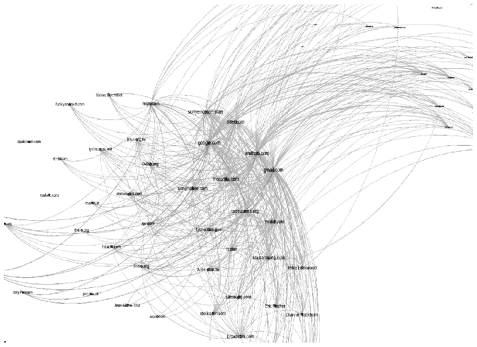
Understanding Software Development in an Open Source Context: Network Analysis of Source Code Repositories



Alma Oručević-Alagić

Doctoral Dissertation, 2016

Department of Computer Science
Lund University

Dissertation 50, 2016
LU-CS-DISS: 2016-2

ISBN 978-91-7623-719-9 (printed version)
ISBN 978-91-7623-720-5 (electronic version)
ISSN 1404-1219

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: Alma.Orucevic-Alagic@cs.lth.se
Web Page: http://cs.lth.se/alma_orucevic-alagic

Typeset using LaTeX
Printed in Sweden by Tryckeriet i E-huset, Lund, 2016

ABSTRACT

Open Source Software (OSS) created a paradigm shift within the software engineering field prompting further research to understand how mature, industry grade, software can be produced in an online milieu with distributed and lightly managed developers contributing source code in their free time. The OSS has become also a major revenue generator for many commercial organizations, and has found its place in closed source products, putting many proprietary software producers in the middle of an OSS community.

The goal of the thesis is two-fold. Firstly, the research focuses on the assessment of the scope of impact the OSS has had on commercial software development. Secondly, the research studies some of the scoped aspects in more depth, such as the applicability of the OSS development practices within the closed software development environment, also known as inner source, as well as the analysis and benchmarking of developers' collaboration networks.

A systematic review of literature was conducted to scope OSS usage within the commercial context, while a case study with focus of understanding applicability of inner source development practices was conducted within a large company. The conducted research has in large part focused on the analysis of source code for over 400 Open Source Software projects, such as Android Open Stack, Apache Software Foundation, Ingres dbms, and a proprietary source code produced by a large branch within Ericsson.

The results of the conducted research show that Open Source Software has impacted the way companies develop software by including OSS components into their proprietary products, implementing OSS business models, participating in OSS communities, and implementing OSS development practices. The research provides guidelines on how to implement inner source, as well as a network theory based approach for assessment and monitoring of software development process along with associated network metrics benchmarks.

CONTENTS

PREFACE

List of Included Publications

The following publications are included in this doctoral thesis:

I **A Systematic Review of Research on Open Source Software in Commercial Software Product Development**
Martin Höst, Alma Oručević-Alagić
Journal of Information & Software Technology, 53:6, pp. 616-624, 2011

II **Usage of Open Source in Commercial Software Product Development - Findings from a Focus Group Meeting**
Martin Höst, Alma Oručević-Alagić, and Per Runeson
International Conference on Product Focused Software Development and Process Improvement (PROFES) 2011, pp. 143-155, 2011

III **A Case Study on the Transformation from Proprietary to Open Source Software**
Alma Oručević-Alagić, Martin Höst,
Extended version of 6th International IFIP WG 2.13 Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, May 30 - June 2, 2010, Proceedings, pp. 367-372, 2010

IV **A Prolonged Two Phase Case Study on Implementation of Open Source Development Practices within a Large Company Setting**
Alma Oručević-Alagić, Martin Höst
Submitted.

V **Network Analysis of a Large Scale Open Source Project**
Alma Oručević-Alagić, Martin Höst
Extended version of 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, August 27-29, 2014

VI **Development Process Monitoring Through Application of Network Analysis on Source Code Repository Data**
Alma Oručević-Alagić, Martin Höst
Submitted.

VII **Benchmarking Apache Software Foundation Projects: Network Analysis of the Contributors' Collaboration Networks**
Alma Oručević-Alagić, Nicklas Johansson, Christian Tenggren, Martin Höst
Submitted.

Contribution statement

The author of this doctoral thesis, Alma Oručević-Alagić, is main contributor of publications III-VII. For these papers, she was the main designer, and responsible for most of the writing and the running of the research process. For the Paper II, the author participated in the design and implementation of the focus-group meeting. The author was involved in the implementation of the research presented in Paper I, the systematic literature review, in particular the articles' selection and assessment process, as well as in the writing of parts of the paper, such as summarizing two subject areas.

List of Related Publications

Related publications that are not included in this doctoral thesis:

I **A Case Study of Open Source Development Practices within a Large Company Setting**
Alma Oručević-Alagić and Martin Höst
ICSET 2014 International Conference on Software Engineering and Technology, Istanbul, Turkey, Sep 29-30, 2014

II **Inner Source Project Management**
Martin Höst, Klaas-Jan Stol, and Alma Oručević-Alagić
Software Project Management in a Changing World, Gunther Ruhe and Claes Wohlin
Springer Berlin Heidelberg, ISBN:978-3-642-55034-8, pages 343-369, 2014

III **Development of Safety-Critical Software Systems Using Open Source Software - A Systematic Map**
Sardar Muhammad Sulaman, Alma Oručević-Alagić, Markus Borg, Krzysztof Wnuk, Martin Höst, and Jose Luis de la Vara
40th EUROMICRO Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2014, Verona, Italy, August 27-29, 2014

ACKNOWLEDGEMENTS

This work was funded by the Industrial Excellence Center EASE - Embedded Application Software Engineering.

"Only when we see ourselves in our true human context, as members of a race which is intended to be one organism and one body, will we begin to understand the positive importance not only of the successes but of the failures and accidents in our lives. My successes are not my own. The way to them was prepared by others. The fruit of my labors is not my own: for I am preparing the way for the achievements of another. Nor are my failures my own. They may spring from failure of another, but they are also compensated for by another's achievement. Therefore the meaning of my life is not to be looked for merely in the sum total of my own achievements. It is seen only in the complete integration of my achievements and failures with the achievements and failures of my own generation, and society, and time." Thomas Merton, No Man is an Island

"We are the best gift to each other ." Hafiz Sulejman Bugari

There are many individuals, family, friends, teachers, colleagues, co-authors, who have helped me in pursuing my professional path, without whose guidance this work would not be possible. Back in 1992, as a high school student in Sarajevo, Republic of Bosnia and Herzegovina, I would had never dreamed that war would break out in my homeland, and send me, an average teenager in love with sports, music, arts, and mathematics, into a world unknown. On this journey, not only have I crossed paths with many truly inspiring souls, I have also acquired three new close families (Bajić Anita & Darinka & Vjeko, Sattler Sue & Elmer, and McNall Natalie & Dan) and was blessed for having had an opportunity to study and work in three other countries; Croatia, USA, and Sweden.

I would especially like to express my utmost gratitude, to all the people, and the organizations that have provided funds for full scholarship and board during the war times, such as the Jerrahi Order of America, Fellowship for Reconcili-

ation Minnesota (FOR), World University Service Austria (WUS), St.Catherine University, and Faith United Methodist Church (St.Anthony, MN) . To name all those who helped me journey, would be impossible, so I will name just those who were directly involved in supervision of the thesis. Thank you Prof. Dr. Martin Höst, for your invaluable guidance, insight, and patience during my journey of knowledge acquirement. It has helped me look at the field of research from a different perspective and increased my awareness on the importance of objective and structured research approach. My thanks also go to Prof. Dr. Per Runeson, whose guidance and leadership by example is an inspiration for everyone who has had an opportunity to work with him. I would also like to thank all my colleagues at the department of computer science at Lund University, especially members of the SERG for the inspiring discussions and helpful advises. I am very grateful for having the opportunity to work with such a wonderful group of people.I would also like to thank the members from the companies supporting EASE project, without whose help much of the research would not be possible.

Special thanks go to my parents, Ramiz and Sabira, and my girls, Aiša, Emina, and Selma, whose sunshine and love are inexhaustible source of inspiration.

Finally, all praises, goodness, and thanks, are due to God alone, the Creator and the Sustainer of this reality that we call life.

POPULAR SCIENCE SUMMARY

Understanding Open Source Software in Industry Context and Knowledge Transfer

Alma Orucevic-Alagic,
Department of Computer Science, Lund University

Ever since it has found its way into offices, homes, hospitals, transportation, and other industries, software has profoundly impacted and reshaped the way of living. Looking for information on the internet, socializing, watching movies, shopping, and other activities that have become a part of daily lives, would have been impossible without software running on internet servers, mobile devices, personal computers, etc.

For the end users, with no formal experience in software development, the intricate world of software development is unfamiliar along with its many problems and solutions, which have been invented during the past decades. Project budget overruns, software that does not fulfill user requirements, poor or malfunctioning software, are just a few problems that the software industry has been trying to solve for decades.

Traditionally, software has been developed in-house, and all code was much guarded and secured, so the secret software bits do not leak out, as in this case, the software would become free and would hold no commercial value for the company. Guarded, closed source software development means also that it cannot be studied openly, on a larger scale which could help improve software development process.

This closed and set world was shaken to its foundation with the emergence of open source software communities. The communities were originally composed of software hobbyists, developing software at free time in an online milieu. While the open source software movement has existed for as long as the software has been developed, it has never gained broad industry acceptance until it produced software that could compete with software produced by industry.

Many mark the emergence of Linux operating system as a product that created positive industry disposition towards open source. After all, using a free operating system, a very complex software product, and building commercial products and services around it was more financially sound than building the whole product in house.

Not only did the industry start using open source products, but it also started participating in OSS development communities. Today, open source software is everywhere, from mobile devices with Android, to personal computers running Linux, even Apple's IOS being based on open BSD software, to over 60% of internet servers running Appache Web server.

The emergence of high quality open source software, distributed under permissive licenses, built by transparent online communities everyone can join, and freely available source code that anyone can access, brought several questions. Firstly, how can self-managed communities organize and produce complex, industry quality software, secondly how open source software affects commercial organizations, and thirdly what open source development practices could be applied within closed development context in order to improve software development. These are also some of the most important questions studied in our research.

We identify different industry roles with respect to an open source community, showing that when industry decides to use an open source product, this creates a long lasting relationship with the community. We also show how and why open sourcing a proprietary software product can help the product regain market share and generate economic benefit for the company. We also zoom into open source code repositories, and examine source code contributions by applying network analysis. The application of this method on the Android source code repository has gained interest in local industry. We have applied the same method on hundreds of different mature open source project and proposed a set of metrics that can be used to assess any development effort. Hence, the method proposed in the research helps companies understand the major contributors and influencers on the project, which can aid them in creating business strategies. The same method was also applied to assess commercial closed source development effort and to monitor organization and process changes. We also define a framework of open source development practices that can be used by a company to examine and align its development practices in order to make their development effort more efficient.

While the software industry has went through a paradigm shift ever since the open source software became mainstream, the field is very dynamic, requiring greater research focus to scope and assess existing and emerging consequences. In the meantime, an average user can enjoy a plethora of high quality software, hoping that this joint development effort called open source movement will create even more value for everyone in the future.

INTRODUCTION

Open source software (OSS), distributed under various permissive licenses, nowadays can be found in a plethora of hardware devices, ranging from personal computers and mobile devices, to the ones used by different industries, such as telecommunication, medical, aviation, etc. While the OSS has become de-facto, mainstream, standard of software industry today, its history has been marked by conflicts between proponents of closed source software and different groups within the open source software movement. According to Larry Augustine [Lin16], a pioneer of the OSS movement [OSI13a], five distinct periods of the OSS can be identified: *games 70s* (Adventure, Rouge), *tools 80s* (GCC [Fre16b], compilers, linkers), *operating systems 90s* (BSD Unix [McK99], Linux [Fou15c]), *database middleware* (MySQL[Ora16], JBOSS[Red16a]), and wide industry grade *applications*, e.g. customer relationship management (CRM), enterprise resource planning (ERP), business intelligence BI, etc. Over the past several decades, the OSS movement and business models have matured moving from software produced by unpaid computer enthusiasts for their own needs, e.g compilers, linkers, games, to software produced by both enthusiasts and industry members, e.g. complete software stacks, such as Open Source Linux Stack, as viable alternatives to proprietary Microsoft and IBM stacks [Aug16].

The emergence and wide spread usage of large scale Open Source Software products, e.g. Linux [Fou15c], made available for the public under permissive licenses has profoundly changed and challenged the traditional way of software development, valuation, and marketing, as explained by Raymond [Ray01a] in 1999. Hence, OSS created a paradigm shift within the software engineering field prompting research for several reasons. *Firstly*, to understand how large, industry grade software is developed in an online milieu with lightly managed, geographically distributed developers participating mostly on volunteer bases. *Secondly*, to assess if there exist OSS development practices that can be beneficial in a closed source development environment. *Thirdly*, what motivates industry participants to use OSS components and participate in OSS process, and, *finally*, what kind of impact does industry involvement with OSS have on the OSS communities' and

general trends in software production.

