that about 50% of the Norwegian software industry integrates OSS into their products, that about 16% participate in the development of OSS products controlled by someone else, and that about 5% provides their own OSS products.
[127]	OSS ideas may be adopted in different ways. Uptake of OSS is high. The interest in OSS is beyond simply using the LAMP stack.
[139]	Discusses several benefits and drawbacks of OSS.
[141]	About 50% of the software companies in the Finnish survey use OSS products in their business.
[153]	Organizations are only willing to collaborate if they get financial gains. Discusses motivations for approaching OSS.
[172]	Analyzes different factors which may influence the adoption of OSS products in software development.
[188]	Motivations for using OSS server software: lower cost, higher reliability, availability of external support. Source code only interesting for integrators.
[189]	Organizations are pragmatic in the adoption of OSS. The influence of ideology could matter in smaller organizations.
[204]	Discusses hybrid (open and proprietary) business strategies in large companies.

or less) discussing the experience from one context with one or a few OSS products. Most of these papers illustrate successful use of one or more OSS products (see Table 8 for an overview of the research papers).

Table 8: Focus and findings from the empirical research papers on deploying OSS

ID	Topic
[91]	Respondents have not performed cost analysis.
[37]	Identifies seven critical success factors for migrating to an OSS desktop: financial motivation, management support, user awareness, planning planning, analysis and testing, training, pilots, and support.
[62]	Overcoming barriers for migration of the desktop environment. Migration of the desktop could be hard work. Stakeholders' attitude may vary and change over time.
[148]	Identifies several enablers and inhibitors of deployment of OSS products
[161]	Monitors the introduction of OpenOffice.org.
[159]	Monitors the use of OpenOffice.org. Illustrates significant adoption.
[160]	Illustrate the public sectors' strong commitment to proprietary document formats, despite using OpenOffice.org
[164]	Monitors the introduction of OpenOffice.org. User's lack of experience is a barrier.
[187]	OpenOffice.org migration is possible, but not problem free. Issues often specific to environment. Training is important. No formal cost analysis was performed.
[190]	Benefits of OSS depend on context. Comparing "OSS" directly with "proprietary software" in general is futile. Respondents did not perform cost analysis. This should be done.

While the deployment of OSS can give cost savings, migrating from one technological platform to another is hard work which often includes customization, adaptation, integration, and testing [62, 187].

The general costs related to such a migration are unclear [62, 187], and there are very few studies showing complete calculations of the true costs and savings of (1) introducing OSS products into organizations, and (2) keeping the OSS products operational over a longer period of time. One paper reports cost savings from an OSS migration project at Beaumont Hospital [81], but it is published just after the initial stage of the project is finished.

Despite this unclarity, many organizations seem to be blinded by the perceived advantages of OSS and have therefore adopted it without performing any cost-benefit evaluations in their own context [91, 187, 190]. The adoption of OSS is furthermore frequently bottom-up, in the sense that it is introduced by engineers rather than strategic top-level decisions [188].

4.2.3. Using OSS CASE Tools

Despite having a close relation to the CASE research field, only seven experience reports discuss the use of OSS CASE tools in the context of organizations. Given the amount of OSS CASE tools available, it is surprising that the use of such tools has not been studied in any empirical research papers.

4.2.4. Integrating OSS Software into Software Systems

Only six empirical research papers discuss issues related to integration of OSS components. There are, however, as many as 16 experience reports which briefly describe the integration of some OSS components.

Table 9: Focus and findings from the empirical research papers on integrating OSS

ID	Topic
[8]	Software organizations can increase productivity and quality through integrating systematic reuse of OSS components. OSS reuse does not require any special skills and experience other than software reuse in general.
[43]	Major cost with OSS is learning and understanding new components. Local knowledge and compliance to requirements were the most decisive factors in choosing components. A high number of components needed fixes or modifications.
[106]	There is a need for human computer interaction experts in the OSS context. Reuse of OSS enables them to spend more time on user interface. It is challenging to decide what to contribute because of licenses and patents, and as user interface code can be considered a competitive advantage.
[123]	There are many similarities in using OSS and COTS. Source code is seldom used. Hard to assess providers' (community) reputation.
[130, 186]	Identifies challenges related to modifying OSS components, and strategies (contributing, snapshot, forking, and initiating a new OSS project) for dealing with these modifications.

The integration of OSS components is one of the most popular ways of adopting OSS, in particular in the software industry. From a sample of 146 OSS firms, 69.5% reported that they had adapted OSS to customer needs [27]. In another sample of 769 companies 33% "provide solutions which are based on OSS" [29]. Moreover, 48% of 62 software companies use OSS in their business [141], and in a sample of 569 software companies, 46.8% integrate OSS in their software systems [98]. These software systems represent a great variety of application areas from all major vertical sectors [98]. Finally, Nokia claims that as much as 75% of the software architecture for its Internet tablet consists of OSS [107].

Ajila et al. [8] and Li et al. [123] claim that the similarities between reuse of OSS and COTS (commercial-off-the-shelf) are significant. No particular skills or experience is needed to integrate OSS components into another product [8].

Li et al. [123] claim that the source code of OSS components is seldom modified. Still, Chen et al. [43] report that it is necessary to modify or adapt many of the OSS products. Mannaert and Ven [130, 186] and Iivari et al. [106] discuss some of the challenges related to modifying and extending an OSS product. These challenges are somewhat special for OSS since any developer has access to the product's source code and since it is possible to return the modifications to the provider of the product. Finally, Ven and Mannaert [130, 186] present four strategies for dealing with these modifications: contributing to the OSS community, relying on a snapshot of the code base, forking the OSS product, or releasing modifications to the OSS product as a new OSS product.

Some of the most significant challenges of OSS-based projects are, according to Chen et al. and Li et al. the cost related to learning and understanding the OSS components, and estimating the time it takes to integrate them [43, 123]. This could be the reason why local expertise is so important in the decision making when selecting an OSS component [43].

4.2.5. Participating in the Development of OSS

As stated previously, software companies and software intensive organizations have started to base their existence on the use of OSS components developed by OSS communities [98]. As many as 16 empirical research papers and six experience reports provide evidence on organizations' participation in OSS communities. These organizations are thereby playing an increasingly important role in many OSS communities.

Most organizations seem to have rather limited contributions to the OSS communities [33, 43, 91, 98]. The most common way of participation is being an active user that reports occasional bugs to the community [43, 98, 99]. Only one of 32 respondents from a sample of tertiary education institutions had participated actively by writing code, while 14 had contributed to an OSS community through reporting bugs [91]. Furthermore, Bonaccorsi and Rossi [32] found that 46.2% of companies using OSS components have not joined any OSS projects, whilst 38.5% have joined five or fewer.

While the majority of organizations contribute on a rather limited level, the total body of all organizations contributes substantially to various OSS products [28, 158]. First, Bonaccorsi et al. [28] report that organizations participated in 97 of the 300 most active projects on SourceForge. Second, in a sample of community controlled OSS projects, paid developers contributed almost 50% of the code [200]. Moreover, the number of organizations contributing to OSS seems to be increasing, for instance the number of companies participating in the Debian GNU/Linux community has increased from 200 (in 1998) to 1500 (in 2005) [158]. Organizations within the Linux Kernel community are central to the community and do in fact have significant influence on it [2]. Organizations may have positioned themselves there because they believe they have to be involved to influence the community [51].

Table 10: Focus and findings from the empirical research papers on participating in the development of OSS

ID	Topic
[2]	Influence in the Linux Kernel community is centered to a small group. Organizations have a significant influence on this group.
[20]	Shows different types of employee participation. OSS projects are not uniform and homogeneous. Could be challenging to get contributions accepted.
[31]	Many companies participate in OSS communities. Companies mainly adapt the OSS product to their needs.
[28]	Companies coordinate, develop code for, or provide libraries to one third of the 300 most active Sourceforge projects. Projects where companies are involved are larger.
[50]	Organizations make use of OSS communities through accessing, aligning, and assimilating the communities.
[51]	Companies strategically sponsor individuals to influence OSS communities. Firms believe they need someone on the inside to influence the communities.
[49]	Organizations may have a symbiotic, commensalistic, or parasitic relationship to OSS communities.
[70]	Shows how a network of companies around an OSS product can collaborate to deliver a ”whole product”.
[101]	Different companies have different reasons for contributing. Code sharing is very heterogeneous. Companies share more than they have to.
[106]	There is a need for human computer interaction experts in the OSS context. Reuse of OSS enables them to spend more time on user interface. It is challenging to decide what to contribute because of licenses and patents, and as user interface code can be considered a competitive advantage.
[129]	Software developers who participate in OSS communities spend close to 50% of this time through their work.
[130, 186]	Identifies challenges related to modifying OSS components, and strategies (contributing, snapshot, forking, and initiating a new OSS project) for dealing with these modifications.
[132]	Time based releases seem to boost development in community-company collaborations.
[158]	A large number of companies give significant code contributions to Debian.
[200]	OSS providers must write most (+90%) of the code themselves. About 50% of the code in community controlled OSS projects was written by paid developers.

Nevertheless, companies are clearly becoming a very important part of the OSS community.

To make sure that an OSS product prospers, and to make sure that modifications to the product's code are maintained, an organization may decide to contribute to the product's community. However, Ven and Mannaert [186] mention several barriers for contributing to an OSS product. The organization must first of all spend resources on getting to know the community. Then they may have modifications which are very specific to their own organization or which influence several OSS projects. If the patch is accepted into an OSS product, the organization may have to spend resources maintaining it as part of the OSS project. Deciding not to contribute can also be risky as one may be forced to maintain a parallel copy of the product. In addition, Iivari et al. [106] illustrate how the fear of losing the competitive edge, and difficulties with licenses and patents, can prevent an organization from contributing.

Dahlander and Magnusson [49, 50] discuss company-community relationships and how organizations can benefit from communities. In [49] they identify three types of organization-community relationships:

- Symbiotic: Both the community and the organization benefit from the relationship.
- Commensalistic: The organization benefits from the relationship but the community is not affected.
- Parasitic: The organization benefits from the relationship but at the same time it damages the community.

Later Dahlander and Magnusson [50] show how organizations use three strategies to benefit from OSS communities. First, organizations access existing communities or start their own communities. Second, they align their strategies with the community. Third, they assimilate the community through dedicating resources to evaluating contributions from the community or by contributing non-strategic code to the community.

4.2.6. Providing OSS Products

Many organizations have over the last years released their software as OSS. In total, 15 empirical research papers and 19 experience reports show how these organizations have developed and provided their own OSS products.

Table 11: Focus and findings from the empirical research papers on providing OSS products

ID	Topic
[7]	Shows a shift from OSS as individual to OSS as a community of organizations. Explores the opportunities organizations have related to community collaboration.
[17]	Presents three cases with robot (hardware) vendors which also provide OSS software for their robots.
[49]	Organizations may have symbiotic, commensalistic, or parasitic relationship to OSS communities.
[50]	Organizations make use of OSS communities through accessing, aligning, and assimilating the communities.
[52]	Analyses the transfer of OSS developed in academia to commercial products.
[82]	Companies with many patents will more likely release OSS. Companies with many software trademarks are less likely to release OSS. Companies with many hardware trademarks are more likely to release OSS.
[110]	Organizations must balance leadership vs. too strict control.
[126]	Of 134 products from 70 Italian companies, 27 are released as OSS. These products are considered at least as innovative as the proprietary ones.
[132]	Time based releases seem to boost development in community-company collaborations.
[144]	Provides recommendation for succeeding with an OSS product: Improve product and documentation, listen to the community, make it easy to download and install it.
[200]	OSS providers must write most (+90%) of the code themselves. About 50% of the code in community controlled OSS projects was written by paid developers.
[202]	Illustrates the evolution of JBoss' business model.
[201]	Discusses "second generation" OSS business models and presents some data from four OSS providers.
[204]	Discusses hybrid (open and proprietary) business strategies in large companies.
[206]	Explores some challenges related to legitimizing the use of an OSS business model.

