NTNU – Trondheim
Norwegian University of
Science and Technology

Collaboration Patterns among Commercial Firms in Community-Based OSS Projects

Terje Snarby

Master of Science in Computer Science

Submission date:	June 2013
Supervisor:	Reidar Conradi, IDI
Co-supervisor:	Daniela Soares Cruzes, IDI Duc Anh Nguyen, IDI

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

PROBLEM DESCRIPTION

Open Source Software projects are increasingly influenced by participation and contributions from commercial firms. Knowledge of how firms communicate and collaborate across organizational boundaries within projects may be highly valuable to practitioners and researchers. However, there exist few studies that examine the subject. As a response, we conducted an exploratory case study of commercial firm participation in Wireshark as specialization project during fall 2012. The study identified communication patterns between firms in Wireshark by analyzing publicly archived information and creating social networks from the data.

In this study, we investigate the challenges and effective practices of technical coordination across organizational boundaries within the social networks of firms. The work in the specialization project is extended to include a descriptive analysis phase. Additionally, the findings from Wireshark are further explored by conducting a case study of a similar OSS project, Samba.

Abstract

Context Open Source Software (OSS) development is a highly distributed and collaborative activity and was initially performed by developers volunteering their time and effort. During the recent years, OSS has transformed to become a more mainstream and commercially viable software development approach. This is reflected in the increasing involvement from commercial firms. Currently, there exist little or no research literature examining how commercial firms collaborate across organizational boundaries and how this may affect community-based OSS projects.

Objective This thesis aims to help fill the identified gap by investigating the influence of organizational boundary on team collaboration and project evolution in community-based OSS projects by answering the following research questions:

RQ1: How do commercial firms collaborate across organizational boundaries in community-based OSS projects?

RQ2: How influential is firm awareness in community-based OSS projects?

Method A two-phased case study approach was adopted, compromising exploratory and descriptive analysis. The cases, Wireshark and Samba, were examined in a quantitative and qualitative approach using found documents and conducting interviews with firm-paid developers. Social network and thematic content analysis were applied.

Results This thesis provides the following three contributions:

C1: Empirical data from two OSS projects providing knowledge related to how commercial firms collaborate with each others across organizational boundaries in community-based OSS projects.

C2: A conceptual model illustrating commercial firm's collaboration across organizational boundaries in community-based OSS projects, extending Crowston's onion model.

C3: A set of recommendations to commercial firms that want to participate and utilize community-based OSS projects.

Conclusion Commercial firms participate and collaborate heavily in community-based OSS projects and are present in all the roles identified in Crowston's onion model. Furthermore, independently of firm size and degree of involvement, the firms commonly have one person acting as a gatekeeper between the community and firm managing code, communication and/or bug issues. The findings indicate that firm awareness is a vital part of the firm cross-collaboration and mainly influence the development in a positive manner.

Oppsummering
(Norwegian abstract)

Kontekst Utvikling av åpen kildekode er en distribuert og samarbeidskrevende aktivitet som i utgangspunktet ble utført av frivillige programvareutviklere. I løpet av de siste årene har åpen kildekode endret seg til å bli en ordinær og kommersielt levedyktig programvareutviklingsmetode, noe som er reflektert i den økende deltakelsen av kommersielle aktører. Det eksisterer lite eller ingen forskningslitteratur som undersøker hvordan de kommersielle aktørene samarbeider på tvers av organisatoriske grenser i fellesskapsgrunnlagte åpen kildekode-prosjekter samt hvordan dette samarbeidet kan påvirke prosjektene.

Formål Målet med denne avhandlingen er å bidra med ny kunnskap til det identifiserte kunnskapshullet. Innflytelsen av de organisatoriske grensene på teamsamarbeidet og prosjektets utvikling i fellesskapsgrunnlagte åpen kildekode-prosjekter blir utforsket ved å besvare følgende problemstillinger:

RQ1: Hvordan samarbeider kommersielle aktører på tvers av de organisatoriske grensene i fellesskapsgrunnlagte åpen kildekode-prosjekter?

RQ2: Hvor innflytelsesrik er firmabevisstheten i fellesskapsgrunnlagte åpen kildekode-prosjekter?

Metode En casestudie med to faser ble benyttet, bestående av utforskende og beskrivende analysemetoder. To åpen kildekode-prosjekter, Wireshark og Samba, ble undersøkt både kvantitativt og kvalitativt hvor dokumenter og transkriberte intervjuer med firmaansatte programvareutviklere ble benyttet som data.

Resultat Denne avhandlingen har tre bidrag, og de er følgende:

C1: Empirisk data som beskriver og gir innsikt i hvordan kommersielle firmaer deltar og samarbeider på tvers av organisatoriske grenser i fellesskapsgrunnlagte åpen kildekode-prosjekter.

C2: En konseptuell modell som illustrerer hvordan kommersielle firmaer samarbeider på tvers av organisatoriske grenser i fellesskapsgrunnlagte åpen kildekode-prosjekter. Den er en utvidelse av Crowstons onion modell.

C3: En liste med anbefalinger til kommersielle firmaer som ønsker å delta og benytte fellesskapsgrunnlagte åpen kildekode-prosjekter.

Konklusjon Kommersielle firmaer deltar og samarbeider i stor grad i fellesskapsgrunnlagte åpen kildekode-prosjekter. De er til stede i alle roller identifisert av Crowston i hans onion modell. I tillegg, uavhengig av firmastørrelse og deltakelsesgrad, har firmaene vanligvis en person fungerende som et bindeledd mellom de enkelte åpen kildekode-felleskapene og firmaene. Bindeleddet har ansvar for flyt av kode, kommunikasjon og/eller feilmeldinger. Resultatene indikerer at firmabevissthet er et viktig element i samarbeidet, og påvirker programvareutviklingen fortrinnsvis positivt.

PREFACE

This thesis is the result of the subject *TDT4900 - Computer and Information Science, Master Thesis* at the Department of Computer and Information Science (IDI) at the Norwegian University of Technology and Science (NTNU).

I would like to thank my main supervisor Professor Reidar Conradi for guidance and support throughout the writing of this thesis. Additionally, I would like to thank my co-supervisors Associate Professor Daniela Soares Cruzes and PhD Candidate Duc Anh Nguyen for helpful advice and feedback. Working with you have been both engaging and educational, and I have really enjoyed the opportunity and experience.

Last, I would like to thank all the software developers that participated in interviews during the research. They shared their time and knowledge with me, and the results would not have been as complete without them.

Trondheim, June 11, 2013.

Terje Snarby

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBRIVATIONS

FLOSS Free/Libre/Open Source Software

FOSS Free and Open Source Software

FS Free Software

FSF Free Software Foundation

GPL General Public License

GSD Global Software Development

GUI Graphical User Interface

LGPL Lesser General Public License

OSI Open Source Initiative

OSS Open Source Software

OSSD Open Source Software Development

SE Software Engineering

Part I

Introduction

CHAPTER 1 INTRODUCTION

This chapter gives an introduction to the motivation of the study, research objective, research questions and presents a summary of the main contributions.

1.1 Motivation

Open source software (OSS) started out mainly as an ideologically driven software development methodology, where volunteer developers formed communities to develop software, contributing their time and effort with no monetary reward in mind. During the recent years this picture has changed significantly. OSS has gained considerable traction, and transformed into a more mainstream and commercially viable form [1]. This is reflected in the increasing involvement of commercial firms in OSS projects, which pay their developers to participate and contribute to the community development [2, 3].

The great success of large OSS projects, including Linux, Mozilla and Eclipse, has proven the viability and capability of the software development approach. Linux, which is probably the most prominent OSS product all-time, is estimated to have 65 Million users[1] and is developed by thousands of programmers. The advantages and capabilities seen in these successful projects give fuel for other OSS projects to be initiated. Well-known OSS hosting sites reports a steady growth of projects, users and contributors. SourceForge is one of these, hosting more than 324 000 projects and having more than 3.4 million registered developers[2]. However, researchers have pointed out that only a minority of these projects are active and stable, operating with a number between 8 % to 10 % active projects [4].

As OSS is growing and increasingly becoming an alternative or complement to proprietary software, commercial firms seek new business models using OSS for creating value and generating profit; changing the way they develop, acquire,

[1] http://linuxcounter.net/main.html, Retrieved: 2013-05-07

[2] http://sourceforge.net/about, Retrieved: 2013-05-07

use and commercialize software. Hauge et al. conducted a systematic literature review on OSS research and identified six distinctly different ways organizations adopt OSS [5]. Two of which involve working with a community of developers to develop software. Lakhani and Wolf found that approximately 40 % of software developers are paid by their firm to participate and contribute to OSS projects [6]. These findings are supported by similar results obtained by Hars and Ou [7]. This trend is intriguing for academic researchers who are interested in exploring various aspects of OSS and its implications for commercial business, which is a relative new and unexplored field. Aksulu et al. conducted a comprehensive review and synthesis of the open source research, and found 618 research articles, remarking that OSS is an ever-changing field, spanning several different research areas and disciplines [8]. Martin Höst conducted a systematic review of the research literature concerning OSS in commercial software development, finding only 19 articles covering the subject [9]. Among theses studies, the main focus have been the business model.

Prior studies have found that individual developers and firms have significantly different motivations and incentives for participating in OSS [10]. Furthermore, the motivations and incentives among firms are also highly heterogeneous. With the large percentage of firm-paid developers in OSS projects, which most likely will keep on growing, and their diverse reasons for participating in the development, it becomes increasingly interesting to examine how they come together and collaborate within OSS communities to develop software collectively. It is well-known that commercial firms working within the same domain are typically competitors in the market, and that best practices and competitive advantages are kept secret. A key question in this context is: Does the presence and awareness of firms in OSS projects affect and influence the collaboration?

Collaboration and communication are important aspects of today's society. Due to higher education, people are increasingly becoming more specialized in their discipline, and knowledge and expertise are highly valued traits. The specialization require a higher level of coordination and collaboration, so that common goals can be achieved [11]. This specialization is also observed in software engineering, where products and solutions are becoming more and more complex, requiring large teams of developers to work together. Studies report that lack of collaboration result in more bugs, lower code quality and less efficient development [12, 13]. In OSS projects, developers are often geographically dispersed around the world and rarely or never meet face-to-face. How these developers communicate and collaborate have been examined in prior studies [14, 15, 16, 17]. However, previous work has primarily investigated the collaboration on an individual or community level, rather than on the firm level. To our knowledge, there exist few or no studies that examine the organizational cross-collaboration between commercial firms in OSS projects, making it an interesting research topic. Hauge remarks this by proposing the following further research question: "*How may organizations successfully collaborate through community- or consortium-based software development?*" [5].

Pisano et al. assert that the leaders in innovation will be those that successfully collaborate with a network of outsiders, which is highly applicable in OSS [18]. A better understanding of this may have a large impact on researchers and practitioners.

The aim of this thesis was to build on existing research and contribute with new knowledge to the identified gap in the research literature through an empirical inquiry. To achieve this, a two-phased case study approach was adopted. Wireshark and Samba were selected as the cases, two typical instances of successful and on-going community-based OSS projects. In the first phase, the cases were exploratory examined using quantitative analysis. The projects' publicly available information archives were used as data sources, mainly the mailing list and bug tracking system, to establish an overview of the cross-collaboration among commercial firms in the OSS projects. In the second phase, the cases were descriptively examined using qualitative analysis. We conducted interviews with firm-paid developers identified in the first phase to explore the validity of the findings and further investigate properties of the collaboration patterns. This dual-phase approach allowed us to create a complete picture of the cross-collaboration by triangulating from multiple data sources, which is an established strategy for improving reliability and validity in case studies [19].

1.2 Key Terminology

Community-based OSS project and collaboration are two key terms used throughout this thesis. Here we give a short description of each term:

Community-based OSS project (also referred to as community-founded OSS project) is an OSS project founded and managed by a distributed group of individuals who do not share the same employer [20].

Collaboration is a process where two or more entities (people, developers, organizations) work together on a task with the objective to realize shared goals. Collaboration includes activities such as communication, coordination and cooperation.

The terms collaboration and communication will be used interchangeably in the remainder of this thesis.

1.3 Research Objective

The goal of this thesis is to investigate the influence of organizational boundary on team collaboration and project evolution in community-based OSS projects. The study would focus on: (1) Identifying the presence of commercial firm participation in community-based OSS projects; (2) characterizing the collaboration pattern between developers from different firms; (3) exploring the influence of organization-cross collaboration on community-based OSS project outcomes and evolution.

1.4 Research Questions

The overall research objective was broken down into two main research questions, which in turn was broken down into more specific sub research questions.

RQ1 How do commercial firms collaborate across organizational boundaries in community-based OSS projects?

- **RQ1.1** How much do firms participate in terms of entries in the developer mailing list, bug tracking system and code repository?
- **RQ1.2** How do firms collaborate within an OSS community observed from the public software informalisms?
- **RQ1.3** How do firms collaborate within an OSS community observed from the participating developer's perspective?
- **RQ1.4** How is the boundary between OSS community and firms managed?

RQ2 How influential is firm awareness in community-based OSS projects?

- **RQ2.1** How do firm-paid developers become aware of other firm-paid developers, and how do they perceive them?
- **RQ2.2** How does firm awareness influence the collaboration practices?
- **RQ2.3** How does firm awareness influence the decision making towards the community-based OSS project?

To answer the questions we used social network analysis and thematic content analysis, which have been successfully applied in similar research in other domains when studying collaboration and people interaction. Description of these are given in the pre-study, see chapter 3.

1.5 Summary of Contributions

Here we present a summary of the four main contributions created by the research conducted in this thesis:

- Empirical data from two OSS projects providing knowledge related to how commercial firms participate and collaborate across organizational boundaries in community-based OSS projects.
- A conceptual model describing the commercial firm's collaboration patterns across organizational boundaries in community-based OSS projects, using Crowston's onion model as foundation. The model captures several aspects of the collaboration and present these in an intuitively manner.
- A set of recommendations to commercial firms that want to participate and utilize community-based OSS projects. The recommendations provide insights which can be used by firms to, for instance, adapt their OSS adoption strategy. Firm awareness and firm perception are key elements of the recommendations.

1.6 Thesis Structure

The thesis is structured as seven chapters and is organized in the following way:

Chapter 1 gives an introduction to the thesis.

Chapter 2 establishes a theoretical background of OSS and its underlying concepts.

Chapter 3 presents a pre-study of essential concepts and a tool selection.

Chapter 4 describes the research design and gives detailed insight into processes and techniques used in the research.

Chapter 5 presents the results of the conducted research.

Chapter 6 discuss the results presented in Chapter 6 and answers the research questions, propose a conceptual model and a set of recommendations for practitioners, and evaluate the threats to validity.

Chapter 7 summarizes the research, state the contributions and provide suggestions for further research.

Part II

Pre-Study

CHAPTER 2

BACKGROUND

This chapter establishes a theoretical background for the research context. The main focus of this thesis will be commercial firms and their involvement in community-based OSS projects, examining their collaboration across organizational boundaries. However, a sound understanding of the underlying concepts of OSS is required for building new knowledge and putting the work in the context of others.

2.1 What is Open Source Software?

Open source software (OSS) is software supplied under licensing terms accepted by the Free Software Foundation (FSF) or the Open Source Initiative (OSI) [21, 22]. The free software movement has since 1983 campaigned for software users' freedoms. Some developers did not agree on all the goals of the movement and decided to split and establish their own movement, the open source movement [23]. We start by describing the FSF briefly and continue with the history of OSS.

The FSF was established by Richard Stallman in 1985, with the intention to ensure software users with the following four basic freedoms [23]:

0. The freedom to run the software as you wish, for any purpose.
1. The freedom to study and modify the software to suit your needs.
2. The freedom to redistribute copies, either gratis or for a fee.
3. The freedom to distribute modified versions of the software, so that the community can benefit from your improvements.

Software complying with these freedoms is regarded as free software. The freedoms allow anyone to use, modify, improve and redistribute the software freely. A precondition to fulfilling these freedoms is that the source code of the software is open and available.

2.1.1 The History of Open Source Software

The rationale behind the establishment of the FSF, and subsequent establishment of the OSI can be illustrated by a brief review of the history of software. Hauge created a timeline consisting of three phases, providing a simplified view of relevant events in the OSS history [24]. We extend this timeline with a fourth phase, including recent relevant events. This timeline is displayed in Figure 2.1, followed by descriptions of each phase.

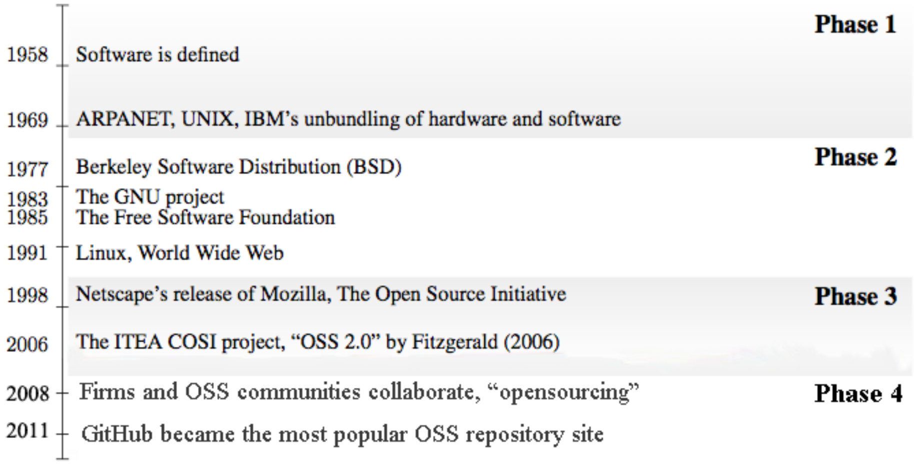

Phase 1
1958 Software is defined
1969 ARPANET, UNIX, IBM's unbundling of hardware and software
Phase 2
1977 Berkeley Software Distribution (BSD)
1983 The GNU project
1985 The Free Software Foundation
1991 Linux, World Wide Web
Phase 3
1998 Netscape's release of Mozilla, The Open Source Initiative
2006 The ITEA COSI project, "OSS 2.0" by Fitzgerald (2006)
Phase 4
2008 Firms and OSS communities collaborate, "opensourcing"
2011 GitHub became the most popular OSS repository site

Figure 2.1: Simplified timeline of relevant events in the OSS history (adapted from Hauge [24]).

Phase 1: Engineers, researchers and hackers

The first phase range from late 1950s to late 1960s. The programmers at the time had their background from engineering and physics, and they used computers mainly to solve problems in their own domain. The computing industry made money selling hardware, and supplied software as an additional service without extra cost [25]. Software was shared open and freely among users with common interests. Programmers of this era were often called hackers. In 1969 IBM decided to unbundle their software from the hardware, ending the first phase [26].

Phase 2: Commercialized proprietary software

Other firms followed IBM's footsteps; unbundling their hardware and software. During the late 1970s and early 1980s, the use of software exploded. Firms started to realize the commercial value of software, and they added technical and formal restrictions to the software to exploit this potential. Accessible source code was no longer supplied with the software and distribution was highly restricted by copyright licenses. Software users now had to pay a fee to acquire and use the software.

In strong contrast to the commercialization and closing-of-source-code evolution, Richard Stallman announced the initiation of the GNU Project in 1983 and established the FSF in 1985 [23, 27]. His main objective was to promote FS, and ensure software users with the four basic freedoms presented above. He believed that use of software should not be restricted. Programmers should not use their time to create duplicate proprietary solution, instead they should use the time for innovation. One of the most prominent OSS products to this date was initiated and developed in this phase; the operating system Linux.

Phase 3: The emergence and rise of OSS

Despite the effort Stallman and the FSF put into emphasizing that the word "free" in free software is referring to freedom, and formulated the distinction as "*free as in free speech, not as in free beer*", the use of the word did not correspond well with commercial firms and business. "Free" was often misinterpreted as free of charge, and firms did not want to associate their products with free. The third phase was initiated in 1997 by the publication of Eric S. Raymond's essay *The Cathedral and the Bazaar*, which received massive attention from software practitioners and researchers [28]. To better comply with the increasing attention from commercial firms, the term "open source" was coined by a group of FS advocates at a strategy meeting in 1998, and the OSI was formed shortly thereafter by Eric S. Raymond and Bruce Perence [29]. Raymond's essay is said to have convinced Netscape, a fortune 500 company, to release its Mozilla web browser as open source and started the commercialization of OSS [30].

Beside avoiding the ambiguity of "free" and creating a more acceptable term to use in business, the goal of the OSI is to create a more pragmatic and business-oriented version of FS. This was achieved by being looser in some aspects. For instance, the OSI has accepted a few licenses that FSF considers as unacceptably restrictive to the user. The effort by OSI has paid off and the adoption of OSS in commercial firm has steadily been rising. In 2006, Fitzgerald reported that OSS has transformed into a "*more mainstream and commercially viable form*", which he has coined as "OSS 2.0" [1].

Phase 4: Opensourcing and GitHub

Adoption of OSS and participation in OSS projects by commercial firms increased throughout phase 3 and continued in phase 4. Ågerfalk and Fitzgerald report that commercial firms and OSS communities collaborate to develop software which is of commercial interest to the firms. They describe this as outsourcing to an unknown workforce, an approach they coin as "opensourcing". A firm can utilize the effort and expertise by an OSS project's community members, which may be firm-paid or volunteering developers. More and more firms adopt OSS and use opensourcing as part of their business model.

In May 2011, GitHub[1] became the most popular OSS repository site [31]. The site is a web-based service for hosting software projects using the Git revision control system. GitHub has the slogan "*social coding (for all)*" and provides social networking functionality including feeds, performance statistics and overview of friends and followers. The emergence of GitHub had a major impact on OSS, making it more decentralized and changing the focus to become less about the project and more about the participating individuals. Coordination, collaboration and contribution became easier and the number of commits to OSS projects has increased significantly.

2.1.2 FS, FLOSS, FOSS and OSS

There are several different understandings of what OSS is, and there exist several similar terms, including: Free Software (FS), Free/Libre/Open Source Software (FLOSS), Free and Open Source Software (FOSS). Despite minor differences in history and ideology, the mentioned terms will be used equivalent with open source software, and no distinction between the terms will be made for the remainder of this thesis.

Nevertheless, it is important to keep in mind that even though projects are addressed with the single term "open source software", there are numerous kinds of OSS projects. Each defined by its own set of principles, practices, culture, and licenses [30]. For example, some OSS projects aim to create software which can be utilized by commercial firms, requiring long-term planning and high source code stability. While other projects are just for fun, mainly driven by individual effort and contributions, making the development unpredictable and vulnerable, which are not business friendly properties. However, two common aspects in all OSS projects are: (1) Transparency of development, which allows anyone to observe and participate in the development, (2) freedom to build more complex systems from already available building blocks. This are aspects that allow rapid development of software without a large initial investment.

2.1.3 Open Source Software Licenses

Use of licenses is a vital part of FS and OSS. Copyright licenses are used to protect intellectual property and to regulate use and distribution. The use of licenses is widespread in the software world, OSS included. However, there is a crucial difference between an ordinary license and an OSS license. In addition to protect and regulate the software, an OSS license include a legal agreement between the creator of the software and the individual user. This agreement grants the user the four freedoms identified by the FSF, but may place some additional restrictions on distribution of the OSS product combined with proprietary software. Commercial

[1] https://github.com/

firms must be aware of the licensing terms associated with the OSS products they adopt or release, and have a clear idea of how they will use the software in the future. Disclosure with the wrong type of license may invalidate business models and prevent future development scenarios [32].

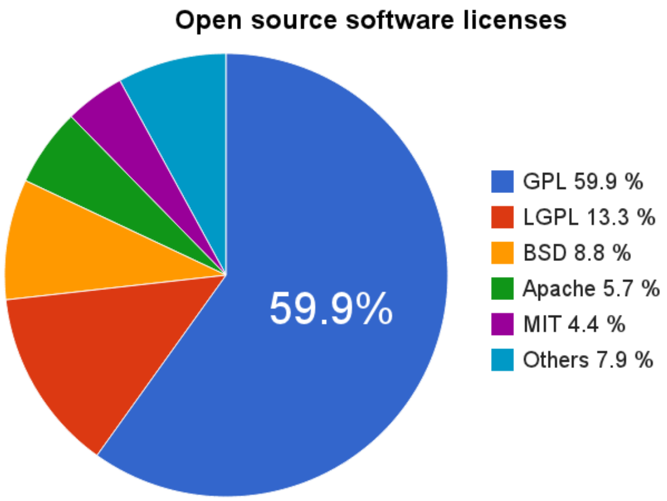


Figure 2.2: Summary of OSI-approved licenses used in OSS projects hosted on SourceForge, Retrieved: 2013-04-03

OSS licenses can be divided in two main categories: Reciprocal and academic licenses [33]. The main difference between the licenses is in the rights they grant to redistribution of derived works. Following is a short description of each category:

Reciprocal Licenses

Reciprocal licenses are often referred to as copyleft, which is a play on the word copyright. The idea behind copyleft is to ensure the same freedoms to all users of all future versions of a piece of software. The GNU Project defines copyleft as:

> "*(...) a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well*" [34]

Acknowledgement of the GNU Project's definition implies that software licensed with a copyleft license enforce all derived works to use the same copyleft license, and this way protect software from becoming private intellectual property [35]. Reciprocal licenses include the General Public License (GPL), Lesser General Public License (LGPL) and Mozilla Public License (MPL). GPL was written by Richard Stallman for the GNU Project, and is by far the most popular OSS license. This is illustrated in Figure 2.2, where GPL has a ratio of 59.9 %.

In addition to providing protection and regulations to software, reciprocal licenses are used by commercial firms to send a signal and create a brand association. Commercial firms are increasingly releasing their software as OSS, and selecting a reciprocal license communicate to everyone that the software will always be open and available for use and development by anyone. Volunteering developers are thus assured that their contributions will benefit the community and not be used by the project owner for profit.

Academic Licenses

Academic licenses are often referred to as permissive licenses. They offer the four basic freedoms and they allow, but don't require, distribution of source code in derivate works. In addition, developers are allowed to create proprietary software from permissive licensed code contributed by other developers, without attributing them. Developers are thus allowed to add restrictions to the software, which is in strong contrast to reciprocal licenses. Academic licenses include the Berkley Software Distribution (BSD), Massachusetts Institute of Technology (MIT) and Apache licenses. The BSD license is the most used academic license (see Figure 2.2).

2.1.4 Open Source Software Project Types

There are three main types of OSS projects [20]:

- A community-based OSS project, also known as community-founded, is an OSS project developed and maintained by a distributed group of individuals independent of their employment context [20]. The project is typically initiated by one or more developers with the aim of solving a problem or creating an open source solution to a proprietary software solution. Initially, the community-based model was the most common OSS model. Prime examples are Linux and Apache. Firms can sponsor the development, however, it is not possible to "buy" positions or commit rights in the project. Commercial firms, sponsoring or not, must build and maintain their role in the project under the same terms as volunteering developers.

- A vendor-based OSS project, also known as sponsored or commercial project, is a project where a sponsor of a software project release internally developed code to the public under an OSS license. Recently, there has been an increase in vendor-based projects. A sponsor could, for instance, be a firm, government agency or a non-profit organization. The intention of a vendor-based project is to attract external developers to join the development, and thus be able to reap the benefits observed in successful OSS projects [36]. Achieving greater software adoption and receiving help from volunteers are among the benefits. Eclipse is an example of a vendor-based OSS project, released as OSS by IBM in 2001.

- A hybrid OSS project is a project which has evolved from a community-based OSS project to becoming a more vendor-based project. Firms use the

OSS as their business model and sell pre-packaged releases, offer training, support and consultancy services based on the OSS. At the same time as the firms are profiting from the software, they receive benefits observed in typically OSS development; external developers help to maintain and develop the software. The firms often develop proprietary add-ons and customizations that they offer to their customers. Examples are Red Hat selling Red Hat Enterprise Linux and HortonWorks selling Apache Hadoop.

2.2 Open Source Software Development

In this section we present the general characteristics of OSS Development (OSSD), mainly focusing on the volunteering development aspect of OSS and the individual developer participation in community-based OSS projects. In the next section the commercialization of OSS is evaluated, focusing on participation of commercial firms and how and why they utilize OSSD.

OSSD is a highly distributed and collaborative activity. Developers are often geographically dispersed, and rarely or never meet face-to-face [37]. On the basis of these characteristics and the success of some OSS projects, OSSD has been established as a hot topic of discussion among researchers. Looking at the extremes, there is at one hand some researchers who describe OSSD as something contrastingly different than traditional software engineering (SE). They believe that OSSD can produce high-quality software on a rapid time scale for free or a very low cost, by utilizing highly talented volunteering developers. This perception infringe with IBM software engineer Fred Brooks' famous principles for SE: "*(...) there is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity*" and "*(...) adding manpower to a late software product makes it later*". On the other hand, some researchers are more disbelieving and characterize OSSD as variable-quality software development approach which often result in lacking documentation, unpredictable stability and reliability, and is formed on a uncertain legal foundation.

The main body of OSS research suggests a more balanced view on OSSD as a software development approach. It is not the "silver bullet" indicated by some, nor is it chaotic and precarious as stated by others. Rather, it is quite similar to traditional SE in many fields, but may provide additional benefits if successfully applied. There are some prominent success stories where category killer applications have been developed in a typical OSSD context. Many researchers claim that the credibility of OSS is biased towards a better perception than it should have, because of the emphasized attention the success stories receive in the research literature. However, it is apparent that developing and maintaining complex software successfully in a *global software development* (GSD) approach is possible.

Fitzgerald predict that OSS will be part of the software landscape in the foreseeing future and suggest that traditional software engineering can draw valuable lessons from it [38]. His prediction is already visible as there has been a significant increase in commercial firms that initialize vendor based-OSS projects and/or adopt OSS in other ways. The characteristics and attributes of OSSD are described in the following subsections.

2.2.1 Motivation

In traditional software development, software is generally developed by a team of developers employed by a commercial firm aiming to make profit. The developers in such a setting is motivated to work as it is part of their job and they receive financial reimbursement for their labor. This is in strong contrast to OSS development where software is developed by a community of developers, many of which are volunteers, contributing their time and effort free of charge. Firm-paid developers can also be regarded as volunteers as they do not receive direct payment from the projects [39]. Several studies have investigated why programmers engage in OSS projects, especially questioning the developer's motivation to do the work for free. Lakhani and Wolf found that motivations are quite diverse and they lists the following items in decreasing order of relevance [6]:

- Intellectual engagement
- Knowledge sharing
- Product creation and development
- Ideology, reputation, and community obligation

Reputation is low on the list of motivations, however, its importance rise as the length of the participation increase. Many of the motivations are related to the act of engaging in collaboration with others that have the same interests and motivations to develop software collectively in communities. Thus, the perception of a community in OSS projects is a major motivation for the participants.

2.2.2 Community

Development of software in community-based OSS projects is typically initiated by a very small collection of developers (referred to as founders) that have a problem they want to solve, "*scratching their own itch*", as Eric S. Raymond famously put it [28]. A successful OSS development can be seen as two phases. During the first phase the development of the software is managed and carried out by the founders. Then, if the ideas of the founders are successfully captured and facilitated in working code, the project is taken to the second phase. Development in the second phase is described as a "creative explosion" where the project is taken public and new developers start to participate and contribute. This can result in quick development of new capabilities and features.

In OSS, like-minded participants come together and form self-sustaining organizations, called communities. The community surrounding an OSS product often compromise the same role as a vendor in commercial software development does. The role often includes but is not limited to responsibility for the following aspects: distribution of the software, support and maintenance, development of new features and providing bug fixes. Commonly, the participants of a OSS project are developers which share technical competencies, values and beliefs. OSS developers are also often end-users of the software, thus develop software they are interested in and are in need of themselves. The members of the OSS community holds the role as software creators and users, and most of the functionality origin from the community.

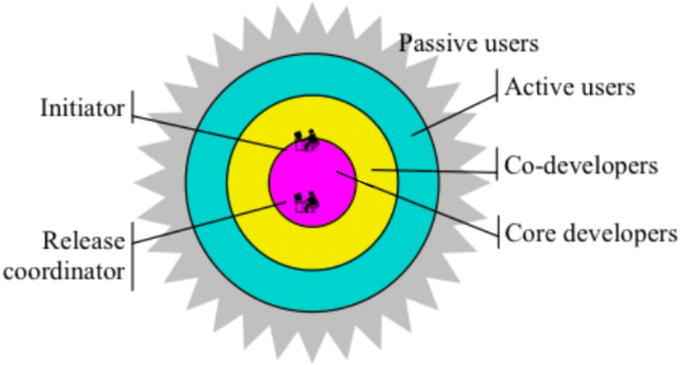


Figure 2.3: Onion-shaped structure of a successful OSS project

The community of a successful and healthy OSS project, which have matured and entered a fairly stable state, is generally onion-shaped [40]. This is illustrated in Figure 2.3. The different layers represent the distinctive roles that developers, leaders and users have in the community. The following bullets give a short description of each role:

Core developers Core developers are located at the center of the onion and are the most experienced and knowledgeable developers. These developers have, through hard work and dedication to the community, earned commit rights to the project's public code repository. This group is generally small, typically consisting of three to ten developers.

Project leaders Along with the core developers, there are one or more leaders at the center of the onion or at the edge of the center. Leaders are often the founders of the project. These serve an important role in managing the community and resolving conflicts that arise. Research findings suggests that the leader is highly influential and respected, and is an important element in the OSS project [41]. Even after the leader has withdrawn and passed on the leader role to one or more of the core developers, he still has a prominent role in the project.

Co-developers The layer surrounding the onion core is the co-developers. These developers are contributors to the project development, however, they have not yet earned commit rights. Their contributions, typically patches and bug fixes, are reviewed by the core developers before they are either accepted or rejected. The co-developers are

Active users The active users are located in the layer surrounding the co-developers. Active users are individuals that use the OSS, and contribute to the development by reporting bugs, testing new releases, writing documentation and participating in mailing list discussions. The active users serve a very important role in protecting the core and co-developers from passive users that need help with setup, build and configuration. Essentially acting as insulation for the co- and core developers. Active users are also a large source to new features and capabilities.

Passive users In the periphery of the onion, the passive users are found. This are users that use the OSS, but do not actively participate in the OSS community.

An important aspect of OSSD is that developers primarily participate and contribute by choice, rather than by assignment. Organization of people and work in OSS development is often done for fun, not for efficiency and productivity. There are no fixed pool of resources and no deadlines to meet as there are in traditional software development. Developers contribute their effort free of charge and have no formalized contract that enforce them to work. Thus, developers come and go as they wish, participate and contribute to parts they are interested in, and do not tolerate imposed work assignment.

Requirements and designs in OSSD are not explicitly stated and not formal. Rather, they are embedded within the software informalisms [37]. Reading, reviewing and reinterpreting informalisms is thus a prerequisite to developing OSS. Traditional software requirement processes includes elicitation, analysis, specification, modeling, and validation. Whereas the OSS requirement processes include post-hoc assertion, sense-making, continually revision, progressive improvements, and transparency. Comparing the two shows that developing software requirements in traditional SE is a technical development process, whereas in OSSD it is a community building process. The community landscape may change rapidly because of these aspects. How communities actually manage to produce successful and complex software, given the constraints, have been subject of many studies [42, 43]. Building social relationships and leveraging the social capital have proven to be key points, along with personal motivation for the project to be successful. Well-functioning collaboration and coordination are thus prerequisites for successful OSS development.

2.2.3 Collaboration

Collaboration has become a vital part of software development and the information technology industry. People are becoming higher educated and more specialized in their field, and the software development process is becoming more and more complex, requiring large teams of developers with key knowledge working together for successful and efficient development. Accordingly, development of OSS is a highly collaborative activity where participation and contributions from the community developers, and interactions among them, are fundamental for the sustainability of a OSS project.

Collaboration, communication and coordination in OSS projects are generally done through *software informalisms* [37]. Software informalisms are information artifacts that allow participants to describe, study and question the events that take place in a software development project, which is well-suited given the characteristics of OSSD. The informalisms capture detailed information as rationales and debates, giving answers to why changes were made and which developers that participated. Software informalisms adhere to the open spirit of OSS; they are publicly accessible and transparent. Anyone can browse the history of the development, and study the current development and work assignments. Crowston and Howison suggest that software informalisms can be important sources when assessing the health of an OSS project, as they promote the activity in the community and give an idea of how developers treat each other and newcomers [40].

Mailing lists, bulletin boards and threaded message discussion forums are the most common informalisms used in OSS projects. They facilitate *asynchronous communication* and allow developers and users of the OSS to observe and participate in discussions in the community. Additionally, the communication is usually publicly archived which allows easy browsing of prior discussions and events that took place several years back in time. Many OSS projects also use bug tracking systems and public code repositories. Bug tracking systems can give detailed information about bugs, including information about the status of bugs (open/closed/resolved), who reported them, comments from developers, and an overview of related bugs. Browsing an OSS project's bug tracker may also give indications of how well the development is going. If there are many open bugs, untouched for a long time or not fixed by developers, the development is probably not healthy and the software may not be reliable. Code repositories can provide detailed information about commit history, giving insight in what was committed, when and by whom. Other informalisms are also used, but to a lesser degree. An example of this is instant messaging or Internet Relay Chat (IRC) and social networking sites [44].