Based on the fore mentioned questions, the initial research focus of the thesis was divided into the following three areas:

1. Assess the usage, development, and business models of commercial organizations with respect to OSS.
2. Identify a framework of the most important OSS development process characteristics and assess its applicability within a closed development setting.
3. Explore applicability of network theory analysis in studying a structure and an evolution of OSS development communities.

As the first step in assessment of usage, development, and business models by commercial organizations, we conducted a systematic review [KC07] on usage of OSS in commercial software development to understand the landscape of the prior research done in the area. The landscape shows four distinct areas that are tightly intertwined. Availability of OSS components that can be included and used in commercial software development motivate creation of *OSS business models* that can benefit from *reusable OSS components* which primarily replace the need to build the software in-house or purchase it as an off the shell component. The connection formed between a company and an OSS community through the usage tends to grow, prompting companies to take an active role in the community to ensure that the component is properly maintained, and that the changes the company might had made to the component, are included in the future releases. As the companies gain experience through their *participation in the OSS process*, an increased understanding is gained into *the OSS development model*. Through a focus group meeting carried out with representatives from large software intensive companies, we identified and discussed the criteria applied by the companies when selecting OSS component for inclusion in their software products. In addition, the motivation for contributing the source code changes the companies have made back to OSS community were discussed and assessed. The role of commercial organizations with respect to OSS was further examined in a case study of Ingres database management system which was developed and sold as closed source before being open sourced by Computer Associates (CA) Inc., one of the world's largest software producers. Here we show how business model for commoditized software is motivated, as well as how the OSS community is set up and run. We also show how community development affects the product in terms of changes in standardized source code quality metrics by applying statistics tools to measure significance of the metrics' changes.

Based on the gained insight on inner workings of OSS communities, the next step in research was to assess how aligned the development practices of a large software intensive company, that bases its products on an OSS product, are with the OSS development practices. For this purpose, a framework defining OSS development practices based on the conducted research by Fogel [Fog05] was proposed,

and the companies practices were assessed. Here we offer insights into which OSS practices companies decides to adopt and the reasoning behind the decision. We also offer insight into potential benefits that could result in the implementation of the OSS development practices. In order to further the research on the inner workings of development communities, network analysis of development communities was performed. Through three distinct research efforts, development communities of Android, Apache, and Ericsson were studied. The research relies on well established theory from the network analysis theory, and proposes construction of developers networks as weighted and directed graphs. We show that such construction of developers' networks provides a more realistic picture of development communities in terms of developers' or companies' influence within the community than previously proposed ones as, e.g. López-Fernández et al. [LF+06] . In addition, we also show how measuring changes in metrics of developers networks can be used to monitor effects of organization or development process changes on developers' interactions. Based on the research done, the concrete contributions of the thesis are:

- A network theory based approach for studying software development networks which can be used in closed source and OSS environment to assess and monitor software development structures, influencers, and general metrics' trends.
- An in depth case study on motivation, setup, and outcomes of commercially led effort to open source a commoditized software product.
- An OSS development framework describing characteristics of a mature OSS development milieu.
- An in depth case studies tracking alignment of in-house software development practices .

1 Background and Related Work

In this section major aspects of OSS evolvement and proliferation into industry are discussed. The presented aspects are related to current research in the field as well as the research done in this thesis.

1.1 Open Source Software: Historical Overview

In the early days of software development, during the 1960's, building large and expensive computing machines required more effort and resources than creation of software that would execute on the machines. This led to the software being shared freely among the scientist. With technology advancement and production of more complex and diversified computing machines, the field of computer programming

yielded new guidelines for creation of more complex software products. Among the first pioneers of the field were Edsger Dijkstra proposing layered architectures in 1968 [Dij83] and David L. Parnas who in 1972 [Par72] introduced concepts of system modularization. Thus, instead of building new solutions from the scratch, which was a common practice in 1970s, programmers started building and using reusable, tested and verified families/architectures, enabling them to create unique software products by introducing variance and specific implementations at appropriate levels. Hence, the ability to share software architectures increases software reuse and, as a result, it improves productivity and reduces cost of software development.

Sharing of the code became especially popular in academic environments. As described by Raymond [Ray01a], the Berkeley Software Distribution license also known as BSD, is a result of years long collaboration on the development of the Unix operating system by the University of California, Berkeley and AT&T labs. In the beginning of the 1980s, especially during the time when the personal computers gained popularity, the decades old concept of software source code sharing, was replaced with proprietary, closed source software products. As a response to the new situation, open source proponents led by the effort of Richard Stallman founded the Free Software Foundation (FSF) [Fre16a]. However, the efforts were not met with a broader public acceptance, especially among the industry participants [Web04b]. According to Raymond [Ray01a], the emergence of the Linux operating system served as catalyzer for the open source movement proliferation especially with industry, as it has demonstrated that a large open source community can produce complex and sophisticated software and that business models can be built around open source software. In 1998 Eric Raymond became one of the founders of the Open Source Initiative (OSI) [OSI13a], a non profit organization with aim to advocate and educate about the benefits of open source. The OSI also issues Open Source Initiative Licence trademark, which according to the organization's mission statement, has a purpose of building trust around all constituencies of an open source community carrying the trademark [OSI13b].

While the need to share source code at the very beginnings of software development was motivated by high cost of computing machines, an interesting question for current times characterized by low cost, high performance computing machines, is what motivates individuals to take part in development of OSS. The question has been answered in the studies at Kiel University [Her+03] and by Boston Consulting group [LW05]. The study at Kiel University examined motivations of Linux kernel project participants from 28 countries. The two leading motivating factors identified in the study are participants' desire to increase their own commercial/market value and personal satisfaction. A study by Boston Consulting Group, done on 525 Source Forge community members, showed results in line with the Kiel University study, with the highest motivating factors as personal belief in OSS, hope of increasing ones commercial value, and an opportunity to enhance skills.

1.2 Industry Perspective: Software as Commodity

A wide industry acceptance of OSS products such as the Linux and the Apache, and the emergence of OSS business models has created a need for systematic study of the OSS phenomenon from commercial perspective. The research study by West [Wes07] discusses aspect of software commoditization. He argues that while there exist some unique and new technology inventions coming from the OSS world, such as the Apache web server, the majority of broadly adopted OSS solutions are counterparts of existing proprietary solutions, such as Linux and MySQL. This finding is in line with the study by Linden et al. [Lin+09a] who also argue that over time software goes through an amortization process where its value decreases to the point where it does not hold any significant differentiating value needed for competitive advantage. According to Perens[Per05] some 90% of software used in the industry is a type of infrastructure software that offers no competitive advantage. Research by Bradley and Porter [BP00] shows that companies can at the same time cooperate and compete, just as in the case of HP and IBM Strategic Group Alliance [Hew16]. Hence, the strategic closed group alliances have been made in the past between many industry competitors, as discussed in the study by Gomes-Caseres [GC94]. The study also explains that as the products grow in complexity and under the pressure to compete globally, the companies are motivated to share the costs of commoditized parts of the products which then enables them to focus development efforts on differentiating parts of the product. In addition, the study offers some guidelines on factors to consider on how to form, lead, and manage strategic alliances.

In this thesis we study the development community of the Android OSS operating system for mobile devices using a network theory based approach. The Android OSS community, led by Google Inc, receives input from the Android Handset Alliance which includes companies from the entire mobile eco-system. Android is an example of an enterprise grade OSS operating system for mobile devices which in the 2010 replaced Symbian, the proprietary operating system developed and used at the time by major market leaders, e.g. Samsung, Sony Ericsson, Nokia. Again, the same motivation lies behind the sharing of development costs for the undifferentiating part of the product. The cost motivated approach to reuse OSS components in favor of developing in-house software products has shown to have a wide impact on both, hardware and software industry, further reaching than in the case of closed strategic alliances. Not only did Google through Android commoditize a complex and large software product. It has also achieved this by building Android on a plethora of preexisting OSS components, among which the most notable is the Linux kernel. Since then, the Symbian operating system has been replaced by Android as market leader, resulting in lower entry costs into the market of mobile devices [Con10]. Thus, while reducing costs of software development, the new circumstances brought a more competitive environment for the associated businesses. Unlike the strategic partnership, a typical OSS commu-

nity is open to anyone who wishes to participate and its product are developed in transparent fashion. Therefore, understanding an underlying fabric of community, the major contributors and dynamics of development in an OSS milieu is of high importance for the participating companies.

Besides using an OSS component as a third party component, companies can also choose to open source a software product which has lost its differentiating value. For example, Computer Associates, a company ranked as one of the five largest software vendors [Tim13], Raymond [Ray01a], open sourced the Ingres database management system. The software is used in some of the company's products, but over the time it has become a non-differentiating technology, and lost its leading edge over other commercial and OSS solutions, e.g., Oracle and MySQL [OAH10]. Hence, the motivation for the Ingres open sourcing is not only sharing of development burden with the community, but also regaining some of the lost market share. The Netscape web browser is an example where a company has lost a large part of its market share due to entrance of another software product, in this case the Internet Explorer web browser distributed by Microsoft corporation. To remedy the situation, Netscape decided to become OSS in 1998 and successfully regained the lost market share[Ray01a].

The Eclipse OSS project develops and maintains Eclipse, leading software development platform, and it is governed by the Eclipse Foundation whose members are largely comprised of leading industry software companies. By participating in the Eclipse OSS community, companies can ensure that necessary functionality for building software products with their proprietary or open source solutions is included in the Eclipse OSS. Hence, by contributing resources to an OSS project, company can also proliferate usage of its own products. This was also the main motivation for IBM to open source Eclipse in 2001, which based on development effort was valued at 40 million dollars at the time [Wes07].

1.3 Fabric and Dynamics of OSS development

Many studies have been conducted to understand the underlying fabric of OSS communities. Some of the first large scale studies of OSS development communities tried to understand the structure of developers in terms of their source code contributions, how the communities are organized and managed, as well as the underlining characteristics of the OSS source code. The nature of the OSS online milieu is such that it provides communication and source code archives which land themselves for data mining and analysis.

Orbitan Software Survey [AGVP00] studied the make up of participants that participate in OSS projects, such as RedHat Linux v 6.1, Linux Kernel sources v 2.2.14, Munitions Cryptography, and some 50% of projects available through Freshmeat, has shown unequal distribution of developers' contributions in 2000. The results of the survey show that 10% of developers, or 1276 of them, contribute to 72% of the code base comprised in a total of 25 million lines of code and 3149

distinct projects. In another study by Crowston and Howison [CH05], the way the participants communicate in projects hosted under Source Forge was analyzed. Even though Source Forge at the time of the study hosted over 50,000 distinct OSS projects, by eliminating projects that have less than 7 developers and less than 100 bug reports, they discovered that only 124 projects or just 0,002% of all hosted projects fit the criteria. The outcome of such selection criteria can point at the importance of understanding the studied OSS communities at an individual level, as there seem to exist many projects hosted by various OSS portals that are not active. The results, based on studying communication channels for 61,068 bug reports, show that the majority of the projects have highly centralized decision and communication structures. Further more, the study found that there exists a predictable relationship between the structure of the code and the organizational structure of the development team.

Some conducted studies, e.g. by Robles et al. [Rob+05] and Godfrey et al. [GT00a], have shown that OSS evolution goes against the fourth law of software evolution as defined by Lehman [LR01] which states that the growth of software is constant and independent of the amount of resources devoted to its development. The research has shown that in some project, e.g. Linux kernel, the growth was super linear, i.e. greater than linear or constant, in contrast to the Lehman's fourth law, but also that this super linear growth was made possible by inclusion of external code that did not require maintenance.

Through the years, an increased industry awareness and participation in the OSS development process has shifted OSS perception from a d̈evelopment playgroundp̈rimarily reserved for software hobbyists to an industry stirred and strategically planned software development endeavor. Various OSS business models described by Raymond [Ray01a] have been further explored through introduction of new OSS distribution licenses [Fit06a]. Hence, the industry involvement served as another motivating factor to find appropriate tools to analyze OSS development effort to understand the underlying development network topologies, identify the most influential contributors, and observe the corresponding changes over the time.

Social networks analysis presented by Wasserman [WF94a] is a part of the field of network theory, as described by Newman [New13], which provides methods for studying social interactions. The initial formal studies in the field of social network analysis, called sociometry in 1934 by Moreno [Mor34], laid a foundation to todays studies of social network analysis. The networks are composed of actors and links between them, usually referred to as nodes and edges. In OSS context the edges represent associations such as communication between a community's participants, or affiliations such as developers who made changes to same source file. A significant amount of research is done by applying network analysis study by Luis et al. [LF+06] which offers relevant guidelines on how to use social network analysis within the context of developers and module networks. Luis et al. propose usage of weighted networks where the edges between developers are given weight of sum of all changes developers have done together on a source

code module. Research presented in this thesis [OAH14b] builds on the research by Luis et al. [LF+06], and proposes construction of weighted and directed developers' networks. The proposed method builds on earlier research by Davis et al.[Dav+41], also referred to as "Southern Women Study", which in the software development context implies that two developers are connected through an event defined as modifying the same source file. In this case the weight or strength of relationship is measured relative to the size of the event, i.e. respective to changes performed on the file by all developers. This approach was applied in this thesis and tested on the Android OSS stack to understand who are the most significant contributors on the project, a relevant investigation as the software stack contains OSS components used by many competing industry members. The same methodology was later applied to study internal development networks at Ericsson as well as to understand if there exist any common trends in the network metrics for over 255 projects managed by the Apache Software Foundation. Some of the results of the studies are in line with similar studies, e.g. by Mohammad et al. [AL13] showing that more experienced developers have tendency to collaborate with new developers, an indicative feature of an emerging and complex network [New13].