In a sample of 368 software companies, 5% said they provided their own OSS products [98]. In a sample of 134 software products, developed by 70 Italian companies, 27 (20%) products were released as OSS [126]. Furthermore, 36 (12%) of the 300 most active projects on SourceForge were founded by companies [28]. Moreover, we observe that Microsoft [133], research institutions [52], and other companies in several domains [19, 34, 104, 202] regularly release new OSS products.

While some provide OSS products to attract and benefit from a community, others merely release OSS to attract attention or to disseminate their software or research results. OSS providers perceive benefits like simpler dissemination of their products, reduced marketing costs, simpler recruitment of new employees, and community contributions in form of bug reports, bug fixes, feature requests, and added functionality [7, 14, 95, 99, 100, 197, 200, 201, 204]. Most of these advantages are related to actually having a community around the product.

Providing an OSS product is described as a global sourcing strategy where "commercial companies and open source communities collaborate on development of software" [7, page 385]. Furthermore, Ågerfalk and Fitzgerald [7] relate their research with offshoring and outsourcing of software development. This suggests that there is some division of labor involved in providing an OSS product. However, based on the observation that the OSS providers do most of the work and write most of the code themselves, Wasserman and Capra [200] claim that OSS is primarily a distribution model and not a development model for organizations.

Other papers report challenges related to similar difficulties such as attracting contributors, involving people at the right time, establishing a common infrastructure and so on [23, 26, 87, 109, 156]. Despite reporting different problems, few focus on identifying and solving these challenges. Another study presents experiences related to establishing a community and how limited continuity and a too strong focus on one stakeholder made it difficult to build a vivid community [109]. The provider must also consider which code to release as OSS and which to retain under proprietary licenses. Providing OSS is in other words no free lunch, and simply making the code available is not enough [100].

One of the exceptions which tries to solve some of these challenges identifies the obligations and expectations an OSS provider has to its community and vice versa [7]. Ågerfalk and Fitzgerald [7] moreover show that the provider and his community have different perceptions of the extent to which they fulfill these obligations. Being an OSS provider is not a static business model. OSS providers have to develop and adapt their business model according to the needs of their customers and communities [144, 202].

4.2.7. Using OSS Development Practices

Table 12: Focus and findings from the empirical research papers on using OSS development practices

ID	Topic
[125]	Illustrates the use of a company internal Sourceforge-like portal.
[135]	Adoption of OSS practices may be a way to standardize development processes. Transparency has both advantages and disadvantages. The OSS phenomenon is adapted to the organizations when it is adopted.

While there is quite a lot of research on specific development processes in OSS communities e.g. [167], there is little research on using these processes and practices inside organizations. We identified two empirical research papers and 11 experience reports which discuss the use of such practices.

Several companies have adopted "OSS practices" internally or within a consortium of partners. Common to the cases reported in the literature is that they are large companies with technologies which are reused in a large-scale, distributed development environment. While the use of "OSS practices" within these large companies shares commonalities, they are labeled somewhat differently e.g. "Progressive Open Source" [60, 61, 135], "inner source" [125, 203], or "Corporate Source"[96, 97].

These attempts of adopting "OSS practices" may have slightly different purposes. First, one may want to improve the collaboration between the people responsible for a core platform and the people reusing it throughout the company [96, 97, 203]. Second, it is possible to increase the visibility of reusable software components by providing them through a common platform, much like an internal SourceForge [60, 61, 125]. Third, to increase transparency and standardize diverse development practices, an organization may create a common development platform as a vehicle for collaboration [61].

The introduction of "OSS practices" is however more a social, rather than a technical change [61]. While the changes could provide benefits for the organization, it may also be painful for the individual and the company to change existing work practices [135]. For instance the experience (or lack of) with OSS could influence the adoption of new "OSS practices" [179], and when adopting such practices in a commercial setting they are shaped to the organization [135].

5. Discussion

In this section we will discuss each of the research questions in the light of the findings from the literature review. Then, we discuss opportunities for future research and finally, possible limitations of this study.

5.1. RQ1: Adoption of OSS in Organizations

In Section 2.5, we asked the following research question: *In what ways are software-intensive organizations adopting OSS?*

The short answer to this question is that there exist several ways of adopting OSS in software-intensive organizations. This literature review provides evidence that such organizations approach OSS in different ways. With a focus on software development, we identified six ways in which organizations adopt OSS (see Section 2.3).

Each of these ways of adopting OSS offers different benefits and challenges. For instance, while access to source code is not that important to organizations which only deploy OSS products [81], it could be a significant advantage for OSS integrators [139]. An organization providing an OSS product may get feedback and code contributions from their community, but at the same time they have to deal with a (potentially) large number of stakeholders [109, 200]. However, relating to a community is not relevant in the same way for an organization which simply integrates OSS components into their own products.

We furthermore see variations in the motivations that organizations have for adopting OSS, the context these organizations are in, the resources they have, and so on. While OSS seems to be an option in almost all kinds of settings, we agree with Glynn et al. [93] and Melian and Mähring [135] in that OSS needs to be understood in the organization-specific situation it is adopted. This is particularly important as OSS adoption may also involve organizational changes [81, 135]. According to Glynn et al., adoption of OSS is a complex situation consisting of an internal and an external environment, individual factors, and a technological context [93].

5.1.1. The Classification Framework: Inter-dependencies and Internal Differences

It must be emphasized that there are interdependencies between the different ways of of adopting OSS, and that an organization may approach OSS in several ways at the same time. These inter-dependencies are further complicated as organizations may evolve their approach to OSS over time [39]. Hence, the categories in our framework are not mutually exclusive.

There are in particular a few categories which are closely related. First, organizations which participate in the development of an OSS product are most likely integrating this product into one of their own systems (see e.g. [107, 130]). Second, organizations adopting OSS development practices are also frequently using OSS CASE tools to facilitate the adoption of these practices, e.g. [125, 203]. Tools like revision control systems, mailing lists, wikis, build-environments, and documentation systems shape the development process and enable the introduction of practices like code sharing, increased transparency, having a core team controlling the core of the product, peer review of code, and so on [87, 128, 131].

Next, the difference between a few of the categories is, as mentioned in Section 2.3, one of degrees rather than orthogonality. Grand et al. [94] have similar observations, in that the dedication to OSS is a matter of degree, or in other words a matter of allocating resources. There are overlapping areas between deploying OSS products and integrating OSS products into a system [3, 4], and between participating in the development of an OSS and providing an OSS product [7, 48]. For example, different organizations may use the same OSS product quite differently. In one case, a sample of organizations simply deployed Linux on their servers [188], while others extended it, integrated it into their own products, and participated in the development of it [101].

There are also internal differences within each of the ways of adopting OSS. Even though different organizations provide OSS products, they have different motivations, resources, and success. Where Bleek et al. [23] describe a public project which is struggling, Watson et al. [202] report the (successful) story of JBoss. Li et al. [124] show that the reuse of OSS components is adapted to an organization's current development process. There are also differences in how organizations participate in the development of OSS products. While some organizations donate large amounts of source code or actively participate in the development of the product [28, 101], others provide more modest contributions like occasional bug reports or forums posts [43, 98].

We observe situations very much alike these in other cases as well. Despite approaching OSS in similar manners, there are clear differences between the various organizations. The classification framework must therefore be understood and used as a tool for identifying and discussing the opportunities and challenges different organizations may find when adopting OSS. It is not an attempt to completely identify all minor variances in how organizations adopt OSS.

Nevertheless, the categories are still valuable when discussing an organization's adoption of OSS. The benefits and challenges related to a specific organization's approach to OSS should be discussed as a combination of the different ways of adopting OSS in the framework. Hence, we recommend placing an organization in all the categories it would fit into, and investigating the specific challenges related to each of the categories in the framework.

5.1.2. Implications

Practitioners need to be aware that there is not just one correct way of adopting OSS. OSS rather offers several opportunities which each have their unique benefits and drawbacks. Each organization should therefore evaluate the implications of approaching OSS in their own context [189]. This is increasingly important when we see that many organizations adopt OSS without knowing concretely if or how they will benefit from it and without having a clear strategy behind this adoption [91, 187, 188, 190].

Practitioners should also be aware that there are most likely others which have adopted OSS in a similar context. Here, we show that OSS is a viable option for many organizations and we provide several references to a variety of successful cases of OSS adoption. These references could be used as a starting point for an organization's adoption of OSS.

Researchers ought to avoid treating OSS and the adoption of OSS as one homogeneous phenomenon. We should acknowledge the individual contexts in which OSS is adopted, precisely describe how the organizations we study approach OSS, and carefully consider how this adoption influences our findings.

5.2. RQ2: Focus of the Research Literature

The 59 empirical research papers and the 53 experience reports included in this review cover a very large span of topics but have not had any particular focus. The research which is specific to how organizations adopt OSS covers topics from OpenOffice.org migration projects, through code contributions to existing OSS communities, to using "OSS development practices". Both within and between these topics there are several unexplored gaps.

Besides the studies which investigate the motivations for adopting OSS products and the migration to OpenOffice.org, there are few overlapping studies discussing closely related topics. Even though there are a few exceptions like [27, 31, 32, 33], [81, 93], [130, 186], and [60, 61, 135], the majority of the papers come from rather fragmented studies. Few studies have been reported in more than one publication and almost no publications report follow-ups or continuations of earlier research.

This lack of focus may be caused by the relatively recent birth of OSS in organizations as a research area. The many variations in how organizations adopt OSS may also contribute to diversify the research. Although similar diversity is seen in research on both software engineering [92] and information systems [192], *OSS researchers* could benefit from intensifying their efforts on a few common problems, rather than exploring an increasingly larger number of issues.

5.3. RQ3: Limitations of Existing Empirical Research

There are relatively few empirical publications on OSS in organizations, and the quality of published work is not good enough. Much of the published research lacks relevance and a clear focus, and does not draw enough support from related literature. These observations are not particular to research on OSS. For instance, Kitchenham et al. [117], Vessey et al. [192], and Zelkowitz and Wallace [214] observe a lack of relevant empirical research of high quality within both the software engineering and information systems fields. Finally, we would also like to see more research from outside Europe and the USA.

5.3.1. Little Empirical Research

Even though OSS is changing how software is developed, acquired, used, and commercialized, relatively few empirical research papers on OSS in organizations are being published. The number of such papers is particularly low in quality software engineering journals (see Appendix A).

Only 86 (12.8%) of the 674 publications relevant to OSS (S2) and five (4.5%) of the 112 papers included in the fifth stage (S5) are published in quality software engineering journals like IST, JSS, EMSE, Software Practice and Experience, Software Process: Improvement and Practice, TOSEM, or TSE. On the other hand, 380 (56.47%) of the papers related to OSS and 75 (67.0%) of the 112 publications included in the Stage 5 are published in the IEEE Software magazine, or through the International Conference on Open Source Systems and the ICSE Workshop on Open Source Software Engineering.

5.3.2. Limited Quality

Many of the empirical research papers achieved a decent score in the quality assessment. Despite this, almost half of the publications we identified were experience reports and many of the empirical research papers were troubled by missing information, low rigor, limited validity, and unclear contributions. These observations are in line with previous work e.g. [66, 71, 92, 180, 213, 214], in the sense that much of the published research has limitations when it comes to planning, execution, and reporting.