Dabbish et al. report the great importance of the transparency created by the open and accessible software informalisms [45]. The learning potential from examining prior work and observing the current development in OSS projects are sources that can radically improve the complex and knowledge intensive activities incorporated in the software development. Transferring and exchanging knowledge is crucial to the success of OSS projects where developers come together and collaborate to create a complex product which they probably could not achieve individually [46]. There are often large distances between developers in OSS projects, which has been reported as a major challenge in traditional software development. One study reveals that 91 % of the interviewed developers express that they have had problems when collaborating with colleagues across geographical locations [47]. Other studies report that tasks take longer time to complete and that there are higher failure rates when collaborating dispersed [48, 49]. In addition, OSS has challenges related to the unreliable nature of developer's motivation and participation. Despite of these evident challenges, many OSS projects have developed large, complex and successful systems. Furthermore, the transparency along with developers participating or spanning multiple OSS projects, has proven to be important in a cross-project perspective as well; best practices and procedures are picked up and implemented across individual project boundaries [37].

Gutwin et al. state that OSSD is almost always collaborative and distributed, and have investigated how developers in OSS projects overcome the mentioned challenges [50]. By maintaining group awareness, distributed developers reduce the amount of effort that is wasted in duplicate development and coordination of work. Group awareness is related to knowledge about which developers that participate in the project, in what area of the code they are working, what they are currently doing, and what their plans for the development are. Findings indicate that developers maintain a general awareness of the whole OSS team and more detailed awareness of the developers they plan to collaborate with [50]. Three mechanisms establish and maintain group awareness in situations where developers collaborate dispersed:

Explicit communication Developers tell other developers explicitly about their activities.

Consequential communication Developers watch other developers work, which provides them with information about activities and plans.

Feedthrough Developers observe changes to the project's artifacts, which indicates which developers who has been doing what.

Awareness created by the explicit communication is the most flexible, but also the most effort intensive awareness mechanism. Although there are several sources of information in OSS projects, the group awareness is primarily maintained through text-based communication using, as mentioned above, software informalisms such as mailing lists or forums [50]. Furthermore, there is additional value in using the software informalisms as the communication is broadcasted to a large audience, both in finding the right developer for a given task and allowing the developers to decide for themselves whether to respond or not [50].

Howison points out the unreliable nature of the collaborators in the context of OSSD. There are difficulties related to planing and monitoring the development as volunteering developers can not be penalized for not completing their tasks or participating continuously in the OSSD. A result of this, is that dependencies between activities in the OSS projects become hard to manage. Yamauchi et al. report that teams of volunteering developers usually solve this by collaborating in a post-hoc coordination approach [51]. In post-hoc coordination a developer's first communication about a coding activity is typically the announcement that the coding is completed. In traditional SE, coding activities are normally announced and discussed in advance before the actual coding is initiated. The post-hoc coordination suggests that developers in OSS try to eliminate dependencies instead of managing them. Consequently, developers in OSS may be developing the software more individually and more modular.

2.3 Commercialization of OSS

The idea of free and open source software originated in the academic community, where scientists and researchers freely shared their code and experience to build on each other's innovations. Essentially, OSS was born as a movement primarily based on contributions from volunteering developers sharing their time and effort free of charge [39]. During the recent years this picture has changed significantly as there has been a tremendous growth of commercial firms that adopt and participate in OSS. The emergence and potential impact of commercial firms in OSS has been reported in many studies. According to Fitzgerald, OSS has transformed into a more mainstream and commercially viable form, which he has coined as OSS 2.0 [1]. Furthermore, a study by Ågerfalk and Fitzgerald reveals an on-going shift from OSS projects driven by communities of individual developers towards OSS projects driven by communities of commercial firms, particularly small and medium-sized firms [52].

2.3.1 Adoption and Business Models

How software-intensive firms adopt OSS has been examined by Hauge et al., which identified the following six ways [5]:

1. Deploying OSS products
2. Using OSS CASE tools
3. Integrating OSS components into their systems
4. Participating in development of OSS products controlled by someone else
5. Providing their own OSS products and relating to their surrounding communities
6. Using OSS development practices in their own software development

Clearly, practitioners have several possibilities when it comes to OSS adoption. Choosing one over the other is highly depended on what purpose the OSS is to fulfill. As with all business decisions there are advantages, risk and challenges involved. The possible advantages and benefits from OSS adoption in commercial firms are many, and are often the reason for the adoption. Some of the most frequently mentioned reasons are: Avoid vendor lock-in, better flexibility, higher degree of innovation, reduced cost, better quality and faster development [53]. Studies have reported many cases where the advantages gained from OSS have been apparent and significant. One example is from an field study of European firms, where the interviewed managers saw the business benefit from OSS as extremely important, especially with regard to the avoided vendor lock-in, and increased collaboration and innovation [54].

Although there are several benefits, it is important to consider the risk and challenges associated with OSS. Licensing has already been mentioned as one pitfall. Lindman et al. state that "*companies should pay special attention when choosing a license in this development scenario: the wrong choice could result in lost revenue or even loss of control over development.*" [32]. This is applicable both when a firm decides to release its own software as OSS, and when a firm decides to adopt an already established OSS product. Another major set of risk factors is associated with the OSS development characteristics. Developers come and go as they wish, and if they lose interest in an OSS project they may stop contributing to it. This implies that developer communities can cease to exist, and further development of an OSS product might be terminated. There is also the possibility that an OSS project might evolve in an undesired direction. In addition, there are several challenges associated with the lack of support, responsibility and predictability. For instance, there is no responsible provider to turn to in case of security flaws, bugs or unfulfilled requirements. Firms aiming to be heavily involved in OSS projects must carefully consider the mentioned risks and challenges to successfully benefit from OSS and secure their longterm investment.

There are many business models on how firms could profit from OSS. Four overreaching models are: Value-added service enabling, market-creating, leveraging community development and leveraging the open source brand [1]. Whereas the first two have existed longest. In the value-added service enabling model firms use OSS as a platform and profit from providing additional services (support) and selling complementary proprietary software. In the market-creating model firms distribute an OSS product for free with the intention to create a large market, and then offer a proprietary solution with better and more functionality for a fee. Leveraging community development as a business model is based on the idea that talented programmers are willing to contribute their time and effort for free in community development, and therefore be used by firms to increase their development productivity. The last model, leveraging the open source brand, is creating associations that the firm is open and conform to OSS values, which are highly valued by some organizations (for instance government organizations).

2.3.2 Opensourcing

In the list of OSS adoption ways presented above, two of the ways are of particular interest:

4. Participating in development of OSS products controlled by someone else
5. Providing their own OSS products and relating to their surrounding communities

The ways are closely related to the idea of utilizing the unknown workforce that is resident in OSS projects, termed opensourcing [52]. Commercial firms can: (1) Create their own project from releasing proprietary code as OSS (create a vendor-based OSS project), which is equivalent to bullet 5 above, or (2) participate and contribute to an existing OSS project, which may be vendor-based or community-based OSS project, equivalent with bullet 4 above. This way, a commercial firm can collaborate with an OSS community to develop software that is of commercial interest for the firm. Lakhani et al. report that approximately 40 % of developers are paid by their firm to contribute to OSS development [6]. Bonaccorsi et al. found similar results in a survey of 300 OSS projects hosted on SourceForge[2], where almost one third of the projects had one or more firms involved in the development [55]. This have changed the way firm develop, acquire, use and commercialize software [5]. Why firms contribute to OSS projects has also been a topic in research, and researchers have concluded that firms have clearly different motivations than individual volunteering developers [56]. Furthermore, researchers report that the motivation to participate among firms are vastly different. The most common motivations mentioned by firms are [56, 57]:

- Selling complimentary services
- Building greater innovative capability
- Cost reduction through opensourcing to an external community
- Increase value of the core product to existing users
- Increase attractiveness for new users

By participating in OSS projects, commercial firms take the software development beyond the boundaries of the organization [57]. The firm-paid developers that engage in collaboration with other developers in the OSS communities have to learn the community work style and adjust to the rhythms and the demands of the OSS development [58]. Daniel et al. have studied the potential impact of the culture clash the firm-paid developers experience [59]. Results of the study suggest that the merging of open and traditional software development can cause stress for the developer as there are conflicting norms. Firm-paid developers come in a situation where they must balance the firms' intellectual property and competitiveness concerns, meanwhile supporting the community development by contributing code and taking part in the development, also in

[2] SourceForge.net

parts which might not be directly beneficial to the firm. The values of the firm and the OSS project can be conflicting. It is important for commercial firms to understand how to engage in the community development, while maintaining the organizational commitment from the firm-paid developers. The OSS values impact the way the developers perceive their employer, hence, firms should be strategic in selecting the developers to participate and contribute to the OSS development on their behalf [59].

Henkel has found that the link between commercial firms engaging in OSSD and the OSS community is established by individual developers [60]. There is a risk that the participation might expose and result in a loss of intellectual property. The increasing participation from firms implies that competitors in the market may be participating within the same OSS project, and thus a greater risk that competitors will benefit from the contributed code. This is a key consideration to make when deciding to adopt OSS. Research indicate that the participating firm's management is overly concerned about the openness, while researchers conclude that a more positive stance towards openness will enable firms to better utilize the benefits of open innovation processes [60].

Some research has examined how commercial firms are able to transform the competition among them into collaboration for shared goals [61]. Of course, the firms commonly have several domains and markets where they are competitors, and collaboration within one OSS project does not transform and eliminate all competition. Rather, the firms may collaborate and become allies in one aspect, and compete in others. Collaboration across organizational boundaries can be hard as the interests, goals and practices of the developers differ. O'Mahony and Bechky have proposed a triadic role structure in OSSD where commercial firms create boundary organizations for establishing and fostering collaboration with community-managed projects, and in turn other firms that can be allies in the OSSD [61]. The boundary organizations consists of participants from the OSS project and firm-paid developers from the firms. The role structure is displayed in Figure 2.4. In such boundary organizations, the interests of the OSS project and the firms converge and the collaboration practices adapt. The boundary organizations do not solve all conflicts related to collaboration and participation from commercial firms in the OSS projects, but it is reported that the volunteering developers are gradually accepting the firms presence and the friction associated with the differences is disappearing [61]. The OSS projects and its members are adapting to the new reality where commercial firms are a major part of the development, also referred to as *OSS 2.0* [1]

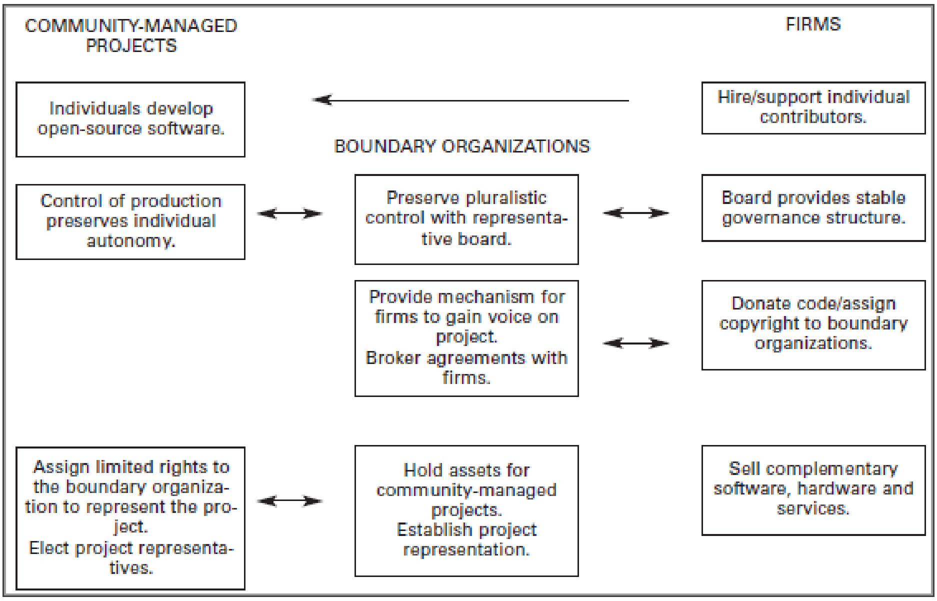

COMMUNITY-MANAGED PROJECTS
FIRMS
Individuals develop open-source software.
Hire/support individual contributors.
BOUNDARY ORGANIZATIONS
Control of production preserves individual autonomy.
Preserve pluralistic control with representative board.
Board provides stable governance structure.
Provide mechanism for firms to gain voice on project.
Broker agreements with firms.
Donate code/assign copyright to boundary organizations.
Assign limited rights to the boundary organization to represent the project.
Elect project representatives.
Hold assets for community-managed projects.
Establish project representation.
Sell complementary software, hardware and services.

Figure 2.4: Triadic role structure illustrating how commercial firms collaborate with OSS projects through boundary organizations. Adopted from [61].

CHAPTER 3

PRE-STUDY AND TOOL SELECTION

This chapter presents a pre-study of the research strategies, data generation methods and analysis used in this thesis. First, a general introduction to the research process and its elements is given. This includes outlining the concepts of case studies, documents, interviews, and quantitative and qualitative data analysis. Lastly, a study of social network analysis along with a tool selection is presented.

3.1 Research Process

A methodical and effective research process is paramount in good research. In addition to giving the researcher a systematic approach to follow, it allows other academics to scrutinize the process and findings when they evaluate the work. Oates provides fundamental insights to research in general, and research process in particular in [62], which was used heavily in the process of writing this thesis. Figure 3.1 is a model adopted from [62], which gives an overview of the research process and its elements.

In this thesis we used the case study strategy as an overall approach to answer the research questions. Documents and interviews were used as data generation methods, producing both quantitative and qualitative data. By using two data generation methods the object of study can be examined in different ways and provide an in-depth understanding of the phenomenon. In addition, the findings can be corroborated and/or questioned by comparing the generated data from the different methods. This is called method triangulation, which is a well-known technique for increasing the validity and reliability of a study. Figure 3.2 illustrate the concept of method triangulation. In the following subsections, a general introduction to the strategies, data generation methods and data analysis used in this thesis will be given.

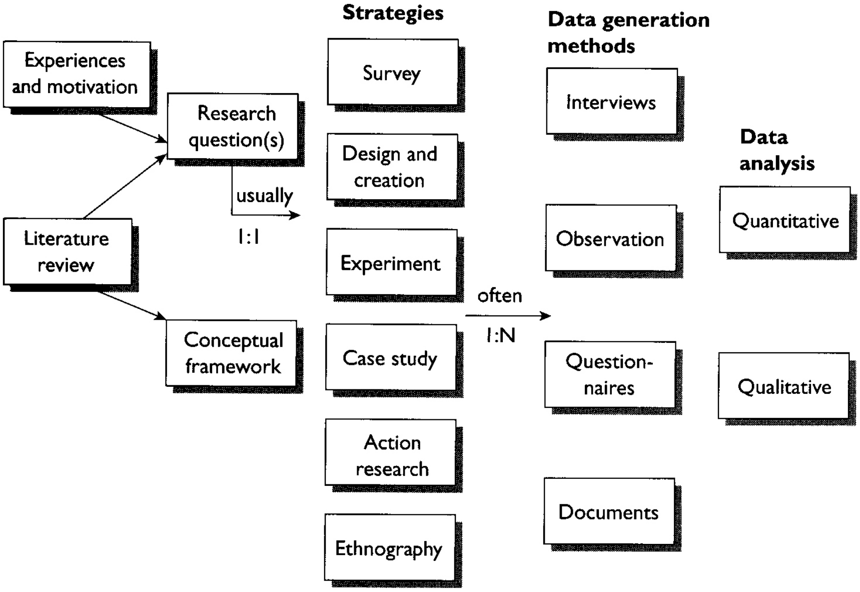


Figure 3.1: Model of the research process (From page 33 in [62]).

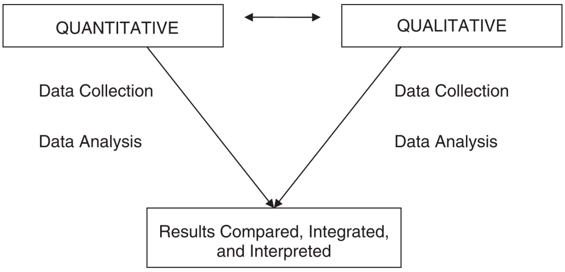


Figure 3.2: Concept of method triangulation using quantitative and qualitative data and analysis.

3.1.1 Case Study

A case study is an intensive analysis of a selected object in its natural setting. This research method may be used to investigate an organization, a community, a computer system and so on. It is an empirical inquiry, using real-life data typically gathered from interviews, documents, observations or questionnaires. Case studies are well suited to answer "why" and "how" questions [63]. The object can be studied in depth by applying a variety of data generation methods and using several sources of data. Both quantitative and qualitative data can be used. The goal of a case study is to acquire a detailed insight in the complex processes of the object.

Yin suggests that there are three basic types of case studies: Exploratory, descriptive and explanatory [19]. An exploratory case study allows the researcher to understand the research problem and subsequently create research questions, which can be addressed by a descriptive case study. Exploratory research often involves a problem that has not been clearly defined and the purpose of the research is to gain knowledge and explore the topic. A pilot study is an example of an exploratory case study. A descriptive case study can be used to obtain a rich and detailed analysis of a phenomenon and its context, providing answers to what events that occurred and how these were perceived. Descriptive case studies are well-suited for exploring fields where a researcher knows which properties to examine and knowledge about the topic is available in prior research literature. The explanatory is an extension of the descriptive study, going further and deeper in explain the events.

3.1.2 Documents

Documents are one of the main data generation methods in research, especially in information systems and computing research where huge amounts of data is resident in web pages, text documents and multimedia documents on the Internet. Information found in documents can provide insights into discussions, decisions, rationales, usage patterns, reports and so on. Documents can be divided in two main categories: Found documents and researcher-generated documents.

Found documents are documents that already existed prior to the research. Examples of this are budgets, manuals, emails, memos and work schedules. Such documents are typically created and stored in archives.

Researcher-generated documents are documents generated by the researcher for the purposes of the research task at hand. For instance, a researcher conducting interviews may generate an interview guide and interview transcripts, which will be important data sources in subsequent analysis and writing of a thesis.

A document-based data generation method may be far easier than an interview or observation-based data generation method. Many documents are already available on the web or in the library, and do not require the researcher to spend much time on planning and execution of the data collection. However, there are some important aspects to consider when using documents in research. It is not always possible to gain access to the desired documents. For example, a company's strategic plans or financial documents are often regarded as secret and kept private, and thus not available to a researcher. Another aspect is the authenticity of the documents. A researcher has to consider these aspects prior to a and during a study, and be prepared to use extensive time to examine the validity of documents and be prepared to not be able to collect all desired data.

3.1.3 Interviews

An interview is a particular kind of conversation between two or more people where a set of underlying assumptions applies. Normally, one person acts as the interviewer and asks questions with the purpose of gaining information and eliciting facts from the interviewee(s). Hence, a researcher conducting an interview has a specific agenda, where questions and topics are planned in advance. The interview will thus not be a normal free-flowing conversation, but rather a guided discussion where the researcher explore topics of interest and control the agenda and proceedings. Interviews can be divided into three categories: Structured, semi-structured and unstructured.

Structured interviews use pre-determined questions, which are standardized and identical for every interviewee. Practically, the researcher is asking the interviewees to answer a questionnaire, where the only difference is that the researcher is writing down the responses. The aim of a structured interview is to collect answers which can be reliably aggregated and allow easy comparison across multiple interviews.

Semi-structured interviews also consist of pre-determined questions and topics, however it is more flexible and less formalized than a structured interview. Questions may be changed or added during the interview based on what the interviewee answers and how the conversation is flowing. The questions may also be open-ended, allowing the interviewee to speak more freely on the specified topics. An interview guide is commonly created to ensure that the semi-structured interview covers the specified topics, meanwhile not constraining the interview to a certain format or set of pre-determined questions.

Unstructured interviews are interviews where the researcher has less control and a less prominent role. A topic or set of topics is introduced by the researcher, and then the interviewee speaks freely and is allowed to develop and explore ideas, while the researcher listens unobtrusively and try not to disturb.

The semi-structured and unstructured interview approaches are well suited for exploration and discovery. Interviewees are allowed to speak their mind and can give insight into fields that are worth exploring further. Though, because the conversation in semi-structured and unstructured interviews can influence the questions and topics covered, these are not useful for research situations where generalizations about the whole population is to be made (especially statistical generalizations).

The role and identity of the researcher is important to consider when conducting interviews, especially with regard to the reliability and validity of the research. Prior studies have reported that people respond differently based on how they perceive the role and identity of the person asking the questions. Hence, the generated data may be influenced by aspects such as sex, age, ethnic origin, behavior, profession and status of the researcher. Some of these aspects can be reduced or eliminated by, for instance, acting professional, polite and neutral, however many of the aspects are not possible to change. This is important to keep in mind while preparing, conducting and evaluating an interview based data generation method.

Selecting participants to interview is an important aspect of interviews as data generation method. Using a proper sampling technique to establish a sampling frame is key as it removes bias from the research. There are two main sampling categories: Probabilistic and non-probabilistic. Probabilistic sampling means that the sample is representative for the whole *population*. Subcategories of probabilistic sampling are random, systematic, stratified and cluster. Non-probabilistic sampling means that the sample may or may not be representative for the whole population (it is unknown to the researcher). Non-probabilistic sampling can be used when the researcher believe it is not feasible or not necessary to have a representative sample. There are four subcategories of non-probabilistic sampling (we elaborate on them because they are highly relevant in this thesis):

Purposive A purposive sample is hand-picked by the researcher where instances are chosen because they are believed to produce valuable and interesting data.

Snowball A snowball sample starts with one instance, and when the researcher has collected data from the instance, he asks for guidance to the next potential instance. This way the researcher is introduced to instances which are believed to be valuable by the interviewees. The researcher can thus be introduced to research objects, which might help in the process of locating instances to interview.

Self-selection A self-selection sample consists of instances which have been recruited through an advertisement from the researcher (i.e. through a forum). They have decided self to participate in the interviews. A useful sampling technique when not knowing who to approach.

Convenience A convenience sample consists of instances which are selected because it is convenient for the researcher. The instances may be easy to reach or willing to participate.

3.1.4 Quantitative Data Analysis

A quantitative data analysis aims to find patterns and draw conclusions from data based on numbers. A researcher can collect numerical data or transform collected or observed data into numerical data. Examples of numerical data can be issue resolution time, visitors to a webpage, activity in a mailing list and so on. Quantitative analysis is commonly used in descriptive and inferential statistics, where a researcher is measuring or counting attributes in a study of a sample or population. By organizing and summarizing the data in tables, graphs and statistics, a quantitative data analysis can be used to illustrate similarities, differences and relationships within the data. The opposite of a quantitative analysis is a qualitative analysis, where the analysis typically involves abstracting themes and patterns from the research data rather than using the data numerically.

There are four different types of data used in quantitative data analysis:

Nominal data describes categorical data, for instance gender, type or activity. Thus, nominal data does not have a numeric value, but refers to a numeric value. An example of this would be to assign the value 0 to walking and 1 to running. This allows a researcher to do arithmetical operations on the data set.

Ordinal data is similar to nominal data, except that the numerical values are allocated to a quantitative scale. Ordinal data is commonly used in questionnaires, where participants respond with “Disagree”, “Neutral” or “Agree”, allowing them to rank their response.

Interval data is similar to ordinal data, but with the possibility to measure differences between the points on the quantitative scale, and thus allowing to state the difference between any data points precisely. An illustration of this could be measuring the traffic on a webpage between 06:00-09:00 and 16:00-19:00, and comparing these numbers to find the difference.

Ratio data is similar to interval data; the only difference is that there exists a true zero to the measurement. An example would be the age of a person or length of an object.

3.1.5 Qualitative Data Analysis

Qualitative data analysis involve abstracting verbal, visual and/or aural themes, and patterns from research data. This includes all non-numeric data such as words, images, sounds and so on. Examples of non-numerical data can be a developer’s feelings towards a new system or a set of users’ perception of a website. Interview transcripts, diaries by researchers, and found and researcher-generated documents are among the main data sources used in qualitative data analysis.

Qualitative analysis differ from quantitative analysis in that there is not always a straightforward task to perform the data analysis. There are less formal procedures, and thus the analysis are more dependent on the skill of the researcher. In addition, words can have different meanings for different people, and must be studied in the context of surrounding words. The gathered data might be overwhelming and cannot easily be displayed in figures and tables as with quantitative data. For instance, an observation of a website’s traffic might result in several hundred textual log files. Consequently, describing how data is analyzed and processed towards a set of conclusions is highly important, especially for putting the research in the scrutiny of other researchers.

3.2 Social Network Analysis

Social network analysis is complex and require in-depth knowledge for successful utilization, thus considerable time was used to study the underlying concepts and selecting the appropriate tools. The analysis approach is described, followed by an evaluation and selection of the toolkits we decided to apply.

3.2.1 Description

Social network analysis is a methodical approach to map and measure relationships and flows between people, organizations, computers, and other connected entities. A social network consists of nodes and edges. The nodes are the entities, for instance people, and the edges are the connection between the nodes displaying their relationship, for instance friendship. A social network analysis can give insight into complex aspects of a network, which cannot easily be observed by just looking at the data. An understanding of the network and its participants is created by evaluating the location of the actors in the network and the overall network structure. The location is measured by calculating the centrality of the nodes, which in turn, give insight in the various roles and groupings in the network. Hence, the centrality measure reveals: (1) Who the connectors, the leaders, the bridges and the isolates are in the network; (2) where there are clusters, and who is in them; (3) who is in the core of the network, and who is in the periphery.

The three most popular centrality measures are degree, betweenness and closeness centrality. To be able to describe the measures mathematically, we need to establish the following commonly used notation: A graph G is a representation of a collection of vertices V and edges E, denoted as $G := (V, E)$.

Degree centrality is based on the idea that an important node is involved in a large number of interactions. Thus the degree centrality of a node is a measure of how many direct connections it has. For a given vertex v in graph G, the degree centrality C_D is mathematically defined as:

$$C_D(v) = \deg(v) \tag{3.1}$$

Betweenness centrality builds on the idea that an important node will lie on a high proportion of paths between other nodes in the network. Hence it is a measure of the number of the shortest paths from all nodes to all others that pass through the node. The equation for the betweenness centrality is:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{3.2}$$

Where σ_{st} is the total number of shortest paths from node s to node t, and $\sigma_{st}(v)$ is the number of paths passing through v.

Closeness centraility is based on the idea that an important node is typically "close" to, and can communicate quickly with, the other nodes in the network. Thus the closeness centrality of a node is a measure of the distance from the node to all others in the network. Mathematically it is defined as:

$$C_C(v) = \sum_{t \in V \setminus v} 2^{-d_G(v,t)} \tag{3.3}$$

Calculating the measures described above is both time-consuming and tiresome, and not viable for large graphs. Fortunately there exist social network analysis tools for automating the process. In addition, most of the tools offer the capability to use the measurement calculations to create a visualization of the social network. Visualization is a powerful technique for presenting complex information in a conceivable manner, while preserving the richness of the data. The following subsections outline the solutions we evaluated and which tool we selected for the analysis.

3.2.2 Available Tools

There is a large collection of social network analysis tools and libraries available online, many of which are open source. We searched for a tool that could calculate the three centrality measures mentioned above, and create informative and accurate visualizations of the social network. After an assessment of the various tools, three of the most promising were selected for a more elaborate test of their capabilities and features. The three tools are described next.

libSNA

libSNA (The library for Social Network Analysis) is an open source Python library for social network analysis. The framework offers several features, including fast processing, easy to use API, scalability, flexible data import and export options, and easy integration. The libSNA website[1] state that "*libSNA is the premier open source library for conducting social network analysis research*".

Gephi

Gephi is an open source network analysis and visualization software package. Through its graphical user interface (GUI), it offers a wide variety of features. Gephi supports networks consisting of up to 50 000 nodes and 1 million edges, the most common metrics for social networks, all the import and export file formats, and give the user numerous network layout options. This software has been used in a number of other research projects [64].

[1] www.libsna.org

NodeXL

NodeXL (Network Overview, Discovery and Exploration for Excel) is a free and open source template for Microsoft Excel that create and visualize social networks. The template offer the following features: Flexible import and export of data and graphs, several different layouts, high flexibility and adjustability, built-in analysis, and graph metric calculation and automation.

3.2.3 Selected Tool

libSNA, Gephi and NodeXL all have the main functionality needed for the social network analysis. The major difference between them are how they are operated: libSNA through code/command line, Gephi through the program's GUI and NodeXL through Microsoft Excel.

We decided to use NodeXL for the social network analysis. The following three reasons justified the choice:

- NodeXL is an Excel template. We have used Excel many times before and have already learned the user interface and the capabilities of the software.
- NodeXL is sponsored by Microsoft Research and other foundations, and is updated regularly. This is in strong contrast to libSNA, which have not been updated since 2008.
- The online software documentation is good and there are many threaded forum discussions where support questions can be asked.

Part III

Research

CHAPTER 4 RESEARCH DESIGN

The purpose of this chapter is to describe the research design used in this thesis. We outline the research process, describe the selected cases and present in detail how the research was designed and conducted. In the end, we elaborate on some of the challenges encountered during the research.

4.1 Research Process

This thesis is a continuation and extension of previous work (see [65]). In the prior study we conducted a case study to investigate how commercial firms collaborate and communicate across organizational boundaries in a well-known community-based OSS project, namely Wireshark. The nature of the study was mainly exploratory and the analysis were purely based on data extracted from the project's public software informalisms. In this thesis we continue the study on Wireshark and include descriptive analysis to investigate the topic in-depth by applying other research methods. Furthermore, an additional community-based OSS project is incorporated in the case study, the Samba project. Description of these projects and a rationale for why they were selected are given in the next section.

There is a considerable body of research conducted in the OSS field, and it is steadily increasing. However, the research of commercial firms and their use and involvement in OSS projects is lacking. This is illustrated by a systematic review of OSS research with respect to OSS in commercial software product development by Höst and Oručević-Alagić, which found only 23 articles covering the subject [9]. In addition, the main focus of the conducted research has been business models and motivation for participating. The main goal of the study presented in this thesis is to contribute with knowledge towards how commercial firms collaborate and communicate across organizational boundaries in community-based OSS projects. This is a field we believe can be valuable for practitioners as it will increase their understanding of how to adopt and successfully utilize the OSS. It can also be interesting for researchers to explore the topic as it may have a large impact on the overall view and knowledge base of OSS.

We decided to adopt a two-phased research approach consisting of an exploratory phase and a descriptive phase. Two reasons contributed to this choice: (1) We found little or no research to use as a starting point and (2) establishing an understanding of the phenomenon was necessary before it could be explored in-depth. A schematic overview of the research approach is displayed in Figure 4.1. Following is a short description of the main elements of the research approach:

Pre-study Prior to conducting the research, a pre-study of the OSS literature, research methodologies and the Wireshark study project was performed.

Phase 1 Data from the OSS project's public software informalisms is used in a quantitative analysis to explore and understand how commercial firms participate and collaborate in the projects, providing the public structure of the collaboration (mailing list, bug tracking system and code repository).

Phase 2 Findings from phase 1 are further explored by conducting qualitative interviews with identified firm-paid developers, providing the perceived structure of the collaboration (developer's perspective). Then results from both phases are summarized to answer the research questions.

Results Four contributions are expected as a result of the conducted research: Empirical data, conceptual model, recommendations and further work suggestions.

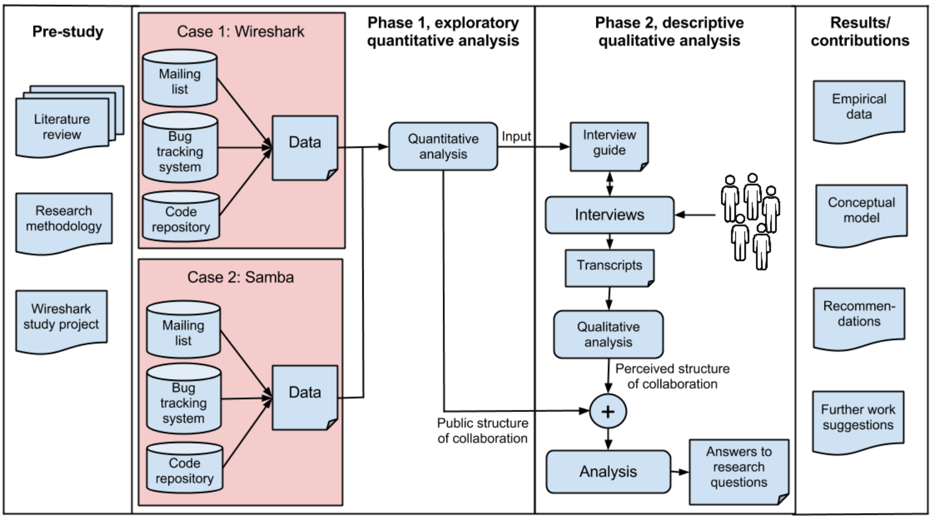


Figure 4.1: Schematic overview of the research approach.

Adopting such a dual-phase approach allowed us to triangulate from multiple data sources, which is an established strategy for improving reliability and validity of a study [19]. Several other studies have successfully used the same approach [52, 66]. Kaplan and Duchon conclude that combining quantitative and qualitative methods in information systems research provide richness and remove potential limitations caused by only using one of the methods [66]. In our case, the use of both allowed us to first study the unexplored field in its entirety and subsequently approach identified commercial firms in the OSS communities to confirm the findings and explore the community cross-collaboration further. It may also be interesting to compare the public structure of the collaboration with the developer's perceived structure of the collaboration.

4.2 Case Study

As mentioned earlier, there exist little or no research literature examining the topic of this study. Seen from a researcher's perspective this is an excellent opportunity to explore and learn about how commercial firms participate in community-based OSS projects, and how they collaborate with each other across the organizational boundaries. However, the lack of knowledge creates a high uncertainty to what the research might reveal. Constructing research questions in advance, without knowing what to search for, is hard. In this context the exploratory case study approach is well suited. It will allow a deep study of the community-based OSS projects in their natural setting, using data from events that have occurred. Collaboration and communication in OSS projects are mainly performed in the projects' software informalisms, i.e. mailing lists and forums. These sources of data are open and accessible for anyone to study, thus facilitating external research: A researcher can build a comprehensive overview of an OSS project from the empirical data stored in the informalisms, without having to interfere with the object of study.

Although the exploratory case study type is a very suitable approach for this study and highly convenient, as all the data is already available through the public software informalisms, it has its shortcomings. By only investigating data found in the OSS project's public archives we cannot fully understand the properties of the collaboration network. One concern is that the firm-paid developers might communicate through other mediums, which are private or hidden, and thus not visible in our results. Then our findings does not reflect an accurate picture of how commercial firms collaborate and communicate with each others in the community-based OSS projects. Another concern is that the software informalisms do not contain detailed enough data for the properties to be explored deeply and to describe the incentives behind the individual firm's participation. We solve this by combining the exploratory and the descriptive case study approach types. As mentioned above, a two phased research approach was adopted. In terms of case study types, phase one corresponds to the exploratory approach and phase two corresponds to the descriptive approach.

4.3 Case Selection

In-depth investigations of collaboration patterns in OSS projects are complex and time-consuming. Yin has stated that the purpose of a case study is not to do a statistical generalization, but rather an analytic generalization [19]. Yin further proclaim that conducting a case study with two or three cases makes it possible to do a literal replication, which enables the establishment of a theoretical framework that can be generalized for new cases. Due to this, we decided to limit the study to include two cases. The case selection procedure was performed by initially selecting an OSS project as the first case from a set of criteria (which was essentially done in the prior study project), and subsequently use the criteria along with attributes from the first case to select the second case.

We started out by searching for one applicable case. With the research objective in mind, we created the following criteria that the first OSS project had to fulfill:

Commercial participation: The selected case should have multiple commercial firms participating in the development. In addition, there must be an adequate way to identify them.

Typical instance: The selected case should be a typical instance of a community-based OSS project. By ensuring this, the research findings can be representative and generalizable to the whole population. This is an important criterion as it will allow the findings to be compared to other OSS projects.

Successful and on-going: The OSS project must be successful and on-going. This implies that the project attracts developers and the development of the software is progressing.

High activity: There must be a high level of activity and temperature in the project. Without activity and interactions among the developers there is nothing to study.

After evaluating different OSS projects with these criteria in mind, the choice ended on the Wireshark project. Many reasons contributed to this choice. First and foremost, the contributor list and community activity reveal high participation and involvement of commercial firms. Wireshark is a typical instance of a community-based OSS project. The project uses software informalisms for development collaboration (mailing list and bug tracking system), the developers are a mix of firm-paid and volunteers, and the software is licensed under the GNU General Public License (GNU GPL). Wireshark is also a very successful on-going OSS project with a high number of contributors and active users consistently pushing the development forward. By examining the mailing list and bug tracking system, we observed a high level of activity between participants. An ideal case for study of communication and collaboration. Lastly, a connection to the project and one of the core developers had already been established in a prior study project. The study project involved creating an add-on for Wireshark with the guidance from the core developer, which in turn was going to be used in the core developer's firm to monitor internal network traffic.

Having selected Wireshark as the first case, we proceeded to identify and select the second case for our study. To be able to do a literal replication, the second case should have similar properties as the first case. This will allow ease of comparison and increase the probability of achieving the same results. Hence, in addition to fulfilling the criteria used in the first case selection, the following attributes given by the Wireshark project should ideally also be fulfilled by the second case, listed in decreasing order of importance:

Time frame: The Wireshark project was established in the 1990s, and in this study we examined the time frame 2006 to 2012. Preferably the second case should have been established in the same period and is required to be active in the same time frame.