1.4 OSS Development to Closed Source Development: Knowledge Transfer

Understanding the development and communication structures of OSS development can provide insights into how online, self-managed communities organize and work to produce software, which according to Raymond [Ray01a], is of higher quality and reusability, and easier to maintain than traditionally produced software. High quality disposition of OSS software that has been argued by now a famous Linus's Law "with enough eyeballs all bugs are shallow" [Ray01a] can be analogously viewed in the context of academic research where the peer review process forms the bases for establishment of quality and merit in research. A study by Fagan [Fag86] shows that the OSS development process assumes peer review of new source code features and fixes, which can be viewed as another way to perform traditional code inspection.

A natural question that arises is if some parts of the OSS development process can be applied within a closed development setting, especially in large software organizations that have distributed development sites. Work by Fogel [Fog05] provides insights on how to run a successful OSS project, while work by Dinkelacker [Din+02] proposes and implements so called *progressive open source* or POS in a closed source setting. The POS distinguishes between three types of source code development, each residing in its own circle with circumference of the circle corresponding to firewall boundaries as depicted in in Figure 1. The most restrictive firewall encircles software projects developed under inner source process, an OSS development process for source code repository accessible only to developers residing behind the company's firewall. The second circle is bounded by a less re-

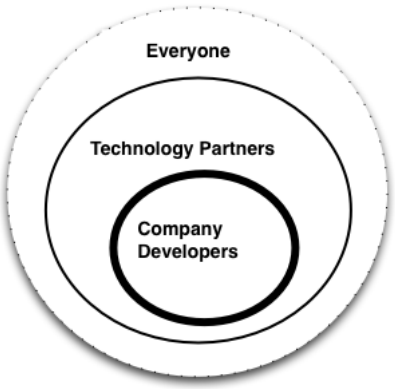


Figure 1: Role Based Source Code Access Enforced by Firewall Restrictions

strictive firewall that guards code which, besides the company's developers, other technology partners can access and maintain. The final circle contains source code accessible to everyone, i.e. this is the source code the company has decided to develop under the OSS process. This approach has shown to have many benefits like standardized development environment and process, increased rapid team redeployment, higher quality of shared components, shared community debugging as well as a milieu for partner technology companies to provide immediate feedback and propose and implement bug fixes [H+14].

Some of the challenges of the inner source process come from the fact that unlike an OSS environment, strategic planning, budget and time limitations include several stakeholders and project owners whose work needs to be highly coordinated and planned. The planned projects have limited development resources of varying skills, unlike the OSS environment where developers contribute code on voluntary bases. Closed source code may also need to restrict access to the company's developers motivated by a need to preserve highly valued bits of the code to those directly involved in its development. A number of studies have been conducted to understand the applicability of the OSS development practices within a closed, commercial environment. The studies conducted in HP [MM08], Lucent [Gur+06], and Nokia [Lin+08b] analyzed development of software products within the company setting. In this thesis we present a two phase study on how aligned company's software development practices are with OSS development practices as presented in Paper IV. A part of this study was also presented in a book chapter on Inner Source [H+14].

2 Research Overview

An overview of work done in this doctoral thesis is presented in the Figure 2. The results of the systematic review presented in Paper I and summarized in Figure 3 are a starting point for the research. While the researched area is rather broad, its assessment was necessary to understand the breadth and depth of the prior research on the usage of OSS components in commercial software development. As discussed in Section 1.2, the primary motivation for the industry to use OSS software components is to reduce development costs for products that do not hold commercial value, which at the same time frees resources to work on value-adding product features. The effect of the motivation is an emergence of different OSS based business models. Thus, besides the initial sharing development burden model, there is, e.g., the augmenting services model where a company actively participates in an OSS community to add services needed to the OSS product so the product can be sold under a more restrictive license. The Android operating system is an example of an OSS based eco system that can lead and change large industry segment.

A company that decides to implement any of the OSS business models [Ray01a], is also driven to be involved with the OSS community for several reasons. For example, in case a company decides to include an OSS component in its product, it needs to keep the product updated for the new releases to ensure that the component still functions properly. Any changes that the company makes to the component, the company should send back to the community in order to ensure that the future releases, especially the ones including fixes to security, also contain any new features the company has added. Thus, implementation of the OSS based business models normally leads to increased industry participation in OSS community process. Working with an OSS community assumes following its development model and adhering to the corresponding governance rules. Through the involvement, the companies gain experience in the OSS way of development, and, as a consequence, might decide to implement internally some of the processes they deem value adding. The OSS industry cycle presented in Figure 3 highlights a need for industry involvement with the OSS community which was a motivation to carry out the studies that could provide an increased understanding of the OSS community process.

The overall research goal of the thesis is to leverage knowledge gained through the study of commercial involvement with OSS development processes and archived OSS data to support large scale commercial software development.

RQ1: What interactions exist between different industry roles with respect to OSS?

RQ2: What are the major characteristics of an OSS development framework?

RQ3: What are the perceived benefits and drawbacks of OSS development process implementation within a closed development environment?

RQ4: How can the proposed network analysis based approach be used to understand and monitor development structures and associated development metrics?

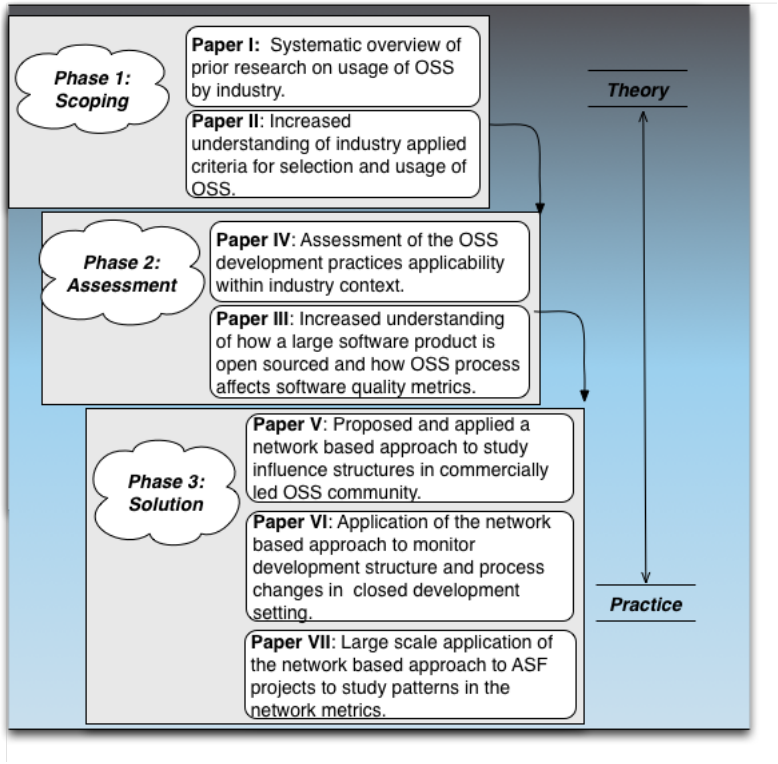


Figure 2: Overview of Contributions of Thesis

The research can be dividid into three phases as presented in Figure 2: the scoping, the assessment, and the solution phase. The scoping phase, besides Paper I, includes Paper II where work was focused on collecting empirical evidence from representatives of large companies which use OSS components through a structured focus group meeting. Paper II further clarifies the criteria industry uses when selecting an OSS component for inclusion in their products as well as the motivation to give back to the community any modifications of the OSS component.

The assessment phase includes two case studies presented in Paper III and Paper IV. Paper III follows a transition of a proprietary database management system, Ingres, from a closed source environment to an open source one. Software quality metrics were calculated and analyzed through application of standard statistics tools and methods for the two versions of code; the most recent closed source version and the latest open source version. The goal was to understand how the

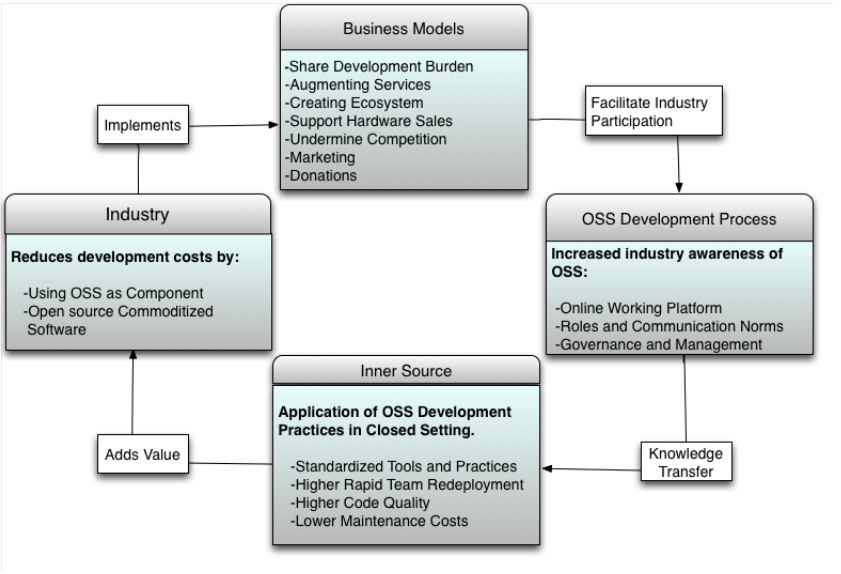


Figure 3: Industry Involvement with OSS Cycle

software quality metrics have changed between the two versions, i.e. how the software metrics were affected under the community development process. The paper also provides a background on why the company has decided to open source Ingres as well as the steps taken to establish and run the Ingres OSS community. The prolonged, two phase, case study presented in Paper IV was carried out in a large, global software company with the goal of understanding how aligned its development practices were with OSS development practices. For this purpose, an OSS development framework was proposed based on the study of Apache Software Foundation [Fou16] and Fogel [Fog05]. The company practices were studied and analyzed, through work at the company as well as through a set of structured interviews and then compared against the framework. Two years upon the completion of the first phase of the study, a focus group meeting was held with the interviewees from the first phase to understand and assess any changes in the alignment. The company was a relevant and interesting subject to study as its core software products are based on OSS components and the company worked closely with the corresponding OSS communities.

Finally the third, or the solution proposal, phase focused on designing appropriate tools to assess and quantify development structures with the aim to create a tool that can provide valuable information on the underlying development communities. The work presented in Paper V proposes a method to construct developers'

networks which was not previously applied in the OSS research. While study by Luis [LF+06] establishes guidelines for the application of network analysis to study developers' and module networks as weighted networks, the weights of the edges are determined by the total sum of changes to same module done by two developers. When this method was applied in the work for Paper V to study the the Android OSS stack, it yielded networks that did not represent the underlying structures and influences well. Instead, the network analysis method applied in Paper V builds on theory from the earlier study by Davis [Dav+41] and shows that calculating edge weights relative to the number of changes done on the module results in more realistic representation of the networks. Development networks constructed in this way can be useful when trying to understand the major community influencers and development structures, which is import information when a company plans to join an OSS project. In Paper VI the same network analysis based approach was applied in a large branch of Ericsson to understand how development structures and corresponding metrics changed after the company underwent major development structure and process changes. The results were validated through a focus group meeting which has indicated that the approach can be useful to monitor development process. Finally, in study presented in the Paper VII the same network analysis was applied on over 250 OSS projects governed by the Apache Software Foundation (ASF). Since the ASF hosts mature and industry used OSS projects, the goal of the study was to uncover patterns in network metrics which could be indicative of a well structured development process. Such metrics can be possibly used to benchmark closed and OSS development structures.

3 Research Methodology

The research carried out in this thesis applies standard research methodology established in the field of software engineering as discussed by Wohlin et al.in [Woh+12, p. 5–8]. A simplistic classification of research methodology according to Glass et al. [Gla+02] was presented earlier in Glass [Gla95], identifying the following *research methods*:

- Scientific - Implies building a model based on the real-world observations.
- Engineering - Studies existing solutions, proposes modifications and evaluates them.
- Empirical - Builds and evaluates models based on empirical studies.
- Analytical - Proposes a formal theory which is then evaluated against the empirical observations.

According to Wohlin et al. [Woh+12, p. 5–8] some of the empirical strategies include formal experiments, case studies of real industry projects, and surveys, e.g.

formal and structured interviews. A study by Easterbrook [Eas07] identifies and compares *five classes of research methodologies* that are applicable in software engineering research:

- Controlled Experiments (including Quasi-Experiments)
- Case Studies (both exploratory and confirmatory)
- Ethnographies
- Survey Research
- Action Research

Research carried out in the form of *controlled experiment* is concerned with finding a cause-effect relationship between independent and dependent variables which are clearly defined and stated in form of testable hypothesis. The attribute controlled implies that all necessary measures need to be taken to ensure that the dependent variables are only affected by the independent ones. In case this is not possible, e.g. data analyzed in time-series way for events that have already occurred in discrete time periods, term quasi experiment is used instead of controlled experiment. Easterbrook [Eas07] advises that in this case a more careful interpretation is required.