5.3.3. Limited Relevance

As many as 19 of the 59 empirical research papers study topics which are not directly related to how organizations adopt OSS. While these papers are valuable for understanding the OSS phenomenon, they are not particularly relevant to the specific problems practitioners face every day, and provide little concrete advice to practitioners.

5.3.4. Lack of Complex and Longitudinal Studies

The research on OSS in organizations is as mentioned a fairly young field. The immaturity may explain why there are few longitudinal studies and few studies looking at complex issues beyond the deployment or integration of a single OSS product. However, Höfer and Tichy [103] made similar observations concerning longitudinal studies on software engineering.

5.3.5. Does not Reflect Context

Parts of the literature neither describe nor reflect over the actual context in which OSS is adopted. This is seen when papers fail to describe how the studied organizations adopt OSS. It is also seen when researchers study samples of organizations without distinguishing between these organizations' different approaches to OSS and try to make the generalization that to all organizations "adopting OSS". This lack of context is problematic since, according to Basili et al. [18], every software development/maintenance environment is different.

5.3.6. Does not Benefit Fully from Related Research Fields

OSS and OSS development has been described as revolutionary and something totally different from software engineering [30, 154, 167]. The history of OSS has furthermore been characterized by contrasts like OSS vs. proprietary or closed source software [149], OSS vs. free software [175], the cathedral vs. the bazaar [154], copyleft vs. copyright [55], OSS development vs. software engineering [59], and so on. These contrasts and the perception that OSS is something different, have contributed to creating a gap between OSS and other research areas.

We agree with Fitzgerald [78] and Fuggetta [84] in questioning whether these differences are significant. Moreover, we find evidence that much of the research on OSS in organizations has in fact profound similarities with other research areas. The review revealed that organizations deploying OSS faced the same challenges as with adoption of any other technology [37, 81, 148, 187], and that these issues were often organizational rather than technical [135, 139]. The adoption of OSS seems to be more depending on the organization adopting it and the situation in which it is adopted, than on the technology being released as OSS. Next, we saw that integrating

OSS components was very much the same as integrating COTS components [8, 123]. Finally, many of the advantages of OSS (like reduced development effort, increased quality, and so on) are really advantages of software re-use [137, 193]. OSS products can clearly give these benefits, but this is not necessarily because these products are licensed as OSS.

Despite these similarities, parts of the literature have still treated OSS as something new and quite different from other information technologies and methods for software development. Treating a novel research area as something totally different from existing areas is not a new phenomenon as "[t]here is a tendency for IS researchers [as well] to treat new technologies as virgin or greenfield, thereby acting on the belief that prior theories or models are not appropriate" Vessey et al. [192, page 169].

Some researchers are looking to related research and theories to explain OSS phenomenons e.g. adoption of IT innovations [93], psychological contract theory [7], and business models and business networks [70]. However, many researchers present an introverted view of OSS, and when treating OSS as something totally unique, they fail to draw valuable support from related literature.

This literature review focuses on empirical studies from the OSS and software engineering literature. By widening its scope it would have encompassed publications that extended literature from a broader spectrum of areas. Nevertheless, we find that many researchers should draw more support from related research fields both when identifying research questions and when discussing their findings.

5.4. Directions for Future Research

Even though the research so far has some limitations there are several opportunities for further work. Based on the literature review and the answers to RQ 2 and 3, we give some recommendations for this work.

5.4.1. General Recommendations

Maturing the research field on OSS in organizations and dealing with some of its limitations may be done through four main steps:

1. Focus research on topics which are relevant to how organizations approach OSS
2. Strive to increase the rigor of the empirical studies
3. Conduct more longitudinal, in-depth studies

4. Align our research with related research fields

There are several unexplored issues in relation to the "adoption of OSS" in general. However, we agree with for instance Charters et al. [42] in that researchers need to pay more attention to issues that are interesting to practitioners. Hence, we recommend focusing on topics related to the ways in which organizations actually approach OSS, and issues which could benefit practitioners, rather than general "adoption issues". Researchers and practitioners should increasingly collaborate to define a common research agenda and study research questions which matter to practitioners. These research questions should be answered through several related studies from different contexts.

The overall rigor of the studies performed on OSS, both within organizations and in general, is furthermore not good enough. Consequently, we should strive to do better work and to present this work in more detail [180]. In particular, we agree with Kitchenham et al. [118] in that the context of the organizations being studied should be given much more attention.

We observed that few of the studies were longitudinal and that few publications focused on providing in-depth details from one or a few organizations. To really understand the profound consequences of approaching OSS, we believe there is a need for both more longitudinal and in-depth case studies.

Finally, we found evidence that OSS is not that different from other information technologies. *OSS researchers* should therefore increasingly rely on research and theories from related fields (see Section 2.4). *Software engineering and information systems researchers* should see OSS as an opportunity to investigate general software engineering and information systems research challenges.

5.4.2. Topics for Future Research

In this section we suggest topics for further research for each of the approaches to OSS. The topics discussed below may work as an initial starting point for discussing and staking out the direction of the research on OSS in organizations. We would in particular recommend investigating two issues: (1) topics related to integration of OSS components and (2) topics concerning participation in organization-community or inter-organizational OSS collaborations. We find these issues important because integration of OSS components concerns most software-intensive organizations [98] and because participation in collaborative software development is increasing [7, 185].

The research could focus on identifying the characteristics of successful approaches to OSS, the challenges these organizations met, and how they solved them.

Deploying OSS: Many claim that reducing costs is one of the advantages of deploying OSS server software, infrastructure, and applications. However, a recent study by Fitzgerald [80] is one of few studies with a longitudinal view on deployed OSS products. This highlights a need for more studies on:

- What are long-term costs and consequences of deploying and keeping OSS products operational?

Using OSS CASE tools: The research on OSS CASE tools has been very limited. However, Wicks and Dewar propose a new agenda for research on tool integration, requesting a more business-oriented approach to future research [207]. The use and development of OSS CASE tools and research on such tools could easily fit into this new agenda. Robbins provides an extensive overview of OSS tools for development, and claims that CASE research has a lot to learn from OSS [157]. OSS should furthermore be particularly interesting to academia since they have access to professional state-of-the-art tools and the tools' source code. This enables them to extend existing tools and test new ideas in collaboration with each other.

Increased participation in OSS projects, increased collaboration between organizations, and increased use of OSS practices will most likely require improved collaborative development tools. Hence, there is a potential for research on:

- What kinds of tools are needed for collaborative software development across organizational and community borders?
- How do organizations collaborate using such software development tools?

Integrating OSS components:

Navigating through the amounts of OSS components and related information available across the Internet is a significant challenge [143]. However, the information offered over the Internet through OSS communities, web forums, and so on, constitutes at the same time a valuable resource. Due to the easy access to reusable software components, we see that software systems are constantly growing. Software developers are integrating an increasingly larger number of OSS and commercial components into their products. In doing so they have to relate, adapt, and possibly contribute to a large number of providers. Therefore, we believe research could focus on the following questions:

- How may organizations most efficiently navigate through available information and select OSS components?
- How may organizations benefit from OSS communities and the resources available over the Internet?
- How can organizations maintain and secure the sustainability of software systems consisting of components from a variety of providers?

While there are a few studies outside the scope of this review focusing on software selection [46, 56, 105, 184] and knowledge sharing within OSS communities [119, 173, 195], none of these are directed towards studying actual practice in organizations. A few studies have started to look at some of the challenges in the borderlands between integrating an OSS component and contributing to the development of it [106, 130, 186], but further research is needed to solve the maintenance challenges facing developers who integrate a large number of components into their products.

Participating in OSS communities: To enable organizations to reap benefits from their participation in OSS communities, the research community should dedicate much more attention to questions concerning this [48, 165]. While there are a few examples [50, 101, 176, 186], more work is needed to aid organizations in participating in communities and collaborating with other organizations through collaboratively working on OSS products [7] and to solve questions like:

- When, how, and with what should organizations participate in the development of OSS products controlled by others (including inter-organizational collaborations)?
- How can one effectively contribute only parts of a product and at the same time retain other parts private?

Providing OSS products: Succeeding at providing an OSS product is not necessarily easy as there are challenges related to collaborating with a community, like attracting and relating to contributors, requirements engineering from a community, balancing focus on community and paying customers, and so on [109, 200]. We hope to see more research on these topics like e.g. [7, 205] on the following topics:

- How are OSS providers able to attract and sustain a community?
- What are the success criteria for incorporating contributions (requirements, code, bug reports/fixes, etc.) from a community?

Using OSS practices: It is more and more difficult to talk about "OSS practices" as the practices used in OSS communities are heterogeneous, and as organizations are increasingly getting involved in, and influenced by, the development of OSS. Nevertheless, there are opportunities for further research on the use of development practices for distributed software development. OSS development in large communities and in and between organizations, are areas where researchers could have an impact on practice. OSS research has so far focused mainly on processes in communities of volunteers [167], but some of this research could turn its focus towards the application of their findings within organizations and questions like:

- How can development practices from OSS communities be adopted within organizations?
- How may organizations successfully collaborate through community- or consortium-based software development?

5.5. Limitations of this Study

Even though this systematic literature review has been supported by a pre-defined study protocol, explorative pilot testing of each of its stages, and continuous interaction between the authors, it has some limitations.

5.5.1. Completeness of the Selected Set of Publications

By basing the review on a clearly defined set of publications (see Section 3.1), we excluded certain types of publications, work published through other channels or outside the defined time frame. We can therefore not claim to have included all relevant publications. However, we based the review on an extensive set of publications from core software engineering and OSS publication channels. The most relevant publications should therefore be included.

5.5.2. Different Search and Data Extraction Facilities

The search and data extraction facilities provided by the various digital publication databases are different and not necessarily developed with systematic literature reviews in mind [36, 68]. This posed two challenges.

First, the data provided by digital libraries is not always reliable. For instance, we found several publications which were to be printed in 2009, registered as published in 2008. This entails that we may have included papers due for publication in 2009 as published in 2008.

Second, we wanted to cover as many publications as possible and decided to use full text search, wherever possible. However, the standard search in IEEE Xplore provides only a meta-data search. Full text search is possible, but we did not discover this until late in the review process. The use of full text search in IEEE Xplore would have implied that the number of database hits (Stage 2) in TSE, ICSE (1998-2005, 2007), ISESE (2002-2005), METRICS, ESEM (2007), and particularly IEEE Software and IEEE Computer would have been significantly higher.

Even though it was possible to do full text search in IEEE Xplore we decided against it because of several reasons. First, using a full text search reduces the precision of the search quite dramatically [58]. Our experience with the other databases confirms this, and a large number of the publications included in Stage 2 were not within the scope of this literature review. The 112 papers from Stage 5 constitute only 7.3% of the papers included in Stage 2. Therefore, Dieste and Padua [58] recommend searching through only abstracts and titles as a good searching strategy. Finally, we randomly reviewed a significant amount of the papers not included in the initial meta-data search. None of these publications would have been included in Stage 4 or 5 of this literature review.

5.5.3. Classification of Papers and Missing Information

In Stage 2 we classified the publications primarily based on their titles and abstracts. This is sometimes hard as many abstracts often omit relevant information [38]. As a consequence, Brereton et al. [36] recommend reviewing also the conclusions of the papers in addition to the titles and abstracts. However, acquiring the whole text for more than 1500 publications and reviewing them was not a viable option. Conducting a systematic literature review is already a very resource demanding exercise.

Reading the papers more closely (Stages 3-5) clearly increased the precision of the classification, but it could still be quite difficult to classify a paper due to the paper's lack of detail and because some papers discuss several topics. This was particularly relevant since much of the OSS research has had a focus on qualitative data. Kitchenham's guidelines for systematic literature reviews recommend contacting the authors of such papers to get the necessary details [115]. Even though this was not done, we achieved a good agreement in the classification process and solved any disagreement by reaching a consensus.