Size of data sources: The size of data available in the public sources, mailing list, bug tracking system and code repository should be of the same magnitude. Especially the number of emails and bugs.

Type of OSS project: Wireshark is a community-based project where the software is developed and managed by a community of users. The second case should ideally also be a community-based project.

Bug tracking system: Both projects should be using the Bugzilla software for the bug tracking system and have the same interface. This will allow reuse of the data collection scripts written for Wireshark.

Mailing list software: Both projects should be using the Mailman software for managing their mailing lists and have the same interface. This will allow reuse of the data collection scripts written for Wireshark.

Firm-paid developers: The proportion of participating firm-paid developers and volunteering developers should be equal in both projects.

OSS License: GNU General Public License

Finding a second case that fulfilled all the criteria was not straightforward. After a long period of searching, we ended up with 3 promising cases that matched the specifications: Horde[1], Samba[2] and Wine[3]. A closer evaluation of these projects allowed us to determine their level of similarity to Wireshark. This is visualized in Table 4.1.

[1] www.horde.org
[2] www.samba.org
[3] www.winehq.org

Table 4.1: Similarity of potential projects compared to Wireshark.

Project:	**Wireshark**	**Samba**	**Wine**	**Horde**
Commercial participation	✓	✓	✓	✓
Typical instance	✓	✓	✓	✓
Successful and on-going	✓	✓	✓	✓
High activity	✓	✓	✓	✓
Time frame, 2006-2012	✓	✓	✓	✓
Size of data sources	✓	✓	✓	
Type: Community-based	✓	✓	✓	✓
Bug system and interface	✓	✓	✓	
Mailing list system and interface	✓	✓	✓	
License: GNU GPL	✓	✓		

From the comparison it is evident that Samba is very similar to Wireshark, and thus we chose it as the second case in our case study. As well as fulfilling all the criteria, there were three additional causes for selecting the Samba project which are worth mentioning:

- Both Wireshark and Samba have an annual conference where developers come together to discuss further development and socialize.
- Many of the same firms are involved and/or contribute to both of the projects.
- Developers in both projects develop software that they are users of themselves.

The cases are very similar as demonstrated above, but it is important to acknowledge that the selection criteria are mainly superficial. How similar the cases really are can only be evaluated after the analysis are performed and the findings are presented. Therefore, a case comparison is included in the discussion chapter to accommodate this. Descriptions of the selected cases are given in the following subsections.

4.3.1 Wireshark

A description of Wireshark as a product is probably best captured by the following quote, obtained from the Wireshark website[4]:

> "*Wireshark is the world's foremost network protocol analyzer. It lets you capture and interactively browse the traffic running on a computer network. It is the de facto (and often de jure) standard across many industries and educational institutions.*"

Wireshark is a free and open source tool developed by a community of networking experts around the world. It is licensed under the GNU General Public License. The tool offers a rich set of features, including: Deep inspection of more than thousand protocols, live monitoring and capturing of data, and offline data analysis. Wireshark is operated through a simple and highly intuitive graphical user interface, and is a cross-platform tool supporting Windows, Linux, Mac OS X, BSD and Solaris.

The development of Wireshark started in late 1990s by Gerald Combs, a computer science graduate at the University of Missouri-Kansas City. Combs was working for a small Internet service provider at the time and was in need of a protocol analyzer tool [67]. Commercial protocol analyzers were both expensive and not compatible with the primary platforms that Combs were working on. Thus, he decided to write his own tool. Initially it was named Ethereal, but due to trademark issues it was changed to Wireshark in May 2006. Since its inception Wireshark has attracted an increasing number of users and developers and has become one of the prime examples of a successful OSS project. Wireshark can be regarded as a *category killer* within its field of application. The availability of other networking monitoring software, both open source and proprietary, has declined or vanished as Wireshark has become the leading software solution. Today, there are more than 800 individual developers listed in the contribution list, and this number is steadily rising. The core contributor group consists of 50 developers.

Gerald Combs started working for CACE Technologies in May 2006, and CACE became the primary sponsor of Wireshark. In 2010 CACE was acquired by Riverbed Technology, which took over as the primary sponsor for the OSS project. Combs is still working for the firm, and he continues to maintain the overall code base of Wireshark and participate actively in the community.

[4] www.wireshark.org, Retrieved:2013-04-23

4.3.2 Samba

Samba is an open source/free software suite that provides file, print and authentication services to all clients using the SMB/CIFS protocol. Samba is licensed under the GNU General Public License, and the Samba project is a member of the Software Freedom Conservancy[5]. The suite is a multi-platform program, running on Linux, Solaris, AIX, BSD, Windows and Mac OS.

The development of Samba was started in late 1991 by Andrew Tridgell. At the time, Tridgell was a PhD student at the Australian National University. As with many other OSS projects the development of the software was started by scratching the developer's personal itch. Tridgell was working with a Unix server and a DOS computer, and was required to run multiple protocols to manage the networking of the computers. This was both laborious and technically challenging, and thus Tridgell decided to write a re-implementation of the SMB/CIFS networking protocol. He published the code in 1992. During the first releases the software suite did not have a specific name and Tridgell just called it "*a Unix file server for Dos Pathworks*". Then, midway through the 1.5-series, the suite was named smbserver. However, the name was already in use by a commercial firm and Tridgell received a trademark notice, demanding him to change the name. Tridgell decided to find a new name by running a dictionary search for words containing the letters S, M and B, which resulted in the name that is still in use today, Samba [68].

The Samba project grew, and by mid-1996 it became too intensive for one person to manage. The OSS project experienced massive code contribution and high activity in the mailing list. Tridgell decided to form a core team, officially named the Samba Team, where core developers were invited to join and received commit access. Today, this team consists of about 40 persons, where 10-15 of them are active in the development. The development of Samba is still going strong and the adoption of the software is widespread.

[5] www.sfconservancy.org/members/current/, Retrieved: 2013-04-19

4.4 Phase 1 – Explorative Analysis

This section describes the first phase of the study. Given the lack of research covering the topic, the first phase was mainly concerned with initially achieving an increased understanding of the phenomenon and providing interesting insights that could be further investigated in phase two. Four main parts make up the research design in this phase: (1) Data extraction procedure, (2) identification of firm-paid developers, (3) filtering of the extracted data and (4) detailed description of the collaboration pattern analysis.

In the analysis of the collaboration patterns in Wireshark and Samba, we wanted to explore the interaction between firms and how the firms were structured in the collaboration network. We decided to use the entries in the mailing lists and the bug tracking systems as a foundation for the analysis. The code repository is also used, but only to give a picture of how much the firms contribute to the projects (there is no collaboration in the repository). These public sources contain massive amounts of data, are easily accessible and give a comprehensive overview of the events in the communities. This overview is not complete, however, as it does not include the perspective of the individual developer, but this is covered by the second phase. Quantitative data analysis allowed us to find patterns and draw conclusions from the data by using visual aids and tables of summarized data.

4.4.1 Data Extraction

We decided to extract data from three data sources, which are commonly used heavily in OSS projects: Developer mailing list (referred to as mailing list), bug tracking system and code repository. These sources contain detailed information about events and activities that have occurred in the communities several years back in time. Table 4.2 gives an overview of when the sources were first used and how many entries they have today[6] in Wireshark and Samba.

Table 4.2: Data sources, date of first entry and total number of entries.

Project	Data source	Date of first entry	Total # of entries
Wireshark	Mailing list	2006-05-31	27230
	Bug tracking system	2005-04-08	7862
	Code repository	1998-09-16	42794
Samba	Mailing list	1997-01-03	90588
	Bug tracking system	2003-04-24	9659
	Code repository	1996-05-04	84699

It is worth noticing that both projects have made changes to their infrastructure during their lifetime, for instance switching the source code version control system from CVS to Subversion and from Subversion to Git. Therefore, the data sources

[6] Numbers collected: Wireshark 2012-11-30, Samba 2013-03-12

found online and used in this study might not date back to the initial start of the projects. It is also evident that the date of first recorded entry in the various data sources are quite different. In the case of Wireshark, the entries in the code repository dates as far back as 1998, while the mailing list used by the developers in the community only dates back to 2006. Before 2006, Wireshark was named Ethereal, and used the Ethereal developer mailing list and bug tracking system (code repository has remained the same). It is possible to obtain entries from these data sources from online archives, but they are of slightly different format and usage pattern that the mailing list and bug tracking system used in Wireshark today. Because of this, we decided to focus on the mailing list and bug tracking system currently in use in Wireshark and the Samba.

In both cases, building an overall picture of the activities in the community from the different data sources, and in turn extracting the collaboration pattern between the commercial firms, required that data was collected from a time frame where all the three sources were in use. In addition, to be able to do a good comparison and ensure the validity of the study, both projects should be analyzed within the same time frame.

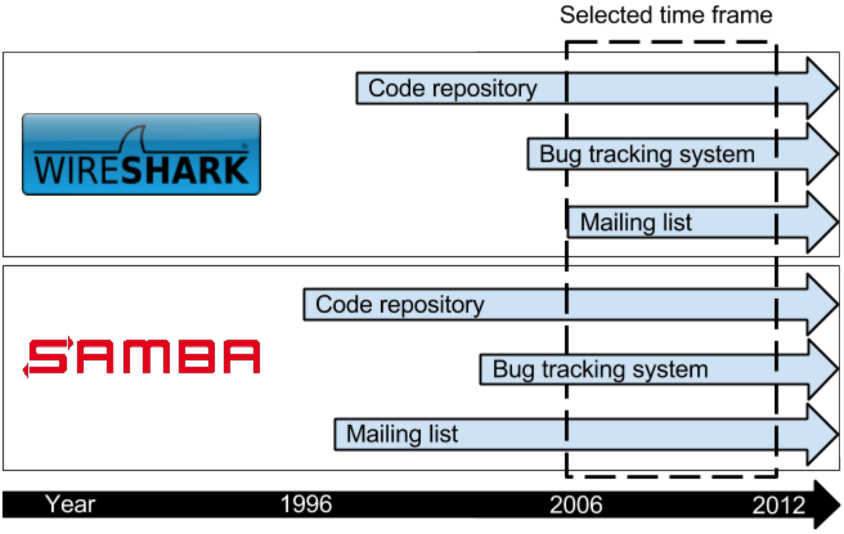


Figure 4.2: The OSS project's data sources, their existence and selected time frame highlighted.

As mentioned earlier, Wireshark was investigated in our prior study and this thesis is a continuation and extension of that work. Consequently, the data sources in Wireshark were first examined and thereby established the time frame for the consecutive study of Samba. The mailing list, having the most recent first entry date, delimit the time frame to begin from 2006. The prior study was conducted during the fall of 2012, thus entries for 2012 were not complete. We therefore decided to include 2011 as the final year of the time frame. This time frame was used as a selection criteria for the second case, which the Samba project fulfilled. To summarize, data was collected over a 6-year period from January 2006 to January 2012 from the three identified data sources in both Wireshark and Samba. Figure 4.2 shows the data sources, their existence and the selected time frame highlighted in dashed lines.

Data Extraction in Wireshark

Collaboration and communication between developers in the Wireshark community mainly occur in two places; the developer mailing list and the bug tracking system. From the characteristics of OSS development, we know that developers may be geographically dispersed and have different work hours because they live in different time zones. Consequently, collaboration, communication and coordination in OSS projects typically occur in public online archives. These archives allow asynchronous communication between developers, and conform to the open spirit of OSS; anyone can access and study the activities in the software development. The data extracted from these two sources will be the foundation for the subsequent analysis of the public structure of collaboration among firms in the community. The commit repository provide a list of the core developers and their contributions committed to the repository. It does not provide any collaboration information, but it is useful as an additional information source and is used in the qualitative phase.

Based on the information required for the analysis we specified a set of data parameters to extract from the data sources. In Wireshark, the mailing list, bug tracking system and code repository are all entry based systems. Each entry being an item posted by a developer or user. Table 4.3 specifies what parameters to extract from each entry in the three different sources.

Table 4.3: Data to extract from entries in the three data sources.

Mailing list	Bug tracking system	Code repository
- Name	- Reporter (name)	- Name
- Time and date	- Assignee (name)	- Date and time
- Mail header	- Time and date	- Commit message
- Mail content	- Description	
	- Comments	

The Wireshark webpage does not provide any procedure to download the full set of data in a convenient way (at least seen from a researcher's perspective). Data had to be collected from various webpages, and downloaded in HTML format. A set of self-written Python scripts were made to automate the preprocessing of the HTML pages. First stripping of the HTML tags and then extract and save the desired data.

Data Extraction in Samba

The Samba project was chosen as the second case because of its similarity to Wireshark. This include how developers communicate, mainly using the project's mailing list and bug tracking system. The structure and format of the software informalisms are also similar; both projects are using the Mailman and Bugzilla software with matching user interface. Hence, the set of data parameters specified in Table 4.3 was also used for the Samba data extraction.

As with Wireshark, there was no convenient way to download the data from Samba. Various webpages were downloaded and preprocessed in the same manner, however, there were some minor differences which required the Python scripts to be modified or rewritten to some extent.

4.4.2 Identification of Firm-Paid Developers

The overall goal of this thesis was to examine the interactions between commercial firms across organizational boundaries in the OSS communities. Information whether a participant is a firm-paid or volunteering developer is generally not available in OSS projects. Consequently, an identification technique had to be defined. In prior research name and email address has been successfully used to do similar identifications [69, 70]. This is a well-known and accepted technique, thus we applied it in this study. The following information was evaluated in the process of identifying the developers:

Email domain The domain in an email address used by a developer can reveal firm association. We regard it as unlikely that a developer use a job email address to participate in an OSS project if it is not related to the job as a paid developer. This measure is the most distinctive identification entity.

Email signature Some developers have their employment firm name as part of their email signature, which they use when posting to the mailing list or bug tracker.

Personal homepage or blog Searching for a developer's name on the web can give directions to a personal homepage or blog that might reveal firm association.

Social networks Searching for a developer's name on social networks like LinkedIn[7] and other professional pages might reveal firm association.

Presentations and conferences Developers that give public presentations commonly include name and firm in the presentation slides, which are easy to find by a web search.

[7] http://www.linkedin.com/

Developer Identification in Wireshark

In the Wireshark mailing list the entries have name and email address of the sender listed, but as the mails are published in online public archives a well-known technique is applied to prevent spam: The domain is replaced by a string of X's. An example of this is: "From: Gerald Combs <gerald@xxxxxxxxxxxxx>". Entries in the bug tracking system have email address listed in full, but no name. The objective of this phase of the study was to create a complete picture of collaboration between commercial firms occurring in both the mailing list and the bug tracking system, and therefore it was important to create a connection between those two data sources. One of them only displays the name, and the other only displays the email address. The only data source we had available with full name and email address pairs was the contributor list, containing 802 developers at the time of this study. The Wireshark homepage provides no information of what effort that is required to become a contributor, but derived from other OSS projects we know that a contributor is a developer participating in development of the product in some way. We decided to base the analysis in the remainder of this study on the members of the contributor list. The 802 entries were examined and each developer were identified as either a firm-paid or volunteering developer. The collection of firm-paid developers formed the foundation for the subsequent analysis of Wireshark.

Developer Identification in Samba

In Samba, there are no concealment of the email addresses in the mailing list; all entries have name and email address visible. The bug tracking system entries have the name listed. Thus, connections between the developers in the mailing list and the bug tracking system can be made using the name as a key. This also implies that all individuals active in the community in the selected time frame could be evaluated and identified, unlike Wireshark where the identification of developers was limited to only include the members of the contributor list. An implication of this was that the set of developers to identify became substantially bigger than in the Wireshark case. The majority of the participants in the mailing list only posted 1 mail, which makes it a waste of time and effort to identify these participant as their contribution towards the firm interaction and software development is minuscule. Consequently, we decided to exclude developers with less than 10 entries in the mailing list or bug tracking system. This significantly reduced the amount of developers to identify and removed insignificant noise in the data set.

4.4.3 Filtering of Extracted Data

The extracted data contained entries by firm-paid developers, volunteering developers and users of Wireshark and Samba. To be able to answer the research questions we had to create a filtering process, so that all entries not made by the firm-paid developers could be excluded. From the approach described above, we obtained a list of firm-paid developers. This list was used as input in a self-written Python script, which iterated through the collected data and outputted the entries made by firm-paid developers. The entries by firms with details were saved to a CSV file and stored. This way we could examine the contributions by firm-paid developers and firms in the mailing list and bug tracking system, on an individual and collective level.

The code repository entries in Wireshark did not contain name or email address of the developers, instead an username/nickname was used. A page on the Wireshark Wiki[8] provided mapping of most of the usernames to the actual developers. To determine what contributions that were done by firm-paid developers in the code repository, we first mapped the usernames to names, then checked if the names were in the firm-paid developer list.

Entries in the code repository in Samba contained both name and email address, which allowed filtering using the same script as for the other data sources.

4.4.4 Collaboration Pattern Analysis

The mailing list and bug tracking system are the public collaboration channels in the Wireshark project. Developers use these software informalisms to discuss problems, features, work assignment, further development, and so on. By investigating these discussions, it is possible to characterize and analyze the collaboration patterns between the firm-paid developers in the community. We applied social network analysis on the filtered data.

4.4.5 Social Network Analysis

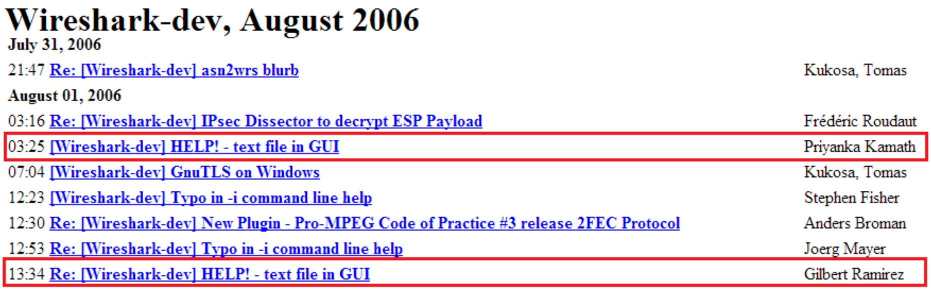
A social network analysis is a well suited approach to investigate the collaboration patterns between firms in the Wireshark and Samba communities. It uncovers which firms that interact with each other and gives a detailed insight in how the firms are structured in the communication network in the community. We used the firms as nodes and the interaction between firms as edges.

Before the analysis of the network could start, we had to define what we meant by "*interaction*". Entries in the mailing list and bug tracking system form several discussions, where each discussion is a set of related entries. Developers taking part in the same discussion are interacting. One problem with this approach was that these discussions were not given explicitly in the extracted data. To solve this problem, we created a set of procedures and Python scripts to build discussion trees of the extracted data corpus. The discussions in the mailing list and the bug tracking system are structured differently, thus different discussion tree procedures were needed. The next subsections describe the procedures used in each data source.

[8] http://wiki.wireshark.org/Developers, Retrieved: 2012-11-28

Mailing List

In this study we collected mailing list entries from archives located at the Wireshark website and at the Samba website. The approach we used for the two projects was so similar that we do not make any distinction here. Technicalities describing how a mailing list system operates and how developers join and post to these lists, are not relevant. Therefore, they will not be covered in this thesis. However basic understanding of how a mailing list is structured in archives is necessary. Each archive is a composition of mails submitted to the mailing list for a given period. In Wireshark and Samba this period is one month. Emails within an archive are sorted by time and date, and are identified by a mail subject and sender name. A screenshot displaying this can be seen in Figure 4.3. When a developer post to the mailing list, he has two choices: (1) Create a new discussion or (2) reply to a on-going discussion. In case (1), the developer compose a descriptive mail subject, which will be used as the identifier. In (2), the mail subject is copied from the initial mail that the developer is replying to and it is appended “Re: ” at the start. This way, developers are quickly able to see which mails that are related and these mails form a discussion. In Figure 4.3 a mail discussion taken fom the Wireshark mailing list is highlighted in red boxes; developer Priyanka Kamath start a new discussion, and Gilbert Ramirez follows up with a reply. Figure 4.4 displays an example discussion in the Samba mailing list. The two mailing lists are formatted differently, the Wireshark mailing list presents mails in a sequential order based on their time-stamp and the Samba mailing list presents the mails structured as discussions.



Wireshark-dev, August 2006

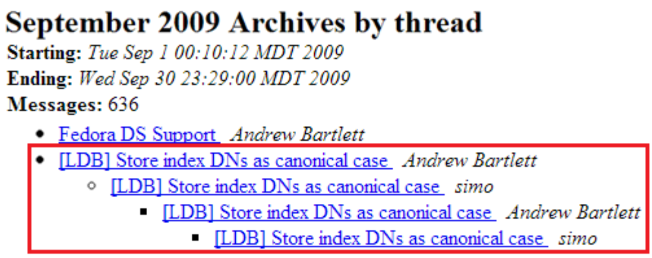
July 31, 2006

21:47 Re: [Wireshark-dev] asn2wrs blurb — Kukosa, Tomas

August 01, 2006

03:16 Re: [Wireshark-dev] IPsec Dissector to decrypt ESP Payload — Frédéric Roudaut
03:25 [Wireshark-dev] HELP! - text file in GUI — Priyanka Kamath
07:04 [Wireshark-dev] GnuTLS on Windows — Kukosa, Tomas
12:23 [Wireshark-dev] Typo in -i command line help — Stephen Fisher
12:30 Re: [Wireshark-dev] New Plugin - Pro-MPEG Code of Practice #3 release 2FEC Protocol — Anders Broman
12:53 Re: [Wireshark-dev] Typo in -i command line help — Joerg Mayer
13:34 Re: [Wireshark-dev] HELP! - text file in GUI — Gilbert Ramirez

Figure 4.3: Screenshot of the Wireshark mailing list.



September 2009 Archives by thread

Starting: *Tue Sep 1 00:10:12 MDT 2009*
Ending: *Wed Sep 30 23:29:00 MDT 2009*
Messages: 636

- Fedora DS Support *Andrew Bartlett*
- [LDB] Store index DNs as canonical case *Andrew Bartlett*
  - [LDB] Store index DNs as canonical case *simo*
    - [LDB] Store index DNs as canonical case *Andrew Bartlett*
      - [LDB] Store index DNs as canonical case *simo*

Figure 4.4: Screenshot of the Samba mailing list.

We use the structure of the mailing lists to extract mail entries that are related, and group them together in "discussion trees". A discussion tree consist of an identifier node, a source node and a set of responder nodes (which can range from none to many). The developer that initiates a discussion is regarded as the source, and the developers that follow-up on a discussion is regarded as responders. The process of building these discussion trees were automated by a Python script, executing the following steps:

1. Create two lists, one for sources and one for responders.
2. Iterate through the input entries, and examine the mail subject field. If the subject starts with "Re: ", the entry is saved to the responders list, else it is saved to the sources list.
3. For each entry in the sources list, create an identifier. Iterate through the responders list and compare the subject field. If the subjects are equal (ignoring the "Re: "), the responder entry is marked as a follow-up of the source entry. When the responders list iteration is complete, the source and its responder entries are saved as a discussion tree.

The process of generating the discussion trees is illustrated in Figure 4.5. To obtain the correct ordering of responders in each discussion, it was important that the input data was ordered by date and time. The script iterates through the entries in sequential order.

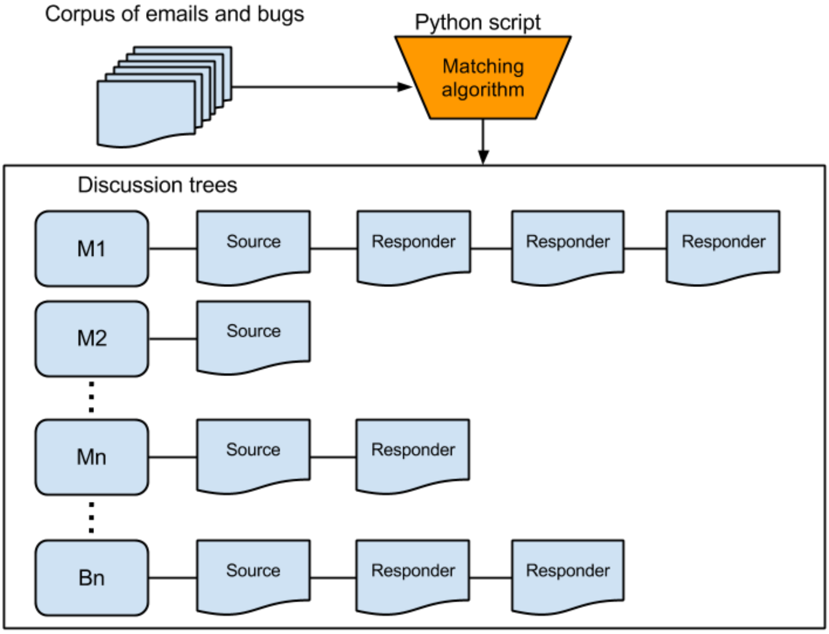
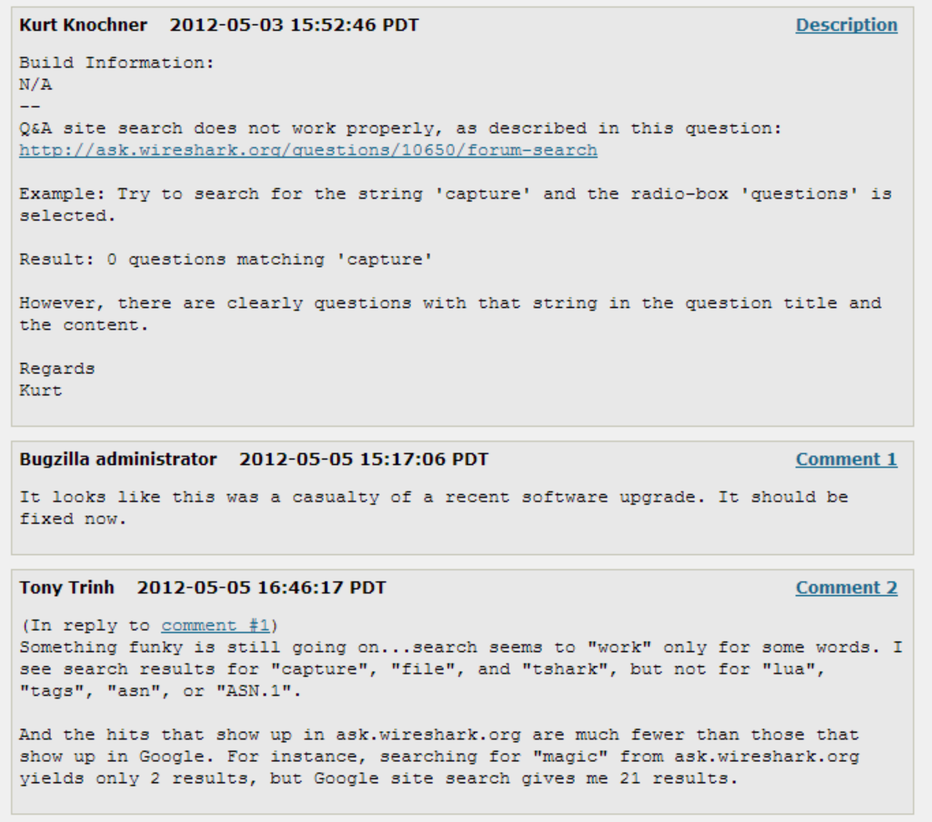


Figure 4.5: The process of creating discussion trees from emails and bug entries.

Bug Tracking System

Interaction between developers in the bug tracking system occurs in each bug entry, rather than across multiple entries as in the mailing list. This simplifies the job of extracting the discussions and generating the discussion trees considerable. A bug entry contain several fields of information, including bug title, time, status, reporter, assignee, importance, description, comments, and many more. Developers collaborate by posting comments in each bug, working together to solve the problem. These comments are the interaction. An example of this can be seen in Figure 4.6. The structure of the bug tracking system was identical in the two OSS projects.



Kurt Knochner 2012-05-03 15:52:46 PDT Description

```
Build Information:
N/A
--
Q&A site search does not work properly, as described in this question:
http://ask.wireshark.org/questions/10650/forum-search

Example: Try to search for the string 'capture' and the radio-box 'questions' is
selected.

Result: 0 questions matching 'capture'

However, there are clearly questions with that string in the question title and
the content.

Regards
Kurt
```

Bugzilla administrator 2012-05-05 15:17:06 PDT Comment 1

```
It looks like this was a casualty of a recent software upgrade. It should be
fixed now.
```

Tony Trinh 2012-05-05 16:46:17 PDT Comment 2

```
(In reply to comment #1)
Something funky is still going on...search seems to "work" only for some words. I
see search results for "capture", "file", and "tshark", but not for "lua",
"tags", "asn", or "ASN.1".

And the hits that show up in ask.wireshark.org are much fewer than those that
show up in Google. For instance, searching for "magic" from ask.wireshark.org
yields only 2 results, but Google site search gives me 21 results.
```

Figure 4.6: Screenshot of a bug and related comments in the Wireshark bug tracking system

Each bug in the bug tracking system were considered a discussion, where the bug reporter was the source node and the comments were responders. A script were made to extract this, and it created discussion trees with the same structure as the discussion trees created from the mailing list, see Figure 4.5.

To correctly build the discussion trees it was important that the unfiltered data was used as input. If not, several source nodes might have been lost in the filtering, resulting in a loss of many interactions. In addition, we wanted to observe the actual response time between entries in the discussions. Only using entries from firm-paid developers would not give the correct response time. The procedure that was used in the preprocessing of the discussion trees is presented in Figure 4.7, and consists of three steps:

1. Fetch a discussion tree and remove the identifier (i.e. M1 or B53).
2. Use the timestamps to calculate the response time between each of the entires. This is calculated in number of hours since the prior entry was registered. For instance, (2012-08-12 12:46) - (2012-08-12 10:44) = 2 hours. The source node response time is set to -1, indicating that it is the first entry in the discussion.
3. Lookup the developers name in the list of firm-paid developers. If the developer is working for a firm, replace the developer name with the firm name. If the developer name is not in the list, he is regarded as a volunteering developer and labeled volunteer. Remove all entries by volunteers (illustrated by a red cross in Figure 4.7), and connect the firm entries and retain the response time.

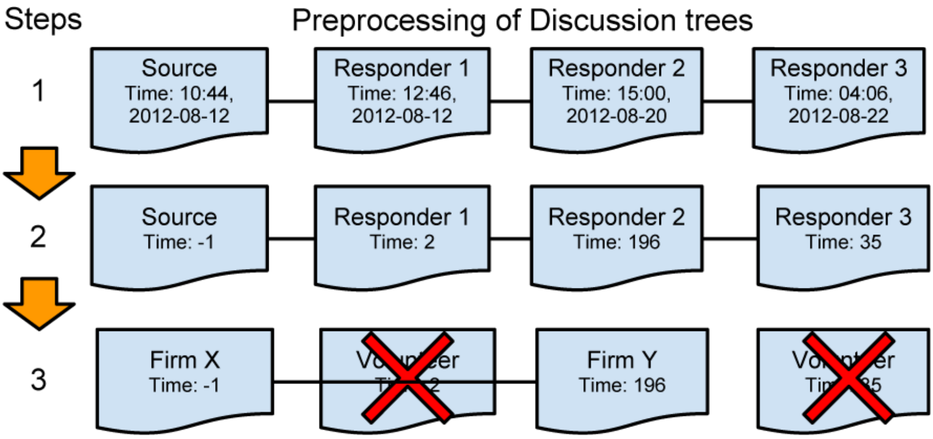


Figure 4.7: Steps in the preprocessing of discussion trees.

An essential consideration that had to be taken, before we could create social networks of the discussions, was how the structure of the discussion trees should be represented in the networks. We decided that developers participating in the same discussion, should all be interconnected (edges between all nodes in a discussion). The reason for this is that it is impossible to know who the developers are actually interacting with. For instance, is the second responder of an email entry interacting with the source or the first responder, or maybe both?

Furthermore, the process of removing volunteer entries breaks the discussion trees as displayed above. Interconnecting the discussion participants solves this problem. The approach we decided to use is depicted in Figure 4.8. The participants of a discussion (discussion tree) is all connected together with nondirectional edges, illustrating the collaboration among the firm-paid developers and their equality. Another choice we had to make was how to represent the firms that only interact with volunteers, and does not collaborate with any of the other firms in the community. As we mainly were interested in examining how firms interact across organizational boundaries, we did not include the discussions where a firm has no interaction with any other firms. Hence, discussion trees with only one entry after the preprocessing, were excluded from the social network analysis. The response time of the interactions with volunteering developers was preserved and included in the response time analysis. This was done so that a complete picture of the response time from a given firm could be evaluated.

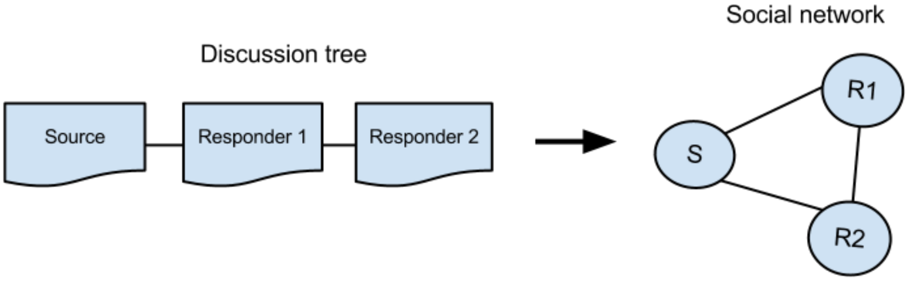


Figure 4.8: From discussion tree to social network.

Before the NodeXL tool could be used to create the social network, we had to filter and convert the discussion trees to the correct format as described in the preprocessing. A set of Python scripts were created to automate the task. The exchange of developer name to their associated firm was essential, so that the interactions could be visualized on firm level, rather than individual developer level. Finally, each discussion tree was converted to NodeXL input format. The list of commercial firms was used as nodes in the creation of the social network, and the converted discussion trees were used as edges. An excerpt of the input used in NodeXL is displayed in Table 4.4 (see next page). The weight in the table is how many times the firms were in the same discussion. It is an interesting number to evaluate, however, we do not use it in the social networks. This is part of making an analytic generalization, not a statistical one, as Yin proclaimed [19].

Table 4.4: Excerpt of the input used in the social network analysis tool to create connections (edges) between firms (nodes).

Node
ascolab
mxtelecom
nsn
Thales
intracom
blue-cable
ngc
wanadoo
Ericsson
Trihedral

Source	Target	Weight
ascolab	mxtelecom	'weight':1
ascolab	Ericsson	'weight':2
ascolab	Trihedral	'weight':3
nsn	Thales	'weight':1
nsn	Ericsson	'weight':13
nsn	intracom	'weight':2
ngc	Ericsson	'weight':1
wanadoo	blue-cable	'weight':1
wanadoo	Thales	'weight':2

4.5 Phase 2 – Descriptive Analysis

This section describes the second phase of the conducted study in this thesis. In the first phase we collected large amounts of data and created social networks to explore the nature of how commercial firms communicate across organizational boundaries in community-based OSS projects. This allowed us to understand and establish a wide overview of the public communication structure. The overview, however, is not complete as it does not include the developer's perspective. The objective of this phase is to further explore and complement the social networks by conducting semi-structured interviews with identified firm-paid developers involved in the OSS projects.

4.5.1 Data Generation Method – Interviews

Interviews, documents and observations are among the most common qualitative data generation methods. Documents were used in the first phase, and mining these for more data would not enable an exploration of the social networks using multiple data sources, and in turn allow us to use method triangulation to confirm the validity of the findings. Observing software developers in global software development setting, which OSS essentially is, is a very time-consuming and impractical task. Especially observing developers in real life as they are located all around the world. Therefore, we decided to carry out semi-structured interviews to collect data directly from a subset of the firm-paid developers identified in the social network analysis.

The semi-structured interview approach is well-suited as it provides flexibility to explore topics observed in the social networks, while not constraining the interviewees to a predetermined set of questions. This way, findings from the social network could be complemented and possibly confirmed with data and analysis from the firm-paid developer's perspective, and at the same time facilitate a deeper exploration of the collaboration patterns and properties.

4.5.2 Participants

Participants for the interviews were selected from a sample frame consisting of the firm-paid developers identified in the first phase. As we did not know much about the population, we aimed for a non-probabilistic sampling technique using a conjunction of purposive and snowball sampling (see subsection 3.1.3).

Two overall selection criteria for developers to interview were that:

1. They participate in the OSS development as part of their paid work
2. They have an adequate degree of participation in the development

Despite the fact that a person work as a software developer and use her work email for participation in an OSS project, does not guarantee that she participate in the development as part of her professional job. The first criterion enabled us to ensure that the participation was work related. The second criterion ensured that the selected interviewees were active in the collaboration network, and hence could contribute with relevant information to this research.

In Wireshark, we used the already established connection to one of the core contributors as a starting point, and asked for suggestion of firm-paid developers that could be interesting to interview next (snowballing). The core contributor pointed out relevant developers for the research topic, and assisted in contacting them by posting our interview invitation on the core contributor mailing list.

In Samba, we did not have any acquaintances to any of the developers. Therefore, we selected relevant firm-paid developers in the OSS project based on the quantitative data and sent interview invitation to these by email. The invitation contained a presentation of the research context and provided guidelines for what topics the interview would cover, thus the firm-paid developers could assess whether to participate and be prepared for the interview. The interview invitation can be seen in Appendix A.