Runeson and Höst [RH09a] present guidelines for conducting and reporting *case studies* which is defined as an empirical study applied when investigating contemporary phenomenon in natural context. They also argue that ethnographic studies [Eas07] are a special case of case study with focus on cultural practices. In addition, [RH09a] argues the case study methodology can be applied for all of the following purposes as defined by Robson [Rob02]:

- Exploratory - Trying to scope nature of investigated phenomenon.
- Descriptive - Portraying an investigated phenomenon.
- Explanatory - Seeking explanation of a phenomenon.
- Improving - Improving some aspects of a phenomenon.

According to Easterbrook [Eas07] the *survey* methodology is applied when characteristics of a broad population of individuals need to be identified. The survey data needs to be collected from a representative sample of a well defined population and can be carried out as unstructured, semi-structured, or structured interviews [Rob02]. Some of the challenges associated with *survey* methodology are ensuring that the questions are well formulated, i.e. they do not lead to ambiguous interpretations, as well as that the sampling bias is minimized so the results can be generalized.

An *action research* methodology is used when trying to solve a real world problem while at the same time studying the experience of solving the problem [Dav+04]. Thus, it can be argued that action research is closely related to case study research with the main difference being that action research also aims to propose a solution related to an investigated phenomenon. Easterbrook [Eas07] points out action research challenges arguing that it is an immature empirical research methodology supported by vague and subjective evaluation framework proposed by Lau [Lau99].

A study by Wieringa and Morali [WM12] proposes *technical action research* in which three distinct roles are defined: artifact designer, artifact developer, client helper with the overall goal of enhancing research rigor and relevance by bridging gap between ideal conditions of an artifact design and concrete conditions that occur in real-world problems.

3.1 Classification of Included Papers

The research presented in this thesis, shown in Figure 4 , is based on the empirical research method, which according to Easterbrook [Eas07] implies that the research questions are related to the class of *knowledge questions*, i.e., the questions focused on the observable and measurable state of the world.

Paper I through Paper V are of exploratory nature, which according to Eeasterbrook, is typical for the early stages of the research, when an attempt is made to understand the studied phenomena. According to Kitchenham et al. [Kit+02], the exploratory studies are an important instrument for formulation of hypothesis questions and an aid in the planning of the future research activities. The exploratory research conducted in this paper is of qualitative and quantitative nature. Since software engineering involves a human factor, according to Seaman [Sea99], qualitative studies are necessary to study complex phenomena such as the ones that involve human behavior. Seaman also recommends to complement qualitative methods with quantitative methods. Qualitative and quantitative empirical software engineering can be conducted in the form of systematic reviews, a surveys, action research, experiments, or case studies.

According to Kitchenham et al.[Bre+07], systematic reviews can be used, among others, to provide a framework to appropriately position new research. A systematic review is divided into three stages; the planning, the execution, and review phase during which review protocol is created and validated, appropriate relevant research is identified, assessed for quality, and synthesized. Paper I is conducted as a systematic literature review to provide background on the current state of research with respect to industry roles within the OSS world.

The research presented in Paper II was conducted in the form of a focus-group meeting, which according to Kontio [Kon+04], is an effective method to obtain qualitative insights and practitioner feedback. However, as the data obtained in such way is limited in time and scope, it is suggested that this type of research be

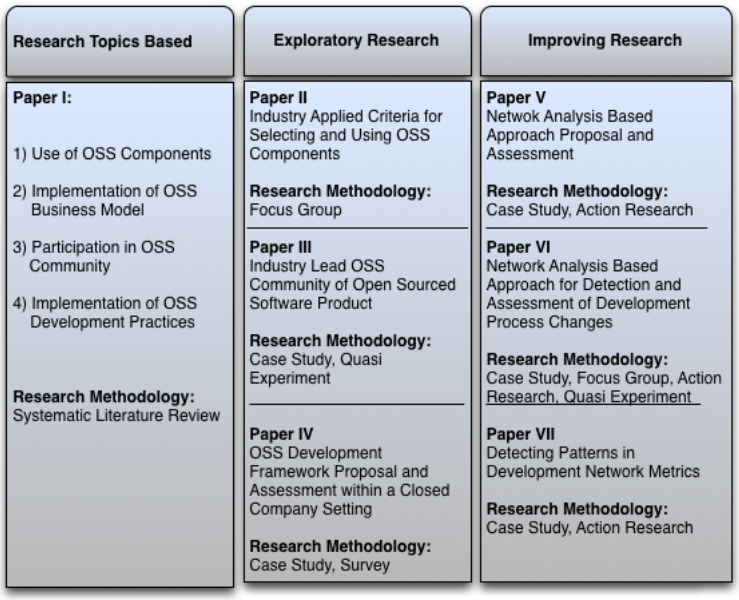


Figure 4: Research methodology and nature

complemented by another more rigorous methodology. The results of the Paper II, have been shown to complement the result obtained from the systematic review presented in Paper I.

The research in Paper III is conducted as a case study with a quasi-experiment component, which according to Eeasterbrook [Eas07], is a variant of an experiment performed when the conditions for a true experiment are not feasible. Since the event of open-sourcing the Ingres solution was performed in the past, and the event could not be recreated, this quasi-experiment methodology was deemed as appropriate to use. Paper IV, is conducted as a case study with survey elements. According to Runeson and Höst [RH09a], case study is appropriate methodology to use when observing a phenomenon within its natural context, and it can be combined with a survey.

The work presented in the Papers V, VI, VII is conducted as case study and action research, which according to Wieringa [Wie12], consists of developing new techniques for software engineering and evaluating them for the purpose of continues improvement. In paper V we propose a new approach to the application of social network analysis for the purpose of assessing committers networks. Hence, the action research is done with respect to finding appropriate approach to construct the committers' networks and then testing the approach with Android. In

Paper VI, the same network analysis method is applied within a large division in Ericsson in order to understand if it can be used to monitor organization and development process changes the division underwent in a one year period. In Paper VII the same network analysis method is used to construct and study development networks for over 250 projects hosted under the Apache Software Foundation with aim of finding patterns in network metrics for all of the studied projects. The purpose of Papers V,VI, and VII is improving with overall goal to help software practitioners in properly assessing underlying development structures and monitoring software development effort.

Table1 provides a summary of research methodology type and research method used. Figure 4 provides an overview of the thesis focus with respect to the research methodology used. More specifically, it shows links between results of earlier studies in context of them being used as research topics in later studies. Hence, e.g., the results of the study presented in Paper I, were used as research topics in Paper II, Paper III, and Paper IV. Further more, findings presented in Paper III and Paper IV motivated research topics of studies presented in Papers V,VI, and VII.

Table 1: Research Type and Method Used in the Papers

Work	Research Type	Research Method
Paper I	Exploratory	Systematic Literature Review
Paper II	Exploratory	Focus-group Meeting
Paper III	Exploratory	Case Study and Quasi-Experiment
Paper IV	Exploratory	Case Study and Survey
Paper V	Improving	Case Study and Action Research
Paper VI	Improving	Case Study and Action Research
Paper VII	Improving	Case Study and Action Research

4 Results

This sections presents the results of the conducted research from each of the included papers.

4.1 Paper I: A Systematic Review of Research on Open Source Software in Commercial Software Product Development

The aim of this study was to conduct a comprehensive systematic review of research on usage of OSS components and OSS development practices (OSDP) within the commercial context as well as the industry participation in the OSS communities. We have identified and reviewed a total of 495 articles, 357 of

which were found through an automated search of INSPECT and COMPENDEX databases which include articles from major conferences, journals, and publishers (e.g. IEEE, ACM, Springer, IEE). The remaining 138 reviewed publications were identified through a manual search and include all, at the time available, articles from the Conference on Open Source Systems. By applying rigorous methodology, we have identified the 23 most relevant publications that can be divided into four categories: OSS as a part of component based software engineering, business models based on OSS, company participation in open source communities, and usage of OSDP within a company setting. The research methodologies used in the identified articles are equally divided between case study and survey. The results of the research by Lundell et al. [Lun+06] show that 75% the companies that use the OSS components for development purposes, also participate in the OSS communities. Another research shows that 6-8% of the Linux Debian GNU code base is contributed by companies [Rob+07]. The conducted research also shows that more research is needed on how OSS communities function [Bon+07]. While there is evidence of successful business models built around OSS, especially for hybrid models [Wes03], there exist many challenges in sustaining development communities. According to Koening [Koe09], open source is conducive to creating competitive advantage by shifting from traditional revenue model based around a program's functionality, and focusing on service oriented revenue models. The results of the systematic review show that companies engage in an informal process when it comes to selecting an OSS component to use internally [Li+09]. Finally, studies conducted on using OSDP within a company setting can facilitate innovation Gurbani et al. [Gur+05], Lindman et al.[Lin+08b].

4.2 Paper II: Usage of Open Source in Commercial Software Product Development - Findings from a Focus Group Meeting

The objective of this study was to get relevant industry views on prerequisites for using OSS components within a commercial setting. For this purpose, a focus group meeting was held with industry representatives, and the meeting's discussion was based around predefined questions. The questions had two primary concerns: to understand the selection process of OSS components, and to understand how modifications of OSS components were handled. The discussion input was collected in form of notes, that were later classified and summarized, and thus results of the research were presented. The main identified topics include:

1. Concerns related to the discovery process of candidate OSS components'.

2. Management of the decision making process pertaining to the selection of best fit OSS candidate with respect to legal, technical, and community support aspects.

3. Practices and issues with respect to management of OSS component modifications.

4. Advantages of participating in the OSS community.

The findings of the research are in line with the findings from the literature review in Paper I. The results also confirm that when a company decides to use an OSS component, it is important to establish ties with the respective OSS community in order to acquire skills needed to maintain the component. In order to ensure compatibility with future releases of the OSS component, it is important for a company to give back changes made to the OSS component to the OSS community.

4.3 Paper III: A Case Study on the Transformation from Proprietary to Open Source Software

Paper III presents a case study on the transition process of the Ingres database management system from proprietary to open source. The research provides an insight into business motivation to open source the proprietary solution, concerns and issues that need to be taken into account during the open-sourcing process, as well as an outcome of a well planned transition process. The focal point of this research is measuring change in software quality metrics, e.g., effective lines of code, cyclomatic complexity, and file function count between the proprietary and the open source community modified version of the product. The software metric changes were tracked for separate groups of the Ingres source code modules grouped by their functionality into the backend, front end, common, and utilities components. The research shows that the majority of the new code added and changed by the Ingres open source community was located in the front end module, while the smallest number of changes were made to the backend module. The software quality metrics such as cyclomatic complexity, effective lines of code were improved after the transition to open source. The open sourcing process resulted also in a 100% increase in customer base, and 32% increase in revenues. While the investigated software presents just one case of the proprietary software product open-sourcing process, and thus can not be freely generalized, it offers a valuable insight into open-sourcing concerns, business motivation, and the effect of community on the company sponsored open source product.

4.4 Paper IV: A Prolonged Two Phase Case Study on Implementation of Open Source Development Practices within a Large Company Setting

Based on previous research [Gur+06], [MM08], [Lin+08b], and motivation to explore the applicability of open source development practices within an industry

context, a prolonged, two phase case study was designed for this purpose. The study was conducted in a large software and hardware developing company that bases its software products on a large open source solution. Through a long involvement with the open source project, the company has modified some of its development practices to be more in line with the open source community development processes, and thus represented a good case candidate to study the applicability of the OSDP in a closed company setting. The purpose of the first phase of the study was two-fold: identification of a common open source development framework based on evidence and practice, and the assessment of the framework compatibility with the case company's development processes. The author of the thesis was granted company and network access to study the company's portal and internal documents related to past and ongoing projects, as well as development practices and guidelines. The results of the portal and documentation review were validated through semi-structured interviews conducted with the case company's employees. The research results show existence of the processes and roles compliant with the open source development, such as common development portal and electronic communication infrastructure, formally are present in the company. However, in practice they are very little used, and the development process seems to fit more traditional development than the open source context. Furthermore, results of the research show existence of roles in line with the roles found in open source communities, but with additional and conflicting tasks, e.g., ensuring soundness of technical solution and not sacrificing it due to a time constraint and being project lead working under the time constraint.

The second phase of the research, conducted as focus group meeting with the same participants that were involved in the first phase, had overall goal of understanding changes and the underlying causes in the level of implementation of OSDP over period of two years. Results of the second phase of the study show that the company's development practices have higher level of alignment with OSDP and that the change was driven by need to more efficiently collaborate with the company's new, globally distributed development site. The case study is relevant as it shows what happens when implementation of OSDP within company is driven primarily by employees who have been working under the OSS process and have recognized its benefits. Hence, since the implementation was not carried out in a systematized way, the benefits of inner source could not be noted in the short run, but rather in the long run when the new development site was opened. The findings of the study are relevant since with growing usage of OSS by industry, more companies could find themselves in the same position, implementing some aspects of the OSDP, and thus can learn from this example.