5.5.4. Quality Assessment

To ensure scientific rigor, we performed a quality assessment by assessing whether or not the publications covered the nine quality metrics defined in Section 3.2. This binary (yes/no) scale was in retrospect not sensitive enough and the quality assessment scores are therefore a bit inflated. Many of the publications barely mention issues related to some of the quality metrics, but were still evaluated to cover them.

Using a three-level scale like Kitchenham et al. [116] could have been a better option. However, this would have required a larger classification effort and even more subjective judgment. Given the amount of literature reviews concluding that many publications lack rigor e.g. [66, 71, 92, 180, 213, 214], spending even more time on detailed quality assessments would hardly have provided any new insight.

5.5.5. Data extraction and author bias

The most challenging part of the literature review is perhaps extracting relevant data or findings from the publications and writing a synopsis like this paper based on such data. Author bias in this process is a potential problem as the extracted data has primarily a qualitative nature and as we must prioritize what to include in the synopsis. To reduce this problem we used a pre-defined study protocol, piloted the various stages of the review, performed most stages individually, and had continuous discussions about the review process.

The systematic literature review is one of evidence based software engineering's key tools for integrating research with practice. One of the objectives of a systematic literature review is therefore creating guidelines for practitioners. This was difficult due to the many limitations of the OSS literature, and is something future research should try to achieve.

6. Conclusion

Unlike existing literature reviews on OSS, this review is a systematic literature review which focuses on OSS in organizations. We provide an extensive overview of the OSS literature, and together with Stol and Babar [180] we introduce systematic literature reviews to the OSS research arena.

Our results show that *organizations adopt OSS in distinctly different ways*. To better understand these ways of adopting OSS, we have provided a classification framework which shows that an organization may (1) deploy OSS infrastructure and applications, (2) use OSS CASE tools, (3) integrate OSS components, (4) participate in external OSS communities and contribute to the development of OSS products which are controlled by someone else, (5) provide their own OSS products, and (6) use OSS development practices.

Known *success stories* from other organizations is an important factor which increases an organization's confidence in OSS, according to Ågerfalk et al. [6] and Glynn et al. [93]. Here, we show that OSS provides organizations with several opportunities, that OSS is widely adopted, and that organizations make up a significant part of the OSS phenomenon. We furthermore provide references to a large number of success stories which could educate practitioners and make them feel more confident about adopting OSS.

We see that while the research community's *attention to OSS in organizations has been limited*, it seems to be increasing. In addition, this literature review reveals that the research literature is rather fragmented and lacks a clear focus, rigor, and longitudinal studies. OSS researchers have furthermore not reflected well enough over how organizations adopt OSS and they have not benefited fully from related research fields.

The research on OSS in organizations has some identified limitations, but there are several *opportunities for future research*. We provide directions for this work, which apply not only to OSS researchers, but also to software engineering and information systems researchers, who want to study contexts in which OSS is developed and used.

From these results we draw several *implications*. Researchers could use these contributions to find new research challenges and align their work with the work of others. They may also use the classification framework to position their work and to describe the context of the organizations they are studying. We furthermore advise researchers to put emphasis on how the studied organizations actually use OSS, and on problems which really matter to practitioners. Practitioners should be open to OSS and see that it offers several opportunities. However, they must first evaluate the implications of adopting OSS in their own context.

Acknowledgements

Claudia Ayala's work was carried out during the tenure of an ERCIM "Alain Bensoussan"Fellowship Programme. We acknowledge Thomas Østerlie and Torgerir Dingsøyr for their comments during the execution and writing phase of this work, and Sari Cunningham for her assistance with language and proofreading.

References

[1] I. Aaen, J. Arent, L. Mathiassen, O. Ngwenyama, A Conceptual Map of Software Process Improvement, Scandinavian Journal of Information Systems 13 (2001) 123–146.

[2] T. Aaltonen, J. Jokinen, Influence in the Linux Kernel Community, in: Feller et al. [77], pp. 203–208. doi:10.1007/978-0-387-72486-7_16.

[3] P. Adams, C. Boldyreff, D. Nutter, S. Rank, Adaptive Reuse of Libre Software Systems for Supporting On-line Collaboration, in: Feller et al. [75], pp. 1–4. doi:10.1145/1082983.1083259.

[4] P. Adams, D. Nutter, S. Rank, C. Boldyreff, Using Open Source Tools to Support Collaboration within CALIBRE, in: Scotto and Succi [168], pp. 61–65.

[5] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Information and Software Technology 51 (6) (2009) 957–976. doi:10.1016/j.infsof.2008.12.005.

[6] P. J. Ågerfalk, A. Deverell, B. Fitzgerald, L. Morgan, Assessing the Role of Open Source Software in the European Secondary Software Sector: A Voice from Industry, in: Scotto and Succi [168], pp. 82–87.

[7] P. J. Ågerfalk, B. Fitzgerald, Outsourcing to an Unknown Workforce: Exploring Opensourcing As a Global Sourcing Strategy, MIS Quarterly 32 (2) (2008) 385–409.

[8] S. A. Ajila, D. Wu, Empirical study of the effects of open source adoption on software development economics, Journal of Systems and Software 80 (9) (2007) 1517–1529. doi:10.1016/j.jss.2007.01.011.

[9] J. Akkanen, H. Demeter, T. Eppel, Z. Ivánfi, J. Nurminen, P. Stenman, Reusing an open source application practical experiences with a mobile CRM pilot, in: Feller et al. [77], pp. 217–222. doi:10.1007/978-0-387-72486-7_18.

[10] C. A. Ardagna, E. Damiani, F. Frati, M. Montel, Using Open Source Middleware for Securing, e-Gov Applications, in: Scotto and Succi [168], pp. 172–178.

[11] F. Attilio, P. Di Nunzio, F. Di Gregorio, A. Meo, A graphical installation system for the GNU/Linux Debian distribution, in: Damiani et al. [53], pp. 337–338. doi:10.1007/0-387-34226-5_35.

[12] L. Augustin, D. Bressler, G. Smith, Accelerating Software Development through Collaboration, in: Tracz et al. [183], pp. 559–563.

[13] N. Ayewah, D. Hovemeyer, J. D. Morgenthaler, J. Penix, W. Pugh, Using Static Analysis to Find Bugs, IEEE Software 25 (5) (2008) 22–29. doi:10.1109/ms.2008.130.

[14] J. Ayre, F. Gasperoni, A Successful Business Model for Free Software, in: Scotto and Succi [168], pp. 135–139.

[15] C. Bac, O. Berger, V. Desbordes, B. Hamet, Why and how-to contribute to libre software when you integrate them into an in-house application?, in: Scotto and Succi [168], pp. 113–118.

[16] M. Banzi, G. Bruno, G. Caire, To What Extent Does It Pay to Approach Open Source Software for a Big Telco Player? , in: Russo et al. [163], pp. 307–315. doi:10.1007/978-0-387-09684-1_27.

[17] P. Barrera, G. Robles, J. M. Cañas, F. Martín, V. Matellán, Impact of Libre Software Tools and Methods in the Robotics Field, in: Feller et al. [75], pp. 1–6. doi:10.1145/1083258.1083261.

[18] V. R. Basili, R. W. Selby, D. H. Hutchens, Experimentation in Software Engineering, IEEE Transactions on Software Engineering 12 (7) (1986) 733–743.

[19] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE: A software framework for developing multi-agent applications. Lessons learned, Information and Software Technology 50 (1-2) (2008) 10 – 21. doi:10.1016/j.infsof.2007.10.008.

[20] E. Berdou, Insiders and outsiders: paid contributors and the dynamics of cooperation in community led F/OS projects, in: Damiani et al. [53], pp. 201–208. doi:10.1007/0-387-34226-5_20.

[21] F. O. Bjørnson, T. Dingsøyr, Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used, Information and Software Technology 50 (11) (2008) 1055–1068. doi:10.1016/j.infsof.2008.03.006.

[22] W.-G. Bleek, M. Finck, Migrating a Development Project to Open Source Software Development, in: Feller et al. [74], pp. 9–13.

[23] W.-G. Bleek, M. Finck, B. Pape, Towards an Open Source Development Process? Evaluating the Migration to an Open Source Project by Means of the Capability Maturity Model, in: Scotto and Succi [168], pp. 37–43.

[24] P. Boccacci, V. Carrega, G. Dodero, Open source technologies for visually impaired people, in: Feller et al. [77], pp. 241–246. doi:10.1007/978-0-387-72486-7_22.

[25] C. Boldyreff, K. Crowston, B. Lundell, A. I. Wasserman (Eds.), Proceedings of the 5th IFIP Working Group 2.13 International Conference on Open Source Systems (OSS2009) - Open Source Ecosystems: Diverse Communities, June 3-6, Skövde, Sweden, Vol. 299/2009 of IFIP International Federation for Information Processing, Springer, Heidelberg, Germany, 2009. doi:10.1007/978-3-642-02032-2.

[26] C. Boldyreff, D. Nutter, S. Rank, Communication and Conflict Issues in Coollaborative Software Research Projects, in: Feller et al. [74], pp. 14–17.

[27] A. Bonaccorsi, S. Giannangeli, C. Rossi, Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry, Management Science 52 (7) (2006) 1085–1098. doi:10.1287/mnsc.1060.0547.

[28] A. Bonaccorsi, D. Lorenzi, M. Merito, C. Rossi, Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research, in: Capiluppi and Robles [41], pp. 1–5. doi:10.1109/floss.2007.3.

[29] A. Bonaccorsi, L. Piscitello, M. Merito, C. Rossi, How is it possible to profit from innovation in the absence of any appropriability?, in: Damiani et al. [53], pp. 333–334. doi:10.1007/0-387-34226-5_33.

[30] A. Bonaccorsi, C. Rossi, Why Open Source Software Can Succeed, Research Policy 32 (7) (2003) 1243–1258.

[31] A. Bonaccorsi, C. Rossi, Contributing to OS Projects. A Comparison between Individual and Firms, in: Feller et al. [74], pp. 18–22.

[32] A. Bonaccorsi, C. Rossi, Intrinsic motivations and profit-oriented firms. Do firms practise what they preach?, in: Scotto and Succi [168], pp. 241–245.

[33] A. Bonaccorsi, C. Rossi, Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business, Knowledge, Technology, and Policy 18 (4) (2006) 40–64. doi:10.1007/s12130-006-1003-9.

[34] G. Bortis, Experiences with Mirth: an open source health care integration engine, in: W. Schäfe, M. B. Dwyer, V. Gruhn (Eds.), Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), ACM, New York, USA, 2008, pp. 649–652. doi:10.1145/1368088.1368179.

[35] P. Brereton, D. Budgen, Component-Based Systems: a Classification of Issues, Computer 33 (11) (2000) 54–62. doi:10.1109/2.881695.

[36] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, Journal of Systems and Software 80 (4) (2007) 571–583. doi:10.1016/j.jss.2006.07.009.

[37] D. Brink, L. Roos, J. Weller, J.-P. Van Belle, Critical Success Factors for Migrating to OSS-on-the-Desktop: Common Themes across Three South African Case, in: Damiani et al. [53], pp. 287–293. doi:10.1007/0-387-34226-5_29.

[38] D. Budgen, B. A. Kitchenham, S. Charters, M. Turner, P. Brereton, S. Linkman, Presenting software engineering results using structured abstracts: a randomised experiment, Empirical Software Engineering 13 (4) (2008) 435–468. doi:10.1007/s10664-008-9075-7.