Some of the developers were not eligible for participating due to various reasons, including that their participation was not work related or they felt that they could not contribute with information on the relevant topics. Another aspect of interviews is the poor response rate to invitations, which is further elaborated in section 4.6.

We ended up with interviewing 7 firm-paid developers from the two OSS projects. Table 4.5 gives an overview of the interviewed developers and their background. The developer ID's are used frequently throughout the rest of this thesis.

Table 4.5: Some details of the developers and their associated firm.

ID	**# Employees**	**Business domain**	**OSS project**
D1	>10 000	Computer and network security	Wireshark
D2	>10 000	Telecom	Wireshark
D3	1400	Computer networking	Wireshark
D4	200	Telecom	Wireshark
D5	4500	Open Source Software	Samba
D6	>10 000	Computer Software	Samba
D7	50	Computer Software	Samba

4.5.3 Interview Guide

An interview guide was designed to assist the researcher in the interviews. The guide enabled a systematical data collection process, and it assured that the relevant topics were covered. As our knowledge of how developers perceived the collaboration network in the community-based OSS projects was lacking, we used the first interview to establish an understanding of the field from the developer's perspective and identified interesting subjects to explore further. The results of this first exploratory interview was subsequently used to develop a more complete and specific interview guide, which was used in the following interviews. Minor changes and modifications were done to adapt the interview guide to the interview object, however, the fundamental themes and ideas stayed the same.

The interview guide consisted of four to five main topics, with both closed and open questions. The closed questions were mainly used in the introduction phase of the interview to solicit background information about the respondent, firm and OSS project context. In addition, closed questions were used to confirm or attribute statements given by other developers. The open questions were used to collect information about: (1) Work process/bridge engineer role, (2) firm awareness/organizational boundary and (3) position in the community/contributions.

The interview guide is presented in the following text. There are several questions in the interview guide, but not all the questions were asked to all developers because of time limitations (one hour was allocated for each interview). Furthermore, answers to some of the questions may be available online, and the interviews were primarily used as confirmation. For instance, how the branching system is managed. The questions asked to less than four developers are marked by a "*". Note: Firm X, Project Y and Conference Z are used as a placeholders. The projects were Wireshark and Samba, along with their corresponding conferences Sharkfest and SambaXP.

Background Information

- What is your background and years of experience as a software developer?
- What are your roles and tasks in Firm X?
- Experience with OSS
    - How many years of experience do you have with OSS?
    - Do you participate/contribute to OSS projects as part of your job, personal interest or both?
    - Do you participate in other OSS projects than Project Y? Which?
    - How did you discover Project Y? Did you introduce it to Firm X, or did they introduce it to you?
- What is your motivation to participate in Project Y?
- What is your role/responsibility in Project Y?
- What is the role of Project Y in Firm X?
- What is the role of Firm X in Project Y? In terms of responsibility, contribution and position?

General Project Y Development Questions

- How are feature requests submitted? How is a feature request accepted/rejected?*
- How frequent are new piece of code (patches or bug fix or new implementation) integrated into the main branch?*

- How is the branching system maintained?*
- Are the similar ways of coding and commenting among different participants? Can you provide examples?
- How are source code commits controlled and reviewed? Who can integrate the different branches into the main one?*
- How is the date of a new release/version decided? Who decide?*
- How is the Project Y community organized? (Flat or hierarchical structure)*
- Do you perceive that someone "owns" the project? If a firm has a large number of developers working in the project, do you feel that they dominate or control the development?

Work Process/Bridge Engineer Role

- Does Firm X have other developers working with Project Y? If yes, do you work together and how? Is the experience and the knowledge within the firm organized such that you know who to ask if you need help?
- Would you say that you act as a gatekeeper/bridge between Firm X and Project Y? Do your coworkers report bugs and errors to you, and then you forward them to the community (Bugzilla or mailing list)?
- Are there differences between how Firm X and the Project Y community develop software? (Testing/coding/requirements/maintenance) Explain.
- Do you have an internal branch of Project Y where you do development for Firm X? What do you do when there is a new major release of Project Y, how do you merge the code you have internally with the new release? Does the internal code base create problems (integration/merging problems)?
- Do you find it more difficult to read/understand code and comments from developers in the community, than from your coworkers?
- If you need help or guidance with Project Y, do you rely on the community? What if you can't manage to solve the problem at hand? (Do you contact developers directly, which you know are experienced in the field of the problem?)
- When do you post to the mailing list? Why do you participate in discussions?

Firm Awareness/Organizational Boundary

- Have you meet other developers from Project Y in real life? Have you participated on Conference Z?
- How do you view other developers, firm-paid or volunteers? (If you know their background?)
- Are you aware of other firms in the Project Y community? Are you aware of developers from partners/competitors participating in the development?
- Do you work with other firms to develop Project Y? Does Firm X form alliances with other firms for Project Y development?
- How do you communicate with other developers within the community? (Partners, coworkers, other firm-paid developers?) Do you sometimes use a private channel (IM, private mail, telephone)? Why?
- Do you think of others as potential partners or competitors? Do you consider that others could use the code Firm X is giving back to the community to gain/recapture technological advantage?
- Do you have code that is proprietary (i.e. plug-ins or extensions of Project Y), and is kept private? How do you decide what code that is kept

proprietary/private, and what is given back to the community?
- When code is given back to the community, do you experience that it is further developed and improved by other developers?

Position in the Community/Contributions

- Do you think it is important to have a certain role/position in the OSS community (a position based on your excellent contributions and status)? For receiving help/attention from the community when you have a problem or want to influence the development?
- Does Firm X have a formal strategy for how they want use the OSS community? With regard to other firms? (Utilizing the knowledge of other developers?) Is there a strategy/policy for what code that is given back to the community? Any comments on your answers?

Other Comments

- Do you have any other comments?
- Is there any important aspects that we have forgotten?

4.5.4 Data Extraction Procedure

The interviews were mainly performed in two ways, by phone or email. Only one interview were performed face-to-face in the developer's workplace. A summary is given in Table 4.6. All the interviews were conducted in English, except for one. The duration of the interviews ranged from 45 minutes to 72 minutes. All the live interviews were recorded to facilitate subsequent analysis and minimize potential data loss due to note-taking. The recordings were thereafter transcribed verbatim.

Table 4.6: Interview participants and procedure overview

OSS Project	ID	Interview Procedure
Wireshark	D1	Face-to-face
	D2	Phone
	D3	Phone
	D4	Phone
Samba	D5	Email
	D6	Email
	D7	Email

In the case of interview performed by email, we modified the interview guide slightly to be a bit more self-explanatory and tailored for the interviewee. Questions were written more verbose, and information about the given developer available online was used to specify the background questions. The interview guide was sent to the developer, which answered the questions and sent it back.

In the interview invitation and conversation before an interview, we communicated that we preferred to do the interview by phone as it would enable us to be more dynamic in the interview process by adding/removing questions and responding to answers given by the interviewee. However, some of the developers insisted to do it by email as they lived in a different timezone and could not set aside one hour for a continuous interview. By email, they could complete the interview in stages when they had spare time. We accepted this, and made an agreement that a follow-up interview could be initiated if we had further questions or wanted to clarify answers given in the first interview.

4.5.5 Thematic Data Analysis

The interviews were prepared for analysis by manual transcription of the audio recordings to text documents, and the email responses were refined to transcripts of the same disposition. This resulted in 55 pages of rich text. The analysis of the qualitative data was undertaken following guidelines stated in [62], and using the recommended steps for thematic synthesis in SE proposed by [71]. This thematic analysis approach allowed the main themes in the text to be summarized in a systematic and transparent way. A basic outline of the process is illustrated in Figure 4.9.

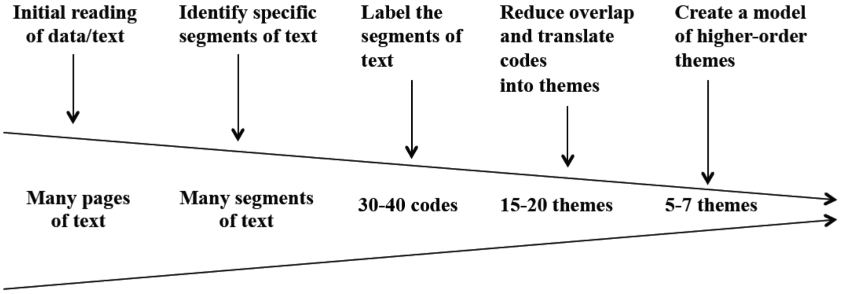

Initial reading of data/text
Identify specific segments of text
Label the segments of text
Reduce overlap and translate codes into themes
Create a model of higher-order themes
Many pages of text
Many segments of text
30-40 codes
15-20 themes
5-7 themes

Figure 4.9: Thematic analysis process (adopted from [71]).

We moved iteratively from left to right between the steps in the thematic analysis process figure. The first step after the interviews was to read all the transcripts to get an overview of the data, and to identify relevant pieces of the text. In the next step, the material was dissected into more compact and meaningful text segments, and was labeled with codes. The codes are descriptive labels that allow the data to be organized, which enable the researcher to group similar data into categories because they fully or partially share some characteristics. Coding is the first step in transforming from concrete statements in the data, to making analytic interpretations. The coding was performed using an integrated strategy, employing both deductive and inductive approaches. Such an integrated strategy was well-suited in our study as it accommodate the establishment of a predefined set of codes from the research objective and research questions (deductive), meanwhile allowing development and refinement of codes as they emerge in the data during the thematic analysis (inductive).

After multiple reviews of the data, we ended up with 52 codes covering the interesting properties from the interview analysis. The following step of the thematic analysis was to translate the codes and the corresponding text segments into themes. A theme in this context is essentially a code in itself, however, a theme is an increased distanciation from the text, and thus an increased level of abstraction. The codes were evaluated and combined to form overreaching themes which compromised the same characteristics and patterns. Lastly, the themes were refined and combined to form a set of higher-order themes.

The higher-order themes, themes and codes were structured hierarchically in a mind map with “organizational cross-collaboration” as core. The mind map was an important tool in systematically visualizing the relevant topics and properties of the cross-collaboration taking place in the OSS projects seen from the participating firm-paid developer’s perspective.

4.6 Challenges

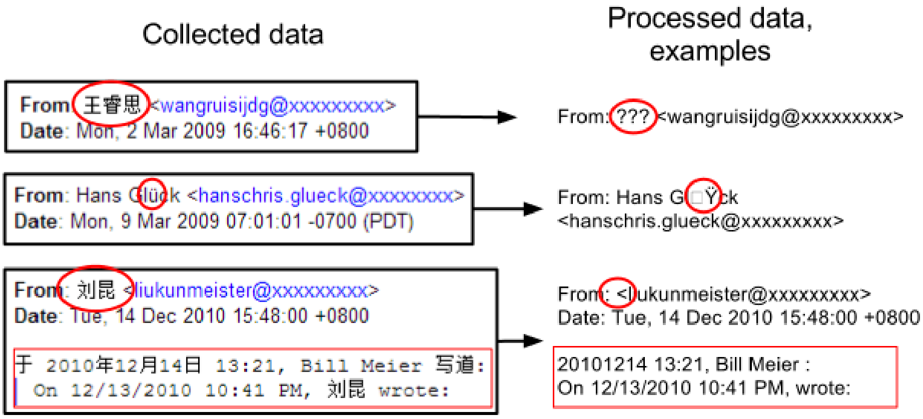
A set of challenges were encountered during the research that may have influenced the results of this thesis. The challenges are outlined, followed by a description of how we resolved them or minimized the possible impact. The measures were important towards ensuring the validity of the study.

4.6.1 Special character encoding

The data collected from the OSS project’s software informalisms contained several special encoded characters. Especially the Russian and Chinese names, but also some of the common European names. This is not a problem in itself, as the characters are properly displayed in the downloaded web-pages. However, the Beautiful Soup[9] and Text2HTML[10] modules used in the Python-scripts to extract and process the data does not support this out of the box. Furthermore, reading and writing to files proved to be problematic without the correct encoding/decoding functionality implemented. Since the names used in the mailing list and bug tracking system were used to match developers with their respective firms, the character encoding was a highly problematic issue. The Python-scripts returned what at first sight seemed as correct results, but after closer investigation the flaw was revealed. An example of how the data may transform in an incorrect encoding handling is displayed in Figure 4.10.

[9] http://www.crummy.com/software/BeautifulSoup/

[10] http://txt2html.sourceforge.net/



From: 王睿思 <wangruisijdg@xxxxxxxxx>
Date: Mon, 2 Mar 2009 16:46:17 +0800

From: Hans Glück <hanschris.glueck@xxxxxxxxx>
Date: Mon, 9 Mar 2009 07:01:01 -0700 (PDT)

From: 刘昆 <liukunmeister@xxxxxxxxx>
Date: Tue, 14 Dec 2010 15:48:00 +0800

于 2010年12月14日 13:21, Bill Meier 写道:
On 12/13/2010 10:41 PM, 刘昆 wrote:

From: ??? <wangruisijdg@xxxxxxxxx>

From: Hans G□Ÿck
<hanschris.glueck@xxxxxxxxx>

From: <lukunmeister@xxxxxxxxx>
Date: Tue, 14 Dec 2010 15:48:00 +0800

20101214 13:21, Bill Meier :
On 12/13/2010 10:41 PM, wrote:

Figure 4.10: Extracted data with special encoded characters and examples of how the processed data may be transformed without correct handling.

The solution to this problem was to implement functions to read the stream of data, decode it and re-encode it to a known selected encoding. Several random-tests and debugging executions were performed to check that the scripts gave the correct results. Although we used considerable time to review the results, the corpus of collected data was so massive that it is not possible to confirm that every special character is handled correctly. This is a potential source of error, and it is important to consider the impact of it when evaluating the validity of this study.

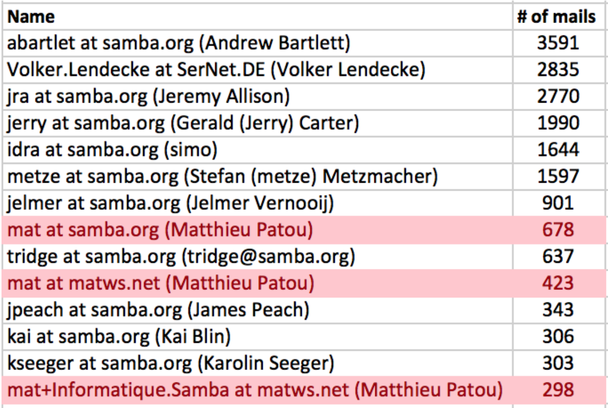
4.6.2 Name, Email Address and Mail Subject Fields

The name, email address and mail subject fields were some of the main data objects used in the quantitative analysis. Each of the objects had challenges attached.

Name and email address

The name and email address were used to provide mapping from developers to firms. A major problem with this approach is that people tend to change email address occasionally, especially in the domain examined in this thesis; the commercial world. Developers change firms, and firms are acquired by other firms. Names also change, not as frequent as email addresses, but it does happen (marriage, personal problems, and so on).

Because of the relative long time frame selected in the quantitative analysis, six years, we observed several instances of name and email address changes in the collected data. To ensure that the data was correctly analyzed additional mappings were manually added. This was a laborious task, and impossible to perform on the whole data corpus. The solution to the problem was to count and



Name	# of mails
abartlet at samba.org (Andrew Bartlett)	3591
Volker.Lendecke at SerNet.DE (Volker Lendecke)	2835
jra at samba.org (Jeremy Allison)	2770
jerry at samba.org (Gerald (Jerry) Carter)	1990
idra at samba.org (simo)	1644
metze at samba.org (Stefan (metze) Metzmacher)	1597
jelmer at samba.org (Jelmer Vernooij)	901
mat at samba.org (Matthieu Patou)	678
tridge at samba.org (tridge@samba.org)	637
mat at matws.net (Matthieu Patou)	423
jpeach at samba.org (James Peach)	343
kai at samba.org (Kai Blin)	306
kseeger at samba.org (Karolin Seeger)	303
mat+Informatique.Samba at matws.net (Matthieu Patou)	298

Figure 4.11: Top 14 participants in the Samba mailing list between 2006-2012, displaying in red 3 different names for the same developer.

sort the data on the number of entries (emails or bugs), see Figure 4.11. This allowed us to create a list of participating firm-paid developers in descending order. The participation seem to conform to Pareto's principle: about 80 % of the entries are done by 20 % of the developers. Consequently, we manually evaluated the top 40 % and added correct mappings (more than 20 % to ensure good matching). This way we minimized the potential error of name and email address changes.

Mail Subject

The mail subject field was used to arrange and connect mails that were related to each other in the Wireshark mailing list. The Samba mailing list provided other means for doing this. The challenge by using the mail subject as identifier is that some subjects are too generic or they are changed during a conversation. This would in turn create incorrect results, as emails that were related are not connected or that emails not related to each other are connected together. Examples of generic subjects are "*Help*", "*Wireshark problem*", and "*<no subject>*". These types of subjects occur from time to time in the mailing list, without being the same conversation. A typical example of a subject change is when the subject "*Problem with protocol X*" is changed to, for instance, "*Problem with protocol X [Patch]*".

First, it is considered inappropriate behavior to not provide an explanatory mail subject when posting to the mailing list. Hence, entries where generic subjects are used can justifiably be excluded without further notice, as they are probably from persons that are not highly involved in the OSS development. Second, changes in the subject are not that common in the Wireshark mailing list. Manual tests were performed on the data corpus to confirm this, which revealed that there was a low number of emails where this happens (less than 1 %). Because of the diminishing influence these mails would have on the end result, no further measures were taken to “fix” this.

As a side note, linking mails together that are not related invalidate the response time analysis. Some response times would be wrongly assigned, and could potentially be assigned to a negative value. This way, we could easily detect incorrect matching.

4.6.3 Interview Participants

Persuading firm-paid developers to participate in the interviews was a tough challenge. Oates proclaims that response rates ranging between 10 % to 30 % for surveys are not uncommon [62]. We believe that the response rate for interviews, at least in our case, is comparable. In total, 56 interview invitations were sent out by email. Furthermore, our Wireshark contact posted the interview invitation on the core developer mailing list.

We did not set a target number on how many interviews to conduct, but was hoping for 10-12. We ended up with 7. Despite being some interviews short, we were satisfied. The firm-paid developers we interviewed proved to be key figures in the communities and firms, and contributed largely with interesting and relevant data. It is important to keep in mind that the sample frame give limitations to how many interview participants we could get hold of.

Another challenge is that firm-paid developers are typically very busy people, and thus not all of them have time for an one hour long live interview. This was particularly true for the Samba developers we interviewed, all of which insisted to perform the interview over email. The problem with interviews over email is that the interviewer is not able to guide the conversation, and respond and adapt to the interviewee’s answers. This may lead to a less satisfactory result. For instance, the developer may lose the direction and/or interesting themes may not be sufficiently explored. We were interested in conducting as many interviews as possible and thus could not afford to reject any developers because they wanted to do it over email. To minimize the potential impact, we made an agreement to conduct follow-up interviews if we had further questions. The potential impact of conducting the interviews over email on the findings of this thesis is further discussed in the validity of the study.

Part IV

Results and Conclusions

CHAPTER 5 RESULTS

This chapter presents the results of the conducted research. The chapter is divided in two main sections, corresponding to the adopted dual-phased research approach: The first section provide results from the exploratory quantitative analysis (phase 1), and the second section provide results from the descriptive qualitative analysis (phase 2).

5.1 Results Phase 1 – Exploratory Analysis

This section presents the results from the first phase. The objective of the phase was to give insight in how commercial firms communicate across organizational boundaries in the communities using the public software informalisms. First, a set of metadata about the software informalisms is presented. This gives an indication of the amount of data that was analyzed and demonstrate how widespread the firm participation and contributions in the selected OSS projects are. Second, the results from the social network analysis is presented.

5.1.1 Results of Data Extraction

In both projects data was extracted from three sources: The mailing list, the bug tracking system and the code repository. All these sources contained entries from participants of the community during the selected time frame, which was from 2006 to and including 2011. The amount of entries extracted from the sources in Wireshark and Samba are visually displayed in Figure 5.1 and Figure 5.2, respectively. Table 5.1 gives an overview of the number of entries and individuals in each project.

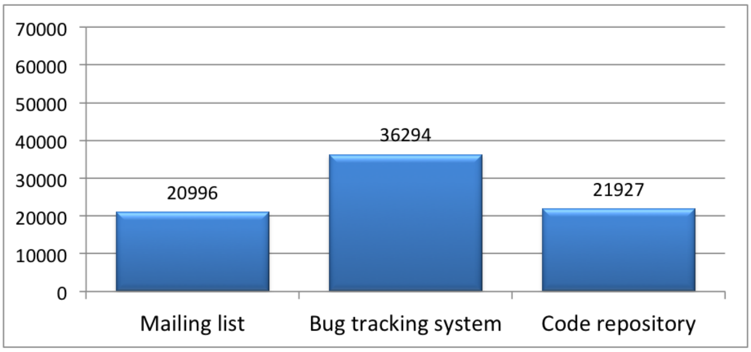


Figure 5.1: Extracted data from mailing list, bug tracking system and code repository in Wireshark.

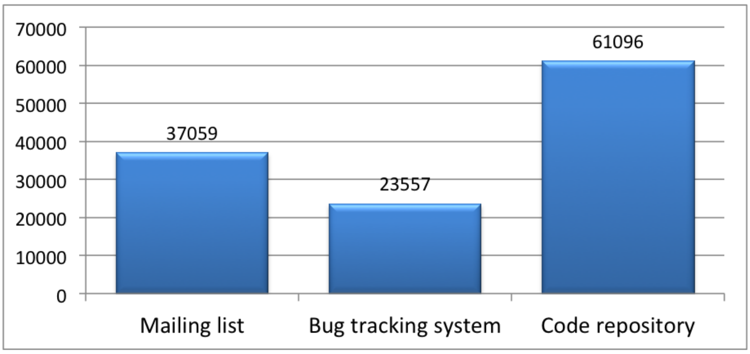


Figure 5.2: Extracted data from mailing list, bug tracking system and code repository in Samba.

Table 5.1: Overview of the extracted data from Wireshark and Samba.

Project	Data source	Entries	Individuals
Wireshark	Mailing list	20996	1918
	Bug tracking system	36294	269
	Code repository log	21927	35
Samba	Mailing list	37059	2749
	Bug tracking system	23557	297
	Code repository log	61096	243

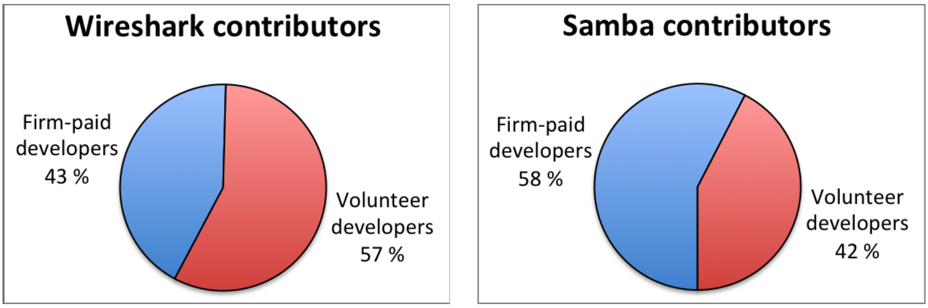
5.1.2 Results of Identification of Firm-paid Developers

By applying the identification techniques described in the research design we managed to classify the developers in the OSS projects as either firm-paid or volunteering. Developers classified as firm-paid were used in all of the subsequent analysis.

Wireshark

In Wireshark we were limited to the contributor list, which contained 802 entries at the time this study was conducted. Out of the 802 developers listed in the contributor list, 342 were classified as firm-paid developers. The remaining 460 were classified as volunteering developers. The division of developers is visualized in Figure 5.3. From the 342 firm-paid developers, 228 unique commercial firms were identified. The top 10 firms participating in the community with regard to number of developers is presented in Table 5.2. Only 8 % of the firms have 3 or more developers participating in the community. Whereas, 78 % of the firms have only one developer participating.

Figure 5.3: Division of firm-paid and volunteering developers in the Wireshark and Samba communities.



Samba

Identification of developers in Samba was not constrained the same way as in Wireshark, however, as described in the research design, we introduced some limitations to reduce the size and remove noise in the data set. In total, 316 developers were evaluated, where 182 were classified as firm-paid developers and 134 as volunteers. The division of developers is visualized in Figure 5.3. The 182 firm-paid developers are representing 90 different commercial firms. As with the Wireshark developers, we have summarized how many developers each firm have participating in the project. This is displayed in Table 5.2. The distribution of developers in Samba and Wireshark is similar; 9 % of the firms have 3 or more developers participating in the community, and 84 % have only one developer.

Table 5.2: Side-by-side comparison of the top 10 firms and the number of developers in the Wireshark and Samba communities.

Wireshark

Firm name	# of developers
Cisco	16
Ericsson	11
Siemens	8
Netapp	6
Citrix	5
Lucent	5
MXtelecom	5
Nokia	5
Axis	4
Harman	4

Samba

Firm name	# of developers
IBM	17
RedHat	14
SerNet	8
SUSE	8
EMC	4
SGI	4
Exanet	3
HP	3
Cisco	3
Canonical	2

5.1.3 Results of Data Filtering

Filtering and cleaning the data significantly reduced the size of the extracted data corpus. The residual data was used as the foundation in the subsequent collaboration pattern analysis. Charts displaying the distribution of emails, bugs and code use percentage and have the same scale for easier comparison across the OSS projects.

Wireshark

In the selected time frame for this study, 224 of the 342 identified firm-paid developers in Wireshark were active in the mailing list, the bug tracking system or both. 224 developers representing 132 different commercial firms. The following paragraphs present the activity by each firm in the mailing list, bug tracking system and code repository.

Mailing List By filtering the mailing list for firm-paid developers the number of entries was reduced from 20996 to 5630. These entires were posted by 78 different firms. There is a high variance in number of emails from each firm. 60 % of the emails come from the three most active firms: Ericsson, Philips and AT&T respectively. The distribution of emails among the top 15 participating firms is visualized in Figure 5.4. Although there are 78 firms participating, we only display the top 15 firms to ensure readability and visibility of the data. The activity of the excluded firms only constitute 9 % of the total activity and the individual firm activity is diminishing.

Bug Tracking System The entries in the bug tracking system were reduced to 14355 after the filtering. 110 of the identified firms were active in the selected time frame. Similar to the activity observed in the mailing list, there is a high disparity in activity. In the bug tracking system, 53 % of the activity is generated by two firms: Ericsson and Thales. The distributed activity can be seen in Figure 5.5. For better readability and visibility, we only include the top 15 firms. The activity of the excluded firms constitute less than 20 % of the total activity.

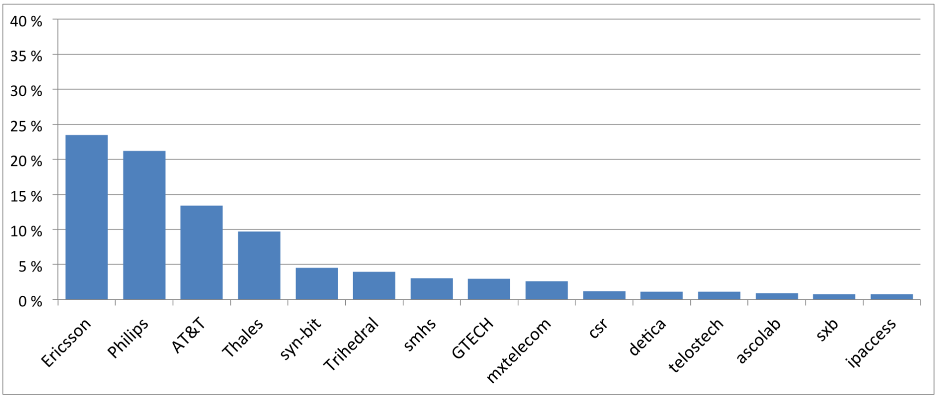


Figure 5.4: Distribution of emails among top 15 firms in Wireshark.

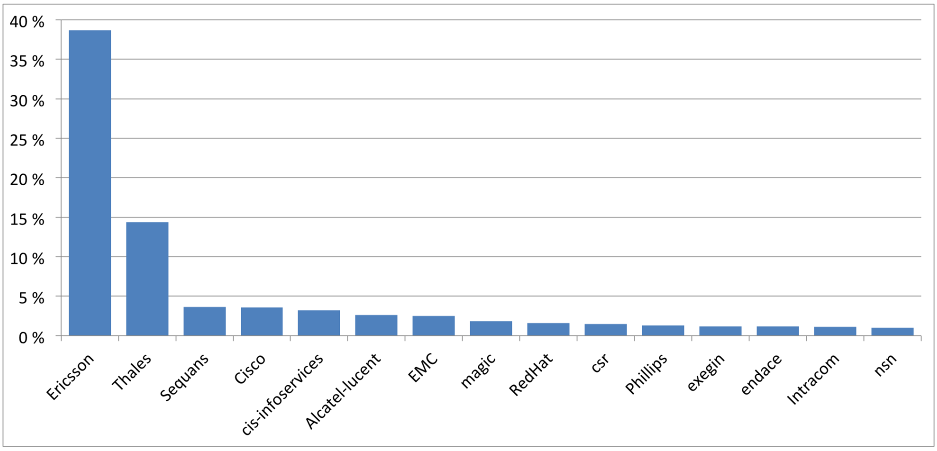


Figure 5.5: Distribution of bugs among the top 15 firms in Wireshark.

Code Repository The code repository contained 21927 entries, where 12053 of them were committed by firm-paid developers. From the total of 35 developers in the commit log, 20 are firm-paid developers. The big pie chart in Figure 5.6 illustrate the proportions of code commits by the represented firms. The little pie chart illustrate the proportion of commits done by the identified firm-paid developers compared to commits done by the volunteering developers.

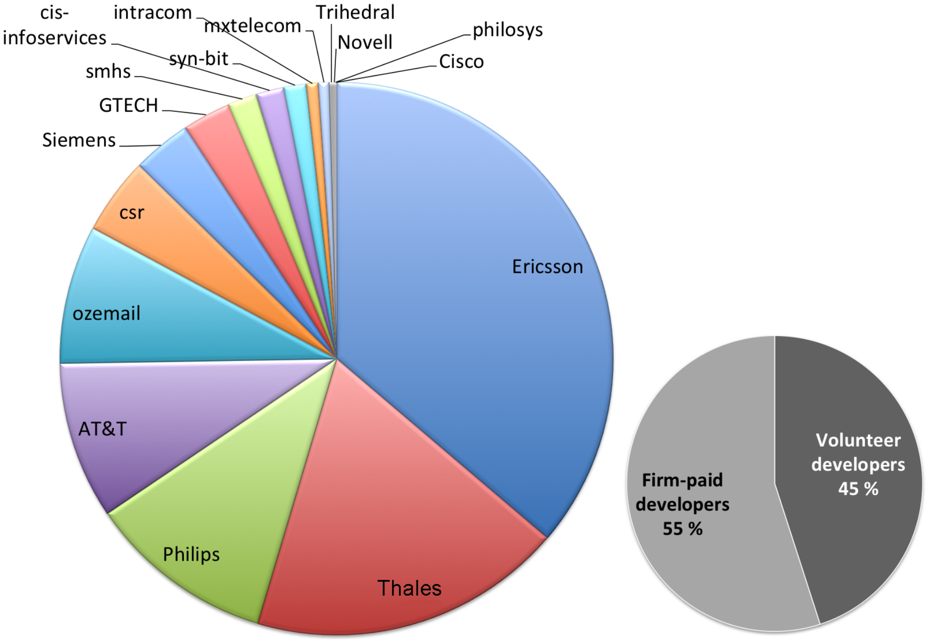


Figure 5.6: Breakdown of the code repository in Wireshark.

Samba

Since the identification of developers in Samba was done differently than in Wireshark, all the 182 identified firm-paid developers from 90 firms were active in one or more data sources in the selected time frame. The following paragraphs present the activity by each firm in the mailing list, bug tracking system and code repository.

Mailing List The number of entries in the mailing list were reduced from 37059 to 18075. The 18075 emails were posted by developers from 73 different firms. The distribution of emails are similar to the distribution observed in Wireshark's mailing list; a minority of the firms post the majority of the emails. The top three firms, Red Hat, SerNet and Google hold 56 % of the posted emails. This is displayed in the breakdown of the emails per firm in Figure 5.7. Only the top 15 firms are displayed to increase readability. The activity of the excluded firms was 12 % distributed across 58 firms.

Bug Tracking System The data entries from the bug tracking system were reduced from 23557 to 15577, which were submitted by 45 different firms. The top three firms, which are the same as the top three in the mailing list, are accountable for 65 % of the bug entries. See Figure 5.8 for an overview of the top 15 firms. The excluded firms holds less than 7 % of the bug entries.

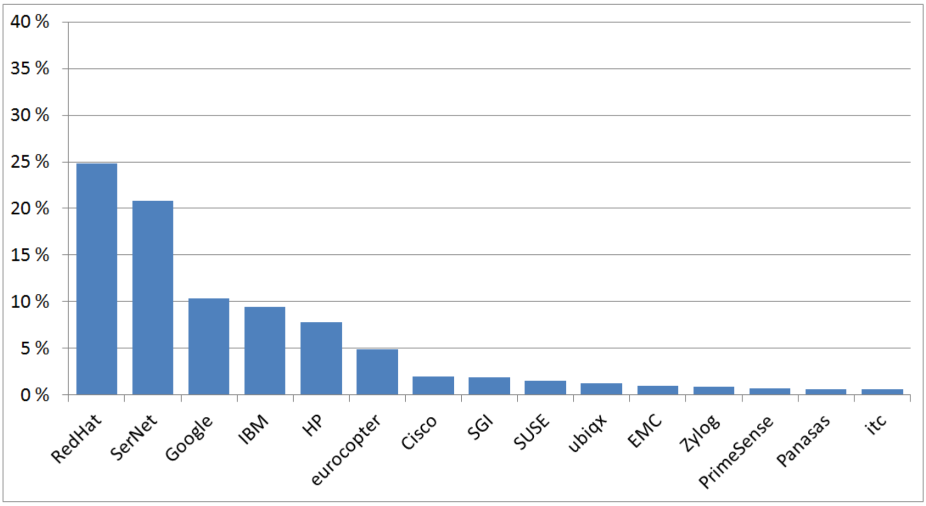


Figure 5.7: Distribution of emails among the top 15 firms in Samba.

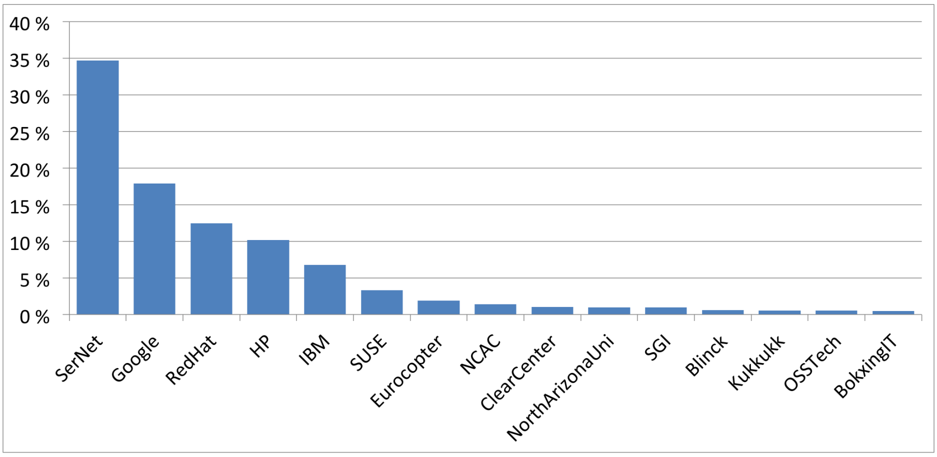


Figure 5.8: Distribution of bugs among the top 15 firms in Samba.

Code Repository List In total there were 61096 commits in the code repository in the selected time frame. The filtering of the data reveals that the majority of these are contributed by firm-paid developers. 34 firms account for 50354 of the commits, whereas, volunteering developers only account for 10868. The breakdown of firms and their commits are displayed in the big pie chart in Figure 5.9, where firms with less than 1 % are excluded. The little pie chart displays the proportion of commits done by firm-paid compared to the volunteering developers.

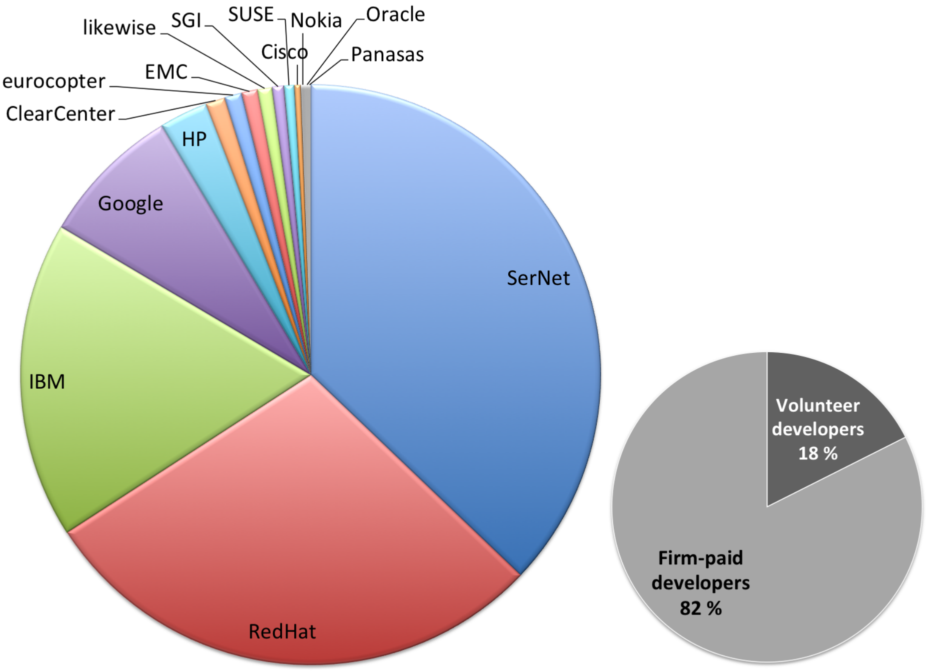


Figure 5.9: Breakdown of the code repository in Samba.