4.5 Paper V: Network Analysis of a Large Scale Open Source Project

Research conducted in Paper V, was motivated by a need to define a quantifiable approach for assessing development communities in terms of source code committers' structure, influence, centrality, and cross project collaboration. The study focuses on the Android Open Source project community, an interesting case to examine, not only because of the OSS product's broad use by almost entire mobile eco-system [Goo13] and its leading market share position [GS13], but also since it is built as an OSS stack, thus including over 150 other open source projects. This work also proposes a new approach for using social network analysis to study open source project committer networks. The new approach is the result of a study on social network analysis theory and the existing research within the field. By applying the existing analysis procedure on the data extracted from the Android source code repository, we came to the conclusion that the results do not accurately represent the studied community. The main reason for such outcome, lies in the fact that the existing social network analysis procedures, study committer networks from a perspective which does not take into account the committers' weights relative to the source code change event. The results of the research show that in an industry sponsored open source project, the company exhibits large control over the source code, even in the other OSS projects that are included in the stack and whose communities are not led by the company. The implications of such finding are relevant as they show one example of how a large company through an OSS project can impact industry participants, as well as assert influence in projects that are not directly under its guidance. This information is relevant for the companies who plan to undertake similar projects or join an OSS community. The proposed approach can also be applied to study structure and evolution of any software development community.

4.6 Paper VI: Development Process Monitoring Through Application of Network Analysis on Source Code Repository Data

Paper VI explores fitness of the network analysis based approach to detect changes in developers' networks in relation to developments' organization structure and process level changes introduced in a large division of Ericsson. For this purpose, the company's source code repository was mined to extract data on all the source code commits and to construct corresponding weighted and directed developers' networks. Network metrics such as weighted average degree, graph density, clustering coefficient, etc, were calculated for three discrete three month long periods as well as for the entire length of the project. The metrics were then presented and discussed through three focus group meetings attended by 5 to 6 company representatives. The company representatives were knowledgable of the process and

organization level changes introduced in the company, and participated either in managerial or technical roles on the development projects. The results of the study show that the calculated network metrics are indicative of the project work carried out during the specified periods, e.g. network modularity metrics properly identified the number of distinct development projects being worked on. The metrics also properly reflect development networks which are sparsely connected, indicating very low degree of collaboration between the developers and very high level of specialization. The feedback received in the focus group meeting validated the fitness of the approach in this one case study, as well as pointed that monitoring of development network metrics could be useful as a monitoring tool for ongoing project work.

4.7 Paper VII: Benchmarking Apache Software Foundation Projects: Network Analysis of the Contributors' Collaboration Networks

The research carried out in Paper VI pointed to absence of network metrics for other similar development projects that Ericsson metrics could be compared to. This motivated work in Paper VII which constructed developers' networks for over 249 mature and wide-industry used projects that were hosted under the Apache Software Foundation and calculates associated network metrics. The metrics were then analyzed using standard statistics tools to detect any correlation between size of the OSS community with calculated metrics such as developers' centrality and degree distributions and clustering coefficients. The clustering coefficient for all the different types and sizes of the project show high values, ranging from 0.62-0.8, given that the maximum value for the clustering coefficient is 1. The results of the research show high correlation of project size, measured by the number of developers that have worked on the project, with developers' betweenness and closeness centralities. The results also show high degree of specialization of the majority of developers on smaller parts of the system. At the same time there exist very few expert developers that bind larger parts of the system together. We argue that such network topology could be indicative of the preferred way to structure development of large-scale software development projects.

5 Synthesis

This section presents a synthesis of the results of the thesis with respect to research questions discussed in the section 2.

RQ1: What interactions exist between different industry roles with respect to OSS?

Based on the systematic literature review presented in Paper I, the industry roles with respect to OSS can be divided into four distinct categories: usage of OSS

components, participation in OSS communities, implementation of OSS business models, and application of open source development practices. Each of the roles as presented in Figure 3 facilitates furthering of the relationship between a company and an OSS community. As discussed in Paper II, the companies using OSS products are motivated to participate and contribute code to an OSS community due to future update of functionality and community support for the product. There exist evidence that a company that has built competence and knowledge of the open source development practices, tends to implement some aspects of the open source development, as is discussed in the Paper IV. Large global companies have recognized business opportunities with respect to OSS, and thus there exist examples of companies, e.g. Red Hat [Red16b], Sugar CRM [Sug16], Ubuntu [Ubu16], [Alf16], that have built successful business models around an OSS product such as the one presented in Paper III.

RQ2: What are the major characteristics of OSS development framework?

The research presented in Paper IV shows that the most important aspects of large and active OSS communities are well organized supporting portal infrastructure, clear communication norms, and governance models. The portal infrastructure enables geographically distributed participants having different roles in the community, to easily access the product as well as get involved in its development. This means that community portals of mature OSS software products are intuitive and easy to navigate and use in order to attract and retain participants. The communication norms are clearly defined and information about product documentation, planned and ongoing work, new releases, security issues, bug reports, and mailing lists and archives are easily accessible and up to date. Community facilitates friendly and pleasant communication environment so participants are not discouraged from participating and contributing to the project. The community governance adheres to meritocracy rules when assigning roles to community members, thus community respect is earned though demonstrated experience and presence on the project.

RQ3: What are the perceived benefits and drawbacks of OSS development process implementation within a closed development environment?

In Paper IV we present a case study of a large and global software and hardware company that bases its products on an OSS product, and which has over years built competence with the OSS community. As a consequence, the company has formally adopted some of the open source development practices, but in reality this partial or superficial adoption has created some issues, such as roles with conflicting tasks. For example, the structure of the organization was changed and code guardian positions were put in place, but besides their primary responsibility of ensuring high code quality, they were also under time pressure to deliver. Such conflicting roles can significantly diminish the primary purpose of the position. In the follow up study with the company, a higher level of alignment was found with respect to the defined open source development framework. However, the increase in the alignment was necessitated by need since the company had opened another

large development office on another continent in the mean time. This speaks for open source development process being facilitating to needs of distributed software development effort.

A company culture that is not acquainted with the operating culture of an OSS community can discourage online communication approach and instead replace it with "go talk to the expert approach". The research presented in Paper IV shows that this behavior can unnecessarily overburden knowledge experts, taking their time from other projects. Some of the experts have resorted to creating online documentation for FAQs, whose existence is one of the standard features of OSS communities. The research conducted in HP [MM08], Lucent [Gur+06], and Nokia [Lin+08b] shows that implementation of open source development within a company setting can be beneficial, especially in terms of standardization of development tools and processes across an organization, higher rapid team redeployment, improved code quality, lower maintenance costs and increased innovation. The main differences observed between traditional and open source development practices exist in transparent communication process and constant feedback loop between the core developers and beta testers of the OSS product. Therefore, as there exist many demonstrated benefits for inclusion of some of the open source development practices, companies tend to implement them only through necessitated need, rather then through a planned process.

Some of the drawbacks stem from the fact that commercial organizations deal with a limited pool of development resources, which is not the case in true OSS environment where a potential development pool is much larger. Some of the companies have resorted to earlier discussed progressive open source (POS), so that a part of development effort is opened for its technology partners, enabling the software partners to partake in continuous feedback loop and report bugs and new feature requests. While decisions on new product features in OSS community are decided through a consensus based on the needs of the community members, e.g. Apache Software Foundation, commercial development effort normally involves higher management structure and product owners, each with their own needs for new feature requests which might not be aligned with the customers' needs.

RQ4: How can the proposed network analysis based approach be used to understand and monitor development structures and associated development metrics?

The results of the research presented in Paper V show that the proposed method for constructing weighted and directed developers' network and its associated metrics can be used to properly assess development structures and influences of OSS communities. The approach was applied in a large division of Ericsson in Paper VI and it was shown to be a very effective tool to monitor changes to developers' networks caused by implemented changes in the development organization and development process. Further more, the network approach when applied on large scale in study VII has shown that some common metrics can be found across 249 distinct, wide-industry used OSS projects.

The results of the implemented network analysis when applied to the OSS communities can help companies better understand the underlying development structures, development trends, and influencers, information that can be useful when making decisions on joining an OSS community, or forming alliances. When applied internally, within a closed development setting, the method can be used to monitor the development process, uncover the most valuable contributors, understand development clusters and cliques. This information can be used when deciding which are the most appropriate resources to be involved in future projects, or for contingency resource planning. Finally, the same method was applied on a large set of mature open source projects, and network metric patterns related to developer centrality metrics were identified. These metrics can be compared against other similar size and type development projects in order to identify differences and analyze their root cause. Large differences could be indicators of software development structures that might not be organized in the most efficient way.

6 Threats to Validity

Identification of the threats that can potentially jeopardize a research validity is of utmost importance, especially in the field of empirical software research context where observations and measurements of the studied phenomena are conducted in a natural context. The research questions presented in this thesis are analyzed for validity threats based on the classification proposed by Wohlin et. al [Woh+12]. The threats to validity category are divided into four main types: *construct, internal, external, and conclusion*.

Construct validity is related to the relationship between the concepts and theories behind the research and observations. Even if it is shown that the causal relationship between the two exists, we need to question weather the measurement tools are appropriate for the investigated subject of the study. There is a risk that terminology used in academia might be misunderstood by industry practitioners or that researchers might lead practitioners to respond in an assumed way. The selection of the interviewees might be biased, and thus the way their answers might be unbalanced or limited. The following steps were taken to lessen the risks:

- *Selection of interviewees:* In order to obtain a balanced set of interviewees, roles covered span a wide range, from upper level management, and middle managers, to more technical roles, such as developers, architects, testers, and source code guardians.
- *Design of interviews:* The interviews were designed based on research questions, and validated by other researchers. The interviews conducted in semi-structured way provided opportunity to discuss and further clarify questions.
- *Prolonged involvement:* The selected companies provided one of the researchers with long-term accommodation. Hence, at least one of the re-

searchers was seated at the companies and given access to necessary company resources. This helped establish a stronger relationship with the companies' participants, based on openness and honesty.

- *Reactive bias:* Presence of a researcher and knowledge of the topic of the study might hinder the results, as correspondents might provide answers in accordance to assumed expectations. To reduce the risc, all interviewees were granted anonymity, and were not given any rewards for their participation.

In Paper IV we base our research on the comparison of the common characteristics of the OSDP with the development practices of the case company. The possible construct validity threats exist in form of inappropriate identification of OSDP characteristics, also referred to as OSDP framework, and inappropriate assessment of the case company's development practices. To reduce the threats, the OSDPs characteristics were defined based on relevant works and assessment of a mature and large open source community. In addition, the author of the thesis spent two months within the company, and was granted access the company's internal electronic resources which is also known as prolonged involvement [Run+11], a practice used to improve validity of the research. The result were validated through a semi-structured survey whose results were coded, buy the second researcher, and a company employee in a senior technical position.

Internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. The case study presented in the Paper III includes a quasi experiment, that examines the effects of the source code modifications, made by the open source community, on the static source code quality metrics. One of Lehman's laws [Leh80] states that a product which is not rigorously adapted or changed will, over a period of time, see decrease in software quality metrics. However, since there is no available data on the average change in software quality metrics for the studied type of the software we can not compare the observed change to some average change value. However, since the transition into the open source community was a major event in terms of software maintenance, it is probable that the transition had affect on the software quality metrics.

External validity is related to the ability to generalize the results of the this study. The research presented in Paper II is based on a focus-group meeting whose participants were industry representatives. Hence, the results of the research are based on the personal opinions, which may not be in line with a view of the organization they represent. Thus, there exist a risk that these results can not be generalized or that they might not be applicable to other organizations. According to Robson [Rob02], the extreme individual opinions tend to be offset by group reactions to them, and group dynamics can be facilitating to focus discussion on relevant issues. In the context of the thesis, the findings of the focus-group are in line with the results from the systematic review presented in Paper I, on the concerns commercial organizations have with regard to selecting an OSS component.

The research presented in Papers V, VI, VII applied weighted and directed network based approach on broad sets of projects which included OSS projects and proprietary ones: Android OSS stack, a closed-source repository in Ericsson, and on over 250 projects hosted under the Apache Software Foundation. This raises the aspect of generalizability.

Conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. The static software quality measures, as presented in Paper II, did not follow normal distribution, and thus in their analysis test of lower statistic power then the t-test were used. However, since the number of the analyzed data points can be considered high, the chance of detecting difference in distributions even when using non-parametric tests, is high as well.

7 Conclusion and Future Work

In the last couple of decades, the OSS software phenomena has increasingly gained support from commercial organizations due to the recognized potential to increase software development efficiency and facilitate creation of new business models. As a result, the perception of the OSS movement has shifted from an informal hobbyist software playground to a viable, mainstream industry value creator. In the systematic literature review presented in Paper I we identify four distinct roles industry takes with respect to OSS communities which further the industry engagement with the OSS community. The commercial organizations include OSS components in their products or use the products for internal purposes as presented in Paper II. The usage tends to create a relationship between the company and the community for the reasons of product support and maintenance. There exists evidence of successful transitions of OSS products from proprietary to open source community, and successful business models built around such communities as presented in Paper III. By participating in OSS communities, a company can develop expertise in open source development process and practices and decide to implement some of the practices as presented in Paper IV.