[39] M. Campbell-Kelly, D. D. Garcia-Swartz, Pragmatism, not ideology: Historical perspectives on ibm's adoption of open-source software, Information Economics and Policy 21 (3) (2009) 229 – 244. doi:10.1016/j.infoecopol.2009.03.006.

[40] A. Capiluppi, P. Lago, M. Morisio, Evidences in the evolution of os projects through changelog analyses, in: Feller et al. [73].

[41] A. Capiluppi, G. Robles (Eds.), Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS 2007), May 21, Minneapolis, USA, IEEE Computer Society, Washington, USA, 2007.

[42] S. Charters, D. Budgen, M. Turner, B. A. Kitchenham, P. Brereton, S. Linkman, Objectivity in Research: Challenges from the Evidence-Based Paradigm, in: C. Fidge (Ed.), Proceedings of the Australian Software Engineering Conference (ASWEC'09), April 14th17th, Gold Coast, Australia, IEEE Computer Society, 2009, pp. 73–80. doi:10.1109/aswec.2009.25.

[43] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, C. Liu, An Empirical Study on Software Development with Open Source Components in the Chinese Software Industry, Software Process: Improvement and Practice 13 (1) (2008) 89–100. doi:10.1002/spip.v13:1.

[44] K. Crowston, J. Howison, The social structure of free and open source software development, First Monday 10 (2).
URL http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1207/1127

[45] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, J. Howison, Self-organization of teams for free/libre open source software development, Information and Software Technology 49 (6) (2007) 564 – 575, qualitative Software Engineering Research. doi:10.1016/j.infsof.2007.02.004.

[46] D. Cruz, T. Wieland, A. Ziegler, Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis, Software Process: Improvement and Practice 11 (2) (2006) 107–122. doi:10.1002/spip.257.

[47] P. Currion, C. de Silva, B. V. de Walle, Open source software for disaster management, Communications of the ACM 50 (3) (2007) 61–65. doi:10.1145/1226736.1226768.

[48] L. Dahlander, L. Frederiksen, F. Rullani, Online Communities and Open Innovation: Governance and Symbolic Value Creation , Industry & Innovation 15 (2) (2008) 115–123. doi:10.1080/13662710801970076.

[49] L. Dahlander, M. G. Magnusson, Relationships between Open Source Software Companies and Communities: Observations from Nordic Firms, Research Policy 34 (4) (2005) 481–493. doi:10.1016/j.respol.2005.02.003.

[50] L. Dahlander, M. G. Magnusson, How do Firms Make Use of Open Source Communities?, Long Range Planning 41 (6) (2008) 629 – 649. doi:10.1016/j.lrp.2008.09.003.

[51] L. Dahlander, M. W. Wallin, A Man on the Inside: Unlocking Communities as Complmentary Assets, Research Policy 35 (8) (2006) 1243–1259. doi:10.1016/j.respol.2006.09.011.

[52] J.-M. Dalle, G. Rousseau, Toward Collaborative Open-Source Technology Transfer, in: Feller et al. [74], pp. 34–42.

[53] E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto (Eds.), Proceedings of the 2nd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2006) - Open Source Systems, June 8-10, Como, Italy, Vol. 203/2006 of IFIP International Federation for Information Processing, Springer, Heidelberg, 2006. doi:10.1007/0-387-34226-5.

[54] E. Davini, E. Faggioni, G. Granatella, D. Tartari, M. Scotto, Open Source in Public Administration, a real example: OSS for e-government Observatories, in: Scotto and Succi [168], pp. 119–124.

[55] P. B. de Laat, Copyright or copyleft?: An analysis of property regimes for software development, Research Policy 34 (10) (2005) 1511 – 1532. doi:10.1016/j.respol.2005.07.003.

[56] J.-C. Deprez, S. Alexandre, Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS, in: A. Jedlitschka, O. Salo (Eds.), Product-Focused Software Process Improvement Proceedings of the 9th International Conference on Product-Focused Software Process Improvement (PROFES 2008), Vol. 5089/2008 of Lecture Notes in Computer Science, Springer, Heidelberg, 2008, pp. 189–203. doi:10.1007/978-3-540-69566-0_17.

[57] M. Di Giacomo, MySQL: Lessons Learned on a Digital Library, IEEE Software 22 (3) (2005) 10–13. doi:10.1109/ms.2005.71.

[58] O. Dieste, A. G. Padua, Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews, in: Juristo et al. [112], pp. 215–224. doi:10.1109/esem.2007.19.

[59] T. T. Dinh-Trong, J. M. Bieman, The FreeBSD Project: A Replication Case Study of Open Source Development, IEEE Transactions on Software Engineering 31 (6) (2005) 481–494. doi:10.1109/tse.2005.73.

[60] J. Dinkelacker, P. K. Garg, Corporate Source: Applying Open Source Concepts to a Corporate Environment, in: J. Feller, B. Fitzgerald, A. van der Hoek (Eds.), Making Sense of the Bazaar: 1st Workshop on Open Source Software Engineering, 15 May, Toronto, Canada, IEE Software Proceedings, 2001, pp. 1–9.

[61] J. Dinkelacker, P. K. Garg, R. Miller, D. Nelson, Progressive Open Source, in: Tracz et al. [183], pp. 177–184. doi:10.1145/581339.581363.

[62] L. Dobusch, Migration Discourse Structures: Escaping Microsofts Desktop Path, in: Russo et al. [163], pp. 223–235. doi:10.1007/978-0-387-09684-1_18.

[63] G. Dodero, K. Lupi, E. Piffero, Comparing macro development for personal productivity tools: an experience in validating accessibility of Talking Books, in: Damiani et al. [53], pp. 247–252. doi:10.1007/0-387-34226-5_24.

[64] B. Donnellan, B. Fitzgerald, B. Lake, J. Sturdy, Implementing an Open Source Knowledge Base, IEEE Software 22 (6) (2005) 92–95. doi:10.1109/ms.2005.155.

[65] T. Dybå, An Empirical Investigation of the Key Factors for Success in Software Process Improvement, IEEE Transactions on Software Engineering 31 (5) (2005) 410–424. doi:10.1109/tse.2005.53.

[66] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and Software Technology 50 (9-10) (2008) 833–859. doi:10.1016/j.infsof.2008.01.006.

[67] T. Dybå, T. Dingsøyr, Strength of Evidence in Systematic Reviews in Software Engineering, in: D. Rombach, S. Elbaum, J. Münch (Eds.), Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08), October 9th-10th, Kaiserslautern, Germany, ACM, New York, USA, 2008, pp. 178–187. doi:10.1145/1414004.1414034.

[68] T. Dybå, T. Dingsøyr, G. K. Hanssen, Applying Systematic Reviews to Diverse Study Types: An Experience Report, in: Juristo et al. [112], pp. 225–234. doi:10.1109/ESEM.2007.58.

[69] C. Ebert, Guest Editor's Introduction: How Open Source Tools Can Benefit Industry, IEEE Software 26 (2) (2009) 50–51. doi:10.1109/ms.2009.38.

[70] J. Feller, P. Finnegan, J. Hayes, Delivering the 'Whole Product': Business Model Impacts and Agility Challenges in a Network of Open Source Firms, Journal of Database Management 19 (2) (2008) 95–108.

[71] J. Feller, P. Finnegan, D. Kelly, M. MacNamara, Developing Open Source Software: A Community-Based Analysis of Research, in: E. M. Trauth, D. Howcroft, T. Butler, B. Fitzgerald, J. I. DeGross (Eds.), Social Inclusion: Societal and Organizational Implications for Information Systems FIP TC8 WG 8.2 International Working Conference, 12-15 July, Limerick, Ireland, Vol. 208 of IFIP International Federation for Information Processing, Springer, 2006, pp. 261–278. doi:10.1007/0-387-34588-4_18.

[72] J. Feller, B. Fitzgerald, Understanding Open Source Software Development, Addison Wesley, 2002, ISBN :0-201-73496-6.

[73] J. Feller, B. Fitzgerald, S. A. Hissam, K. R. Lakhani (Eds.), Taking Stock of the Bazaar: 3rd Workshop on Open Source Software Engineering (WOSSE 2003), 2003.

[74] J. Feller, B. Fitzgerald, S. A. Hissam, K. R. Lakhani (Eds.), Collaboration, Conflict and Control Proceedings of the 4th Workshop on Open Source Software Engineering (WOSSE 2004), 2004.

[75] J. Feller, B. Fitzgerald, S. A. Hissam, K. R. Lakhani, W. Scacchi (Eds.), Open Source Application Spaces: Proceedings of the Fifth Workshop on Open Source Software Engineering (WOSSE 2005), ACM, New York, 2005.

[76] J. Feller, B. Fitzgerald, K. R. Lakhani, S. A. Hissam (Eds.), Perspectives on Free and Open Source Software, The MIT Press, Cambridge, Massachusetts, 2005.

[77] J. Feller, B. Fitzgerald, W. Scacchi, A. Sillitti (Eds.), Proceedings of the 3rd IFIP Working Group 2.13 International Conference on Open Source Software (OSS2007) - Open Source Development, Adoption and Innovation, June 11-14, Limerick, Ireland, Vol. 234/2007 of IFIP International Federation for Information Processing, Springer, Heidelberg, Germany, 2007. doi:10.1007/978-0-387-72486-7.

[78] B. Fitzgerald, Has Open Source Software a Future?, in: Feller et al. [76], pp. 93–106.

[79] B. Fitzgerald, The Transformation of Open Source Software, MIS Quarterly 30 (3) (2006) 587–598.

[80] B. Fitzgerald, Open Source Software Adoption: Anatomy of Success and Failure, International Journal of Open Source Software & Processes 1 (1) (2009) 1–23.

[81] B. Fitzgerald, T. Kenny, Developing an Information Systems Infrastructure with Open Source Software, IEEE Software 21 (1) (2004) 50–55. doi:10.1109/ms.2004.1259216.

[82] A. Fosfuri, M. S. Giarratana, A. Luzzi, The Penguin Has Entered the Building: The Commercialization of Open Source Software Products, Organization Science 19 (2) (2008) 292–305. doi:10.1287/orsc.1070.0321.

[83] A. Fuggetta, A Classification of CASE Technology, Computer 26 (12) (1993) 25–38. doi:10.1109/2.247645.

[84] A. Fuggetta, Open source software–an evaluation, Journal of Systems and Software 66 (1) (2003) 77 – 90. doi:10.1016/s0164-1212(02)00065-1.

[85] C. Gacek, B. Arief, The Many Meanings of Open Source, IEEE Software 21 (1) (2004) 34–40. doi:10.1109/ms.2004.1259206.

[86] A. Galatescu, V. Florian, L. Costea, D. Conescu, Issues in Implementing an Open Source-based XML Repository Manager for Application Maintenance and Adaptation, in: Feller et al. [73], pp. 57–62.

[87] K. Gary, L. Ibanez, S. Aylward, D. Gobbi, M. B. Blake, K. Cleary, IGSTK: an open source software toolkit for image-guided surgery, Computer 39 (4) (2006) 46–53. doi:10.1109/mc.2006.130.

[88] D. M. German, The evolution of the GNOME Project, in: Meeting challenges and surviving success: the 2nd Workshop on Open Source Software Engineering (WOSSE 2002), 25 May, Orlando, USA, 2002, pp. 1–4.

[89] R. A. Ghosh, Study on the Economic Impact of Open Source Software on Innovation and the Competiveness of the Information and Communication Technologies (ICT) Sector in the EU, Tech. rep., UNU-MERIT (2006).
URL http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf

[90] P. Giacalone, OSS implementation solutions for Public Administration applications, in: Scotto and Succi [168], pp. 259–262.