5.1.4 Result of Collaboration Pattern Analysis

Here we present the results of the collaboration pattern analysis. Three elements from each OSS project are presented: (1) Social networks of the mailing list and bug tracking system, (2) tables presenting key figures which were used to create the social networks, and (3) charts displaying the firm's average response time in the analyzed sources.

Wireshark

The social network analysis tool, NodeXL, was used to visualize the social networks of the mailing list and the bug tracking system in Wireshark. These are displayed in Figure 5.10 and Figure 5.11, respectively. The structure of the networks are generated by NodeXL based on discussions among the firms in the two analyzed sources.

An extract of the key figures used to create the visualizations is shown in Table 5.3 and Table 5.4, displaying the top 15 firms in the mailing list and the bug tracking system. The full lists of figures were too long to include in the thesis (consisting of approximately 100 entries), however, the figures presented are the most influential in designing the social network. The degree is the most interesting measure, showing how many connections the the firm has to other firms in the community. An explanation of the various measures and their importance is given in the pre-study, see section 3.2.

In addition to the social networks created from the discussions, the response time of each entry was captured. The number of emails and total response time were used to compute the average response time for each firm. As earlier, only the top 15 firms are included in the presentation here for better readability and visibility. The chart in Figure 5.12 display number of mail entries and average response time in hours for the top 15 firms. The chart in Figure 5.13 display number of bug entries and average response time in hours for the top 15 firms.

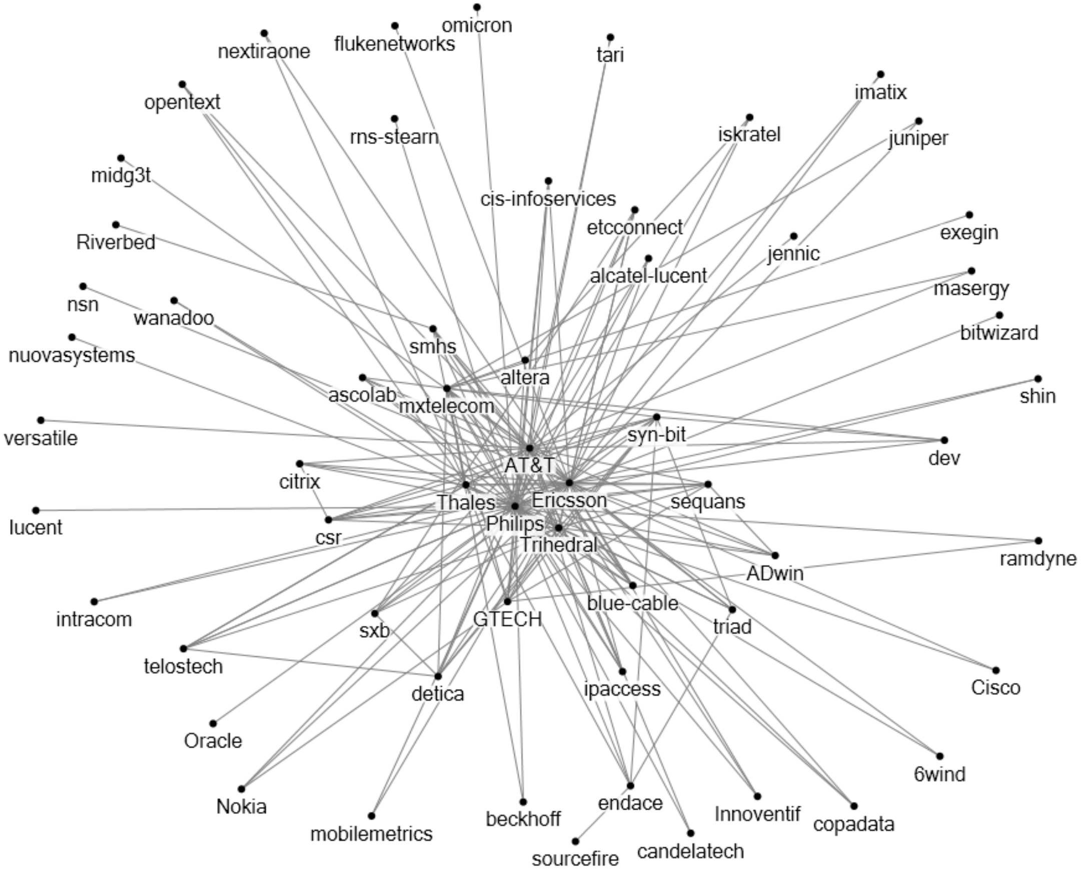


Figure 5.10: Social network from mailing list in Wireshark.

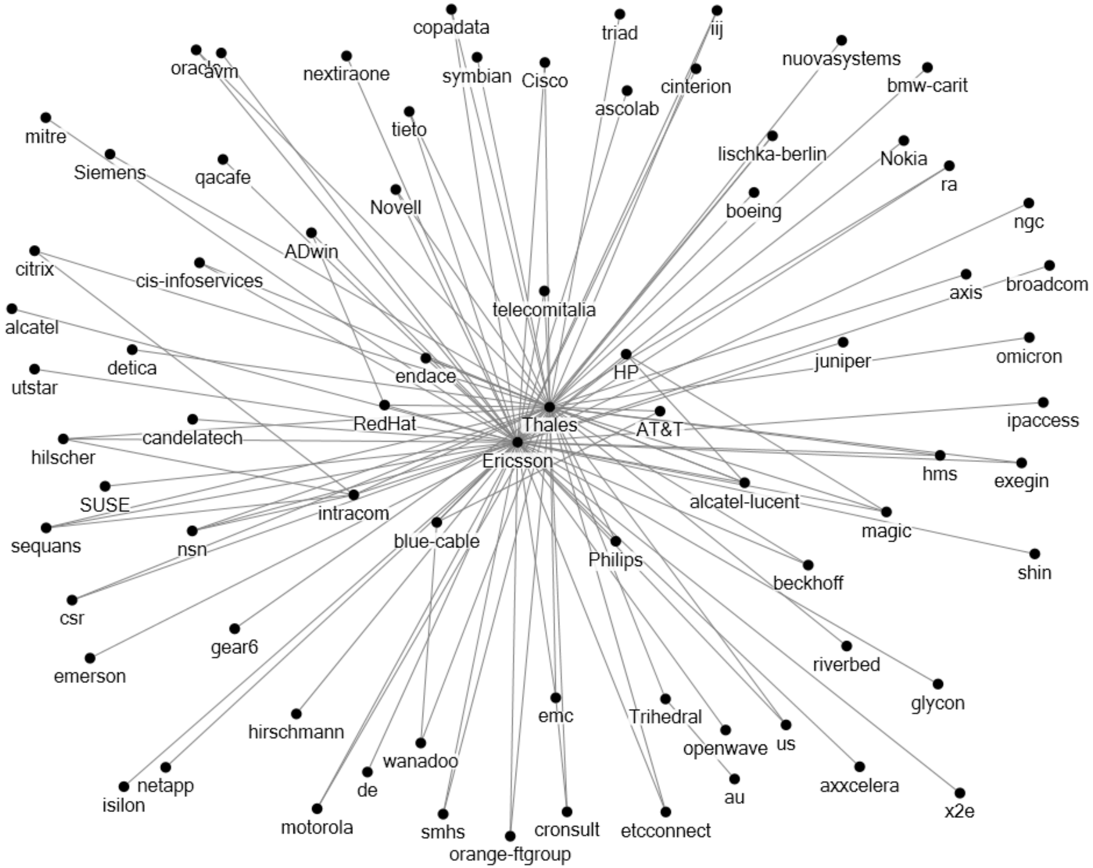


Figure 5.11: Social network from bug tracking system in Wireshark.

Table 5.3: Firm, number of entries, and centrality measures for the top 15 firms in the Wireshark mailing list.

Firm	Entries	Degree	Betweenness	Closeness
Philips	1195	48	0.275	0.640
Ericsson	1322	39	0.141	0.560
AT&T	756	34	0.118	0.533
Trihedral	222	21	0.011	0.457
Thales	548	19	0.019	0.453
mxtelecom	149	19	0.042	0.444
syn-bit	253	14	0.042	0.427
GTECH	165	13	0.006	0.419
detica	64	10	0.000	0.408
csr	67	10	0.000	0.404
sequans	31	10	0.000	0.404
smhs	170	9	0.020	0.404
telostech	64	8	0.000	0.397
sxb	43	8	0.000	0.397
endace	19	8	0.020	0.390

Table 5.4: Firm, number of entries, and centrality measures for the top 15 firms in the Wireshark bug tracking system.

Firm	Entries	Degree	Betweenness	Closeness
Ericsson	5551	61	0.323	0.567
Thales	2060	51	0.232	0.518
intracom	162	8	0.003	0.365
blue-cable	72	7	0.007	0.363
alcatel-lucent	376	6	0.001	0.360
sequans	522	5	0.000	0.358
EMC	359	5	0.002	0.358
magic	260	5	0.001	0.358
RedHat	232	5	0.001	0.358
endace	170	5	0.000	0.358
NSN	144	5	0.000	0.358
HP	143	5	0.000	0.319
hilscher	52	5	0.000	0.358
Cisco	511	4	0.000	0.356
cis-infoservices	460	4	0.000	0.356

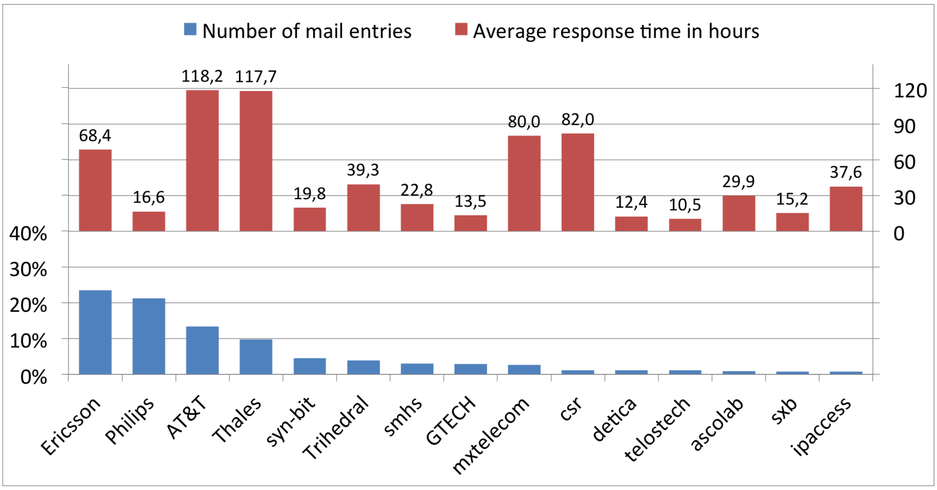


Figure 5.12: Number of mail entries and average response time in hours by the top 15 firms in Wireshark.

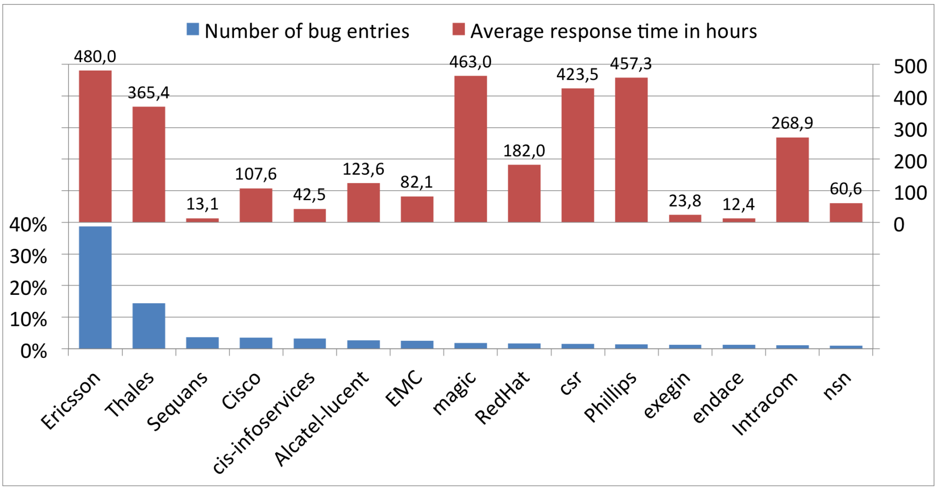


Figure 5.13: Number of bug entries and average response time in hours by the top 15 firms in Wireshark.

Samba

The social networks from the mailing list and bug tracking system in Samba are displayed in Figure 5.14 and Figure 5.15, respectively. Followed by key figures used to create the networks in Table 5.5 and Table 5.6. An interesting property of the key figures in Wireshark and Samba is that the number of entries and centrality degree are not linear. This is further discussed in the next chapter.

Last, charts displaying the average response time of the top 15 firms in the mailing list and the bug tracking system are shown in Figure 5.16 and Figure 5.17.

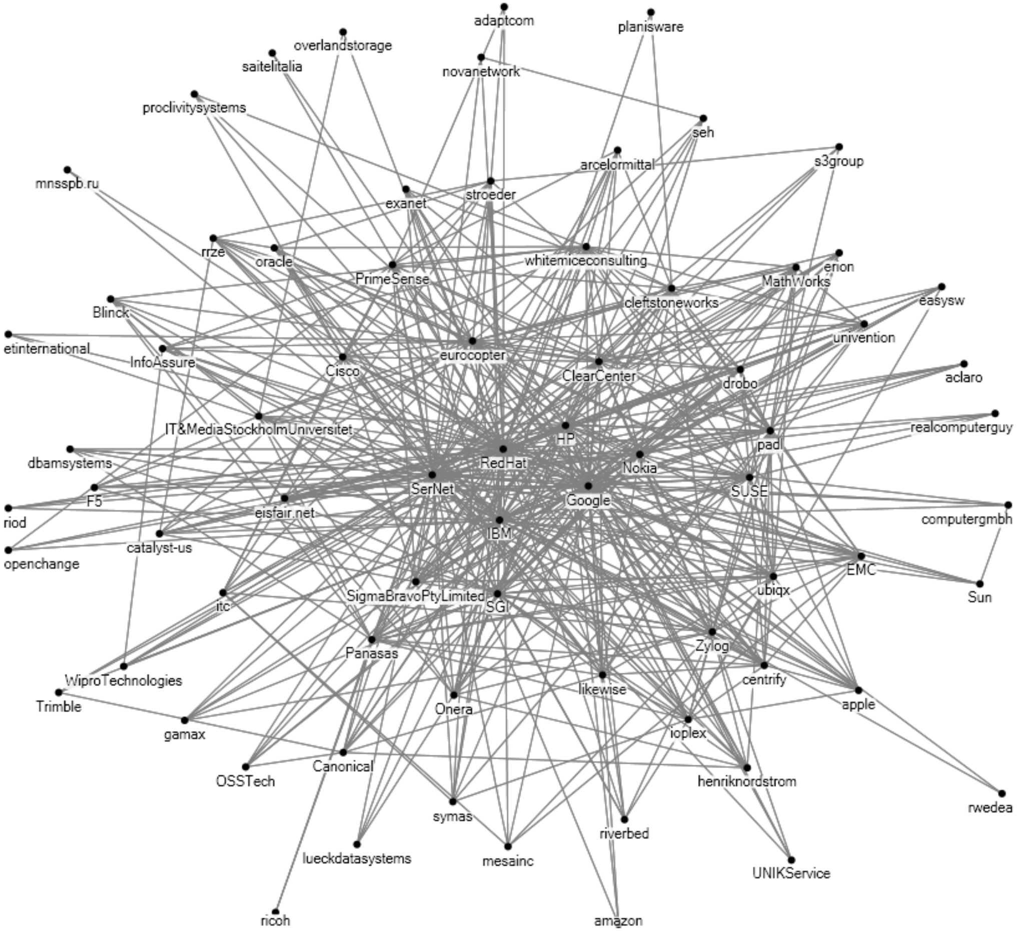


Figure 5.14: Social network from mailing list in Samba.

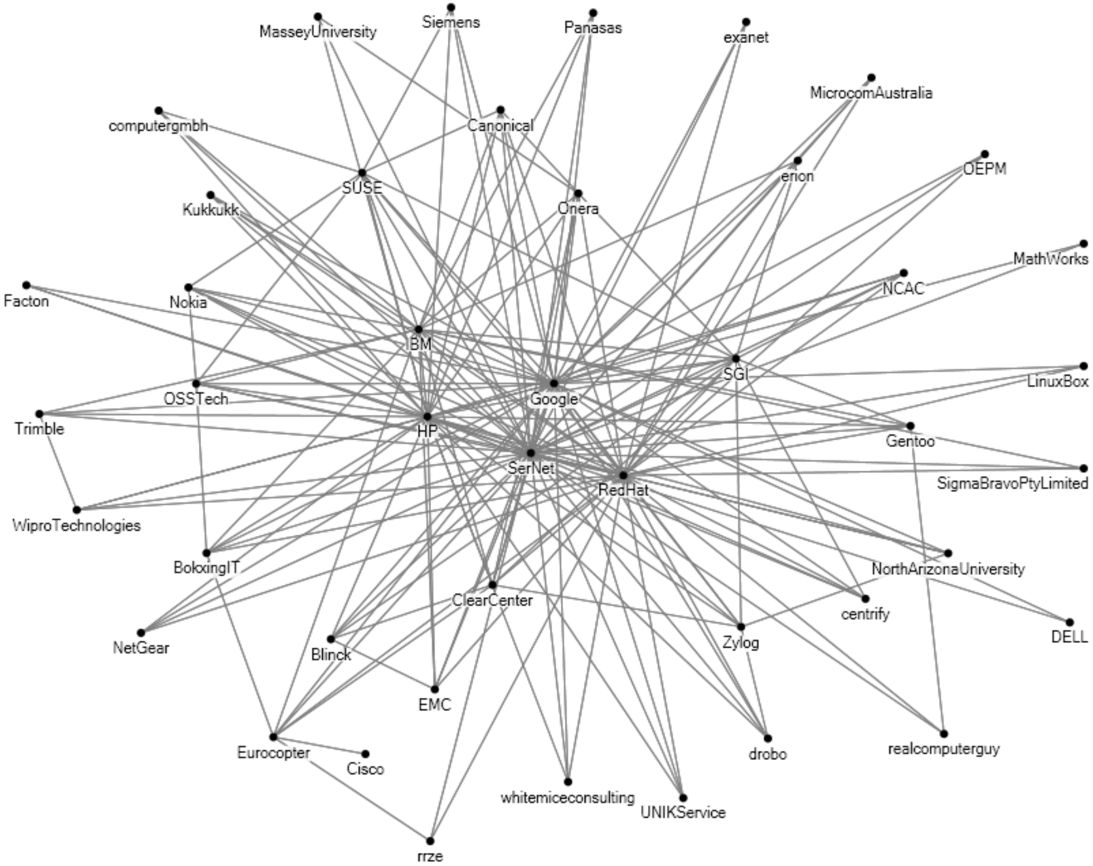


Figure 5.15: Social network from bug tracking system in Samba.

Table 5.5: Firm, number of entries, and centrality measures for the top 15 firms in the Samba mailing list.

Firm	Entries	Degree	Betweenness	Closeness
RedHat	4480	71	0.278	0.948
SerNet	3765	66	0.184	0.890
google	1835	57	0.115	0.802
IBM	1701	48	0.061	0.730
HP	1408	44	0.050	0.702
eurocopter	874	35	0.027	0.646
SGI	335	29	0.012	0.613
padl	82	29	0.010	0.613
Zylog	159	28	0.019	0.608
Nokia	104	28	0.008	0.608
Suse	277	27	0.006	0.603
ClearCenter	58	27	0.008	0.603
cleftstoneworks	40	25	0.009	0.593
ubiqx	218	24	0.004	0.589

Table 5.6: Firm, number of entries, and centrality measures for the top 15 firms in the Samba bug tracking system.

Firm	Entries	Degree	Betweenness	Closeness
SerNet	5260	44	0.304	0.955
Google	2789	38	0.179	0.840
RedHat	1936	35	0.143	0.793
HP	1585	33	0.106	0.750
IBM	1052	23	0.039	0.646
SGI	128	16	0.009	0.575
SUSE	520	13	0.004	0.552
ClearCenter	161	12	0.003	0.552
Onera	73	11	0.002	0.538
Eurocopter	298	10	0.045	0.538
Zylog	71	10	0.001	0.531
OSSTech	87	10	0.000	0.531
Nokia	19	9	0.000	0.525
BokxingIT	74	9	0.000	0.531
Canonical	40	9	0.000	0.525

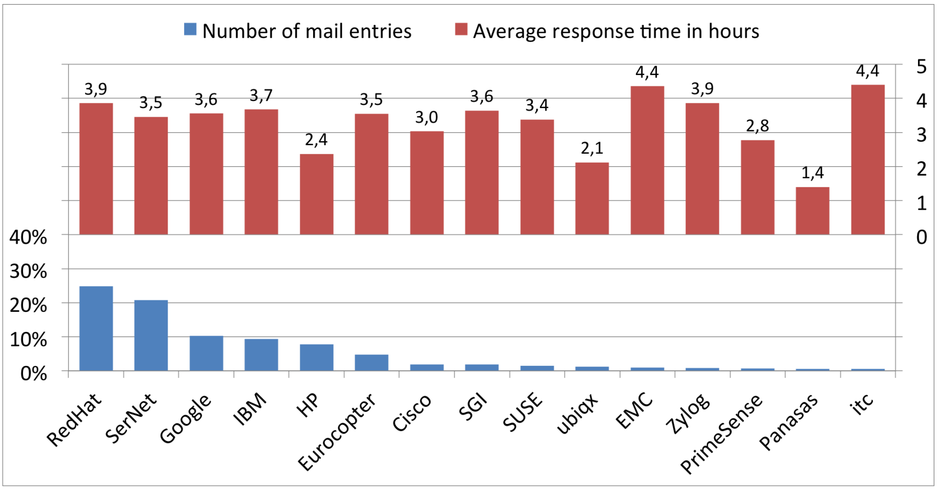


Figure 5.16: Number of mail entries and average response time in hours by the top 15 firms in Samba.

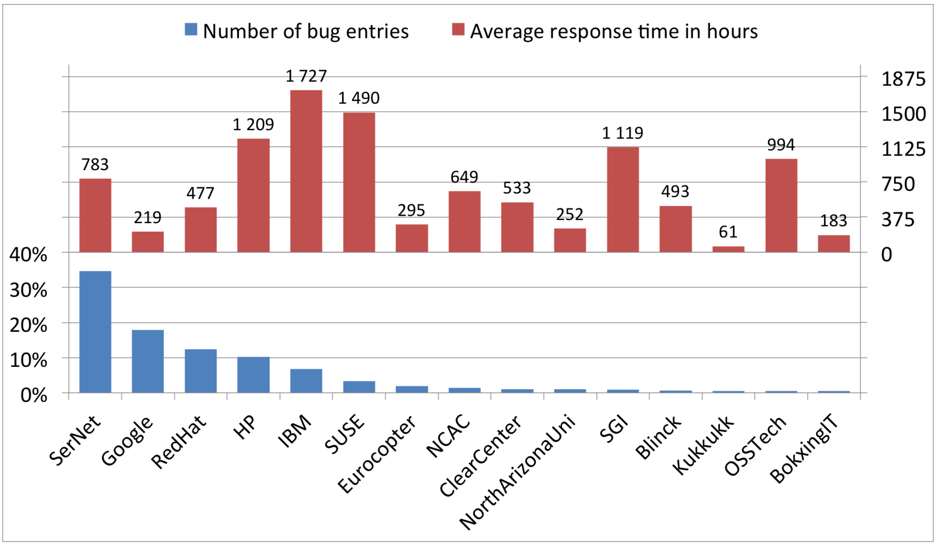


Figure 5.17: Number of bug entries and average response time in hours by the top 15 firms in Samba.

5.2 Results Phase 2 – Descriptive Analysis

Here the findings from the second phase of the study is presented. A mind map was generated from the results of the thematic analysis of the interview transcripts. The mind map was divided in two parts for better readability. Part one is displayed in Figure 5.18 and part two in Figure 5.19. In the following subsections we describe each higher-order theme in the mind map along with the underlying sub themes and codes. The themes are presented in decreasing order of importance with regard to the research questions. At the end we include a brief summary of interesting issues found in the thematic analysis, which were outside the scope of our research but worth to mention. The findings are presented along with quotes from the interviews.

The mind map contains themes, sub themes and codes expressed by the developers in the interviews. Behind each leaf node (code) there is a small box figure along with a list of which developers that expressed the code. The developer IDs created in the research design is used in the lists, see Table 4.5. The box figure indicates which project the developers participate in, displaying 1 for Wireshark, 2 for Samba, and 1 and 2 for both projects.

As the interviews were conducted in a semi-structured fashion, it is important to remark that the interviewees were not necessarily asked the exact same questions. Questions were added, removed or adjusted to the interviewee's answers. Therefore, although only one developer mentions a certain element in the organizational cross-collaboration, it does not imply that the others do not agree or have the same perception. Due to this, we do not present the findings using the split of developers expressing that, for instance "*five out of seven developers said ...*". Rather, we elaborate each interesting point mentioned in the interview and at times emphasize that a finding was mentioned by developers from both projects. This is in line with Yin's "*analytic generalization*", implying that the primary role of a case study is to expand and generalize theories, not to provide representative numbers for a particular population [19].

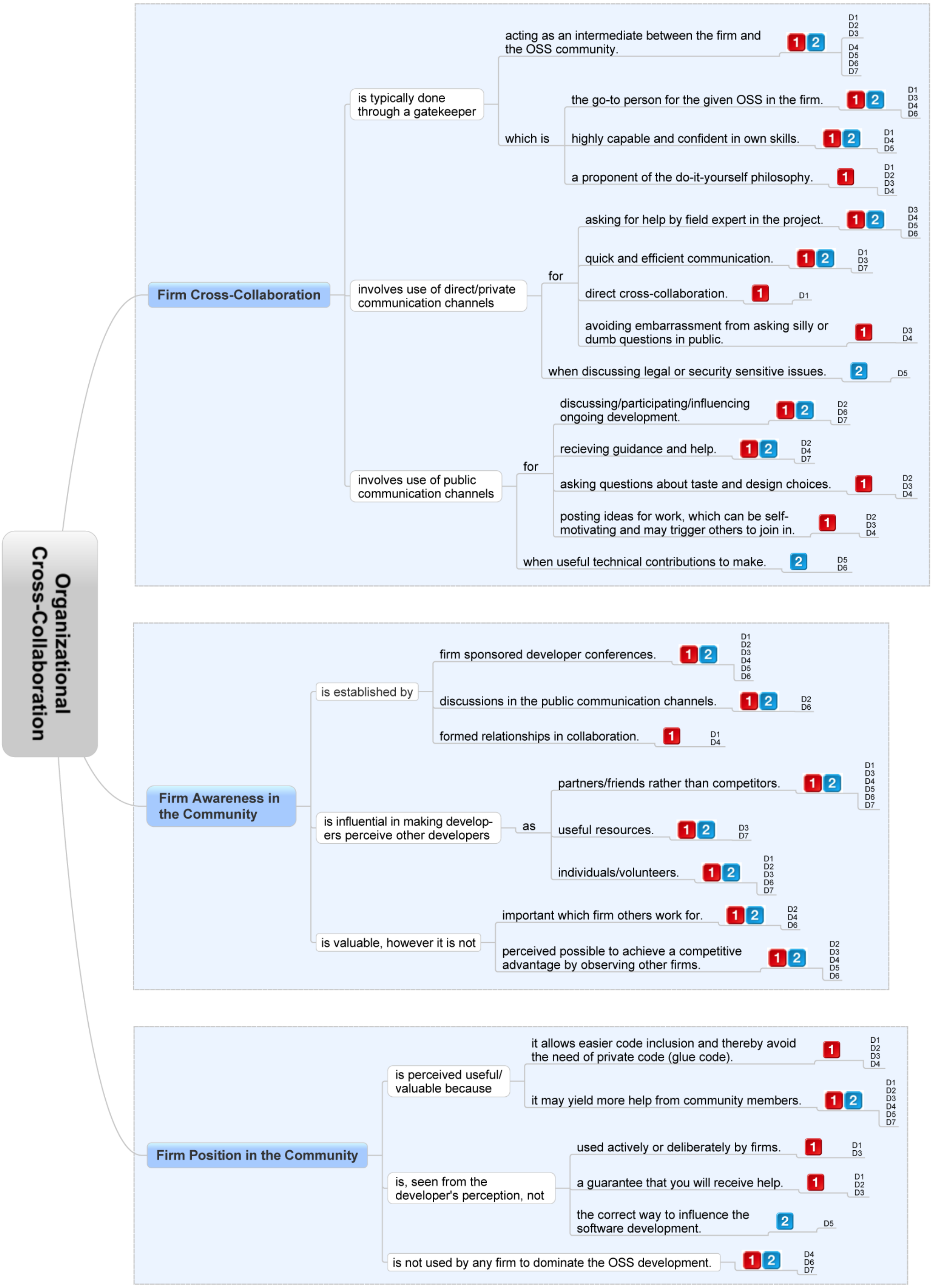

Organizational Cross-Collaboration
Firm Cross-Collaboration
is typically done through a gatekeeper
acting as an intermediate between the firm and the OSS community. 1 2 D1 D2 D3 D4 D5 D6 D7
which is
the go-to person for the given OSS in the firm. 1 2 D1 D3 D4 D6
highly capable and confident in own skills. 1 2 D1 D4 D5
a proponent of the do-it-yourself philosophy. 1 D1 D2 D3 D4
involves use of direct/private communication channels
for
asking for help by field expert in the project. 1 2 D3 D4 D5 D6
quick and efficient communication. 1 2 D1 D3 D7
direct cross-collaboration. 1 D1
avoiding embarrassment from asking silly or dumb questions in public. 1 D3 D4
when discussing legal or security sensitive issues. 2 D5
involves use of public communication channels
for
discussing/participating/influencing ongoing development. 1 2 D2 D6 D7
recieving guidance and help. 1 2 D2 D4 D7
asking questions about taste and design choices. 1 D2 D3 D4
posting ideas for work, which can be self-motivating and may trigger others to join in. 1 D2 D3 D4
when useful technical contributions to make. 2 D5 D6
Firm Awareness in the Community
is established by
firm sponsored developer conferences. 1 2 D1 D2 D3 D4 D5 D6
discussions in the public communication channels. 1 2 D2 D6
formed relationships in collaboration. 1 D1 D4
is influential in making developers perceive other developers
as
partners/friends rather than competitors. 1 2 D1 D3 D4 D5 D6 D7
useful resources. 1 2 D3 D7
individuals/volunteers. 1 2 D1 D2 D3 D6 D7
is valuable, however it is not
important which firm others work for. 1 2 D2 D4 D6
perceived possible to achieve a competitive advantage by observing other firms. 1 2 D2 D3 D4 D5 D6
Firm Position in the Community
is perceived useful/ valuable because
it allows easier code inclusion and thereby avoid the need of private code (glue code). 1 D1 D2 D3 D4
it may yield more help from community members. 1 2 D1 D2 D3 D4 D5 D7
is, seen from the developer's perception, not
used actively or deliberately by firms. 1 D1 D3
a guarantee that you will receive help. 1 D1 D2 D3
the correct way to influence the software development. 2 D5
is not used by any firm to dominate the OSS development. 1 2 D4 D6 D7

Figure 5.18: Mind map part 1 of 2 displaying the thematic analysis results.

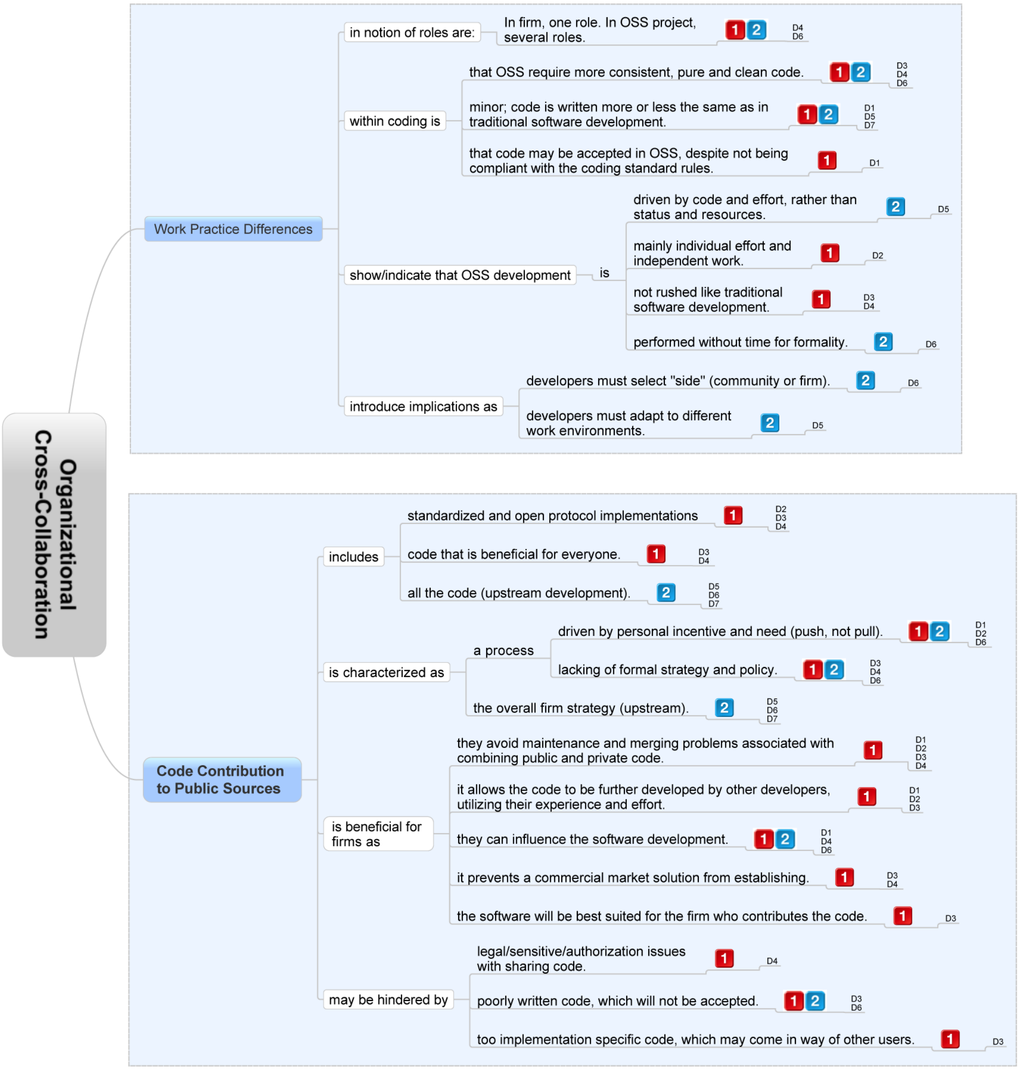

Organizational Cross-Collaboration
Work Practice Differences
in notion of roles are:
In firm, one role. In OSS project, several roles.
1 2
D4 D6
within coding is
that OSS require more consistent, pure and clean code.
1 2
D3 D4 D6
minor; code is written more or less the same as in traditional software development.
1 2
D1 D5 D7
that code may be accepted in OSS, despite not being compliant with the coding standard rules.
1
D1
show/indicate that OSS development
is
driven by code and effort, rather than status and resources.
2
D5
mainly individual effort and independent work.
1
D2
not rushed like traditional software development.
1
D3 D4
performed without time for formality.
2
D6
introduce implications as
developers must select "side" (community or firm).
2
D6
developers must adapt to different work environments.
2
D5
Code Contribution to Public Sources
includes
standardized and open protocol implementations
1
D2 D3 D4
code that is beneficial for everyone.
1
D3 D4
all the code (upstream development).
2
D5 D6 D7
is characterized as
a process
driven by personal incentive and need (push, not pull).
1 2
D1 D2 D6
lacking of formal strategy and policy.
1 2
D3 D4 D6
the overall firm strategy (upstream).
2
D5 D6 D7
is beneficial for firms as
they avoid maintenance and merging problems associated with combining public and private code.
1
D1 D2 D3 D4
it allows the code to be further developed by other developers, utilizing their experience and effort.
1
D1 D2 D3
they can influence the software development.
1 2
D1 D4 D6
it prevents a commercial market solution from establishing.
1
D3 D4
the software will be best suited for the firm who contributes the code.
1
D3
may be hindered by
legal/sensitive/authorization issues with sharing code.
1
D4
poorly written code, which will not be accepted.
1 2
D3 D6
too implementation specific code, which may come in way of other users.
1
D3

Figure 5.19: Mind map part 2 of 2 displaying the thematic analysis results.