Given the broad impact of the OSS movement on the industry, the ability to understand the structure and the evolution of an open source community is an important factor that should be considered when a commercial organization plans to work with an OSS community. Paper V proposes a new network theory based approach to study development communities and applies the approach to the study of the Android OSS project. The paper also demonstrates the importance of properly forming the edge weights between the developers relative to the total number of changes done on a file. When the weights are formed as mere sums of common contributions done to a file, in communities where a strong influence of very few major contributors exists, the resulting networks do not accurately represent the underlying development community. Paper VI applies the weighted and directed

approach to proprietary source code base in a branch of Ericsson detecting changes in the underlying development network that were introduced by reorganization of development teams. Paper VII shows that by applying the same network analysis approach on a set of 249 projects hosted under the Apache Software Foundation yields a set of associated network metrics that indicate existence of patterns with respect to the sizes of the projects. The uncovered patterns also indicate that across the OSS development communities, development work is carried so that the majority of developers are specialized on smaller parts of the project, with only few developers with high centrality metrics, serving as a "glue" connecting distinct parts of the projects. More work is needed to understand how the data from such analysis can be used to predict the future behavior of development communities.

The industry acceptance of the OSS has changed the way software is produced and marketed. From the research stand point, the rich and growing OSS communities provide much of the archived data that lends itself to further study. The contribution of this thesis is demonstrated effectiveness of the proposed network analysis approach to study development structures and deliver results that can be used by business in creation of their own OSS strategies. Understanding structures of development communities, the main influencers and changes in associated metrics is valuable when companies plan to include OSS products. This more so, as the research has shown that inclusion of OSS components into company products creates a long lasting relationship between the company and the community. The company depends on the community for the new versions, especially the ones containing security patches, while at the same time wants to ensure that any modifications it has made to the product are included in the latest releases. Hence, understanding the development fabric is equivalent to understanding the community leaders and trend setters.

Besides aiding business in understanding the OSS community structure influencers, the network analysis approach can also be used to improve monitoring of internal development efforts. Development structures not only uncover the most valuable resources, but can also show how development structures and influences change over time. This information can be also valuable when planning future projects as it brings raised understanding on best fit resources for the planned work.

Finally, the network based approach can be used to benchmark any development effort against the results of the study presented in Paper VII. Scarce networks, with centrality and degree metrics not following power a law distribution may indicate development structures that are not organized in the most efficient way.

More replicated studies on greater number of projects are needed to validate the research results as well as to fully understand the potential applicability of the metrics provided through network analysis based approach. Future work should also consider studies of the metrics though formal time series analysis methods which could uncover some evolutionary patterns in associated metrics. Ideally, an automated tool should be created to extract, analyze and present the results of the network analysis.

As previous research has shown that team structures and code architecture modulate each other, a potential interesting area of future work could be application of the used weighted and directed approach on the source code, so that networks are created with classes as nodes and in and out links constructed and weighted based on the predefined inter-class dependancies. Then, the two network topologies could be further analyzed for similarities and differences on larger number of projects. If indeed a strong correlation is shown between the two network topologies, then this information could serve as a base for planning and monitoring the team structures for preferred architectures.

Through the work done in this thesis, we have shown that OSS has in its rather short life span penetrated mainstream software development arena, which was just until a decade ago considered a place that was primarily reserved for large-scale closed source development effort. Not only did the OSS development practices found their natural fit in a world of globally distributed software development, but the availability of mature OSS products served as catalyzer for value creation though various business models. Dynamics of the OSS and industry synergy require proper formal methods to assess the OSS development structures and processes so the information can be used to improve planning of development effort and market positioning.

References

[AGVP00] Rishab Aiyer Ghosh and Vipul Ved Prakash. “The Orbiten Free Software Survey”. In: *First Monday, Peer Reviewed Journal on the Internet* 5.7 (2000).

[Alf16] Alfresco Software, Inc. *Alfresco*. `http://www.ubuntu.com`. 2016.

[AL13] Mohammad Y. Allaho and Wang-Chien Lee. “Analyzing the social ties and structure of contributors in open source software community”. In: *Advances in Social Networks Analysis and Mining 2013, ASONAM ’13, Niagara, ON, Canada - August 25 - 29, 2013*. 2013, pp. 56–60.

[Aug16] Larry M. Augustine. *Hewlett Packard Enterprise Development LP*. `http://site05.goscon.org/presentations_2005/Larry\%20Augustin-Medsphere.pdf`. 2016.

[Bon+07] Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Cristina Rossi. “Business Firms’ Engagement in Community Projects - Empirical Evidence and Further Developments of the Research”. English. In: *Proc. International Workshop on Emerging Trends in FLOSS Research and Development*. 2007, pp. 57 –61.

[BP00] Stephen Bradley and Kelley Porter. “eBay, Inc.” In: *Journal of Interactive Marketing, no. 4* (2000).

[Bre+07] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. “Lessons from applying the systematic literature review process within the software engineering domain”. In: *Journal of Systems and Software* 80.4 (2007), pp. 571–583.

[Con10] Andreas Constantinou. *Symbian is dead. Long live Symbian*. `http://www.visionmobile.com/blog/2010/10/symbian-is-dead-long-live-symbian/`. 2010.

[CH05] Kevin Crowston and James Howison. “The social structure of free and open source software development”. In: *First Monday* 10.2 (2005).

[Dav+41] A. Davis, B. Gardner, and M.R. Gardner. “Deep South; a social anthropological study of caste and class.” In: *Chicago, IL, US: University of Chicago Press Deep South; a social anthropological study of caste and class*. 1941, p. 558.

[Dav+04] Robert M. Davison, Maris G. Martinsons, and Ned Kock. “Principles of canonical action research”. In: *Inf. Syst. J.* 14.1 (2004), p. 65.

[Dij83] Edsger W. Dijkstra. “The structure of the multiprogramming system”. In: *Communications of the ACM* 26.1 (Jan. 1983), pp. 49–52.

[Din+02] Jamie Dinkelacker, Pankaj K. Garg, Rob Miller, and Dean Nelson. “Progressive open source”. In: *Proceedings of the 22rd International Conference on Software Engineering, ICSE 2002, 19-25 May 2002, Orlando, Florida, USA*. 2002, pp. 177–184.

[Eas07] Steve M. Easterbrook. “Empirical research methods for software engineering”. In: *IEEE/ACM International Conference on Automated Software Engineering*. 2007, p. 574.

[Fag86] Michael E. Fagan. “Advances in Software Inspections”. In: *IEEE Trans. Software Eng.* 12.7 (1986), pp. 744–751.

[Fit06a] Brian Fitzgerald. “The Transformation of Open Source Software”. In: *MIS Quarterly* 30.3 (2006), pp. 587–598.

[Fog05] Karl Fogel. *Producing open source software - how to run a successful free software project*. O’Reilly, 2005, pp. I–XX, 1–279.

[Fou15c] Linux Open Foundation. *Linux Operating System*. `http://www.linuxfoundation.org`. 2015.

[Fou16] The Apache Software Foundation. *Apache Project*. http://www.apache.org. 2016.

[Fre16a] Free Software Foundation, Inc. *About FSF*. `http://www.fsf.org/about/`. 2016.

[Fre16b] Free Software Foundation, Inc. *GCC, the GNU Compiler Collection*. `https://gcc.gnu.org`. 2016.

[GS13] Market Analysis Gartner and Statistics. *Market Share: Mobile Devices, Worldwide, 1Q12*. `http://www.gartner.com/newsroom/id/2017015`. 2013.

[Gla95] Robert L. Glass. "A structure-based critique of contemporary computing research". In: *Journal of Systems and Software* 28.1 (1995), pp. 3–7.

[Gla+02] Robert L. Glass, Iris Vessey, and Venkataraman Ramesh. "Research in software engineering: an analysis of the literature". In: *Information & Software Technology* 44.8 (2002), pp. 491–506.

[GT00a] Michael W. Godfrey and Qiang Tu. "Evolution in Open Source Software: A Case Study". In: *2000 International Conference on Software Maintenance, ICSM 2000, San Jose, California, USA, October 11-14, 2000*. 2000, pp. 131–142.

[GC94] Benjamin Gomes-Casseres. "Group vs. Group: How Alliance Networks Compete." In: *Harvard Business Review* (1994).

[Goo13] Google. *Android Open Handset Alliance Members*. `http://www.openhandsetalliance.com/oha_members.html`. 2013.

[Gur+05] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. "A case study of open source tools and practices in a commercial setting". In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), pp. 1–6.

[Gur+06] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. "A case study of a corporate open source development model". In: *ICSE*. 2006.

[Her+03] Guido Hertel, Sven Niedner, and Stefanie Herrmann. "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel". In: *Research Policy* 32 (2003), pp. 1159 –1177.

[Hew16] Hewlett Packard Enterprise Development LP. *CHP and IBM Software Group Alliance*. `http://h22168.www2.hp.com/us/en/partners/ibm/`. 2016.

[H+14] Martin Höst, Klaas-Jan Stol, and Alma Oručević-Alagić. "Inner Source Project Management". English. In: *Software Project Management in a Changing World*. Ed. by Gunther Ruhe and Claes Wohlin. Springer Berlin Heidelberg, 2014, pp. 343–369.

[Kit+02] Barbara A. Kitchenham, Shari Lawrence Pfleeger, David Hoaglin, Khaled El Emam, and Jarrett Rosenberg. "Preliminary Guidelines for Empirical Research in Software Engineering". In: *IEEE Transactions on Software Engineering* 28 (8 2002), pp. 721–734.

[KC07] Barbara Kitchenham and S. Carter. *Guidelines for performing systematic literature reviews in software engineering, v. 2.3*. Tech. rep. Keele University and University of Durham, 2007.

[Koe09] John Koenig. *Seven Open Source Business Strategies for Competitive Advantage*. `http://www.cs.up.ac.za/cs/aboake/sws780/references/designapproaches/collaborative/Koenig-SevenOpenSourceStrategies.pdf`. 2009.

[Kon+04] Jyrki Kontio, Laura Lehtola, and Johanna Bragge. "Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences". In: 2004, pp. 271–280.

[LW05] Karim Lakhani and Robert Wolf. "Why Hackers Do What They Do: Understanding Motivation and Effort in Free Open Source Software Projects." In: *Perspectives on Free and Open Source Software*. Ed. by Joesph Feller, Brian Fitzgerald, Scott Hissam, and Karim Lakhani. MIT Press, 2005.

[Lau99] Francis Y. Lau. "Toward a framework for action research in information systems studies". In: *IT & People* 12.2 (1999), pp. 148–176.

[Leh80] Meir M. Lehman. "On understanding laws, evolution, and conservation in the large-program life cycle". In: *Journal of Systems and Software* 1 (1980), pp. 213–221.

[LR01] Meir M. Lehman and Juan F. Ramil. "Rules and Tools for Software Evolution Planning and Management". In: *Ann. Software Eng.* 11.1 (2001), pp. 15–44.

[Li+09] Jingyue Li, Reidar Conradi, Christian Bunse, Marco Torchiano, Odd Petter N. Slyngstad, and Maurizio Morisio. "Development with Off-the-Shelf Components: 10 Facts". In: *IEEE Software* 26.2 (2009), pp. 80–87.

[Lin+09a] F. van der Linden, B. Lundell, and P. Marttiin. "Commodification of Industrial Software: A Case for Open Source". In: *Software, IEEE* 26.4 (2009), pp. 77–83.

[Lin+08b] Juho Lindman, Matti Rossi, and Pentti Marttiin. "Applying Open Source Development Practices Inside a Company". In: *International Conference on Open Source Systems, OSS* (2008), pp. 381–387.

[Lin16] Linux Foundation. *Linux Foundation Board Directors*. `http://www.linuxfoundation.org/about/board-members`. 2016.

[LF+06] Luis López-Fernández, Gregorio Robles Robles, Jesús M. González-Barahona, and Israel Herraiz. "Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects". In: *International Journal of Information Technology and Web Engineering* 1.3 (2006), pp. 27–48.

[Lun+06] Björn Lundell, Brian s Ling, and Edvin Lindqvist. "Perceptions and Uptake of Open Source in Swedish Organizations". In: *nternational Conference on Open Source Systems, OSS*. 2006, pp. 155–163.

[McK99] Marshall Kirk McKusick. "Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributablet". English. In: *Open Sources: Voices from the Open Source Revolution*. Ed. by Chris DiBona, Sam Ockman, and Mark Stone. O'Reilly Media, 1999. Chap. 3.

[MM08] Catharina Melian and Magnus Mähring. "Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard". In: *OSS*. 2008, pp. 93–104.

[Mor34] Jacob Levy Moreno. "Who Will Survive". In: *Beacon House, Beacon, NY* (1934).

[New13] Mark Newman. *Networks*. Oxford University Press, 2013.

[OSI13a] OSI. *Open Source Initiative Board*. `http://opensource.org/board`. 2013.

[OSI13b] OSI. *Open Source Initiative Mission Statement*. `http://opensource.org/about`. 2013.