[91] D. G. Glance, J. Kerr, A. Reid, Factors affecting the use of open source software in tertiary education institutions, First Monday 9 (2).

[92] R. L. Glass, I. Vessey, V. Ramesh, Research in software engineering: an analysis of the literature, Information and Software Technology 44 (8) (2002) 491–506. doi:10.1016/s0950-5849(02)00049-6.

[93] E. Glynn, B. Fitzgerald, C. Exton, Commercial Adoption of Open Source Software: An Empirical Study, in: J. Verner, G. H. Travassos (Eds.), Proceedings of International Symposium on Empirical Software Engineering (ISESE 2005), IEEE Computer Society, Los Alamitos, USA, 2005, pp. 225–234. doi:10.1109/ISESE.2005.1541831.

[94] S. Grand, G. von Krogh, D. Leonard, W. Swap, Resource allocation beyond firm boundaries: A multi-level model for Open Source innovation, Long Range Planning 37 (6) (2004) 591–610. doi:10.1016/j.lrp.2004.09.006.

[95] M. Gschwind, D. Erb, S. Manning, M. Nutter, An Open Source Environment for Cell Broadband Engine System Software, Computer 40 (6) (2007) 37–47. doi:10.1109/MC.2007.192.

[96] V. K. Gurbani, A. Garvert, J. D. Herbsleb, A Case Study of Open Source Tools and Practices in a Commercial Setting, in: Feller et al. [75], pp. 1–6. doi:10.1145/1083258.1083264.

[97] V. K. Gurbani, A. Garvert, J. D. Herbsleb, A Case Study of a Corporate Open Source Development Model, in: Osterweil et al. [147], pp. 472–481.

[98] Ø. Hauge, C.-F. Sørensen, R. Conradi, Adoption of Open Source in the Software Industry, in: Russo et al. [163], pp. 211–222. doi:10.1007/978-0-387-09684-1_17.

[99] Ø. Hauge, C.-F. Sørensen, A. Røsdal, Surveying Industrial Roles in Open Source Software Development, in: Feller et al. [77], pp. 259–264. doi:10.1007/978-0-387-72486-7_25.

[100] F. Hecker, Setting Up Shop: The Business of Open-Source Software, IEEE Software 16 (1) (1999) 45–51. doi:10.1109/52.744568.

[101] J. Henkel, Selective revealing in open innovation processes: The case of embedded Linux, Research Policy 35 (7) (2006) 953 – 969. doi:10.1016/j.respol.2006.04.010.

[102] G. Hertel, S. Niedner, S. Herrmann, Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel, Research Policy 32 (7) (2003) 1159 – 1177, open Source Software Development. doi:10.1016/s0048-7333(03)00047-7.

[103] A. Höfer, W. Tichy, Status of Empirical Research in Software Engineering, in: V. R. Basili, D. Rombach, K. Schneider, B. A. Kitchenham, D. Pfahl, R. W. Selby (Eds.), Proceedings of the International Workshop on Empirical Software Engineering Issues. Critical Assessment and Future Directions, Dagstuhl, June 26th-30th, Castle, Germany, Vol. 4336/2007 of Lecture Notes in Computer Science, Springer, 2007, pp. 10–19. doi:10.1007/978-3-540-71301-2_3.

[104] T. Hoffman, SchoolTool: Defining Our Niche in the Open Source Architecture of Schools, in: Scotto and Succi [168], pp. 334–337.

[105] O. Hummel, W. Janjic, C. Atkinson, Code conjurer: Pulling reusable software out of thin air, IEEE Software 25 (5) (2008) 45–52. doi:10.1109/ms.2008.110.

[106] N. Iivari, H. Hedberg, T. Kirves, Usability in Company Open Source Software Context - Initial Findings from an Empirical Case Study, in: Russo et al. [163], pp. 359–365. doi:10.1007/978-0-387-09684-1_33.

[107] A. Jaaksi, Experiences on Product Development with Open Source Software, in: Feller et al. [77], pp. 85–96. doi:10.1007/978-0-387-72486-7_7.

[108] J. Jacobson, M. Lewis, Game engine virtual reality with CaveUT, Computer 38 (4) (2005) 79–82. doi:10.1109/mc.2005.126.

[109] J. Järvensivu, T. Mikkonen, Forging a community not: Experiences on establishing an open source project, in: Russo et al. [163], pp. 15–27. doi:10.1007/978-0-387-09684-1_2.

[110] C. Jensen, W. Scacchi, Collaboration, Leadership, Control, and Conflict Negotiation in the Netbeans.org Community, in: Feller et al. [74], pp. 48–52.

[111] M. Jørgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Transactions on Software Engineering 33 (1) (2007) 33–53. doi:10.1109/tse.2007.3.

[112] N. Juristo, C. B. Seaman, S. Vegas (Eds.), First International Symposium on Empirical Software Engineering and Measurement (ESEM'2007), Madrid, Spain 20-21 September, IEEE Computer Society, Los Alamitos, USA, 2007.

[113] V. B. Kampenes, T. Dybå, J. E. Hannay, D. I. K. Sjøberg, A systematic review of quasi-experiments in software engineering, Information and Software Technology 51 (1) (2009) 71–82. doi:10.1016/j.infsof.2008.04.006.

[114] C. Kapser, M. W. Godfrey, Cloning considered harmful considered harmful: patterns of cloning in software, Empirical Software Engineering 13 (6) (2008) 645–692. doi:10.1007/s10664-008-9076-6.

[115] B. A. Kitchenham, Guidelines for performing Systematic Literature Reviews in Software Engineering, Tech. rep., Software Engineering Group, School of Computer Science and Mathematics, Keele University, and Department of Computer Science, University of Durham, eBSE Technical Report, EBSE-2007-01 (2007).

[116] B. A. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering - A systematic literature review, Information and Software Technology 51 (1) (2009) 7–15. doi:10.1016/j.infsof.2008.09.009.

[117] B. A. Kitchenham, T. Dybå, M. Jørgensen, Evidence-based software engineering, in: A. Finkelstein, J. Estublier, D. Rosenblum (Eds.), Proceedings of the 26th International Conference on Software Engineering (ICSE 2004), May 23th28th, Edinburgh, Scotland, IEEE Computer Society, Los Alamitos, USA, 2004, pp. 273–281.

[118] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, J. Rosenberg, Preliminary Guidelines for Empirical Research in Software Engineering, IEEE Transactions on Software Engineering 28 (8) (2002) 721–734.

[119] K. R. Lakhani, E. von Hippel, How open source software works: 'free' user-to-user assistance, Research Policy 32 (6) (2003) 923 – 943. doi:10.1016/s0048-7333(02)00095-1.

[120] J. R. Landis, G. G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (1977) 159–174.
URL http://www.jstor.org/stable/2529310

[121] N. Lesiecki, Applyinq AspectJ to J2EE application development, IEEE Software 23 (1) (2006) 24–32. doi:10.1109/ms.2006.1.

[122] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. P. N. Slyngstad, M. Morisio, Development with off-the-shelf components: 10 facts, IEEE Software 26 (2) (2009) 80–87. doi:10.1109/MS.2009.33.

[123] J. Li, R. Conradi, O. P. N. Slyngstad, C. Bunse, M. Torchiano, M. Morisio, An Empirical Study on Decision Making in Off-The-Shelf Component-Based Development, in: Osterweil et al. [147], pp. 897–900. doi:10.1145/1134285.1134446.

[124] J. Li, M. Torchiano, R. Conradi, O. P. N. Slyngstad, C. Bunse, A State-of-the-Practice Survey of Off-the-Shelf Component-Based Development Processes, in: M. Morisio (Ed.), Proceedings of the 9th International Conference on Software Reuse (ICSR'06), June, 12th-15th, Torino, Italy, Vol. Volume 4039/2006 of LNCS, Springer, 2006, pp. 16–28. doi:10.1007/11763864_2.

[125] J. Lindman, M. Rossi, P. Marttiin, Applying Open Source Development Practices Inside a Company, in: Russo et al. [163], pp. 381–387. doi:10.1007/978-0-387-09684-1_36.

[126] D. Lorenzi, C. Rossi, Assessing Innovation in the Software Sector: Proprietary vs. FOSS Production Mode. Preliminary Evidence from the Italian Case, in: Russo et al. [163], pp. 325–331. doi:10.1007/978-0-387-09684-1_29.

[127] B. Lundell, B. Lings, E. Lindqvist, Perceptions and Uptake of Open Source in Swedish Organisations, in: Damiani et al. [53], pp. 155–163. doi:10.1007/0-387-34226-5.

[128] S. Lussier, New Tricks: How Open Source Changed the Way My Team Works, IEEE Software 21 (1) (2004) 68–72. doi:10.1109/MS.2004.1259222.

[129] B. Luthiger, C. Jungwirth, Pervasive fun, First Monday 12 (1).

[130] H. Mannaert, K. Ven, The Use of Open Source Software Platforms by Independent Software Vendors: Issues and Opportunities, in: Feller et al. [75], pp. 35–38. doi:10.1145/1083258.1083266.

[131] K. Martin, B. Hoffman, An Open Source Approach to Developing Software in a Small Organization, IEEE Software 24 (1) (2007) 46–53. doi:10.1109/MS.2007.5.

[132] J. Martinez-Romo, G. Robles, J. M. González-Barahona, M. Ortuño-Perez, Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company, in: Russo et al. [163], pp. 171–186. doi:10.1007/978-0-387-09684-1_14.

[133] J. Matusow, S. McGibbon, D. Rowe, Shared Source and Open Solutions: e-Government Perspective, in: Scotto and Succi [168], pp. 263–266.

[134] D. McIlroy, Mass Produced Software Components, Scientific Affairs Division NATO, Brussels, Belgium, 1969, pp. 138–151.

[135] C. Melian, M. Mähring, Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard, in: Russo et al. [163], pp. 93–104. doi:10.1007/978-0-387-09684-1_8.

[136] A. Mockus, R. T. Fielding, J. D. Herbsleb, Two case studies of open source software development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology 11 (3) (2002) 309–346. doi:10.1145/567793.567795.

[137] P. Mohagheghi, R. Conradi, Quality, productivity and economic benefits of software reuse: a review of industrial studies, Empirical Software Engineering 12 (5) (2007) 471–516. doi:10.1007/s10664-007-9040-x.

[138] M. Montesi, P. Lago, Software engineering article types: An analysis of the literature, Journal of Systems and Software 81 (10) (2008) 1694 – 1714. doi:10.1016/j.jss.2007.11.723.

[139] L. Morgan, P. Finnegan, Benefits and Drawbacks of Open Source Software: An Exploratory Study of Secondary Software Firms, in: Feller et al. [77], pp. 307–312. doi:10.1007/978-0-387-72486-7_33.

[140] L. Morgan, P. Finnegan, How Perceptions of Open Source Software Influence Adoption: An Exploratory Study, in: H. Österle, J. Schelp, R. Winter (Eds.), Proceedings of the Fifteenth European Conference on Information Systems (ECIS 2007), June 7-9 2007, St. Gallen, Switzerland, University of St. Gallen, 2007, pp. 973–984.

[141] U. Nikula, S. Jantunen, Quantifying the Interest in Open Source System: Case South-East Finland, in: Scotto and Succi [168], pp. 192–195.

[142] J. Noll, What Constitutes Open Source? A Study of the Vista Electronic Medical Record Software, in: Boldyreff et al. [25], pp. 310–319. doi:10.1007/978-3-642-02032-2_27.

[143] J. S. Norris, Mission-critical Development with Open Source Software: Lessons Learned, IEEE Software 21 (1) (2004) 42–49. doi:10.1109/MS.2004.1259211.