5.2.1 Firm Cross-Collaboration

Understanding how commercial firms cross-collaborate in OSS projects was fundamental in our research context. The results of the interview transcript analysis within this theme were divided in three main categories as shown in the mind map.

Gatekeeper

We start by presenting one of the most noteworthy findings; describing how the commercial firms are connected to the OSS community and how the information is flowing between the community and the firms. From the quantitative analysis of the firm's participation in the community, in terms of code contribution and activity in the mailing list and bug tracking system, it was apparent that most firms have: (1) One developer participating in the community or (2) multiple developers participating in the community, but the majority of activity was generated by one of the firm's developers. This lead us to introduce the term *gatekeeper* to describe the role of the participating firm developer, suggesting that the flow of code, bug reports and communication between the OSS community and the firm goes through this one developer. This is illustrated in Figure 5.20.

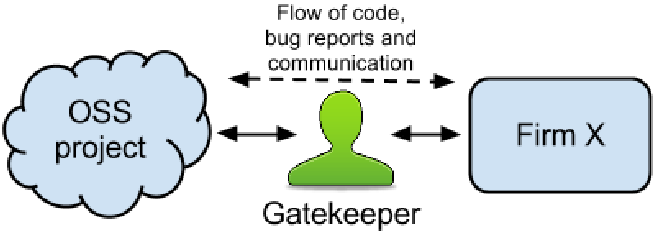


Figure 5.20: The gatekeeper developer acting as an intermediary between firm X and the OSS project.

All the interview respondents acknowledged this perception, and they provided additional comments describing the gatekeeper's role. Developer D1 stated that when coworkers find bugs in the OSS, they report them to him and he reports them to the community. Developer D7 expressed a similar perception and said: "*Yes, I act as a bridge between Firm X and Samba and forward bugs/errors to the community.*" From the interviews, it is apparent that communication and code is done in the same fashion.

A key characteristic of the gatekeeper is that he is highly capable and confident in his own skills when it comes to the OSS. In addition, the gatekeeper is typically the go-to person for the OSS in the firm, and he is a proponent of the do-it-yourself philosophy. The latter referring to gatekeeper, implying that he would rather do it himself than bring it to the community. This is illustrated by the following statement given by developer D4: "*... I have never seen anybody in Firm X reporting bugs directly to the Wireshark team or anything like that. Usually they just come to me and I'll fix it.*" Developer D2 expressed a similar view: "*Bugs,* *yes. New functionality, yes some, not that much. (...) Bug fixes I do myself. It is usually not that complicated.*" The following comment by developer D3 confirms the prior statements:

> "*I don't think I have ever raised a bug for something someone has brought to me. I have just made the change myself. I can't remember anyone asking to raise a bug, instead of me looking at it.*"

The gatekeeper is typically a core developer or co-developer in the OSS community, and is responsible for providing key functionality to the other developers in the firm. The following statements demonstrate this:

> "*Many of the other core developers are working for smaller companies, and have responsibility for the internal protocols that their company needs. (...) I think most developers work individually, and have the role of providing Wireshark functionality to the other developers in the firm.*" – Developer D2

> "*Yes, everybody definitely knows that I am the Wireshark guy. All the developers, testers and customer support people know that they can come to me if they have Wireshark issues. Partially because I have done some internal training, showing people what Wireshark is able to do and train them for it. I also build our own custom version for internal use. We have some internal protocols that the normal version of Wireshark doesn't understand. I have built custom versions that people can use to understand those protocols and stuff like that.*"
> – Developer D4

A very interesting point uncovered in the analysis is that despite the firm-size and number of participating developers in the OSS community, all firms have a gatekeeper. It is natural that the role of the gatekeeper is multifaceted as firms have different needs and work practices, however, in all the cases that the gatekeeper operate as an intermediate between the firm and the OSS project in some way. Even firms participating in the development *upstream*, which is working within the community to develop the OSS, have a developer they regard as the gatekeeper. An example of this is captured in the following statement made by developer D5:

> "*In general when it comes to contributing patches upstream each Firm X developer is independent and can directly approach the upstream project. (...) The Firm X Samba package maintainer usually has the task of being the gatekeeper for those bugs that have been reported against Firm X products by the customers or the support teams, ...*"

In the case of this firm, which is developing upstream, each individual developer contributes code independently to the project. The bugs, on the other hand, is managed by the gatekeeper which submits bug reports on behalf of the firm into the OSS project's bug tracking system.

Private Communication Channels

Communication between firm-paid developers in the OSS community is another major aspect of the firm cross-collaboration. In the quantitative phase the OSS projects' public software informalisms provided an overview of the communication in the communities. However, this overview was only showing the communication taking place in public, and thus we asked developers how they communicate with other community members. From the analysis of the interview transcripts, it became apparent that developers use a mix of public and private communication channels. The public and private channels each have strengths and weaknesses, and is used differently to fulfill communication needs by the developers. First, we elaborate what developers use the direct and private communication channels in the cross-collaboration for. Second, in the next section we describe what developers use the public communication channels in the cross-collaboration for.

The direct and/or private communication channels used by developers in the OSS projects are mainly e-mail and instant messaging. Developer D2 said that he has used telephone, but that is rare. Use of private channels for communicating is widespread among the interviewed developers. The mind map shows that developers in both projects mentioned that they have used direct and/or private communication channels for asking for help by the field expert in the project. Developer D3 said:

> "*I have done it [contacted developers directly] different times in the past. Not just as a general 'I am stuck, can you help', but because it would be an area I knew the other guy was working on.*"

Developer D6: "*Usually I tend to do researches and development myself. I usually seek review of my work. Should I need assistance, I am going to address directly a developer in the community.*" Furthermore, the developers stated that a private channel is a quick and efficient communication medium. Developer D7 explained that he uses instant messaging for contacting developers in the community when he wants feedback quick.

Developer D3 and D4 said that they have contacted other developers directly to avoid asking silly or dumb questions in public. Developer D4:

> "*I got relationships with other developers and sometimes we don't want to ask the mailing list cause it is a really stupid question and you don't want to ask the whole mailing list, so you just ask the guy you trust.*"

Despite the widespread use of private communication, the majority of the developers emphasized that they try to keep most of the communication and discussions in public, which allow others to keep track of the development and join in. Transparency in OSS is important for successfully utilizing the community workforce.

Developer D5 mentioned that when discussing legal or security sensitive issues he uses a private communication channel. The nature of such issues invoke use of private channels as posting it in the public channels may result in security breaches or similarly undesirable bad situations. Although none of the other developers said anything about the use of direct channels for such issues, we believe that it is common procedure in most OSS projects.

Public Communication Channels

The public communication channels used in the studied OSS projects were the mailing list and bug tracking system. These channels are heavily used in OSS development for collaboration, communication and coordination. Developers mentioned several incentives for using the channels in the interviews. Wireshark and Samba developers said that they use the public channels for discussing, participating and/or influencing the ongoing development. The developers gave the following statements when asked when they post to the mailing list or participate in discussions:

> "*Basically, the times when I need guidance or I have a problem, or answering other people's questions, whether it is other developers or users or whatever. Or if I have an idea about something. (...) I made a suggestion 'hey maybe we should do something to catch this problem automatically in the build-bots rather than ...' Anyway, just making suggestions and putting them out basically.*" – Developer D4

> "*Usually all discussions are done on the mailing list (...) this way we have a history of all discussions. I participate in discussion either to help someone with Samba or to make my point in area of my interest at the moment.*" – Developer D6

Influencing the development by participating in the public communication channels is one of the incentive expressed by developer D1:

> "*If they are working on something that I see as usable for us internally, we find it interesting. It is smart to participate in the discussions when they are doing the development, and not come in afterwards. That is because while they are doing the changes and the development, they are more open for suggestions for changes and improvements.*"

Asking for guidance and help in the mailing list is common, however, developers underline that they do not ask for the solution to a problem. Rather, they ask for receiving useful advice and a push in the right direction. Developer D3: "*Sometimes I have sent emails to the development list and said that I am confused by this, can someone shed some light on it.*" Developer D4 expressed a similar approach:

> "*More often I'll ask people 'OK, I have this problem and I'm trying to solve it. I can see two ways to solve it, does anybody have an opinion on which way is the better way?' More guidance than asking them to solve the problem.*"

This behavior is consistent with the characterization of the gatekeeper given above. They regard problems as challenges, and try to understand and find solutions themselves.

Developer D2, D3 and D4 said that they ask questions about taste and design choice in the public channels. Developer D3:

> "*I usually try to make it a question of taste, rather than ask or post to the mailing list for help because 'I don't understand this'. I much prefer to understand it, but say what would people rather I did, I could do this or that. Yeah, I'd rather make it a question of taste.*"

Posting ideas for work is also common, and some of the interviewed developers find it motivating to describe their idea and approach to the other community members. This way the others are expecting the work do be done, and it allow others to come with suggestions and join the development. Developer D5 and D6 support this by stating: "*I tend to participate in discussions where I feel I have a useful technical contribution to make.*" and "*I participate in discussion either to help someone with Samba or to make my point in area of my interest at the moment.*"

5.2.2 Firm Awareness in the Community

Another aspect of the organizational cross-collaboration in OSS projects is the awareness of participating firms in the community. We have used the term *firm awareness* in this context, which is an extension of the term group awareness (see subsection 2.2.3).

Establishment of Firm Awareness

From the interviews, we see that the firm awareness is established in various ways. Both projects have a yearly developer conference, named Sharkfest and SambaXP for Wireshark and Samba respectively. The conference is the most prominent way developers meet and learn about other firms and their participation in the community development. All of the interviewed developers, except one, have attended the conference. Developer D1 pointed out the importance of the social network building taking place in these events, and he supported the statement by adding that firms commonly cover the developer's expenses associated with the conference. The other interviewees confirm D1's perception and verify that the firms sponsor their attendance at the conference.

Developer D2 expressed multiple ways the firm awareness may be established in the following quote:

> "*Thanks to Sharkfest I have actually meet many of the developers in real life, and hence I know them better and know where they work. Besides from that, much information about other developer's employment can be observed through the developer mailing list. In addition, there are some information exchanges done in the core mailing list and private channels, which are not accessible to the*

> *public. But most of the information is available at the developer and user mailing lists.*"

Accordingly, developer conferences, discussions in the public communication channels and relationships formed in collaboration are all means of establishing firm awareness in the community. Attendance at the conference is not critical for firm awareness as other channels are available. This is illustrated by D7, which has not attended any of the developer conferences, but still has a good awareness of developers and their corresponding firms in the community.

Influence of Firm Awareness

The firm awareness leads to questions related to the individual firm-paid developer's perception of other firm-paid developers in the OSS community. Firms working within the same business domain are often competitors in the market, and thus it is interesting to see how influential the firm awareness is when firms come together in community-based OSS projects to develop software collectively. Surprisingly, the firm-paid developers said that they perceive other developers as partners and/or friends rather than competitors. Developer D5 pointed out that he has met many of the developers at the developer conferences, and that he regards many of them as friends. Developer D1 explained that he does not make any distinction between a firm-paid developer and a volunteering developer, and said: "*I think of them as developers, and not about which firms they represent.*" Developer D7 said that he perceive others as partners. Developer D6: "*I've always thought of others as partners. Even more - I think about them as colleagues.*" The other developers gave similar statements and dismiss the perception of other firm-paid developers as competitors:

> "*Actually, I guess as things have evolved we do actually compete in some respect with some of these people at this point. But that haven't really occurred to me much. The people that are seen contributing, in the mailing list or reporting bugs, I have usually thought of as, like I have noticed more people who tend to be customers of ours, rather than true competitors. We might be competitors within some areas, but I have never really thought about it I guess.*" – Developer D4

> "*Yes. And you think about other firms as your competitors, but I don't think that really comes in to my interactions really. We have a competitor, and I have sometimes seen contributions to Wireshark from their developers, but I think that is good. (...) No, I don't think of them as companies. I think of them as individuals.*"
> – Developer D3

The perception of other developers as individuals or volunteers was confirmed by developer D7 and by D6 which said: "*Always as individuals. We were never discussing the companies we work for. We were always discussing the project.*" Developer D2 complemented this perception by giving the following statement:

> "*I think I see all developers as volunteers, because I believe that there is none that are actually paid primarily to develop Wireshark. Rather, I*

> *think most of the developers have persuaded their employer to actually allow them to spare some hours of their time for development of Wireshark.*"

Developer D3 and D7 said that they perceive other firm-paid developers as useful resources, which can be utilized in their day-to-day job. Developer D3:

> "*(...) because it would be an area I knew the other guy was working on. Or even better, when it is your understanding of a protocol that you have both worked on, and it is a question of 'what do you think is the spec here?' or 'do you think this is correct, or do you think that is correct?' The other developers are a good resource for helping you with work really, and the way things work. As well as with actual coding, it is very useful to be able email someone and say 'you know, some of our customers do this and some of our customers do that. What is your reading of the spec? What is your experience?'. That is very useful. It is a good way to find people who are interested in the same things, or help you with work questions.*"

Value of Firm Awareness

The firm awareness in the community is perceived as valuable. However, developers remarked that it is not the knowledge of what firm others work for that is valuable, rather it is the knowledge of what business domain they are working within. Developer D2: "*Yes, but I don't know that much about the firms of the other developers. They typically say that they work for firm xxx, and that's it. What firm they are working for is not that important to me.*" Developer D3 emphasized the potential value of having firm awareness in the following quote:

> "*No, I don't think of them [other firm-paid developers] as companies. I think of them as individuals. I know that D2 may have some role as a contact for Firm X. I think he has made that role himself, and I think a lot of people within Firm X at times would contact him with specific requests or make log files available for him.* ***I know that D2 may be someone who is good at getting log files for specific things****. In the past when I was working with voice over IP, I think sometimes he was able to give me some log files from within his company, but I don't really think of him as the company representative. I think of him as a company person who may be able to get logs for me, like he does.*"

Clearly, the firm awareness in the case of developer D3 is highly valuable as he knows what other firm-paid developers in the OSS community to contact when he is in need of specific artifacts, for instance, log files.

The interviewees were asked about the competitiveness between firms in the OSS community. Additionally, they were asked if they consider that their contributions can be used by others firms to gain or recapture competitive advantage. The majority of the developers provided statements that dismiss this perception. Developer D4: "*I don't think that it is really of much concern. I mean, with something like Wireshark I don't envision people getting a competitive advantage when everybody can use it.*" Developer D2:

> "*As Firm X does not directly control Wireshark, I guess we have to be a bit careful when we are in contact with other developers. (...) I believe, in the general case, that you gain more from contributing to the development, that retaining your code from the community.*"

A remark made by developer D5 complete the view of the competitiveness: "*Although there may be some competition between companies, as engineers we seek collaboration for mutual benefit. We already know any advancement will be used by everybody, that's not a problem, we get back as much as we give out.*"

5.2.3 Firm Position in the Community

The quantitative analysis revealed that firms are structured differently in the communication network among firms in the OSS community. Some firms hold central positions in the network, and they are highly involved and connected to many of the others. Meanwhile, most firms are located between the core and the periphery of the network and have a varying degree of participation. In the interviews, we asked questions to examine properties of the firm position in the community and to investigate the developer's perception of it.

Value of Firm Position

Developers from both projects replied that they perceive a position within the community as useful and valuable. From the interviews it became apparent that a central position in the community is closely related to being a core developer in most cases. Two concrete benefits mentioned by the interviewees with regards to the firm position were: (1) Easier code inclusion and thereby avoid the need of having a private code repository and (2) receiving more help from other community members. Developer D1 elaborated the value of his position within the community in the following statement:

> **Developer D1:** "*Yes, I have my position as a core developer and that enables me to make any change I want.*"
> **Researcher:** "*But do you think that it is an important position for firms to have in OSS communities?*"
> **Developer D1:** "*I don't know. Yes, because when we are doing changes, we can incorporate them into Wireshark pretty quickly. We don't have to maintain our own code base and synchronize it. We just commit code to the source and have it there. If we hadn't had the commit access as easy as I do, we could have had our own version of Wireshark and the sources, but then we would have to do more work in merging our version with the new releases of Wireshark. I don't see it as a problem, not having commit access, but it is good to have and it makes the work easier.*"

Developer D1, along with many of the other developers, stated that a position within the community may yield more help from other community members.

Developer D2: "*When you have meet people, you become friends, and then it is easier to ask for help or a favor. It is also easier to give a favor.*" The following three developer statements confirm and describe the potential value of a community position:

> "*I think it[having a position] helps a lot. I think there is a difference if, lets say, D2 asks for help, then I'll help him if I can. But if Joe Schmoe from I have never really heard of, is asking for help then my level of effort is usually lower. And part of that is because I know D2 personally, and part of that is because I know that he does a tremendous amount of work. My view is that if he needs help he deserves the help. And I think it goes the other way too, if people are more likely to help me because of the contributions I have made and they know that I have been contributing for a long time. I think it helps to have some sort of status within the community.*"
> – Developer D4

> "*I think there might be a sense of karma in that if I ask a question, other developers that know me might try to look in to it. They might see that I sent it and think: 'Well, he wouldn't be asking unless he already looked in to it', or they may remember when I helped them sometime and hopefully try to repeal that. I think they see me rather than the company, when they see an email for something that I am doing in the company.*" – Developer D3

> "*It is in the natural order of things that an authoritative figure, a prime contributor, wields more influence and gets attention more easily from others. That said we try to look at the merit of technical proposal or requests. We try to be a meritocracy.*" – Developer D5

Perception of Firm Position

Even though developers regard a position in the community as valuable, they stated that it is not something they think of, or use actively or deliberately. When asked about the firm position and if they keep it in mind when working in the community developer D1 answered: "*Not really. We are not using it actively.*" Furthermore, having a position in the community may yield more help, but there is no guarantee. Developer D2: "*No. If you receive help, it is good. If not, you just have to accept that.*"

The developers also pointed out that building up a firm position in the community is not the correct way to influence the software development. When asked if using the position within the community would yield more influence developer D6 replied:

> "*In this particular community I think the answer is mostly 'NO'. Anyone can influence the project. You just need to convince others. This is not easy - many times it is even impossible. But there is single point of decisions in the team.*"

Thus, code contribution influence the development, not status or position in the community. Developer D4 confirms this with the following statement:

> "*The thing with Wireshark for example is that if company A wants to come with an influence direction. The only way the can do it, is to actually do it. If they want Wireshark to do something new, they can come on the mailing list and suggest it and say that it is a great idea. But if nobody is interested they would have to do it themselves, it is not like they can apply pressure to make Wireshark do a certain thing.*"

Utilization of Firm Position

Developers in both OSS projects commented that the firm position is not used by any firm to dominate the OSS development. Developer D6 shared an interesting observation:

> "*Before working on Samba I used to think that big companies may have big influence in OSS project simply by 'buying' core developers. Now, that I know most of the people working on Samba, I know that this is not feasible.*"

Hence, having a position, or "buying" one, is not the way firms relate to nor influence the OSS development.

5.2.4 Work Practice Differences

OSS development has been characterized as something different than traditional software development by some researchers. Here we describe some of the work practice differences observed in the interviews.

Role Differences

The interviewed developers hold various roles in their respective firms, including: Software developer, software architect, system tester, technical customer support and requirement responsible. In OSS, formal processes as system architecture, system design and requirement specification are often omitted. Developers working within these areas are thus not using their core competencies when they participate in the OSS projects. Developer D4, normally working with requirements, said:

> "*In any open source space I am just developing. From that perspective, I just work as a developer basically. Wireshark does not really have sort of formal requirements or any of that kind of stuff.*"

Developer D6 expressed that he undertakes several different roles in the OSS community, whereas when working internally in the firm he generally holds only one role:

> "*As a Samba developer it was solely my responsibility to research/implement/test and make sure that everything is fine against Windows clients/servers. In the other hand, when working on other project in Firm X, I could request people for testing.*"

Thus, there is a difference in the notion and number of roles between firm and OSS community.

Coding Differences

Some work practice differences within coding were mentioned by the developers. The first difference, which were mentioned by developers from Wireshark and Samba, is the requirement of more consistent, pure and clean code in the OSS project. Both projects have well-defined coding standards which the community participants must follow, according to the interviewed developers. Developer D6: "*Samba has an official coding style guide and it is followed closely - otherwise your patches won't be accepted.*" Developer D4 explained how other community members expect the code to be clean and why there is such a difference between OSS and commercial software development:

> "*And honestly, one of the things I always liked about open source as opposed to closed source implementation is that in open source if somebody contributes something that you can't read and understand, the other people get grumpy. They take it out, or they say 'can you fix that, so we can understand what the heck it is doing?'. It is a lot more peer review, I guess. (...) there is a lot of interest in keeping code clean, whereas in work the primary thing is to ship it on budget, on time, you got deadlines basically. Any human when faced with the problem of do it on time or do it pretty chooses do it on time the vast majority of the time. I have seen it happen where people work on something, work on something and they are 75 % complete and they realize that the code is turning in to an ugly monster. You can't go back and rewrite it and design it away because you only got so much time until the end of the project.*"

Developer D4 remarked that there is so much code produced internally and only four or five developers to review it, whereas the OSS projects typically have more than ten active developers and thousands of developers and users looking through the code. They do not read all the code, but eventually somebody will find the poorly written code and may request a clarification, and then re-factor it and make it more readable. Developer D3 said that he would rather hold back code if it is poorly written, than contribute it to the public sources:

> "*I used to write things that wasn't good enough to be submitted. Part of a standard protocol, yes, but not good enough to submit. (...) I don't want to submit something, actually commit it, if it is rubbish. There is nothing wrong with saying 'this was kind of interesting, but not done well enough' and put it in a bug request, and if any of the developers might be interested enough they could pick it up and make it pretty, but I have never really done that.*"

When asked about internal firm coding rules, D3 stated: "*Well, maybe within some areas, but if you need to look at something that you are not familiar with it can look quite alien.*" Developer D1 stated that he consider the coding rules at his firm as stricter than the Wireshark rules.

Some of the developers expressed that they think that there are only minor coding differences between the firm and the OSS community. Developer D5:

> "*During the years we tried to converge on common coding standards. The code has a long history so it is not all homogeneous, and there are certainly differences in how various team members write code, but nowadays the differences in style for new code are minimal.*"

Despite well-defined coding rules, there are situations where these rules are relaxed in the OSS projects. Developer D1:

> "*We try to use our own coding standard rules whenever it is possible, but because of the high number of participants it is very hard to enforce the rules. It is more important to have the features, than following the coding rules.*"

The developer continued by stating that without the contributed code the software could have an unresolved bug or missing piece, which would not be more beneficial than having all the code complying to the rules.

Development Differences

An essential work practice difference is that the development in the OSS projects is driven by code and effort, rather than status and resources. Developer D5: "*(...) we try to look at the merit of technical proposal or requests. We try to be a meritocracy.*" Despite how experienced a developer is, there may be situations in an OSS project where the developer may be confronted and told that the code is not well enough written to be included in the public sources. Developer D3:

> "*I think some developers who considered themselves as very experienced probably wouldn't enjoy to be told no several times before the source is ready to be committed. There is usually not that kind of scrutiny within company source control.*"

This work practice difference is an aspect the gatekeeper developer must keep in mind when working in the different work environments.

Developers working to develop the OSS are often working individually, this is especially true for the Wireshark project. The firms typically have one developer, the gatekeeper, who is responsible for providing the other firm employees with help, features and fix bugs. When asked if the firms have more than one developer working to develop the OSS (both internally or externally) a common answer was "*no*". Large firms may have more than one developer working with the OSS, but the developers are rarely located at the same office in such cases. Developer D2:

> "*The problem is that we are located at different offices. I am working with IP telephony, and I don't know of any other developers from my office that work with Wireshark as I do. Sad, but true. We are trying as best as we can. There was an initiative some years back, that we were to gather the developers working with Wireshark in some way.*"

Some developers expressed that the software developed in the OSS projects may be slow, but the development is done properly and not rushed to make a deadline. Developer D3 described the development differences in the following statement:

> "*I like the way that Wireshark is not rushed; people don't take horrible compromises to get something working. It kind of means that it is very slow, but it seems like it is more careful, and there are more reviews of changes that go in. They are very different processes. (...) there are some really good people that have made changes and worked on it in way that you probably won't get inside a company. People would go back and fix things they don't like. You often don't get the chance to do that when you are working commercially.*"

Implications of the Differences
The differences between the way commercial firm and community-based OSS develop software introduce some implications. Developers holding a role in both the firm and the OSS project may come in a situation where they must select side in a conflict. Developer D6 explained that he believe developers would hold the OSS projects interests above the firms interests:

> "*Core DEs in Samba are very hard to manage in the way corporations work. Thus, even if some company has great deal of Samba DEs working for them, most of those DE are going to work mainly for the Samba project interests. At least this is my feeling about those people - I started to work this way myself.*"

Another implication is that developers must adapt to different work environments. Many of the differences developers encounter in OSS are outlined above. Developer D5: "*Different communities may have different requirements so each Firm X engineer simply adheres to what the upstream project rules are.*"

5.2.5 Code Contribution to Public Sources

Code contribution is an essential part of the cross-collaboration in OSS projects. The code base is the cornerstone of any software development and developers collaborate to develop it further. The interviews uncover that commercial firms have different approaches to how they develop the OSS, and how they contribute code to the public sources. All the firms represented by the interviewed developers have their own code repository where development of the OSS code is done. This repository can be public or private, depending on the firm's OSS adoption strategy. The code inclusion procedure in the public sources are quite similar in both OSS projects: Core developers contribute code to the official sources by pushing their changes to the master branch. Co-developers and others contribute code by attaching the code to a bug report and ask for a review and inclusion by a core developer.

Code Contribution Includes
Not all the code written by the firm-paid developers and stored in the firms code repository is contributed back to the public sources. The developers describe three different practices for deciding what code that should be contributed to the OSS project. In the first practice code that is standardized and open protocol implementations are contributed, and the proprietary code is kept

private. The following statements were given by the developers, describing the code contribution practice where open and standardized code is contributed, and code that is beneficial for everyone:

> "*The majority of the stuff I have written for Wireshark has been pushed up. When I wrote the M2PA dissector I sent it in to Wireshark and everybody now have it. And it makes sense because it is a standardized protocol that is used throughout the world. But you sort of draw a line in the stuff that is obscure enough to not push. The only people who should be looking at our proprietary protocol should be us. (...) Everything that I do for the core of Wireshark goes back. Part of that is because I don't want to maintain a branch and part of it is there is no point. Unless I was making incompatible changes that other people would not like, it is actually better for everybody if push it up to Wireshark.*" – Developer D4

> "*Mainly protocol dissectors for protocols used in our equipment, if the protocol is based on open protocol descriptions from 3GPP, ITU or IETF(RFC) it is considered OK to make an individual contribution to OSS. (...) My understanding is that we should give back all that is open protocol. Then we avoid maintaining it ourselves, and hopefully someone else contributes with functionality or improvements on top of our implementation.*" – Developer D2

> "*I think the criteria were the same then. Some internal management protocol for a product no one might ever buy, I didn't think that should be in Wireshark. But if in our testing we found that there where a problem with SIP or an enhancement to SIP, which everyone would benefit from, then that should go in. The criteria has always been the same really; if it is an enhancement for a protocol that lots of people would find useful, then I have never held it back from submitting it. (...) I'm doing everything I can to make it useful for people, but there are some things that you do with it that are just too specific to you. For instance, if we want to add something to a menu that only makes sense to us, then that is part of our own distribution. The sources are not secret, but there is no point of giving it to everybody as part of the standard distribution.*" – Developer D3

The statements above are made by developers from the Wireshark community. One of the Samba developers, developer D5, referred to the practice as *open core*. In the open core model the community collaborate to mainly develop the core and the open/standardized code. In Samba, developers collaborate on all the developed code. Developer D5 expressed this in his statement:

> "*The Samba community choose the GPL license because we believe cooperation is paramount. In Firm X we also do not believe in the so called 'Open Core' model. We believe in providing full access to our code.*"

The prior statement leads to the second code contribution practice which includes contributing all the developed code to the public sources. This practice is referred to as upstream development, and it is applied by all the interviewed Samba developers. Developer D5 described the upstream development approach by his firm:

> "*Firm X contributes directly to upstream communities. (...) all code is not just given back to the community, it is primarily developed in open communities. We believe strongly in the upstream first mantra. (...) In general Firm X's philosophy is to develop upstream first and then back-port changes that have been approved by the upstream community into the Firm X products. We stay very involved in the communities and try to keep the differences between Firm X packaged software and upstream software to the minimum necessary.*"

The two other Samba developers confirmed this practice. Developer D6: "*All code is public. Either in official Samba repo or published in other public repos.*" Developer D7: "*Yes, I have an internal branch of Samba software (CIFS Linux kernel client) but it's open from the web and patches from here are on their ways to upstream.*"

The third and last contribution practice is not employed by any of the interviewed developers, however, we observed its presence across all the interviews: Some firms adopt the OSS and use it extensively, without contributing any code back to the official sources. Firms employing this practice are working on private channels and private code repositories, and may be collaborating directly with other firms to develop the OSS. These firms establish a commensalistic relationship to the OSS project, which implies that one of the two entities is benefiting from the relationship without affecting the other. The firms get the OSS and give nothing back. Some of the developers are not satisfied with this arrangement, especially developer D1 and D2. Developer D4 expresses another view:

> "*Yeah, but as far as I know there are millions of people using Wireshark and I consider Wireshark as quite fortunate to have as many people giving back as it they are. It is a fairly active community and so forth. If people don't want to give back, I am okay with that.*"

Code Contribution Process

The code contribution to the public sources is characterized as a process driven by personal incentive and need by some developers, and it is a process which lacks formal strategy and policy from the firm. Developers stated that their firms do not control or supervise what code that is contributed to the OSS projects, and there are no strategies for how the code contribution shall: (1) Utilize other community developers, (2) assist to develop the OSS further and (3) benefit the firm. Developer D2: "*Firm X does not practice particular control over what is contributed back to GPL Wireshark.*" Even though firms use the OSS extensively, the connection between firm and OSS community is typically maintained by a single developer which ensure to contribute code to the project based on personal incentive and a desire to keep the community healthy.

In the projects where firms do not develop upstream (Wireshark), the individual developer makes the decision of what to contribute. When asked if it is hard to decide what to contribute, developer D4 replied:

> "*I haven't had too much trouble with that. The standardized protocols are not a though on my mind, I just contributed them because I knew our company was using the protocols and a lot of other companies were soon going to be using the protocols. And not only the companies that are developing the stack, like us, but also the users would want to deal with these protocols. My view of it really was anything that is in wide use should be pushed. Wide use is relative, and telecom world is quite small compared to web world, but it is a judgment call whether it is general use or not, I guess.*"

Hence, the gatekeeper developer is often responsible for making decisions towards the OSS community. Firms developing the OSS upstream have the code contribution to public sources as part of the overall firm strategy.

Code Contribution Benefits

There are several benefits for firms that contribute code to the public sources. One benefit, which have been mentioned above already, is avoiding maintenance and merging problems associated with combining public and private code. This is expressed by developer D2:

> "*The problem is that if you are to make a change in the core, and you want to keep it private, you will have to fork the project and maintain it yourself. (...) That is a balancing act; if you give back your great function, perhaps there is someone that helps to fix bugs or build it further with functionality you did not think of and such. I believe, in the general case, that you gain more from contributing to the development, that retaining your code from the community.*"

Another evident benefit is that the contributed code may be furthered developed by other developers, utilizing their experience and effort for "free". This is evident in the following statement by Developer D1:

> "*When I post my patches and changes some other people might find bugs in it or enhance it further, making it more and better. So I make some small things, and other people come in and make more upon that. Then I may change some bits, which in the end gives us a better product.*"

Developers stated that the most effective way to influence the development is to contribute code, which might be beneficial to the commercial firms in their effort to build new features they are in need of.

Some of the developers gave an interesting comment when discussing the commercial firm participation in OSS. By contributing code an establishment of a commercial solution is effectively prevented. Developer D3:

> "*The way I think of it is that if we contribute to it, and they contribute to it, then there is no market for protocol analyzers. No one sell protocol analyzers and you work on your product. I wouldn't want it to be my job to work on a protocol analyzer that is a rival to Wireshark. Working in Wireshark is good, and I don't want our company to compete on who that has the best protocol analyzer. (...) I would rather want Wireshark to be free, protocol analyzers were available to anybody, and we compete on higher things. (...) No, and as I said, it is almost as by supporting Wireshark you are avoiding a market from happening.*"

Developer D3 provided a genuine benefit from contributing code:

> " *(...) we benefit from it as much as anybody. And you can't deny that if we add it first, then we know it will work well with the way we work. We don't want to make a network with someone else's. It will probably always work slightly better with ours.*"

Accordingly, the software will be best suited for the firm who contributes or is part of writing the code.

Code Contribution Impediments

There are some cases where code contribution is hindered. Legal, sensitive or authorization issues are mentioned as one of these cases. Developer D4:

> "*The stuff we don't send in is stuff that is not of interest to anybody except us. (...) And the other part is that I don't think the company would be thrilled by a publication of these protocols. In order to push those things to Wireshark I would need to get authorization.*"

Contribution may be hindered by poorly written code, which would probably not be accepted (as discussed earlier). Developers sometimes make quick hacks to fix a problem and despite the usefulness of this code, they would not contribute it. Going back to rewrite it may take too long time, and developers refrain from contributing code that does not fulfill the coding standards. Developer D3 stated:

> " *... one thing I used to do a lot in the early days was to really hack something to answer that question. Maybe spend half a day creating something that would do some dirty printfs at the end of a test or something, and so I used to write things that wasn't good enough to be submitted. Part of a standard protocol, yes, but not good enough to submit.*"

Some of the developers said that they do not contribute all the code to the public sources because it is too implementation specific code, which may come in the way of other users. Developer D3:

> "*It is always a bit difficult, because I am working on LTE, certain parts of LTE, and what I have committed to the Wireshark sources is a lot for LTE. But I always try to keep the implementation specific we have done in our product. I don't want them to make it less useful*

> *for someone else. So the parts that I would not commit tend to be the parts that would be confusing or wrong for other people. I do as much as I possibly can, that applies to everybody that would be interested in seeing MACPD (MAC protocol dissector) in use.*"

5.2.6 Additional Issues

Here we briefly elaborate on some additional issues observed in the interviews, which were interesting but outside our research context.

Strategy with OSS Several of the developers mentioned that their participation in the OSS project is not an official part of their job. Developer D4: "*I mean, it is not an official part of my job, but a lot of the developers, testers and the customer support people use Wireshark extensively.*" Accordingly, commercial firms use and rely heavily on the software from these projects, but lack formal strategies for: (1) How developers shall participate and develop the OSS, (2) what code that shall be contributed back to official sources and (3) how to maintain the OSS knowledge base within in the firm.

Compliance with OSS license All the firms have their own code repository where they develop the OSS. Many of the firms keep the repository private as they have proprietary code they use integrated with the OSS. This is not a violation of the GPL license, according to developer D4: " *(...) with GPL any of my changes that I make need to be available to the people who get the binaries, and it is because the people who get the binaries are internal users.*"

Low contributing firms From the quantitative and qualitative analysis it is apparent that some firms use the OSS extensively, but contribute very little back to the OSS project in terms of participation and code. Developers suggest that these firms have commensalistic relationship with the OSS community; the firm benefits from the OSS, however the OSS does not benefit from the firm. A commensalistic relationship normally evolves to a parasitic relationship, where one of the parties suffer from the relationship. In this case it will be the OSS project, which may die out because of the lacking collaboration among the community members.

> "*I see surprisingly little participation/contribution from other telecom companies, for instance Firm X and Firm Y. Keeping the protocol decoders up to date should be beneficial to them. (...) Some of the developers are working over Gmail and such, and then we can't keep track of what they are working on really.*" – Developer D2

CHAPTER 6

EVALUATION AND DISCUSSION

This chapter evaluate and discuss the findings of the study. First, the results presented in chapter 5 will be used to answer the research questions. Second, the results as a whole are discussed. Third and last, the reliability and threats to validity are evaluated.

6.1 Answering the Research Questions

The objective of this thesis was to investigate how commercial firms collaborate across organizational boundaries in community-based OSS projects, and to explore the influence of their collaboration on evolution and outcome in the projects. To understand this phenomenon we created two main research questions with subquestions and conducted a case study of Wireshark and Samba. In the following sections we use the quantitative and qualitative results to attempt to answer the research questions presented in section 1.4. Each main research question is answered as a general summary of the more detailed answers given in the following subquestions. Table 6.1 provides an overview of where to find the results used to answer the different research questions.

Table 6.1: Answers to the research questions.

Question	Subquestion	Status
RQ1	RQ1.1	Exploratory analysis, mainly subsection 5.1.3
	RQ1.2	Exploratory analysis, mainly subsection 5.1.4
	RQ1.3	Descriptive analysis, section 5.2
	RQ1.4	Descriptive and exploratory analysis
RQ2	RQ2.1	Descriptive analysis, primarily subsection 5.2.2
	RQ2.2	Descriptive analysis, primarily subsection 5.2.1
	RQ2.3	Descriptive analysis

6.2 Organizational Cross-Collaboration

Research Question 1 - Main Question:

> *RQ1: How do commercial firms collaborate across organizational boundaries in community-based OSS projects?*

The collaboration across organizational boundaries is a composition of several elements. Answering the research question involve viewing the collaboration from two perspectives: (1) How firms as entities cross-collaborate and (2) how individual firm-paid developers cross-collaborate. We elaborate on each of these perspectives and complete the answer with a short summary.