[Ora16] Oracle Corporation, Inc. *MySQL*. `https://www.mysql.com`. 2016.

[OAH10] Alma Oručević-Alagić and Martin Höst. "A Case Study on the Transformation from Proprietary to Open Source Software". In: *OSS*. 2010, pp. 367–372.

[OAH14b] Alma Oručević-Alagić and Martin Höst. "Network Analysis of a Large Scale Open Source Project". In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, August 27-29, 2014*. 2014, pp. 25–29.

[Par72] David Lorge Parnas. "On the Criteria To Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* 15.12 (1972), pp. 1053–1058.

[Per05] Bruce Perens. "The emerging economic paradigm of Open Source". In: *First Monday 10(special issue 2: Open source)* (2005).

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[Red16a] Red Hat, Inc. *JBoss Development Community*. `http://www.jboss.org/technology/`. 2016.

[Red16b] Red Hat, Inc. *Rad Hat Open Source Project*. `http://www.redhat.com/en`. 2016.

[Rob+05] Gregorio Robles, Juan José Amor, Jesús M. González-Barahona, and Israel Herraiz. "Evolution and Growth in Large Libre Software Projects". In: *8th International Workshop on Principles of Software Evolution (IWPSE 2005), 5-7 September 2005, Lisbon, Portugal*. 2005, pp. 165–174.

[Rob+07] Gregorio Robles, Santiago Dueñas, and Jesús M. González-Barahona. "Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time". In: *International Conference on Open Source Systems*. 2007, pp. 121–132.

[Rob02] Colin Robson. *Real World Reserach*. 2:nd. Blackwell Publishing, 2002.

[RH09a] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164.

[Run+11] Per Runeson, Martin Höst, Austin Rainer, and Björn Regnell. *Case Study Research in Software Engineering*. Wiley, 2011.

[Sea99] Carolyn B. Seaman. "Qualitative Methods in Empirical Studies of Software Engineering". In: *IEEE Transactions on Software Engineering* 25.4 (1999), pp. 557–572.

[Sug16] Sugar CRM, Inc. *Sugar Content Resource Management*. `https://www.sugarcrm.com`. 2016.

[Tim13] Time, Inc. *Fortune 500, Sorted by Industry*. `http://fortune.com/fortune500/`. 2013.

[Ubu16] Ubuntu. *Sugar Content Resource Management*. `http://www.ubuntu.com`. 2016.

[WF94a] Stanley Wasserman and Katherine Faust. *Social Network Analysis. Methods and Applications*. Cambridge University Press, 1994.

[Web04b] Steven Weber. *The Success of Open Source*. Harvard University Press, 2004.

[Wes03] Joel West. "How open is open enough?: Melding proprietary and open source platform strategies". In: *Research Policy* 32.7 (2003), pp. 1259 –1285.

[Wes07] Joel West. "Value Capture and Value Networks in Open Source Vendor Strategies". In: *Hawaii International Conference on System Sciences*. 2007, p. 176.

[Wie12] Roel Wieringa. “Designing Technical Action Research and Generalizing from Real-World Cases”. In: *International Conference on Advanced Information Systems Engineering, CAISE*. 2012, pp. 697–698.

[WM12] Roel Wieringa and Ayse Morali. “Technical Action Research as a Validation Method in Information Systems Design Science”. In: *Design Science Research in Information Systems. Advances in Theory and Practice - 7th International Conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012. Proceedings*. 2012, pp. 220–238.

[Woh+12] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. *Experimentation in Software Engineering*. Springer, 2012, pp. I–XXIII, 1–236.

INCLUDED PAPERS

PAPER I

A SYSTEMATIC REVIEW OF RESEARCH ON OPEN SOURCE SOFTWARE IN COMMERCIAL SOFTWARE PRODUCT DEVELOPMENT

Abstract

Context: The popularity of the open source software development in the last decade, has brought about an increased interest from the industry on how to use open source components, participate in the open source community, build business models around this type of software development, and learn more about open source development methodologies. There is a need to understand the results of research in this area

Objective: Since there is a need to understand conducted research, the aim of this study is to summarize the findings of research that has ben carried out on usage of open source components and development methodologies by the industry, as well as companies' participation in the open source community.

Method: Systematic review through searches in library databases and manual identification of articles from the open source conference. The search was first carried out in May 2009 and then once again in May 2010.

Martin Höst, Alma Oručević-Alagić
Information & Software Technology, 53:6, 616-624, 2011.

Results: In 2009, 237 articles were first found, from which 19 were selected based on content and quality, and in 2010, 76 new articles were found from which 4 were selected. 23 articles were identified in total.
Conclusions: The articles could be divided into four categories: open source as part of component based software engineering, business models with open source in commercial organization, company participation in open source development communities, and usage of open source processes within a company.

1 Introduction

Traditional software development is often perceived as a proprietary, in-house software development, with developers working in a geographically centralized or distributed company's location. Open source software is developed free of charge through a community driven development process, and as such, it is also provided to public at no cost, but under certain usage and distribution conditions. Many of the traditional software companies have tried to take advantage of the free software, not just by using the software, but also by creating business models and strategies around the open source software.

For example, in the mobile industry there are several attempts to form open source communities for development of software, such as the Android project[1] and the Symbian project[2]. Using and relying on open source software can be seen as an alternative way to reduce development costs and stay competitive. Hence, in a way, it can be compared to other similar business methods and strategies, such as outsourcing or acquirement of off the shelf components.

This open source business ecosystem, which has been growing over the past two decades, is quite complex and there exists a need to better understand many of its aspects. Some of the aspects are interesting in at least two different ways. Firstly, an organization can include open source components in its proprietary software product. This is comparable to including any other third party component, although the difference is that the component is now obtained from an open source community instead from a commercial organization. Secondly, an organization can provide its own proprietary software to open source community and that way reduce development costs in long run, reposition itself on the market, create a new source of income through new services, etc.

Already in 2001, Lerner and Tirole [LT01] identified "opening proprietary code" as an important research area, and observed that large open source projects often start based on software provided by "academic or semi-academic institutions". This motivates systematically investigating what research has been published in the area.

[1] `http://www.android.com/`
[2] `http://www.symbian.org/`

The outline of this paper is as follows. In Section 2 background on open source software and some related work is presented. In Section 3 the methodology with respect to search strategy and inclusion and exclusion criteria are presented, and the resulting set of articles is presented in Section 4. Finally, there is a discussion in Section 5, and conclusions presented in Section 6.

2 Background and related work

2.1 Open Source Software

Open source software has been around since the very beginning of electronic computing. In the early days of information technology it was quite natural and financially sound for developers to share source code among very few and very expensive computing machines. As the machines became smaller, more diversified, and cheaper, the number of developers grew, and the source code, in general, became more complex. Development of free software was especially flourishing in the academic environments. Barkley Software Distribution (BSD) is a license developed for distribution of the BSD version of the Unix operating system developed by the University of California, Berkeley, from 1977 to 1995 in collaboration with AT&T labs, as described in [Ray01b]. At the beginning of the development, code was shared between AT&T and Berkeley. Due to anti-monopoly laws at the time, AT&T could not sell software, but as the company was using it to sell phone-related services, it had vested interest in improving the software. During the beginning of the 1980:s and the market deregulation, AT&T was granted the right to sell software. In order to continue distribution of BSD Unix, a lot of code that was not developed by University of California Berkeley had to be backed off and rewritten.

Since the beginning of 1980s, the idea of close-sourced/proprietary software became mainstream, taking the place that free software sharing has held for a long time. The open source supporters went to found their own organizations such as free software foundation (FSF) founded by Richard Stallman, as described in [Web04a]. The FSF did not have desired impact on bringing back open source software development to the mainstream. However, this situation was about to change with the successful release of the Linux kernel. The system was initially developed by Linus Torvalds as part of an academic project, and with the support of the developer community it became a very complex, sophisticated software that was free for everyone to use. Eric Raymond was very much inspired by this set of events, and in his now famous book ”The Cathedral and the Bazaar” [Ray01b] he talks about the importance of Linux, as it was the very first time the open source developer community showed that not only complex and sophisticated software can be built in such way, but also that business models can be built around such way of software development and distribution.

In 1998, Raymond was one of the main contributors to the Open Source Initiative (OSI), an organization that is envisioned as open source educational and advocacy organization. Many companies have followed the suit, and decided to open source a piece of their proprietary software as a part of business strategy to deal with the competition. Thus, among the initial suitors we can find Netscape corporation, who by open sourcing Netscape internet browser tried to compete against closed source and free distribution of Microsoft's Internet Explorer [Ray01b].

In the past ten years, many companies have entered the open source business arena, using some of the business models proposed in [Ray01b]. Unfamiliar with the environment, companies had very quickly to readjust their way of doing business in order to ripe some perceived benefits of open source trends. Besides open sourcing software, companies tend to participate and contribute to open source projects, as well as adopt some of software development methodologies such distributed and voluntary based development community as open source utilizes.

2.2 Related Work

Stol and Ali Babar [SAB09] have made a review of the broad area of "open source" from the conference on Open Source Systems, OSS. They manually selected empirical papers from the conference and investigated them. The scope of the review that we present in this paper is more narrow (open source in commercial organizations) but we searched a broader set of articles (we searched articles in library databases with a search string, and we searched articles from the OSS conference manually, as explained in more detail below).

Stol and Ali Babar [SAB10] have also compiled a list of challenges in using open source as components in product development, based on a literature review. In this review, where they did not require any empirical grounding of the findings, they identified 21 challenges.

In [HOA10] earlier results of this work is presented, based on a search in biblographic databases that was carried out 18 May 2009. The search that is the basis for this paper was conducted 14 May 2010, and it resulted in 4 additional articles.

3 Review Method

This research is carried out as a systematic literature review, based on the guidelines presented in [KC07].

3.1 Research Questions

The objective of this research is to understand the result of the research that has been carried out on the usage of open source software and open source software development in proprietary software development organizations. Before the review, this was broken down to the following research questions:

1. What approaches and processes are applied by commercial organizations to introduce open source products in their proprietary products?
2. What approaches and processes are applied by commercial organizations to provide their software products to the open source community?
3. What experience is available from identified approaches and processes, for example, with respect to quality of the software products, cost of development for the providing organization, time taken to introduce new functionality, etc.?
4. What are the main motivations and business incentives for the procedures and processes identified in question 1 and question 2?

That is, we address the need to understand both what research that has been done in the area, what methods and approaches that exist, and what experience is available for the different methods. It should be noted that the objective of the research has not been to derive quantitative knowledge of which methods perform the best. The objective is more to understand which methods are used and how well the methods work in a qualitative way. If the field was more mature, and it could be expected to find a large number of empirical studies investigating the performance of alternative methods, it would of course be interesting to synthesize this knowledge. However, it is not realistic to find this many studies of this type. The objective of this work is instead to review the research that has been conducted, and in particular what kind of experience that is available for these kind of questions. That is, the research has elements of a mapping study (see for example [KC07]). However, since the objective is to summarize the findings and to understand the total result of the research that has been conducted, and the focus is not merely on identifying published research we classify this as a systematic review.

Open source software in commercial organization is related to a number of research questions that to some extent are relevant to the review, but where it was necessary to decide whether to include them in the study or not. One aspect that is often mentioned concerning open source is the importance of ”legal aspects”, such as licensing, intellectual property, etc. For example, there is a large number of licenses, all complying with the definition of open source described in [FF02], and the implications of choosing different licenses could be an important research field. This is an important and interesting field, which can affect both the adoption of open source practices and open source software components. However, for this study it was seen as out of scope for two main reasons. Even if software engineering is a multi-disciplinary field which includes legal aspects, we thought that it is of another kind than more traditional software engineering topics. To some extent the research questions that have to do with legal aspects are not the same as traditional research questions. If legal aspects were included, then there are other areas that also would be reasonable to include, such as marketing and sales. Second, it would probably require extensive cooperation with researchers

in legal aspects to make sure that the correct search terms were used, and that the right publication fora were searched. These aspects in combination mean that legal aspects were not included in the study.

An area where there are a number of research results available is on comparisons of usage of open source software, such as Open Office, and similar proprietary software systems. This was not seen as highly related to the research questions in this study and therefore excluded. It is, of course, interesting understand the differences, but it was not seen as relevant enough to the question of transforming developed software to open source or to the inclusion of open source software in developed software. Neither are studies on adoption of open source programs, for example as presented by Goode [Goo05], included. That is, this study is more on development of software than on the usage of existing software. In the same way it was decided not to include research results on usage of open source tools, such as Eclipse, in software development.

Another area that is not included, but still interesting and of potential interest to commercial organizations, concerns how open source practices can be transferred to hardware development. Only a few articles on this topic exist [AB09]. This area was not included since it is not mainly concerning software development.

3.2 Search methodology

Two main sources were searched for relevant articles: a broad search in academic databases; and a manual search through all articles of the Conference on Open Source Systems.