[144] A. Onetti, F. Capobianco, Open Source and Business Model Innovation. The Funambol Case, in: Scotto and Succi [168], pp. 224–227.

[145] The Open Source Initiative - Open Source Licenses (2009).
URL http://opensource.org/licenses

[146] T. Østerlie, L. Jaccheri, A Critical Review of Software Engineering Research on Open Source Software Development, in: W. Stanislaw (Ed.), Proceedings of the 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, Gdansk University Press, 2007, pp. 12–20.

[147] L. J. Osterweil, D. Rombach, M. L. Soffa (Eds.), Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), ACM Press, New York, USA, 2006.

[148] B. Özel, U. Jovanovic, B. Oba, M. van Leeuwen, Perceptions on F/OSS Adoption, in: Feller et al. [77], pp. 319–324. doi:10.1007/978-0-387-72486-7_35.

[149] J. W. Paulson, G. Succi, A. Eberlein, An Empirical Study of Open-Source and Closed-Source Software Products, IEEE Transactions on Software Engineering 30 (4) (2004) 246–256. doi:10.1109/TSE.2004.1274044.

[150] A. Persson, B. Lings, B. Lundell, A. Mattsson, U. Ärlig, Communication, Coordination and Control in Distributed Development: an OSS Case Study, in: Scotto and Succi [168], pp. 88–92.

[151] J. Ploski, W. Hasselbring, J. Rehwinkel, S. Schwierz, Introducing Version Control to Database-Centric Applications in a Small Enterprise, IEEE Software 24 (1) (2007) 38–44. doi:10.1109/ms.2007.17.

[152] C. Puschmann, P. Reimer, DiPP and eLanguage: Two cooperative models for open access, First Monday 12 (10).

[153] P. Ravesteyn, G. Silvius, Willingness to Cooperate Within the Open Source Software Domain, in: Russo et al. [163], pp. 367–373. doi:10.1007/978-0-387-09684-1_34.

[154] E. S. Raymond, The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly, Sebastapol, CA, 2001.

[155] E. S. Raymond, Up from Alchemy, IEEE Software 21 (1) (2004) 88, 90. doi:10.1109/ms.2004.1259228.

[156] P. C. Rigby, D. Cubranic, S. Thompson, D. M. German, M.-A. Storey, The challenges of creating open source educational software: the Gild experience, in: Scotto and Succi [168], pp. 338–340.

[157] J. E. Robbins, Adopting Open Source Software Engineering (OSSE) Practices by Adopting OSSE Tools, in: Feller et al. [76], pp. 245–264.

[158] G. Robles, S. Dueñas, J. M. González-Barahona, Corporate involvement of libre software: Study of presence in debian code over time, in: Feller et al. [77], pp. 121–132. doi:10.1007/978-0-387-72486-7_10.

[159] B. Rossi, B. Russo, G. Succi, A study on the introduction of Open Source Software in the Public Administration, in: Damiani et al. [53], pp. 165–171. doi:10.1007/0-387-34226-5_16.

[160] B. Rossi, B. Russo, G. Succi, Open Source Software and Open Data Standards as a form of Technology Adoption: a Case Study, in: Feller et al. [77], pp. 325–330. doi:10.1007/978-0-387-72486-7_36.

[161] B. Rossi, M. Scotto, A. Sillitti, G. Succi, Criteria for the non invasive transition to OpenOffice, in: Scotto and Succi [168], pp. 250–253.

[162] C. Rossi, A. Bonaccorsi, Why Profit-Oriented Companies Enter the OS Field?: Intrinsic vs. Extrinsic Incentives, in: Feller et al. [75], pp. 47–51. doi:10.1145/1083258.1083269.

[163] B. Russo, E. Damiani, S. A. Hissam, B. Lundell, G. Succi (Eds.), Proceedings of the 4th IFIP Working Group 2.13 International Conferences on Open Source Software (OSS2008) - Open Source Development Communities and Quality, September 7-10, Milano, Italy, Vol. 275/2008 of IFIP International Federation for Information Processing, Springer, Heidelberg, Germany, 2008. doi:10.1007/978-0-387-09684-1.

[164] B. Russo, P. Zuliani, G. Succi, Toward an Empirical Assessment of the Benefits of Open Source Software, in: Feller et al. [73], pp. 117–120.

[165] C. Santos Jr., Understanding Partnerships between Corporations and the Open Source Community: A Research Gap, IEEE Software 25 (6) (2008) 96–97. doi:10.1109/ms.2008.167.

[166] W. Scacchi, Free and Open Source Development Practices in the Game Community, IEEE Software 21 (1) (2004) 59–66. doi:10.1109/ms.2004.1259221.

[167] W. Scacchi, J. Feller, B. Fitzgerald, S. A. Hissam, K. R. Lakhani, Understanding Free/Open Source Software Development Processes, Software Process: Improvement and Practice 11 (2) (2006) 95–105. doi:10.1002/spip.255.

[168] M. Scotto, G. Succi (Eds.), Proceedings of The First International Conference on Open Source Systems (OSS2005), July 11-15, Genova, Italy, 2005.

[169] N. Serrano, S. Calzada, J. M. Sarriegui, I. Ciordia, From Proprietary to Open Source Tools in Information Systems Development, IEEE Software 21 (1) (2004) 56– 58. doi:10.1109/MS.2004.1259219.

[170] R. Shatnawi, W. Li, The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process, Journal of Systems and Software 81 (11) (2008) 1868 – 1882. doi:10.1016/j.jss.2007.12.794.

[171] D. I. K. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg, A. C. Rekdal, A survey of controlled experiments in software engineering, IEEE Transactions on Software Engineering 31 (9) (2005) 733–753. doi:10.1109/tse.2005.97.

[172] S. Y. Sohn, M. S. Mok, A strategic analysis for successful open source software utilization based on a structural equation model, Journal of Systems and Software 81 (6) (2008) 1014–1024. doi:10.1016/j.jss.2007.08.034.

[173] S. K. Sowe, I. Stamelos, L. Angelis, Understanding knowledge sharing activities in free/open source software projects: An empirical study, Journal of Systems and Software 81 (3) (2008) 431–446. doi:10.1016/j.jss.2007.03.086.

[174] D. Spinellis, Global software development in the FreeBSD project, in: P. Kruchten, D. Moitra, W. Strigel, C. Ebert (Eds.), Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner, ACM Press, 2006, pp. 73–79. doi:10.1145/1138506.1138524.

[175] R. M. Stallman, L. Lessig, Free Software Free Society: selected essays of Richard M. Stallman, Free Software Foundation, 2002.

[176] W. Stam, When does community participation enhance the performance of open source software companies?, Research Policy 38 (8) (2009) 1288–1299. doi:10.1016/j.respol.2009.06.004.

[177] I. Stamelos, L. Angelis, A. Oikonomou, G. L. Bleris, Code quality analysis in open source software development, Information Systems Journal 12 (1) (2002) 43–60.

[178] M. Staples, M. Niazi, Experiences using systematic review guidelines, Journal of Systems and Software 80 (9) (2007) 1425–1437. doi:10.1016/j.jss.2006.09.046.

[179] K. Staring, O. Titlestad, Networks of Open Source Health Care Action, in: Damiani et al. [53], pp. 135–141. doi:10.1007/0-387-34226-5_13.

[180] K.-J. Stol, M. A. Babar, Reporting Empirical Research in Open Source Software: The State of Practice, in: Boldyreff et al. [25], pp. 156–169. doi:10.1007/978-3-642-02032-2_15.

[181] F. Tiangco, A. Stockwell, J. Sapsford, A. Rainer, Open-source software in an occupational health application: the case of Heales Medical Ltd., in: Scotto and Succi [168], pp. 130–134.

[182] K. Toth, Experiences with Open Source Software Engineering Tools, IEEE Software 23 (6) (2006) 44–52. doi:10.1109/ms.2006.158.

[183] W. Tracz, J. Magee, M. Young (Eds.), Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), ACM, New York, USA, 2002.

[184] M. Umarji, S. E. Sim, C. Lopes, Archetypal Internet-Scale Source Code Searching, in: Russo et al. [163], pp. 257–263. doi:10.1007/978-0-387-09684-1_21.

[185] F. van der Linden, B. Lundell, P. Marttiin, Commodification of Industrial Software: A Case for Open Source, IEEE Software 26 (4) (2009) 77–83. doi:10.1109/ms.2009.88.

[186] K. Ven, H. Mannaert, Challenges and strategies in the use of Open Source Software by Independent Software Vendors, Information and Software Technology 50 (9-10) (2008) 991–1002. doi:10.1016/j.infsof.2007.09.001.

[187] K. Ven, D. Van Nuffel, J. Verelst, The Introduction of OpenOffice.org in the Brussels Public Administration, in: Damiani et al. [53], pp. 123–134. doi:10.1007/0-387-34226-5_12.

[188] K. Ven, J. Verelst, The Organizational Adoption of Open Source Server Software by Belgian Organizations, in: Damiani et al. [53], pp. 111–122. doi:10.1007/0-387-34226-5_11.

[189] K. Ven, J. Verelst, The Impact of Ideology on the Organizational Adoption of Open Source Software, Journal of Database Management 19 (2) (2008) 58–72.

[190] K. Ven, J. Verelst, H. Mannaert, Should You Adopt Open Source Software?, IEEE Software 25 (3) (2008) 54–59. doi:10.1109/ms.2008.73.

[191] V. Venkatesh, M. G. Morris, G. B. Davis, F. D. Davis, User Acceptance of Information Technology: Toward a Unified View, Mis Quarterly 27 (3) (2003) 425–478.

[192] I. Vessey, V. Ramesh, R. L. Glass, Research in Information Systems: An Empirical Study of Diversity in the Discipline and Its Journals, Journal of Management Information Systems 19 (2) (2002) 129–174.

[193] P. Vitharana, F. M. Zahedi, H. Jain, Design, Retrieval, and Assembly in Component-based Software Development, Communications of the ACM 46 (11) (2003) 97–102. doi:10.1145/948383.948387.

[194] E. von Hippel, G. von Krogh, Open source software and the "private-collective" innovation model: Issues for organization science, Organization Science 14 (2) (2003) 209.

[195] G. von Krogh, S. Spaeth, S. Haefliger, Knowledge Reuse in Open Source Software: An Exploratory Study of 15 Open Source Projects, in: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS 2005), IEEE Computer Society, Washington, DC, USA, 2005, p. 198.2. doi:10.1109/hicss.2005.378.

[196] G. von Krogh, E. von Hippel, The Promise of Research on Open Source Software, Management Science 52 (7) (2006) 975–983. doi:10.1287/mnsc.1060.0560.

[197] J. Waldo, Alive and well: Jini technology today, Computer 33 (6) (2000) 107–109. doi:10.1109/2.846324.

[198] D. A. E. Wall, Using Open Source for a Profitable Startup, Computer 34 (12) (2001) 158–160. doi:10.1109/2.970592.

[199] J. Warsta, P. Abrahamsson, Is open source software development essentially an agile method?, in: Feller et al. [73].

[200] A. I. Wasserman, E. Capra, Evaluating Software Engineering Processes in Commercial and Community Open Source Projects, in: Capiluppi and Robles [41], pp. 1–5. doi:10.1109/floss.2007.6.

[201] R. T. Watson, M.-C. Boudreau, P. T. York, M. E. Greiner, J. Donald Wynn, The Business of Open Source, Communications of the ACM 51 (4) (2008) 41–46. doi:10.1145/1330311.1330321.

[202] R. T. Watson, D. Wynn, M.-C. Boudreau, JBoss: The Evolution of Professional Open Source Software, MIS Quarterly Executive 4 (3) (2005) 329–341.