Commercial firms participating in community-based OSS projects collaborate in various ways across the organizational boundaries. Most of the cross-collaboration occurs within the OSS project's public communication channels, typically mailing list and bug tracking system. These are established communication channels, which facilitate asynchronous communication and transparency of the development. The commercial firms value these traits and use the channels extensively, generating large proportions of the activity in the various sources. In addition to the public channels, there is a widespread use of private communication channels among the firms when cross-collaborating. Most firms emphasize that private channels are only used occasionally and normally in situations where they want quick feedback or want to develop more efficiently. For instance, private channels are used when asking other firms for help or guidance, or when discussing sensitive or legal issues.

By evaluating how commercial firms participate in the OSS projects, we observe three distinct cross-collaboration approaches: Upstream, open-core and direct. Firms participating in an upstream approach contribute all the code they develop to the OSS project's public sources, and collaborate exclusively within the OSS project to develop the software. The firms participating in an open-core collaboration approach contribute code that is (1) related to the core of the OSS and (2) code that is regarded as open and/or standardized, and collaborate within the community to develop the code they contribute. Open-core firms typically have private repositories where they have code related to the OSS which is proprietary and thus retained from the public sources. Firms adopting the direct collaboration approach develop the OSS directly with other firms using public and/or private communication channels and code repositories. The open-core and direct collaboration approaches can be combined.

How the individual firm-paid developers cross-collaborate is an interesting finding. From the results it is apparent that all firms have a developer participating in the OSS community acting as intermediary between the firm and the community. This developer is essentially operating as a gatekeeper and is managing the boundary between the firm and the OSS project. The responsibilities of the gatekeeper is influenced by the firm's collaboration approach, but typically includes management of code, communication and/or bug issues flowing between the two entities.

In summary, commercial firms collaborate across organizational boundaries mainly in two ways: Through the public communication channels in the OSS project or through direct communication channels. The collaboration is determined by the firm's participation approach; upstream, open-core or direct. The collaboration is often driven by convenience, rather than being planned and formalized as in traditional software development. Firms with the same interests and needs collaborate within the community to solve the problems at hand. The problems which the community developers collaborate on, are typically problems which are beneficial for everyone to be solved. A gatekeeper developer, acting as an intermediary between the firm and the community, is typically the developer that engage in collaboration with other firm-paid developers in the project. The gatekeeper is a key element in the cross-collaboration as he is familiar with the project's culture, how to relate to other community members and has extensive knowledge of the overall code base.

6.2.1 Level of participation

> *RQ1.1: How much do firms participate in terms of entries in the developer mailing list, bug tracking system and code repository?*

Answering this and the following questions required knowledge of the participating developers in the OSS community to be able to separate firm-paid developers from volunteering developers. This was achieved by applying identification techniques on information located in the OSS projects public sources. Several commercial firms were identified in the OSS projects. The results of the quantitative analysis provide good insight in how much firms participate in terms of development and maintenance effort. The results are a composition of the three investigated data sources: Mailing list, bug tracking system and code repository. We elaborate on firm participation in each source:

Mailing list The collection of identified commercial firms constitute a large fraction of the activity in the mailing list in both projects, approximately 27 % in Wireshark and 47 % in Samba. However, the individual firm contribution range from low to very high. The top three firms account for 60 % and 56 % of the mails in Wireshark and Samba, respectively.

Bug tracking system The contribution from commercial firms in the bug tracking system conform to the same pattern as in the mailing list; significant, but highly diversified. In total, the bug activity by commercial firms constitute 39 % in Wireshark and 66 % in Samba.

Code repository A small subset of the identified commercial firms commit code to the projects. There might be several reasons that this, for instance the fact that it is a privilege to have commit rights (a privilege earned by hard work and dedication in the community). However, the collection of firms that do contribute code account for a substantial fraction of the commits. In Wireshark, 55 % of the total number of commits in the selected time frame are done by firms. In Samba, astonishing 82 % of the commits are done by firms.

Overall, commercial firms participate in the OSS projects in all investigated channels and the aggregated firm participation accounts for a substantial part of the activity in the communities. However, we see a resemblance spanning the data sources; the majority of the activity is generated by a small subset of the firms, and that the remaining firms participate with little to none. The subset of firms can be further divided in three: (1) A small number of firms that are very active and account for large proportions of the development, (2) some firms that are moderately active and participate at a significant level, and (3) multiple firms that are little or very little active in the data sources. Naturally, the most active firms are the same across the different sources. Interestingly, the activity by firms in Samba is significantly larger than the activity by firms in Wireshark, especially in the code repository. This is further addressed in section 6.4.

Crowston's onion model can be used to describe the level of participation and roles firms undertake in community-based OSS projects [40]. An illustration of this is given in Figure 6.1. Firms are distributed in the model according to Crowston's characterization: The most active firms are located at the core (denoted as 1) of the onion taking the place as core developers, which typically is a small number. Surrounding the core are the moderately active firms acting as co-developers (denoted as 2). Outside the co-developers is the layer where the low active firms are located holding the role as active users (denoted as 3), which there are many of. In the periphery of the model, the firms which are very little active or not active at all resident. These are the passive users in the community and is denoted as 4 in the model.

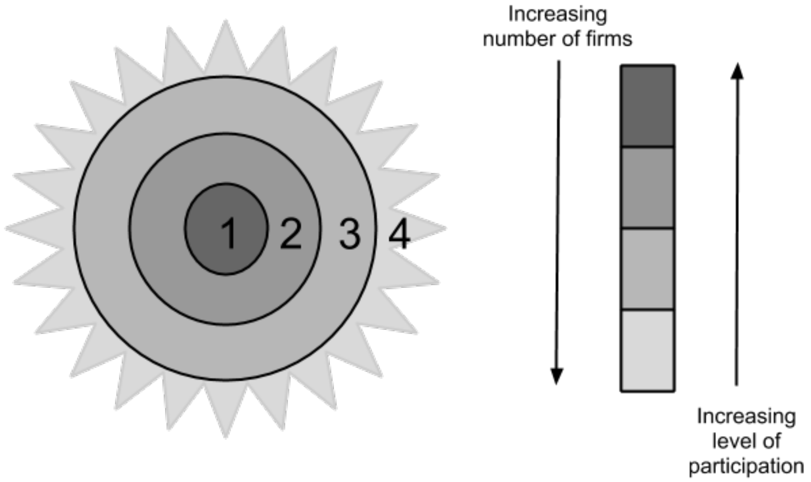


Figure 6.1: Level of firm participation in community-based OSS projects.

6.2.2 Collaboration Observed from Public Informalisms

> *RQ1.2: How do firms collaborate within an OSS community observed from the public software informalisms?*

By applying social network analysis to the discussions in the mailing list and bug tracking system, we were able to extract and visualize the structure of the commercial firms in the OSS communities. With the social network data and the visualization, we managed to identify key aspects of the firm cross-collaboration observed from the public software informalisms. Four social networks were presented in the result chapter (see subsection 5.1.4), two from each project displaying the networks in the mailing list and the bug tracking system. The first impression was the varying degree of collaboration among the firms, similar to the firm participation as described above: Some firms are highly connected and have links to almost all firms displayed in the social network. Whereas most firms are moderately or low connected to other firms.

By conducting a closer investigation of the networks, we found several interesting properties of the commercial firm cross-collaboration in Wireshark and Samba. We see that the visual structure across the networks are very similar, and that the structure can be divided in three layers. The layers are named by their position in the network. Following is a short description of each of layer:

Center The center layer is located at the core of the network. It consists of nodes that are highly connected to the others in the network (high centrality degree).

Middle The middle layer is surrounding the center layer and consists of nodes that are typically connected to more than one other node and is “close” to the center nodes.

Periphery The periphery layer is surrounding the middle layer and consists of nodes that only have a few connections (in most cases only one) to others in the network (low centrality degree).

The layer descriptions emphasize the importance of the centrality degree. The social network numbers presented in tables 5.3, 5.4, 5.5 and 5.6 clearly display this. The numbers also demonstrate that the position in the social network is not purely based on the number of entries. A firm with few entries can have a central position. This is exemplified by Trihedral in Table 5.3, which is ranked as the fourth firm with regard to the centrality degree and have 222 mail entries, while the firm ranked as fifth, Thales, have 548 mail entries. This is because Trihedral, despite the relative low number of mails, has participated in discussions with many other firms and is thus connected to them in the social network. Determining whether it is better to have a more central position from collaborating little many firms, or to have a less central position but collaborating closely with a few firms is not evident from our finding, but may be a purposeful question to pursue in further research.

The social networks from Wireshark with layers applied are illustrated in Figure 6.2 and Figure 6.3, and from Samba in Figure 6.4 and Figure 6.5. The red is the center layer, yellow is the middle layer and the rest is the periphery layer. Note that the applied layers on top of the social networks are only intended as illustrations, no mathematical algorithms or tools were used to determine where the layers starts and ends. We observe that the size and number of nodes in each layer are noticeable different. The center layer is the smallest, and consist only of a few nodes. The middle layer is slightly bigger than the center layer in terms of size and number of nodes. Last, the periphery layer is the biggest and contains the majority of the nodes. Note that firms who are not connected to any other firms are excluded from the social network.

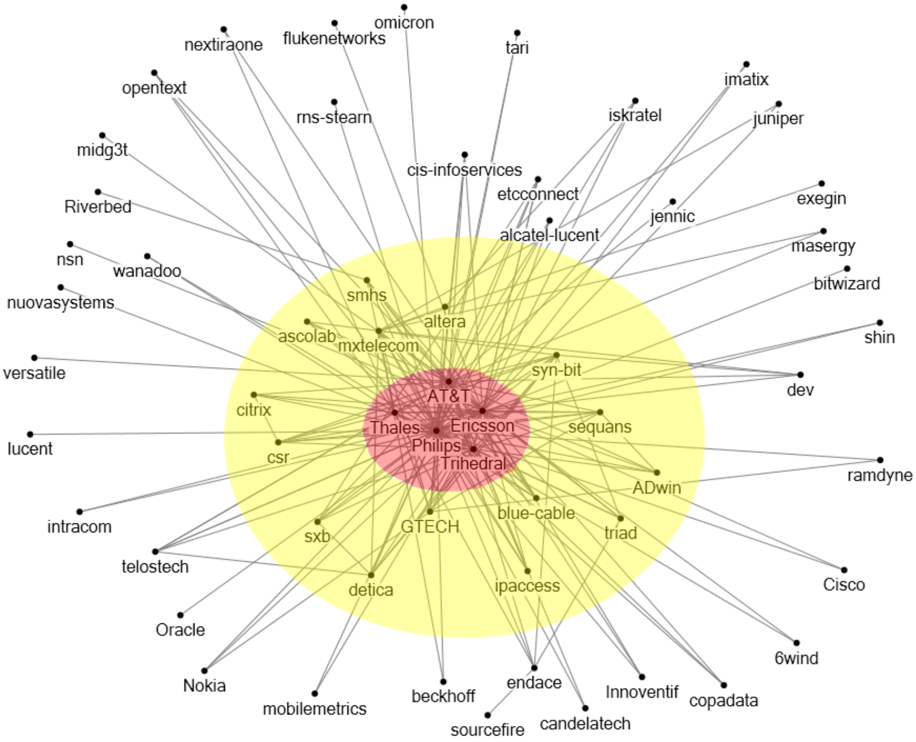


Figure 6.2: Social network from the Wireshark mailing list with layers.

To summarize, commercial firms are layered in three layers: Center, middle and periphery. Which layer a firm is located in, gives insight in how connected the firm is with the other firms in the network. A firm in the center layer is highly connected, which implies that it collaborate with many other firms. We see that the distribution of firms in the social networks correlate, both in terms of structure and content. The firms located in the center layer in one network are

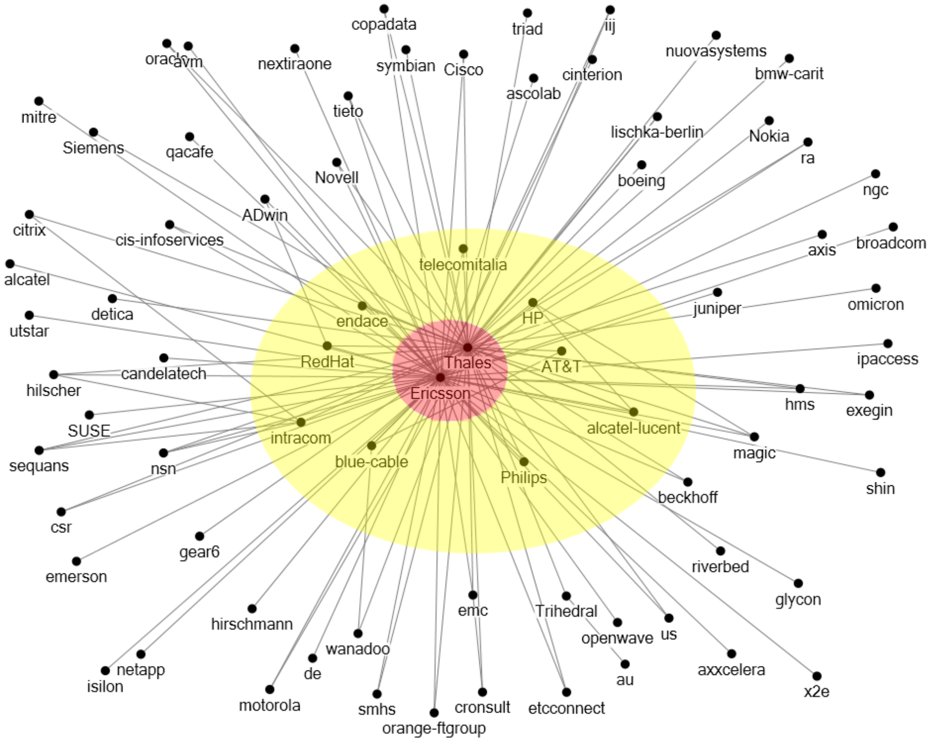


Figure 6.3: Social network from the Wireshark bug tracking system with layers.

likely to be in or close to the center in the other network. From the networks, we see that the center layer consists only of a small fraction of the firms (two to six), the middle layer consists of some firms (10-14), and the periphery layer consists of a high number of firms (approximately 60 firms). The results of the social network analysis are very interesting. The structure of the firms and the layers we identified in the collaboration network can be compared with Crowston's onion-model. The core developers are the center layer, followed by co-developers as middle layer and active users as periphery layer. We did not include the firms that did not have any connection to other firms in the social network, however, they surround the periphery layer, and resemble the passive users in the onion model. It is evident that commercial firms are present in all roles of the OSS community. We see a clear distinction between firms in terms of contributions and connectivity. There is a high number of firms that are located in the periphery layer and do not contribute to the project, neither in terms of development or maintenance effort, or collaboration between firms.

We see that a small fraction of firms are located in the center of the network. These firms contribute substantially to the project and are highly connected with the other firms in the network. These firms play an important role in the

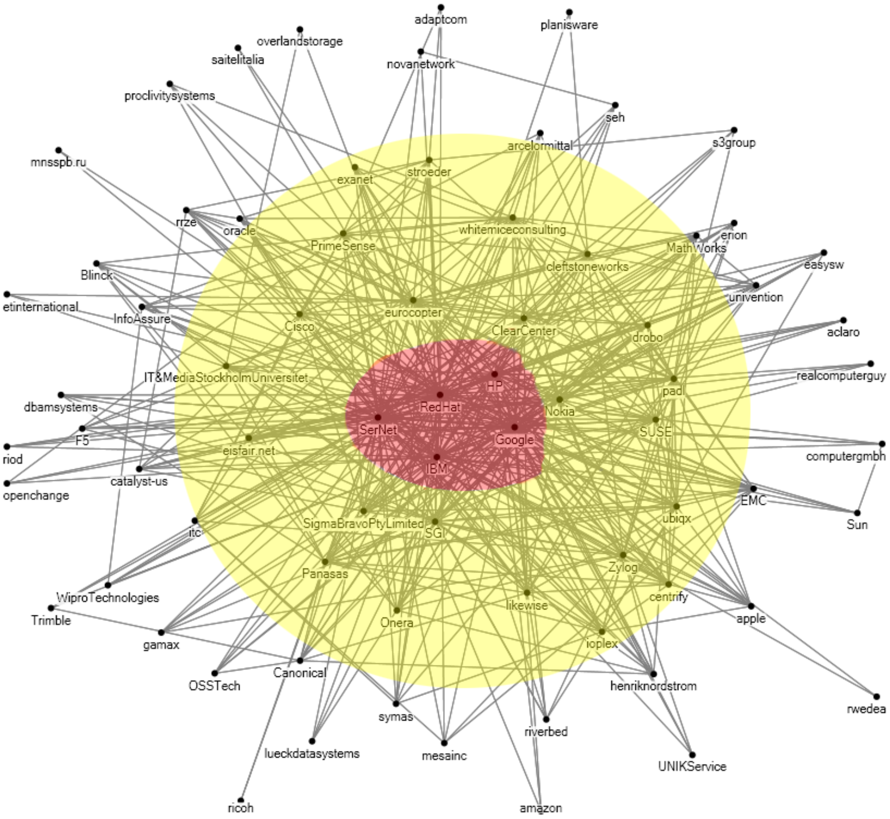


Figure 6.4: Social network from the Samba mailing list with layers.

community collaboration as they create indirect connections between the firms located in the middle and periphery level. The interactions by the center layer firms can be very valuable, as they:

- Connect the various firms with different knowledge, thus leveraging the social capital of the involved participants of the OSS project. This can happen in two different ways: (1) the center layer firms acquire knowledge from the periphery layer firms, and apply it in interactions with other firms (creating an indirect connection for knowledge sharing); (2) the center layer firms point out which firms that, for instance have the same problem or feature request, and then these firms can work together (creating an direct connection for knowledge sharing and collaboration).

- Share experiences, knowledge, best practices, and maintain an awareness of the other developers and what they are currently working on. The latter is crucial for avoiding duplicate programming, which can be regarded as wasted effort.

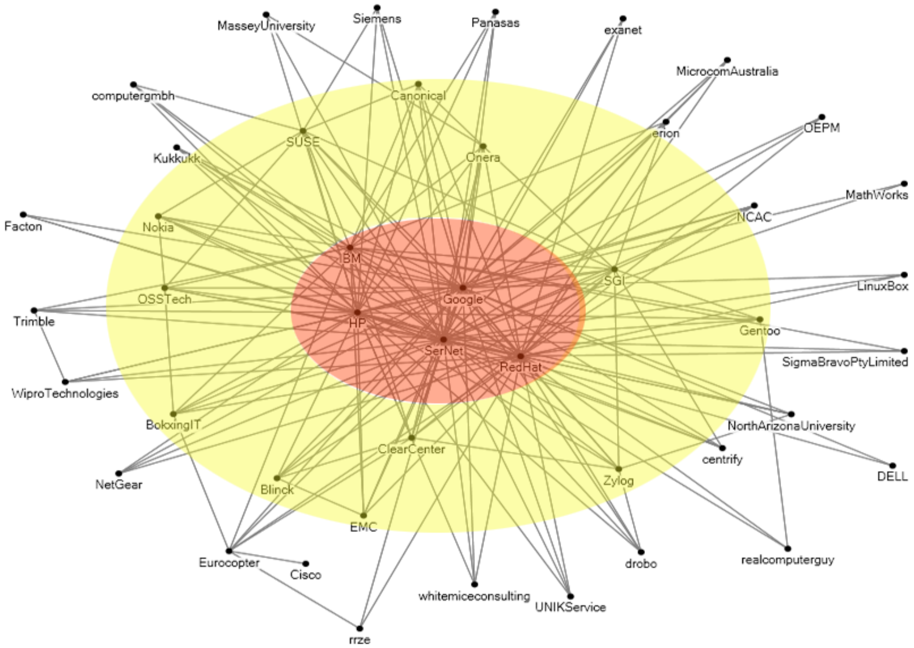


Figure 6.5: Social network from the Samba bug tracking system with layers.

- Promote and attract participation and contributions from other firms. When other firms observe the participation of center layer firms, they can infer that the development is healthy and the community accepts commercial firm participation.

To answer the question, commercial firms collaborate at a highly varying degree within OSS communities observed from the public software informalisms. A small set of the firms collaborate with almost all the other participating firms and effectively undertake the role as hub nodes in the collaboration network. Some firms collaborate with a subset of the participating firms in the community, sometimes acting as an intermediary between firms. The majority of firms collaborate with few other firms, typically one to three.

6.2.3 Collaboration Observed from the Participating Developer's Perspective

> *RQ1.3: How do firms collaborate within an OSS community observed from the participating developer's perspective?*

Analysis of the interview transcripts provide answers to how firms collaborate observed from the participating developer's perspective. Firm-paid developers collaborate with other firm-paid developers in various ways to develop the OSS.

We summarize the cross-collaboration in four practices. The practices are not mutually exclusive, however, developers tend to adhere to one.

Open-Core Collaboration
Several firms work within the OSS community in an open-core collaboration approach to develop the OSS further. The open-core collaboration approach involves contributing code that is: (1) Related to the core of the OSS and (2) code that is open and standardized. Firms participating in an open-core approach typically have a private code repository where the firm-paid developers work on the OSS code base and contribute code to the project in a backporting fashion. That is, first modifying the OSS internally and then committing the changes to the official sources. Not all the code that is written in the firm is contributed back to the OSS project. Some of the code is regarded as proprietary and is retained in the firm's private code repository. Reasons for retaining code are mainly that the firm's internal code is too implementation specific, others would be required to have the same environment to make use of the code, or that there are legal and/or authorization issues with sharing the code. Firms do not retain code because it is secret, gives a competitive advantage or is poorly written.

Firms collaborating in an open-core fashion manage their flow of information, code and communication through a gatekeeper developer (the gatekeeper is described and evaluated in subsection 6.2.4). The gatekeeper is the developer representing the firm in the OSS community and collaborate with other community members by participating in discussions and contributing code to the project. The open-core collaboration approach is characterized as casual and is not driven by any formal strategies or alliances between the participating firms.

Upstream Collaboration
The upstream collaboration approach is similar to the open-core collaboration approach in most aspects: Firms collaborate within the community to develop the OSS and they contribute code to the official sources. A major difference is that firms working upstream are committed to contribute all the code they write to the official sources, and they backport changes from the official source to their internal code repository, as opposed to open-core where the process is done the other way around. There is also a difference in how developers within the firm participate in the development. Each firm-paid developer in an upstream firm is working within the OSS community, and write code and contribute directly to the project's official sources. Hence, there is no notion of a gatekeeper managing the code flow between an upstream firm and the OSS project, but other aspects of the collaboration is channeled through a gatekeeper. The official firm communication and bug issues are examples of these other aspects.

The upstream collaboration approach is slightly more formal than the open-core approach. The firms have committed to contribute all code and work within the community, but there are no apparent formal strategies or alliances for collaboration among the firms. Collaboration occurs when firms are interested in the same features or have coinciding ideas.

Direct and/or Private Collaboration

Firms can collaborate directly with other firms to develop the OSS, either through private or public communication channels. In addition, they may collaborate on the software development in private repositories, as opposed to the public code sources. A direct collaboration approach may be preferred over the open-core or upstream approaches as it is faster, more efficient and may prevent problems related to legal or security sensitive issues. The developed code can be contributed to the OSS project's official sources, but is dependent on the firm's attitude towards contribution to the project. Many firms do not bother use effort to clean the code and make it modular enough to be accepted into the main branch of the OSS project. Collaboration in a direct and/or private fashion is more formal than the two prior collaboration practices. Firms form alliances to develop the OSS and have specific goals in mind that they want to fulfill by the collaboration.

It is perfectly possible for firms to have direct collaboration with some firms, and in addition collaborate open-core or upstream with other firms in the OSS community. The direct collaboration can also take place within the community, and thus appear as open-core or upstream. The difference is that the developers have agreed to collaborate. Code developed in direct collaboration seems to conform to similar contribution practices as the open-core approach; open and standardized code is contributed back to the project.

No Collaboration

None of the firms represented by interviewed developers employ the "no collaboration" practice, however, the developers state that some firms use the OSS extensively and do not participate nor contribute back to the official sources. These firms are resident in the OSS community, but do not take part in any collaboration in the project.

To summarize, firms mainly collaborate within an OSS community in three ways: Open-core, upstream or directly. The approach determines the level of formality and code contribution. The most casual collaboration is the open-core where firms collaborate in the project without any formal alliances and participate in the parts they find interesting. Upstream collaboration involves a commitment to contribute all code and participate in the project with both the project's and the firm's interests in mind. Direct collaboration allows firms to share development cost by initiating a formal collaboration of the OSS development, either inside or outside the OSS project.

6.2.4 Boundary Between OSS and Firm

> *RQ1.4: How is the boundary between OSS community and firm managed?*

Most firms have a developer internally they regard as the go-to person for the specific OSS. This person is typically responsible for training the other firm developers in how to use the software and provide updates, technical support, new features, fix bugs, maintain the code and, if needed, collaborate within the OSS community. In effect, this person act as an intermediary between the firm and the OSS community. We have named this developer the *gatekeeper developer* or just the *gatekeeper*, describing the role this go-to developer undertake. The gatekeeper is managing the boundary between firm and OSS community by having two roles; one as a developer in the given firm and one as a developer in the OSS community. The gatekeeper must thus adopt to different work environments which compromise differences in how developers work, collaborate, communicate and coordinate tasks. There are several reasons why firms have a gatekeeper managing the organizational boundary. Managing the OSS code is one of them. Firms benefit from incorporating their code into the official sources by avoiding maintenance and merging problems with the code when new versions of the OSS is released. Additionally, the code can be further developed and improved by other community developers, and the gatekeeper can influence the development by contributing code and participating when other developers work on it. This way the firms can tailor the OSS functionality to their needs and share the cost of the development. Another reason, is the value of having a position in the OSS community. A position may provide the firm (through the gatekeeper) commit access to the official sources, which entail that the firm developed code can be included directly into the project's source code tree without having to go through a screening process. A community position may also yield more help from other community members when guidance or help is needed.

The gatekeeper perception is confirmed by the quantitative and the qualitative results. First, in the quantitative analysis we observe that most firms only have one developer or a small number of developers participating in the OSS community, which is true for the highly active developers acting as hub nodes as well (Google, Thales and AT&T). Second, the qualitative analysis reveals that all firms have a developer they regard as a gatekeeper. The role of the gatekeeper, as reported above, is influenced by the firm's collaboration approach. In most cases, the gatekeeper has responsibility for managing the code, communication and/or bug issues that cross the organization's boundaries. Furthermore, the gatekeeper is highly involved in the decision making towards the OSS project, a point that is part of the discussion in RQ2.3.

6.3 Firm Awareness in OSS

Research Question 2 - Main Question:

> *RQ2: How influential is firm awareness in community-based OSS projects?*

The presence and influence of firm awareness in community-based OSS projects are important aspects of the complex software development process. Firm awareness is established mainly through developer conferences and interactions among the developers in the OSS community. There is a widespread awareness of firms and their developers in the OSS projects we investigated. In spite of this awareness, firm-paid developers do not perceive other firm-paid developers as firm representatives, but rather as individuals or volunteers.

The firm awareness can be influential in two ways. First, the firm awareness can be engaging as knowledge of other firms in the community may trigger participation and collaboration. This is part of utilizing the unknown workforce in an OSS community, also termed opensourcing by Ågerfalk and Fitzgerald [52]. Firms with similar interests can collaborate within the OSS community to share the implementation tasks and developer expertise, thus reducing development time and cost. In addition, relationships to other firms may be established this way, which can be useful in subsequent development. Second, the firm awareness may have a preventive effect. Firms may resist to participate and/or collaborate because of competition in the market. Helping other firms is not desirable seen from a business perspective, especially if there is not a mutual benefit.

All these aspects of the firm awareness are influential parts of the decision making commercial firms have to consider when participating in an OSS project. The results indicate that the firm awareness is experienced as more influential towards better collaboration and more efficient work, rather than impacting the cross-collaboration in a negative and less productive fashion. This finding is in line with conclusions made by [60], which state that openness will allow firms to utilize the benefits of open innovation more successfully.

6.3.1 Developer Awareness and Perception

> *RQ2.1: How do firm-paid developers become aware of other firm-paid developers, and how do they perceive them?*

Firm-paid developers become aware of other firm-paid developers in community-based OSS projects in various ways, including but not limited to: Firm sponsored developer conferences, discussions in the public communication channels, and formed relationships in collaboration. The ways are not mutually exclusive and a combination is probably the best awareness builder. A common denominator is that the firm awareness is built from firms having the same interests and which share their knowledge for achieving better results in the development.

Developer conferences are powerful means for establishing social networks among the OSS community developers. Developers in OSS development are often geographically dispersed and rarely meet face-to-face. By hosting an annual or biannual conference, developers can meet, form and maintain connections to other developers. In turn, these connections can be utilized in the on-going OSS development. For instance, a developer may know who to contact for a specific question based on expertise or trust, or who to ask for specific artifacts like logs or pieces of code. Developers perceive this as a highly valuable and strategic element in the collaboration taking place in the OSS projects. Firms share the developer's perception and thus normally cover the developer's expenses associated with the conference, and in some cases firms sponsor the conferences directly. The following list of reasons for attending Sharkfest, found at the

OSS project's website[1], captures several important aspects the commercial firm awareness in this thesis. Pay special attention to the last reason, which is outlined in bold text:

1. Receive Wireshark immersion training in a relaxed, informal setting
2. Interact with core developers in a 4-day hackathon
3. Influence Wireshark project direction
4. Network with peers and industry experts to exchange best practices for network and protocol analysis, network security, and troubleshooting
5. Participate in hands-on labs with world-class veteran network consultants and trainers
6. Three keynote addresses by illustrious networking industry influencers
7. WCNA Boot Camp with Laura Chappell: Pre-certification training held on campus available to SHARKFEST attendees at a discounted rate
8. **Forge new partnerships and business relationships with large enterprises, federal and local government agencies, educational institutions, network consultants, and Riverbed strategic partners**

Discussions in the public channels contribute to the firm awareness in the OSS community. The discussions are typically conversations of how to implement new features, solve problems or general discussions of relevant topics. Firm-paid developers participate when they feel they have technical contributions to make or they want to influence the development on behalf of their firm. In the discussions the firm association may be disclosed directly by the developer expressing that he works for a given firm or indirectly by stating firm association in an email signature or using a firm associated email.

Relationships formed in collaboration can be a continuation of discussions in the public channels. Firm-paid developers sharing a goal or need can collaborate and collectively share the cost of the software development, and in the process establish a relationship. The collaboration involves fulfilling requirements to the OSS made by the developer's firm, and thus firm association along with the actual requirements are normally clarified upfront.

The second part of the question is how firm-paid developers perceive other firm-paid developers. The findings suggests that developers have the following three perceptions of other developers: (1) Partners and/or friends, rather than competitors, (2) useful resources, and (3) individuals and/or volunteers. The first perception is especially interesting. Commercial firms are typically business competitors and strive to create and maintain persistent competitive advantages in the market. Collaboration in such a setting must incorporate a condition of mutual usefulness and equality among the firm-paid developers, and among the firms. This implies that if one firm contribute code which is beneficial for the others, an expectation that the others will return the favor is created. Core

[1] http://sharkfest.wireshark.org/, Retrieved: 2013-05-24

developers work this way, and are thus partners in developing the OSS collectively. The second perception is closely related to the first. Perceiving other firm-paid developers as useful resources coincide with the idea of a community workforce, which can be utilized to reduce development time and cost. The first and second perception is closely related to the opensourcing concept [52]. Knowledge and experience, implementation and design support, and expertise in certain areas are some of the useful benefits other firm-paid developers may provide.

The perception of firm-paid developers as individuals and/or volunteers demonstrate the equality among all members of the OSS community. The majority of firm-paid developers make no distinction between firm-paid and volunteering developers when collaborating in the OSS community. Developers explain this perception by stating that few or no developers are paid to participate in the OSS project. Hence, no developers are perceived as firm representatives.

6.3.2 Firm Awareness and Collaboration

> *RQ2.2: How does firm awareness influence the collaboration practices?*

Collaboration in OSS development is a complex process and is influenced by multiple factors. Answering how firm awareness influences the collaboration practices is very difficult, especially how it influences the collaboration directly. Consequently, we attempt to answer the question by giving suggestions of how the firm awareness may (indirectly) influence the collaboration practices.

Firm awareness includes knowledge about which firms that participate in the project, where in the code they are working, what they are using the code for and what their plans are. As mentioned earlier, firms are typically competitors in the market and contributions to the OSS project which may exclusively benefit the competitors is not desirable seen from a business perspective. We observe two distinct ways the firm awareness may influence the collaboration practices:

1. Firm awareness may influence the collaboration in a positive manner by creating incentives for a more active and open collaboration. Knowing which firms that participate and how they participate can have a reassuring effect. Commercial firms are driven by profit, and if they participate in OSS it is because they perceive it as valuable and worthwhile their investment. An OSS project with many participating firms is likely to persist for a long time. This is all part of assessing the health of the OSS community, an important process according to Crowston and Howison [40]. Another aspect of the firm awareness is the perception of developers as partners and useful resources. Firm-paid developers are educated and trained professionals. This may influence firms to collaborate more openly within the community so that they can draw from other professionals in making the code better and more useful. Observing what firms are currently working on may also prosper collaboration; engaging in an early stage of the development imply more influence and more suitable code for the

participating firms. Collaboration in an OSS community may also create links between commercial firms, which may be used in other settings in the future.

2. Firm awareness may influence the collaboration in a negative manner and have a preventive effect on the collaboration. Knowing that the developers asking for help in the mailing list are employees at a competitor firm, may prevent collaboration from taking place. The awareness of firms that use the OSS extensively but only participate little or nothing, may influence other firms to do the same, creating a vicious circle. No firm wants to give and give, but not get anything in return for their effort. In such a case, firms rather develop the OSS internally. As reported above, firm-paid developers regard other OSS community members as volunteers or individuals regardless of their firm association. Our findings suggest that the firm awareness only have minor negative influences on the collaboration.

6.3.3 Firm Awareness and Decision Making

> *RQ2.3: How does firm awareness influence the decision making towards community-based OSS projects?*

From the prior subquestions we know that (1) there is firm awareness in community-based OSS projects and (2) the collaboration practices may be influenced (at least indirectly) by the firm awareness. How these properties may influence the decision making towards community-based OSS projects can be evaluated with regards to code contribution and collaboration.

Code contribution

Determining whether to contribute code to the official sources of an OSS project or not is part of the decision making a firm has to consider. Furthermore, deciding which pieces of the code to contribute is another. There are several benefits from contributing the code, including but not limited to: Avoiding maintenance and merging problems associated with combining public and private code, allowing other developers to develop the code further and making it better, and influencing the development towards the given firm's desire. Additionally, participation and contributions to an OSS project may prevent the establishment of a commercial market solution. However, there are some implications that firms have to consider. There may be legal or security sensitive issues hindering the contribution of code. Too implementation specific code is also a problem, which may come in the way of other developers, thus should not be contributed. All these considerations must be taken into account when adopting OSS, and this way the firm awareness influences the decision making to become a balancing act.

The firm awareness plays an important role in the decision making. Learning by observing the behavior of the best is applicable in most cases. Decisions related to code contribution may be influenced by how the core development firms in the OSS community contribute. If firms experience the other firms in the community as partners and see that their contributed code is developed further, they are more likely to contribute their code.

Collaboration
Another part of the decision making is deciding whether to collaborate with others to develop the OSS (i.e. the OSS community), who to collaborate with and how the collaboration shall be performed. Although collaboration within an OSS community is typically informal and not planned, there are matters that have to be decided upon. For instance, when there is a new post in the mailing list, the developer has to decide whether to engage in the discussion with the others or not (essentially collaborating with them). The awareness of other firms in this aspect may prosper the collaboration. Firm-paid developers with similar needs and interests can collaborate and draw on each other's abilities. Knowing that a developer works for a certain firm, and that he can provide certain code artifacts also influences the collaboration. Establishing relationships to such valuable developers through collaboration is key. There is a strong desire to return favors and honor developer's position by assisting them when they need help in community-based OSS communities.

Many commercial firms adopt OSS, but do not participate nor contribute to the OSS project. Some of these firms collaborate directly with others to develop the OSS further, with or without participating in the OSS community. How to perform the collaboration is an aspect firms have to decide. As described above, the collaboration can take place within the OSS community using public or private communication channels, or outside the community using private channels and private code repositories.

6.4 Case Comparison

In this section a case comparison is performed, describing commonalities and differences between the two community-based OSS projects studied in this thesis. The aim of the study was to perform an analytic generalization of the research topic by conducting case studies of two community-based OSS projects. The involved projects had to be typical instances of OSS projects, and they had to be quite similar to allow conclusions to be drawn based on data collected across the cases. A careful selection procedure ensured this, however, the similarities used as criteria were mostly superficial. Having performed the case studies we are now able to describe the commonalities and differences between the two cases with regards to how commercial firms collaborate across organizational boundaries.