[203] J. Wesselius, The Bazaar inside the Cathedral: Business Models for Internal Markets, IEEE Software 25 (3) (2008) 60–66. doi:10.1109/ms.2008.79.

[204] J. West, How open is open enough?: Melding proprietary and open source platform strategies, Research Policy 32 (7) (2003) 1259 – 1285. doi:10.1016/s0048-7333(03)00052-0.

[205] J. West, S. O'Mahony, The Role of Participation Architecture in Growing Sponsored Open Source Communities , Industry & Innovation 15 (2) (2008) 145–168. doi:10.1080/13662710801970142.

[206] A. Westenholz, Institutional Entrepreneurs and the Bricolage of Intellectual Property Discourses, in: Damiani et al. [53], pp. 183–193. doi:10.1007/0-387-34226-5_18.

[207] M. N. Wicks, R. G. Dewar, A new research agenda for tool integration, Journal of Systems and Software 80 (9) (2007) 1569–1585. doi:10.1016/j.jss.2007.03.089.

[208] Y. Yang, J. Bhuta, B. W. Boehm, D. N. Port, Value-Based Processes for COTS-Based Applications, IEEE Software 22 (4) (2005) 54–62. doi:10.1109/MS.2005.112.

[209] Z. Yang, M. Jiang, Using Eclipse as a Tool-Integration Platform for Software Development, IEEE Software 24 (2) (2007) 87–89. doi:10.1109/ms.2007.58.

[210] T. S. Yoo, M. J. Ackerman, Open Source Software for Medical Image Processing and Visualization, Communications of the ACM 48 (2) (2005) 55–59. doi:10.1145/1042091.1042120.

[211] L. Yu, Understanding component co-evolution with a study on Linux, Empirical Software Engineering 12 (2) (2007) 123–141. doi:10.1007/s10664-006-9000-x.

[212] L. Yu, Self-organization process in open-source software: An empirical study, Information and Software Technology 50 (5) (2008) 361 – 374. doi:10.1016/j.infsof.2007.02.018.

[213] M. V. Zelkowitz, An update to experimental models for validating computer technology, Journal of Systems and Software 82 (3) (2009) 373–376. doi:10.1016/j.jss.2008.06.040.

[214] M. V. Zelkowitz, D. R. Wallace, Experimental Models for Validating Technologies, IEEE Computer 31 (5) (1998) 23–31.

[215] L. Zhao, S. Elbaum, Quality assurance under the open source development model, Journal of Systems and Software 66 (1) (2003) 65 – 75. doi:10.1016/s0164-1212(02)00064-x.

[216] S. Ziemer, Ø. Hauge, T. Østerlie, J. Lindman, Understanding Open Source in an Industrial Context, in: A. Dipanda, R. Chbeir, K. Yetongnon (Eds.), Proceedings of the 4th IEEE International Conference on Signal-Image Technology & Internet-Based Systems (SITIS'2008), November 30-December 3, Bali, Indonesia, IEEE Computer Society, Los Alamitos, USA, 2008, pp. 539–546. doi:10.1109/SITIS.2008.99.

APPENDICES

A. Included Publication Sources

Table 13 contains an overview of the journals/magazines and the workshop and conference proceedings included in this review, the databases from which the publications were identified, and the number of publications included in each stage (S1-5).

Source	Database	Number of publications						
		S1	S2	S3	S4	S5		
						R	E	N
ACM Transactions on Software Engineering and Methodology (TOSEM)	ACM Digital Library	207	22	2	0	0	0	0
Communications of the ACM	ACM Digital Library	3252	217	25	9	1	2	6
Empirical Software Engineering (EMSE)	Springerlink	272	37	11	0			
First Monday	Firstmonday.org	965	247	76	13	2	1	9
ICSE Workshop on Open Source Software Engineering (WOSSE)	opensource.ucc.ie (2001-2004), ACM Digital Library (2005)	95	95	90	20	7	7	6
ICSE Workshop on Emerging Trends in FLOSS Research and Development (FLOSS)	IEEE Xplore	15	15	12	3	2	0	1
IEE Review	ietdl.org	999	5	3	2	0	0	2
IEE/IET Software Proceedings	ietdl.org	323	12	6	0	0	0	0
IEEE Computer	IEEE Xplore	2638	30	23	8	0	5	3
IEEE International Symposium on Empirical Software Engineering (ISESE)	IEEE Xplore (2002-2005), ACM Digital Library (2006)	177	8	5	1	1	0	0
IEEE International Symposium on Software Metrics (METRICS)	IEEE Xplore	249	7	6	0	0	0	0
IEEE Software	IEEE Xplore	1439	76	50	21	1	14	6
IEEE Transactions on Software Engineering (TSE)	IEEE Xplore	900	16	16	0	0	0	0
Information and Software Technology (IST)	Sciencedirect	1089	81	14	2	1	1	0
Information Systems Journal	Wiley InterScience	279	9	7	1	0	0	1
International Conference on Open Source Systems (OSS)	oss2005.case.unibz.it (2005), Springerlink (2006-2008)	210	209	175	55	27	19	9
International Conference on Software Engineering (ICSE)	ACM Digital Library (2006 and 2008), IEEE Xplore (1998-2005, 2007)	1324	105	32	5	1	4	0
International Symposium on Empirical Software Engineering and Measurement (ESEM)	IEEE Xplore (2007), ACM Digital Library (2008)	151	18	12	0	0	0	0
Journal of Database Management	ProQuest	294	18	9	3	2	0	1
Journal of Industrial Economics	Wiley InterScience	304	2	2	0	0	0	0
Journal of Systems and Software (JSS)	Sciencedirect	1567	119	23	2	2	0	0
Knowledge Technology and Policy	Springerlink	364	28	13	7	1	0	6
Long Range Planning	Sciencedirect	1017	19	2	2	2	0	0
Management Science	ProQuest	1787	17	12	1	1	0	0
MIS Quarterly	ProQuest	406	7	3	1	1	0	0
MIS Quarterly Executive	Misqe.org	146	1	1	1	1	0	0
MIT Sloan Management Review	ProQuest	836	27	3	0	0	0	0
Organization Science	ProQuest	674	10	3	1	1	0	0
Research Policy	Sciencedirect	1232	52	18	4	4	0	0
Software Practice and Experience	Wiley InterScience	763	10	8	0	0	0	0
Software Process: Improvement and Practice	Wiley InterScience	315	21	12	1	1	0	0
						59	**53**	**50**
Total		**24 289**	**1540**	**674**	**162**	**112**		na

Table 13: Included journals and conferences

B. Characteristics of the Sample of Publications

The tables below contain overviews of the number of publications per year (Table 14, the contexts described in these publications (Table 15 and 18), the research methods used in the empirical research papers (Table 16), and the quality assessment (Table 17).

We use the following abbreviations: Stages 1 to 5 (S1-5), empirical research paper (R), experience report (E), non-empirical paper (N). The categories from the classification of the papers are abbreviated as follows: adoption of OSS in general (A), deploying OSS (D), using OSS CASE tools (T), integrating OSS (I), participating in OSS communities (PA), providing OSS products (PRO), and using "OSS development practices" (PRA).

Year	**S2**	**S3**	**S4**	**S5**		
				R	**E**	**N**
2008	302	109	26	18	6	2
2007	254	110	23	10	9	4
2006	262	106	28	13	7	8
2005	226	117	41	11	17	13
2004	156	64	17	5	6	6
2003	119	63	7	2	1	4
2002	86	37	5	0	3	2
2001	74	37	6	0	2	4
2000	26	9	2	0	1	1
1999	24	15	6	0	1	5
1998	11	7	1	0	0	1
Total	1540	674	162	59	53	50

Table 14: Number of publications in each stage distributed per year

	A	**D**	**T**	**I**	**PA**	**PRO**	**PRA**
Several private organizations	[6, 27, 32, 33, 94, 98, 99, 127, 139, 141, 153, 162, 172, 188, 189, 204]			[8, 43, 123]	[31, 49, 50]	[17, 49, 50, 82, 204]	
OSS community					[2, 20, 28, 51, 70, 101, 129, 132, 158]	[110, 126, 132]	
One private organization				[106, 130, 186]	[106, 130, 186]	[144, 201, 202, 206]	[125, 135]
Public	[91]	[62, 91, 148, 159, 160, 161, 164, 187]					
Mixed public and private	[29, 93]	[37]			[200]	[7, 52, 200]	
Unclear		[190]					

Table 15: The type of contexts described in the empirical research papers

	A	D	T	I	PA	PRO	PRA
Case study	[94, 99, 139, 188, 189]	[37, 62, 159, 160, 164, 187, 190]		[106, 130, 186]	[20, 49, 50, 70, 101, 106, 130, 186]	[17, 49, 50, 52, 110, 144, 201, 202, 206]	[125, 135]
Survey	[27, 29, 32, 33, 91, 98, 127, 141, 153, 162, 172]	[91, 148]		[8, 43, 123]	[31, 129, 200]	[200]	
Data analysis					[2, 28, 51, 132, 158]	[82, 132]	
Experiment		[161]				[126]	
Field study	[204]					[204]	
Grounded theory	[6]						
Case study and survey	[93]					[7]	

Table 16: Research methods used

QA score	Papers
2	[29, 202]
3	[52, 110, 164, 201, 206]
4	[17]
5	[91, 200, 204]
6	[32, 37, 106, 127, 130, 148, 159, 160]
7	[20, 62, 93, 94, 132, 135, 141, 162]
8	[2, 6, 28, 31, 33, 49, 99, 101, 123, 139, 144, 153, 158, 172, 187]
9	[7, 8, 27, 43, 50, 51, 70, 82, 98, 125, 126, 129, 161, 186, 188, 189, 190]

Table 17: Quality assessment: Distribution of research papers

	D	T	I	PA	PRO	PRA
Private		[131, 143, 151, 198, 209]	[9, 87, 90, 107, 121, 143, 169, 181, 198]	[9, 107, 128]	[14, 16, 19, 34, 87, 95, 100, 131, 133, 197, 210]	[12, 60, 61, 96, 97, 128, 131, 179, 197, 203]
Public	[24, 57, 63, 81]	[13, 182]	[10, 15, 54, 152, 182]	[11, 15]	[22, 23, 47, 104, 108, 156]	[182]
Research project	[3, 4]		[3, 64, 86]		[26]	
Community				[88]	[109]	

Table 18: The type of contexts described in the experience reports

Appendix B

Interview Guides and Questionnaires

Table B.1 contains an overview of the interview guides and questionnaires that were used to gather parts of the data for this thesis. All the documents referred to in the table are available at: `http://www.oyvindhauge.net/phd/`.

Table B.1: Interview guides and questionnaires used as in this thesis

Study	Description	Developed by	Paper
1	COSI Questionnaire	Øyvind Hauge, Andreas Røsdal and Carl-Fredrik Sørensen	P1
1,2	COSI Interview guide	Øyvind Hauge, Andreas Røsdal and Carl-Fredrik Sørensen	P1, P4
2	Norwegian survey interview guide	Marinela Gerea, Carl-Fredrik Sørensen, Øyvind Hauge, and Claudia Ayala	P4
2	Norwegian survey questionnaire	Øyvind Hauge and Carl-Fredrik Sørensen	P2
5	Norwegian questionnaire	Ketil Sandanger Velle, Tron André Skarpenes, Øyvind Hauge, and Reidar Conradi	P7
5	Norwegian interview guide	Ketil Sandanger Velle, Tron André Skarpenes, Øyvind Hauge, and Reidar Conradi	P7
6	Norwegian/Spanish interview guide	Claudia Ayala, Øyvind Hauge, Reidar Conradi, Xavier Franch, and Jingyue Li	P5