6.4.1 Commonalities

There are several commonalities between the Wireshark project and the Samba project. We present each commonality with a detailed description:

Firm participation Both the projects have a high number of firms participating in the development in the OSS community. The firms are structured similarly in the communities with regard to Crowston's onion model; a few firms are at the core layer, some at the middle layer, and many firms at the outer layer and the periphery.

Gatekeeper Firms in both projects have a gatekeeper developer which act as an intermediary between the firm and the OSS community. The gatekeeper holds a vital role in the collaboration across organizational boundaries within the OSS communities. The responsibility of the gatekeeper varies (see the gatekeeper bullet in the case differences).

Developer perception and attitude The firm-paid developers in Wireshark and Samba have similar perceptions and attitudes towards other firm-paid developers. The competitiveness among participating firms is minuscule and the firm-paid developers work closely together to create a better product which benefits all involved parties. The firm awareness is mostly positively influential.

Communication channels The OSS project's public software informalisms are the main communication channels used in the collaboration among the participating firms. Use of private channels is widespread in both projects, however, the firm-paid developers emphasize that they mainly use the mailing list and the bug tracking system because of the transparency and openness.

Firm awareness The firm awareness in Wireshark and Samba is mainly established at annual firm sponsored developer conferences. Firm-paid developers meet and form relationships to other firm-paid developers, which can be utilized in collaboration within the communities. Firm-paid developers in both projects underline that what firm others work for is not important, and developers are not seen as firm representatives.

6.4.2 Differences

The cases have some differences. A major difference between firms in Wireshark and Samba is how the firms collaborate within the community.

Collaboration approach The firms in Wireshark are participating in the OSS project in an open-core approach. They have private code repositories where they develop proprietary and public code. The public code is pushed to the official sources subsequently. The firms in Samba, on the other hand, participate in an upstream approach. They contribute all code to the public sources and back-port changes to their internal code repository. This is illustrated in Figure 6.6.

Firm activity The level of activity by the firms in the two OSS projects are vastly different. In Samba, the firm-paid developers are much more active with regard to discussions and code contribution. This may be a result of the previous bullet; firms in Samba mainly work within the community to develop the OSS. In Wireshark only the code related to the core or open/standardized is contributed.

Response time The average response time by the commercial firms in the projects are vastly different, especially in the mailing list. Firm-paid developers in the Wireshark mailing list have an average response time between 10,5 and 118,2 hours (see Figure 5.12), whereas in Samba the numbers range between 1,4 and 4,4 hours (see Figure 5.16).

Gatekeeper responsibility The responsibility of the gatekeeper developer in Wireshark and Samba is different. The difference in collaboration approaches among firms is already mentioned. Developers in upstream firms (Samba) contribute code directly to the project's public sources, and thus the gatekeeper is not responsible for the code flow between the firm and the project. A gatekeeper in an upstream firm is mainly responsible for channeling the (formal firm) communication and bug issues. A gatekeeper in an open-core firm is responsible for code, communication and bug issues.

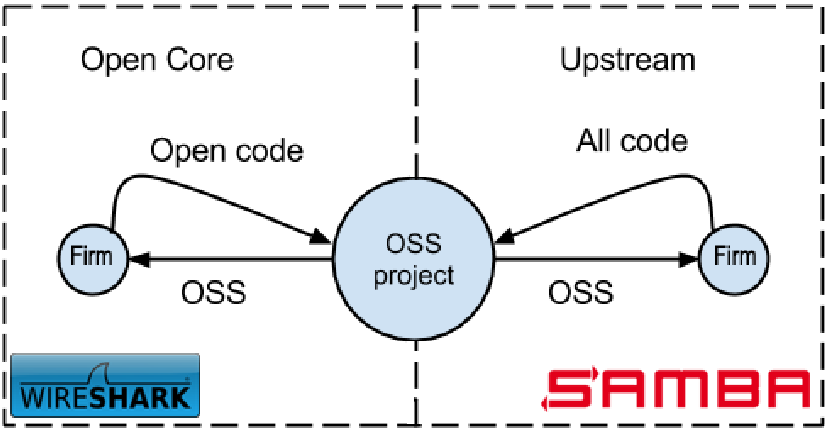


Figure 6.6: Main difference between the two OSS projects.

6.5 Conceptual Model

A conceptual model is derived from the synthesized findings of this study, see Figure 6.7. The model is a two-dimensional view displaying an OSS community layer and a commercial firm layer, and present a comprehensive illustration of how the various ways firms engage in organizational cross-collaboration in community-based OSS projects. The model compromise three vital parts: OSS community layer, commercial firm layer and organizational boundary. Each of these are described closer in the following sections. Lastly, a conceptualization of the organizational cross-collaboration is given.

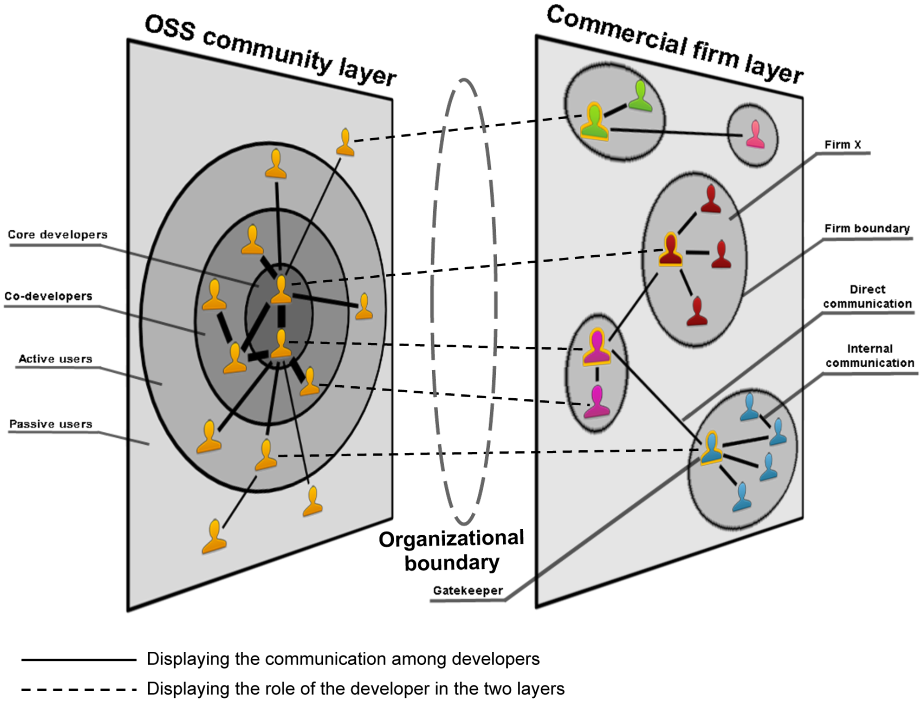


Figure 6.7: Conceptual model

6.5.1 OSS Community Layer

The OSS community layer is mainly based on the quantitative analysis, displaying how firms are structured in the collaboration network, and prior OSS research, especially Crowston's research and his onion model describing the structure of a healthy OSS community [40]. Developers are distributed in the OSS community layer according to their level of participation and connectivity to other developers in the community. There are four roles in this layer (cf. Crowston's onion model): Core developers, co-developers, active users and passive users. Edges connect the developers in the network, displaying their collaboration. The weight of the edge is determined by the connectivity between the given developers. Developers close to the core have higher connectivity, whereas developers in the periphery have low connectivity. All developers in this layer are given the same color (yellow), demonstrating their equality. This conform to the perception firm-paid developers have of other developers regarding them as individuals and volunteers, not firm-representatives or competitors.

6.5.2 Commercial Firm Layer

The commercial firm layer is built from the quantitative and qualitative analysis, and illustrate the structure of commercial firms involved in the OSS projects and their boundaries. Developers and their respective firm are outlined by a firm boundary. Edges within the firms display internal communication among the developers, and edges between firms display direct communication across the firm boundaries. Each developer is highlighted in a distinct color describing which firm he is representing. In the commercial firm layer, the gatekeeper developer is drawn with a yellow silhouette in addition to the firm's color. The yellow silhouette corresponds with the color developers in the OSS community layer have, which represent the two roles a gatekeeper holds.

6.5.3 Organizational Boundary

The organizational boundary is the boundary separating the OSS community layer and commercial firm layer. In the model, the organizational boundary is illustrated by a circle of dashed lines. When firm-paid developers collaborate with other firm-paid developers in the OSS community, they are working outside their organization's boundary. The crossing of this boundary has several implications: Developers must adjust to different work and collaboration practices, and if their firm does not have a formal strategy or policy towards the participation, the developers must make decisions on behalf of the firm. The dashed lines display the role of the same developer in the two layers, operating across the organizational boundary.

6.5.4 Conceptualization of Firm Cross-Collaboration

The model encapsulate various interesting aspects of the organizational cross-collaboration, which are elaborated on in the following bullets:

- Most firms in the model have one developer connecting the firm and the community. This developer is the gatekeeper, providing services to his firm as described earlier. There are two exceptions in the model. First, one firm consisting of two developers, which both have dashed lines. This is an illustration of a firm collaborating upstream in the community, where both developers have central positions in the community. There is still a notion of a gatekeeper, however, the code the firm produces can be contributed to the project by both of the developers. Second, one of the firms consisting of one developer (upper right corner), is not participating in the OSS community. This is an illustration of a firm working, for instance, as a consultant company. The consultant company may develop the OSS on behalf of the firm it is connected to (green). The code may be proprietary and not contributed to the OSS for that reason, or it is possible that the code is contributed by the firm buying the consultancy.

- The two roles of the gatekeeper developer are apparent. One in the firm as the go-to person for the given OSS, and the other one in the community as part of the onion model. These environments may be significantly different in terms of how developers work and collaborate. This is an important aspect to consider when adopting and participating in an OSS project. This is highlighted in prior research [59].

- The location of a firm-paid developer in the OSS community illustrate the position the developer has in the community. Developers close to the core are highly connected and collaborate with many other firm-paid developers. The firm-paid developers do not perceive the position in the community as especially valuable, but there is a greater chance to receive help and guidance from the other community members. In addition, the position may be very influential towards how the other firms participate and collaborate.

6.6 Recommendations for Practitioners

Here we present a set of recommendations to commercial firms that want to utilize and participate in OSS projects based on the findings in this study:

- Establishing a position or status within an OSS community is not necessary in order to influence the development. Proper contributions and effort towards the desired change is more valuable.

- Firms participating in OSS development do not perceive other firms as competitors, rather as partners and helpful resources. This could be leveraged into a collaboration effort by firms, however it require intimate understanding of the project and respect of other developer's time and commitment.

- Firms that want new features or find bug must be prepared to do most of the work themselves. When new functionality or bugs becomes common interest, the other community members will help. It is possible to persuade others to implement features that are good, however, it is both laborious and time-consuming .

- It is more likely to collaborate and communicate with other firms in the mailing list than in the bug tracking system.

- Not all communication goes through the public channels in OSS projects. Legal and security sensitive issues commonly go through private or closed channels because of their nature. In addition, firm-paid developers occasionally contact other firm-paid developers directly for more efficient development or to avoid asking stupid questions in public. Learning how to use the channels properly is a good recommendation to firms interested in participating in OSS development.

- Participation in OSS project's developer conferences is an important measure for firm-paid developers to become aware of other firms participating in the development and build networks across organizational boundaries. In time, this can/may result in better and more efficient development, benefiting both OSS community and commercial firms.

6.7 Validity and Reliability

The validity of a study is related to the trustworthiness of the results and how representative the results are for the population of interest [72]. A crucial part of any research is to assess the quality of both a study and its results. In the following sections we evaluate the validity of this study according to guidelines proposed by [73], which are tailored for case study research in software engineering. The work in this thesis was conducted with the guidance of two experienced researchers, which is an important contributor to higher validity. Ideas, rationales, processes, results and conclusions were all discussed in plenary.

6.7.1 Construct Validity

Construct validity refers to whether observations and measurements actually represent the concept being studied. It can be seen as the legitimacy of, for instance, a test or scale in measuring a theoretical concept that it is intended to evaluate. An example of this could be to what extent an IQ questionnaire actually measures a person's intelligence.

We see two threats to the construct validity of this study. The first threat is the assumption that developers collaborate when they participate in the same discussion in the public software informalisms, which was an essential part of the quantitative analysis. The threat was addressed and minimized by presenting and discussing the assumption with the interviewed firm-paid developers, which confirmed it. The second threat is the use of numbers acquired in the quantitative analysis and to some degree in the qualitative analysis, which were presented in tables and charts. However, the numbers were only used to evaluate and determine differences between the cases. There were no use of statistical tests or statistical significant relationships based on the numbers.

6.7.2 Internal Validity

Internal validity concerns whether cause-effect or causal relationships discovered in a study are truly caused by the involved variables and not affected or biased by other variables. For instance, when a researcher examines whether one variable affects an investigated variable, there is a risk that the investigated variable is affected by a third and potentially unknown variable. If the researcher is not aware of the third variable, and does not understand its influence on the investigated variable, there is a threat to the internal validity.

The data used in the quantitative analysis was collected from the public software informalisms in the two OSS projects. We assume that the entries in the informalims are archived correctly with regard to name, content and structure (i.e. mails related to each other are correctly linked together). A threat to the internal validity is that the entries are not handled properly by the Python scripts written to analyze the collaboration. An improper handling may create relationships that do not exist and result in cascading errors that may affect the whole analysis. Several measures were taken to prevent this, including: Debugging and test executions of the Python scripts, handling of special characters and multiple comprehensive reviews of the results (see section 4.6 for some of the challenges met during the analysis).

The identification of firm-paid developers is a threat to the internal validity of this study. Information whether a developer is participating in the OSS projects on behalf of a firm or not was collected from online sources, and lack or misinterpretation of the information may lead to a wrong classification. We addressed this threat by applying an identification approach successfully utilized in similar research. Several sources of information were assessed during the quantitative analysis for assuring a correct classification. Subsequently, the interviewees in the qualitative analysis elaborated on many of the firm-paid developers, thus confirming that our approach and results of the identification were good. This is part of the method triangulation for achieving higher validity, which was employed in most of the research in this thesis. In addition, some of the interviewed developers evaluated the social networks and acknowledged the presence of firms and their respective developers. We believe the majority of the firm-paid developer were correctly classified.

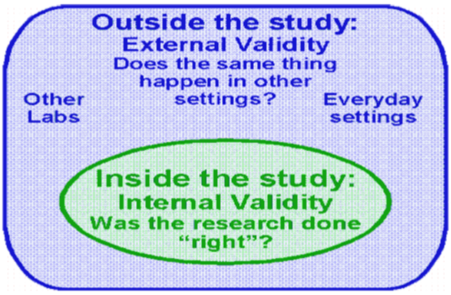


Figure 6.8: The relationship between internal and external validity.

There are two major internal validity threats related to interviews: (1) Data collector characteristics and (2) data collector bias. The first is related to characteristics of the interviewer that might influence the data, for instance, gender, age and ethnicity. A threat that is hard to eliminate. The second is related to unconscious distortion of the data during the data collection process. This includes phrasing questions differently or asking questions which leads to the answer the researcher wants. This validity threat was minimized by (1) being aware of the threat before conducting the interviews, (2) asking the questions the same way to all interviewees and (3) acting professionally.

6.7.3 External Validity

External validity refers to the extent to which results of a study is possible to generalize or extend to other cases. Accordingly, high external validity ensure that the same results can be predicted for subsequent analysis to and across individuals, settings and times. Evaluating the external validity is important for determining whether results are of relevance for other cases. The relationship between internal and external validity is displayed in Figure 6.8.

Yin state that the intention of a case study is to enable an analytic generalization, not a statistical, and thus the results can be extended to cases with similar characteristics [19]. The two cases in this study were selected because of their coinciding characteristics, and because they are typical instances of community-based OSS projects. We selected the cases with the intention to ensure high external validity, allowing the findings to be generalized and applicable for other OSS projects. An illustration of how the cases are similar and different is given in section 6.4, which give an indication of how generalizable the results are across OSS projects.

6.7.4 Reliability

Reliability is concerned with the stability and consistency of the data and analysis performed. High reliability involves reducing errors and biases by, for instance, providing clearly defined data collection procedures and traceable steps in the data analysis. Hypothetically, if another researcher conducted the exact same study, the obtained results and the conclusions should be the same [19].

The work conducted in this thesis is transparent and all involved phases are well documented. The documentation includes a complete pre-study, research design, result and discussion part. The data sources are available at the OSS projects public software informalisms, and are also made public by the researcher[2]. How the data collection and analysis were preformed are described in detail, and how we arrived at the conclusions is both traceable and apparent. It should be perfectly viable to perform a replication of this study by another researcher to verify our results.

[2] folk.ntnu.no/snarby/datasources

CHAPTER 7 CONCLUSION AND FURTHER WORK

This chapter summarizes the research, presents the main contributions and propose subjects for further research.

7.1 Summary

The objective of this study was to investigate how commercial firms participate in community-based OSS projects, examine the organizational cross-collaboration among the firms, and explore the influence of the collaboration on community-based OSS project's outcome and evolution. A two-phased case study approach was adopted allowing to study Wireshark and Samba, two typical instances of successful and on-going community-based OSS projects, using exploratory and descriptive analysis. By investigating the collaboration found in the public software informalisms and conducting interviews with firm-paid developers, we were able to answer: (1) How commercial firms participate and contribute, (2) how commercial firms collaborate across organizational boundaries, and (3) how the collaboration affects the development.

This study illustrate the complexity and importance of the commercial firm collaboration across organizational boundaries as a research subject. Furthermore, it provides new understanding of how commercial firms and OSS communities interact and relate to each other on a collective level. The results are primarily related to tangible elements observed in both the quantitative and qualitative analysis, but also reveal other aspects which may be interesting to pursue in further research. A conclusion would be that this is a topic which will increase in relevance as OSS adoption among commercial firms increase. For firms, understanding how to most effectively participate and fulfill individual firm objectives by leveraging the OSS community workforce are key elements in achieving the benefits observed in successful OSS projects. For OSS projects, facilitating and attracting commercial firms to commit to the project will ensure a healthy and sustainable software development environment, which in the end will benefit all participants.

7.2 Main Contributions

The contributions from this study can be divided in three main parts:

1. Empirical data from two community-based OSS projects providing knowledge related to how commercial firms participate and collaborate across organizational boundaries in community-based OSS projects.

2. A conceptual model describing the commercial firm's collaboration patterns across organizational boundaries in community-based OSS projects. The model captures several aspects of the collaboration and present these in an intuitively manner. The most noteworthy property of the model is the connection between the OSS community layer and the commercial firm layer expressing how a firm-paid developer holds two roles and act as an intermediary between the the firm and community. The model adopts and extends Crowston's onion model to describe how the firms are structured in the OSS community layer and displays the firm's flow of information across organizational boundaries.

3. A set of recommendations to commercial firms that want to participate and utilize community-based OSS projects. The recommendations provide insights which can be used by firms to, for instance, adapt their OSS adoption strategy. Firm awareness and firm perception are key elements of the recommendations.

7.3 Further Research

An increasing number of commercial firms adopt and utilize OSS as part of their business, a trend that is likely to persist. Hence, establishing an understanding of how commercial firms operate, influence and affect OSS projects in various aspects can be beneficial and valuable, both to researchers and practitioners. There is clearly a need for more research related to the collaboration across organizational boundaries and research evaluating how commercial firms operates as entities in an OSS community. The latter referring to a proposal to investigate how the combined effort of firms and the firm-paid developers impact the OSS projects, as opposed to the main body of OSS research where the main focus has been the individual level. Several new questions for further research can be proposed based on the results in this thesis:

- *How does the firm collaboration practices influence the organizational cross-collaboration in community-based OSS projects?* We observe that the average response time in the mailing list in Samba, where essential firms participate upstream, are considerable lower among firms compared to Wireshark. Does this imply that the cross-collaboration among firms working upstream are more important, and that the firms are working closer and more interactive?

- *Are there any correlation or relationship between position in the community and the ability to influence the development?* This topic is addressed in this thesis, however, it is exclusively based on the perception obtained from the firm-paid developers. It is still a very interesting question. There is a need for collecting empirical data from the software informalisms to confirm or disprove the developer's perception.

- *How do firm-paid developers perceive the collaboration with other firm-paid developers compared to collaboration with volunteering developers?* We observe direct collaboration between firms in the OSS projects. Would a direct collaboration with a volunteering developer be equally relevant? Can a firm's participation and dedication be more trusted as the firm is dependent on the OSS, whereas a volunteering developer typically participate because it is fun and educational. Does the motivational aspect have a crucial role in the cross-collaboration?

- *How does the management in commercial firms perceive the participation and contributions to the OSS community?* From the interviews it is apparent that the firm-paid developers operate by themselves and make decisions on behalf of the firm. If management was more included, would there be introduced major changes in the firm participation and contribution?

- *How valuable is the connectivity to other firms in the OSS communtiy compared to the level of collaboration?* The social network analysis reveals that firms can have high connectivity (high centrality degree) despite having a relative low number entries in the public software informalisms. Is it better to have a more central position from collaborating little with many firms, or is it better to have a less central position but collaborating closely with a few firms?

In addition to the questions, there are many other research opportunities. Replicating and verifying the results from this study by conducting more case studies is one of them. We see two applicable approaches:

1. Extending the work in this thesis by conducting case studies of similar OSS projects using the same research approach and analysis. This would provide a wider foundation for comparison and enable greater generalization across cases. A potential approach would be to explore firm-based OSS projects using the same approach to see differences and/or similarities between community-based and firm-based OSS projects with regard to the topic.

2. Conducting a set of case studies exploring other properties of the firm participation and the organizational cross-collaboration in other OSS projects using the findings of this study as a basis. This would provide more empirical data on the subject, and could possibly be used to validate the findings of this thesis. Collecting more empirical data is very important, especially in subjects where there exist little or no knowledge. This will in turn enable researchers to draw a more generalized picture of how commercial firms participate and collaborate in OSS projects, and possible

form new fields of research. The knowledge might be of great interest for researchers and practitioners.

Furthermore, a research opportunity is to use other research methods to verify and explain our results. This could involve, for instance, observations or questionnaires exploring similar properties as this study. Firm participation and cross-collaboration in community-based OSS projects have been investigated in an exploratory and descriptive approach in this study, identifying key concepts and providing an overview of the topic, which is recognized as a field lacking research. Collecting data by observing a small set of firm-paid developers while using the identified concepts and overview can provide detailed insights and explain elements of the topic, which possibly cannot be explored properly by other research methods. Questionnaires based on the properties in this thesis can be used to collect data widely across the population of firm-paid developers, and thus provide a comprehensive and reliable overview of the topic.

Part V

Glossary, Bibliography and Appendices

GLOSSARY

Asynchronous communication Communication between entities where the flow is not concurrent (intermittently rather than steady stream)

Category killer A product or brand that has established such a sustainable competitive advantage that competitors find it practically impossible to compete for profit in the same market

Collaboration A process where two or more entities work together on a task with the objective to realize shared goals

Commercial firm An organization operating in business driven by commercial incentives

Community-based OSS project An OSS project founded and managed by a distributed group of individuals who do not share the same employer

Construct validity Refers to the degree to which a test measures what it claims or is intended to measure

Copyleft The practice of using copyright to ensure the same distribution and use rights for all subsequent users of a given software

External validity Refers to which extent the findings of a study can be generalized to other situations, settings or people

Firm awareness Knowledge about which firms that participate in the project, where in the code they are working, what they are using the code for and what their plans are

Gatekeeper Person acting as an intermediary between a firm and an OSS community responsible for the information flow between the two entities

Global software development Development of software across geographically distributed locations within or across organizations

Group awareness Knowledge about who is on the project, where in the code they are working, what they are doing, and what their plans are

Interaction Two or more entities that have a relation and an effect upon each other

Internal validity Refers to which extent the casual conclusion of a study reflects the actual phenomenon studied

Open-core collaboration Collaboration within and outside the OSS community and sharing of open code

Organizational boundary An imaginary boundary surrounding an organization that distinguish the organization and its members from the external population

Participation The act of taking part or sharing something

Population The collection of all entities of the same group or category

Qualitative study Concerned with information as text either written or spoken

Quantitative study Concerned with information as numbers quantifying a relationship or comparing two or more groups

Reliability Concerned with the stability and consistency of the data and analysis performed

Software informalism Information resources and artifacts that participants use to describe and coordinate what is taking place in the development. Typically informal and narrative, and publicly available to persons interested in joining or browsing the project's development history

Upstream collaboration Collaboration exclusively within the OSS community and sharing of all developed code

BIBLIOGRAPHY

[1] B. Fitzgerald, "The transformation of open source software," *Mis Quarterly*, pp. 587–598, 2006.

[2] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *2007 Future of Software Engineering*, pp. 188–198, IEEE Computer Society, 2007.

[3] M. M. Rahman and G. Ruhe, "Resource allocation and activity scheduling: bug fixing perspective," tech. rep., Technical Report, Software engineering decision support laboratory, University of Calgary, 2010.

[4] C. Daffara, "How many stable and active libre software projects?." http://robertogaloppini.net/2007/08/23/estimating-the-number-of-active-and-stable-floss-projects/, 2007.

[5] Ø. Hauge, C. Ayala, and R. Conradi, "Adoption of open source software in software-intensive organizations–a systematic literature review," *Information and Software Technology*, vol. 52, no. 11, pp. 1133–1154, 2010.

[6] K. R. Lakhani and R. Wolf, *Why hackers do what they do: Understanding motivation and effort in free/open source software projects.* MIT Press, 2005.

[7] A. Hars and S. Ou, "Working for free? motivations for participating in open-source projects," *International Journal of Electronic Commerce*, vol. 6, pp. 25–40, 2002.

[8] A. Aksulu and M. Wade, "A comprehensive review and synthesis of open source research," *Journal of the Association for Information Systems*, vol. 11, no. 11, pp. 576–656, 2010.

[9] M. Höst and A. Oručević-Alagić, "A systematic review of research on open source software in commercial software product development," *Information and Software Technology*, vol. 53, no. 6, pp. 616–624, 2011.

[10] A. Bonaccorsi and C. Rossi, “Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business,” *Knowledge, Technology & Policy*, vol. 18, no. 4, pp. 40–64, 2006.

[11] S. Ludvigsen, A. Lund, and I. Rasmussen, *Learning across sites: New tools, infrastructures and practices.* Taylor & Francis US, 2011.

[12] D. Šmite and Z. Galviņa, “Socio-technical congruence sabotaged by a hidden onshore outsourcing relationship: lessons learned from an empirical study,” *Product-Focused Software Process Improvement*, pp. 190–202, 2012.

[13] R. Grewal, G. L. Lilien, and G. Mallapragada, “Location, location, location: How network embeddedness affects project success in open source systems,” *Management Science*, vol. 52, no. 7, pp. 1043–1056, 2006.

[14] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social coding in github: transparency and collaboration in an open software repository,” in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1277–1286, ACM, 2012.

[15] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, and J. Howison, “Self-organization of teams for free/libre open source software development,” *Information and software technology*, vol. 49, no. 6, pp. 564–575, 2007.

[16] P. Giuri, M. Ploner, F. Rullani, and S. Torrisi, “Skills, division of labor and performance in collective inventions: Evidence from open source software,” *International Journal of Industrial Organization*, vol. 28, no. 1, pp. 54–68, 2010.

[17] Y. Kamei, S. Matsumoto, H. Maeshima, Y. Onishi, M. Ohira, and K.-i. Matsumoto, “Analysis of coordination between developers and users in the apache community,” *Open Source Development, Communities and Quality*, pp. 81–92, 2008.

[18] G. P. Pisano and R. Verganti, “Which kind of collaboration is right for you,” *Harvard Business Review*, vol. 86, no. 12, pp. 78–86, 2008.

[19] R. Yin, *Case study research: Design and methods*, vol. 5. Sage Publications, Incorporated, 2008.

[20] J. West and S. O’Mahony, “Contrasting community building in sponsored and community founded open source projects,” in *System Sciences, 2005. HICSS’05. Proceedings of the 38th Annual Hawaii International Conference on*, pp. 196c–196c, IEEE, 2005.

[21] GNU Operating System, “What is free software?.” `http://www.gnu.org/philosophy/free-sw.en.html`, 2001. Retrieved: 2013-02-16.

[22] Open Source Initiative, "The open source definition." `http://opensource.org/osd`. Retrieved: 2013-02-16.

[23] R. M. Stallman *et al.*, "The gnu project." `http://www.gnu.org/gnu/thegnuproject.html`, 1998. Retrieved: 2013-02-16.

[24] Ø. Hauge, *Adoption of Open Source Software in Software-Intensive Industry.* PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2010.

[25] G. Von Krogh and E. Von Hippel, "Special issue on open source software development," *Research Policy*, vol. 32, no. 7, pp. 1149–1157, 2003.

[26] B. Grad, "A personal recollection: Ibm's unbundling of software and services," *Annals of the History of Computing, IEEE*, vol. 24, no. 1, pp. 64–71, 2002.

[27] R. M. Stallman, "Initial announcement of the gnu project." `http://www.gnu.org/gnu/initial-announcement.html`, 1983. Retrieved: 2013-02-16.

[28] E. S. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, 1999.

[29] M. Tiemann, "History of the OSI." `http://opensource.org/history`, 2006. Retrieved: 2013-02-16.

[30] E. Raymond, "The cathedral and the bazaar-musings on linux and open source by an accidental revolutionary (revised edition)," 2001.

[31] K. Finley, "Github has surpassed sourceforge and google code in popularity." http://readwrite.com/2011/06/02/github-has-passed-sourceforge, 2011. Retrieved: 2013-05-27.

[32] J. Lindman, M. Rossi, and A. Paajanen, "Matching open source software licenses with corresponding business models," *IEEE software*, vol. 28, no. 4, pp. 31–35, 2011.

[33] M. Olson, "Dual licensing," *Open Sources*, vol. 2, pp. 71–90, 2005.

[34] GNU Operating System, "What is copyleft?." http://www.gnu.org/copyleft/. Retrieved: 2013-02-25.

[35] M. Mustonen, "Copyleft–the economics of linux and other open source software," *Information Economics and Policy*, vol. 15, no. 1, pp. 99–121, 2003.

[36] J. West and S. Gallagher, "Key challenges of open innovation: lessons from open source software," *San Jose State College of Business, mimeo*, 2004.

[37] W. Scacchi, "Free/open source software development: Recent research results and methods," *Advances in Computers*, vol. 69, pp. 243–295, 2007.

[38] B. Fitzgerald, “Open source software: Lessons from and for software engineering,” *Computer*, vol. 44, no. 10, pp. 25–30, 2011.

[39] K. Crowston, K. Wei, J. Howison, and A. Wiggins, “Free/libre open-source software development: What we know and what we do not know,” *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, p. 7, 2012.

[40] K. Crowston and J. Howison, “Assessing the health of open source communities,” *Computer*, vol. 39, no. 5, pp. 89–91, 2006.

[41] P. Giuri, F. Rullani, and S. Torrisi, “Explaining leadership in virtual teams: The case of open source software,” *Information Economics and Policy*, vol. 20, no. 4, pp. 305–315, 2008.

[42] S.-W. Chou and M.-Y. He, “The factors that affect the performance of open source software development–the perspective of social capital and expertise integration,” *Information Systems Journal*, vol. 21, no. 2, pp. 195–219, 2010.

[43] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, “Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects,” *Management science*, vol. 52, no. 7, pp. 984–999, 2006.

[44] E. Shihab, Z. M. Jiang, and A. E. Hassan, “Studying the use of developer irc meetings in open source projects,” in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pp. 147–156, IEEE, 2009.

[45] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social coding in github: transparency and collaboration in an open software repository,” in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1277–1286, ACM, 2012.

[46] N. Lertpittayapoom, S. Paul, and P. Mykytyn Jr, “A theoretical perspective on effective interorganizational knowledge,” in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 187b–187b, IEEE, 2007.

[47] J. A. Espinosa, J. N. Cummings, J. M. Wilson, and B. M. Pearce, “Team boundary issues across multiple global firms,” *Journal of Management Information Systems*, vol. 19, no. 4, pp. 157–190, 2003.

[48] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, “Does distributed development affect software quality?: an empirical case study of windows vista,” *Communications of the ACM*, vol. 52, no. 8, pp. 85–93, 2009.

[49] M. Cataldo and J. D. Herbsleb, “Factors leading to integration failures in global feature-oriented development: an empirical analysis,” in *Software Engineering (ICSE), 2011 33rd International Conference on*, pp. 161–170, IEEE, 2011.

[50] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pp. 72–81, ACM, 2004.

[51] Y. Yamauchi, M. Yokozawa, T. Shinohara, and T. Ishida, "Collaboration with lean media: how open-source software succeeds," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pp. 329–338, ACM, 2000.

[52] P. J. Agerfalk and B. Fitzgerald, "Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy," *MIS quarterly*, vol. 32, no. 2, p. 385, 2008.

[53] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. Slyngstad, and M. Morisio, "Development with off-the-shelf components: 10 facts," *Software, IEEE*, vol. 26, no. 2, pp. 80–87, 2009.

[54] L. Morgan and P. Finnegan, "Benefits and drawbacks of open source software: an exploratory study of secondary software firms," *Open Source Development, Adoption and Innovation*, pp. 307–312, 2007.

[55] A. Bonaccorsi, D. Lorenzi, M. Merito, and C. Rossi, "Business firms' engagement in community projects. empirical evidence and further developments of the research," in *Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on*, pp. 13–13, IEEE, 2007.

[56] M. Andersen-Gott, G. Ghinea, and B. Bygstad, "Why do commercial companies contribute to open source software?," *International Journal of Information Management*, vol. 32, no. 2, pp. 106–117, 2012.

[57] J. Bosch, "Software ecosystems: Taking software development beyond the boundaries of the organization," *Journal of Systems and Software*, vol. 85, no. 7, pp. 1453–1454, 2012.

[58] P. Anbalagan and M. Vouk, "On predicting the time taken to correct bug reports in open source projects," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pp. 523–526, IEEE, 2009.

[59] S. Daniel, L. Maruping, M. Cataldo, and J. D. Herbsleb, "When cultures clash: Participation in open source communities and its implications for organizational commitment," in *Proceedings of the International Conference on Information Systems*, 2011.

[60] J. Henkel, "Champions of revealing – the role of open source developers in commercial firms," *Industrial and Corporate Change*, vol. 18, no. 3, pp. 435–471, 2009.

[61] S. O'Mahony and B. A. Bechky, "Boundary organizations: Enabling collaboration among unexpected allies," *Administrative Science Quarterly*, vol. 53, no. 3, pp. 422–459, 2008.

[62] B. J. Oates, *Researching information systems and computing.* Sage Publications Limited, 2006.

[63] I. Benbasat, D. K. Goldstein, and M. Mead, "The case research strategy in studies of information systems," *MIS quarterly*, pp. 369–386, 1987.

[64] K. Leetaru, "Culturomics 2.0: Forecasting large-scale human behavior using global news media tone in time and space," *First Monday*, vol. 16, no. 9-5, 2011.

[65] T. Snarby, "Communication patterns among commercial firms in oss projects." `http://folk.ntnu.no/snarby/TDT4501/fordypningsprosjekt.pdf`, 2012. TDT4501, Student specialization project, NTNU.

[66] B. Kaplan and D. Duchon, "Combining qualitative and quantitative methods in information systems research: a case study," *MIS quarterly*, pp. 571–586, 1988.

[67] "Q&a with the founder of wireshark and ethereal." `http://www.protocoltesting.com/gerald_combs_interview.html`, 2008. Protocol Testing.

[68] Andrew Tridgell and the Samba Team, "A bit of history and a bit of fun." `http://www.rxn.com/services/faq/smb/samba.history.txt`, 1997. History of Samba. Retrieved: 2013-04-19.

[69] C. Bird and N. Nagappan, "Who? where? what? examining distributed development in two large open source projects," in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pp. 237–246, IEEE, 2012.

[70] A. Nguyen Duc, D. Cruzes, C. Ayala, and R. Conradi, "Impact of stakeholder type and collaboration on issue resolution time in oss projects," *Open Source Systems: Grounding Research*, pp. 1–16, 2011.

[71] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pp. 275–284, IEEE, 2011.

[72] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering.* Springer, 2012.

[73] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

APPENDIX A INTERVIEW DOCUMENTS

A.1 Interview Invitation

The following interview invitation were sent to firm-paid developers, inviting them to participate in our research:

> Hi,
>
> My name is Terje Snarby, and I am a fifth year computer science student at the Norwegian University of Science and Technology (NTNU). Currently I am working on my Master's thesis, investigating communication patterns among commercial firms in OSS projects. As part of the thesis, I want to conduct interviews with developers that participate in OSS communities as part of their paid work. Prior to this study, I have examined the Wireshark community and created an overview of the activities and communication between firm participants.
>
> OSS development is a highly distributed and collaborative activity, and is increasingly influenced by commercial firm participation and contribution. These firms have different motivations, work styles and collaboration practices. With regard to these differences, I would like to have an interview with you about:
>
> 1. Work process variety between firm and OSS community.
> 2. Organizational boundaries within OSS communities.
> 3. Potential differences in collaboration practices between firm and OSS community.
>
> Your participation would be very valuable to me. Furthermore, findings from the research may be of great importance for

practitioners and researchers by creating a better understanding of OSS development and firm participation.

Please note that a transcription of the interview would be sent to you for approval. In addition, all sensitive information will be filtered and participants will be kept anonymous. I sincerely appreciate your understanding and cooperation for taking part in the research.

I am looking forward for your reply.

Kind regards,
Terje Snarby