Searched academic databases

The INSPEC and the COMPENDEX databases were searched. Both of these databases intend to provide complete coverage of the area, and include articles from all major conferences, journals, and publishers (e.g. IEEE, ACM, Springer, and IEE). We believe that these two databases give a good coverage of articles in ”computer science” and ”electrical engineering and electronics”, which includes typical questions in software engineering, at least in more well known journals and conferences. However, the coverage of more business-related articles and articles on legal aspects is, as described above, more uncertain. Both databases were accessed through Engineering Village[3].

The following search string was used:

```
(({open?source} wn ALL) OR
 (opensource wn ALL) OR
 (libre wn ALL) OR
```

[3] http://www.engineeringvillage2.org

```
 (OSS wn ALL) OR
 (FLOSS wn ALL))
AND
((proprietary wn ALL) OR
 (commercial wn ALL) OR
 ({non?open?source} wn ALL) OR
 ({non?opensource} wn ALL))
AND
((empirical* wn ALL) OR
 (experiment* wn ALL) OR
 ({case?study} wn ALL) OR
 (survey wn ALL))
```

The search string contains three main parts, separated by `AND`-clauses. The first part states that the article must include the term ”open source” or some other synonym term that is often used, such as ”OSS”. The second part states that the article must include terms about commercial software development. The third part makes sure that the article is empirical, by searching for terms like ”empirical” and ”experiment”. The intention of the ”`*`” after `experiment` is to include also search terms as ”experimental” and ”experimentation”. According to [Die+07] this should be sufficient in order to find most relevant articles in this respect.

A few more terms and details in the search string may have to be explained. The `?`-sign denotes any character, which, for example, means that both articles with the term ”case study” and the term ”case-study” are found. The term `wn` means that the phrase left of it should be found in the entity to the right of it, in this case `ALL`, which means all fields of database entries, such as title, abstract and key words. It would have been possible to list other entities such as abstract and title, but in this case `ALL` was chosen. Text within `{}`-parentheses are searched as phrases and a search is not case sensitive.

Manual identification of relevant articles

In addition to the database searches, all articles in all OSS-conferences (International Conference on Open Source Systems[4]) were inspected. The conference has been held annually since 2005 and all articles are available in full text (2005 online and from 2006 onwards from library databases). The selection was based on the formulated research questions in Section 3.1, which thereby means that articles matching the same kind of content as the identified with search string presented in Section 3.2 were found, but there was no check that the articles matched exactly.

[4]For more information see `http://www.ifipwg213.org/`

3.3 Selection of relevant articles

Articles are selected in a number of steps. First, all articles identified from the databases with the search string were listed with title and abstract. Since the articles have been selected with a search string in a database there are many articles that are not relevant. Therefore, articles that are not relevant, based on our interpretation of title and abstract, were removed. That is the articles that were not relevant compared to the research questions wee removed.

After this, the remaining articles were downloaded and read in full text. Based on this, more articles were seen as non-relevant according to the same criteria as for the title and abstract, and therefore removed.

After analysis of articles selected with the search string, articles from the OSS conference were selected manually. Since this selection was carried out after the analysis of articles identified with the search string, it was possible to use knowledge that was gained from analysis of articles identified with the search string.

All articles that so far have been selected were analyzed with respect to research methodology implementation and presentation. Three different classes were used for this:

Class A: In this type of article the research is presented in a way that makes it very likely that it was conducted according to normal requirements on empirical research methods in software engineering.

Compared to the quality assessment criteria presented used by [DD08] and [Che+08] the answer is positive to most evaluation questions, especially concerning whether it is research or merely experience report, if there is a clear statement of aims, if there is an adequate context description, if the research design is appropriate for the questions, if the data collection was appropriate for the questions, if the data was analyzed with sufficient rigor, and if there is a clear statement of findings.

This class includes both articles that do reference empirical software engineering research method descriptions, such as [RH09b], and articles that do not explicitly reference this kind of descriptions.

Class B: This class of articles may not be presented as a typical article on empirical software engineering even if the overall impression of it is that it was carried out in this way. That is, all aspects of a typical article of class A may not be included, but the main impression of the paper is that the research was carried out according to normal requirements on empirical software engineering.

Class C: For this type of article our interpretation is that the researchers have not followed any traditional research method during the research. The reason may be that the presentation forum is not suitable for presentation of structured research methods or there may be other reasons.

These steps are further presented in Section 4 and illustrated in Figure 1.

3.4 Data Extraction and Synthesis

Articles of class A and class B were treated equally, while articles of class C were not further included in the review. Data from the identified articles were derived by defining categories of articles and summarizing the research in each category. Both authors first defined categories individually and then a final set of categories was defined based on discussion between the authors. The summaries were developed first by one author and then updated based on discussion between the authors.

3.5 Phases

The search was conducted in two phases. First in phase 1 the databases were searched in 2009, and the result was summarized and presented in [HOA10]. Then the same search string was used again in phase 2 in 2010, and the results were updated with the new articles that were identified.

4 Results

In this section the actual results of the steps presented in Section 3 are presented. The results are summarized in Figure 1 and below. The research was conducted in two major phases as described in Section 3.5.

4.1 Phase 1

First the academic databases were searched on 18 May 2009, which resulted in 357 articles. However, among these articles there were a number of duplicates because the searches were made in different databases. Duplicates were identified with a simple java-program based on titles. After this, 237 articles remained (i.e. the result of step 1 in Figure 1).

After this, unrelated articles were removed based on both title and abstract. First an attempt was made to remove articles based only on title and then based on abstract, but it was not seen as possible to remove an article only based on the title. Therefore both titles and abstracts were studied during this process. After this, 45 titles remained.

After this, one additional duplicate was identified where the title was written slightly differently in different databases (”&” instead of ”and”), which means that 44 titles remained (i.e. the result of step 2 in Figure 1).

The first author of this paper first conducted these steps, and then the second author reviewed the result. None of the previously excluded articles were reintroduced.

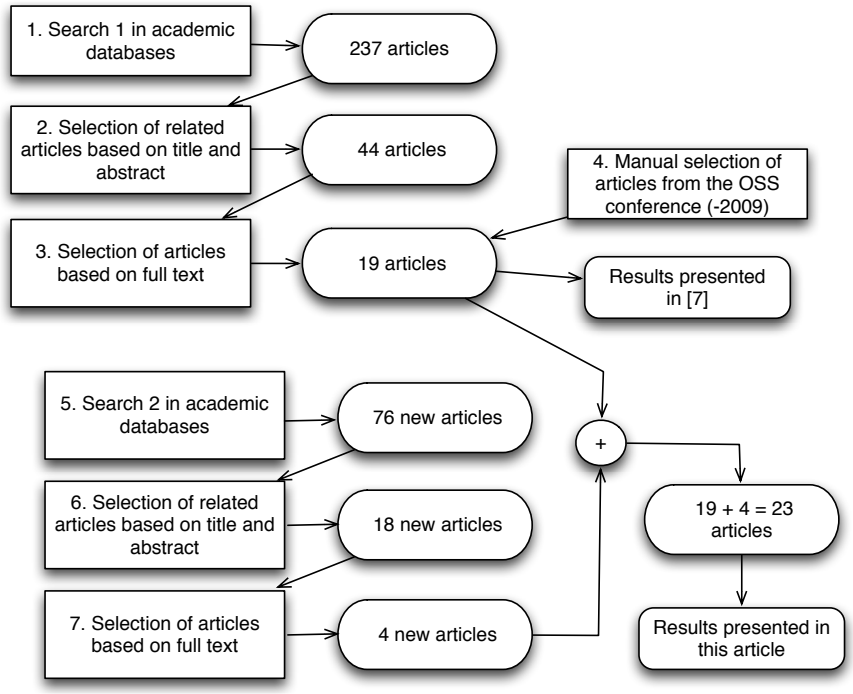


Figure 1: Summary of article selection process.

In these steps, some articles were found for which it was hard to decide whether to keep them or not based on the title and abstract. In these cases we decided to keep them to the next step instead of removing them. That is, articles that were hard to judge based on only title and abstract were kept to the next step, where the whole articles were read.

After this, the identified articles were obtained from the library database, and they were reviewed in full text. Here, some articles were removed since they were not really related to the research questions or because they were not available in full text from the databases.

Some of the removed articles were about developing open source in general, which was not seen as relevant for this work. Some were about using open source software in general, without seeing the context as an IT system that is built.

To this list of articles, relevant articles from the OSS conference were added. There was no overlap between these manually found articles and the articles that were found through the search in the databases. After this, a final set of 19 articles remained (i.e. the result of step 3 in Figure 1).

In the analysis of the papers it was clear that both anticipated and unanticipated areas were covered by the identified articles. That is, some papers dealt with questions that we thought of before, and therefore were aware of when the research questions were formulated. Other areas were more unexpected, mainly the articles about transferring the open source development process to the internal work in a non-open source product. Articles of both types were of course included in the study as long as they were seen as relevant compared to the formulated research questions.

One paper for which it was hard to judge the relevance for this study is the paper by [Kri06], which concerns motivation of developers, to some extent discussing both unpaid and paid developers. Even if the question of motivation for open source developers in general is out of scope of the review, the discussion about paid and unpaid developers makes it more relevant. However, we decided not to include the paper since it was seen as a too small part of the article. Also, concerning the paper by [Lea+02] it could be argued that this type of article should be included. The main focus of it is on design of a modular system for data analysis, but they also conclude that working with the system as OSS improves the possibility of collaborating between different universities, government agencies, and private industry, both nationally and internationally. However, this was stated as a minor part of the paper, which means that the article was not included.

4.2 Phase 2

The databases were searched with the same search string as in phase 1 once again 14 May 2010. This resulted in 76 new articles that were not identified in phase 1 (i.e. the result of step 5 in Figure 1).

The title and abstract of the articles were studied in order to remove articles that were not of interest. This was done as a cooperation between the two authors. After this step, 18 of the new articles were kept (i.e. the result of step 6 in Figure 1).

The next step was to download all articles and study them in full text in order to decided whether they really are of interest with respect to the research questions, and of type A or type B. Some articles could not be downloaded from the library databases, but most could. After this step 4 of the new articles remained.

4.3 Analysis of identified articles

After phase 2 there were in total 23 articles, i.e. 19 from phase 1 and 4 additional from phase 2.

In the rest of this paper, the selected articles are referred to with the keys that are presented in bold in appendix. For example, the first identified article is referred to as [Arhippainen03].

Table 1: Identified articles

Article	Research methodology	Class	Identification phase
[Arhippainen03]	case study	A	1
[Ayala09]	survey	A	1
[Bonaccorsi05]	survey	A	1
[Bonaccorsi06]	survey	A	1
[Bonaccorsi07]	survey	A	1
[Gaughan09]	case study	B	2
[Gurbani06]	case study	A	1
[Harison10]	survey	A	2
[Henkel08]	interviews and survey	A	2
[Hauge07]	survey	A	1
[Hauge09]	case study	A	1
[Li05a]	survey	A	1
[Li05b]	survey	A	1
[Li06a]	survey	A	1
[Li06b]	survey	A	1
[Li09]	summary	A	1
[Lindman08]	case study	A	1
[Lindman09]	case study	A	1
[Lundell06]	survey	A	1
[Munga09]	case study	B	2
[Robles07]	case study	A	1
[West03]	case study	B	1
[Westenholz06]	case study	A	1

The research methodologies that were used in the identified articles are summarized in Table 1. This is our interpretation of the chosen methodology after reading the articles. In some cases it was very clear which methodology that was used, but in other cases it was somewhat harder. For example when a set of interviews was conducted in different organizations we classified this as a survey, since we wanted to classify according to commonly used methods. However, it could had also been classified as something like ”interview study”.

In Table 1 we also present our interpretation of the classification of the method implementation (A or B). Our experience is that it is hard to evaluate articles in this way. Since no difference has been made between the two classes in the analysis, this classification should only be seen as our interpretation of the article for the purpose of this review.

The article [Li09] requires some further explanation. Since it is a summary of

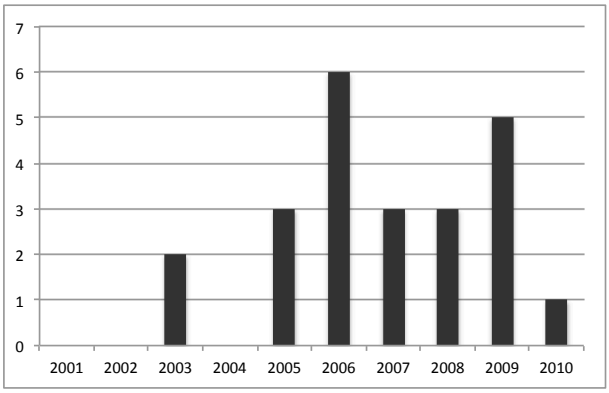


Figure 2: Publication year for included articles

the other articles by the author ([Li05a]–[Li06b]), the methodology is presented as ”summary”, and even if it is presented as a popular science article we have classified it as class A and thereby included it in the study based on the contents of the other articles.

In Figure 2 it can be seen that the oldest identified article is from year 2003, and that the most recent is from 2010. Here it should be noted that the last search in the database was conducted in may 2010, which means that there may be more articles published in 2010. It can be noted that all the identified articles are published rather recently (i.e. since 2003). There were rather many articles published in 2005 and 2006. Four of these were by the same author on the same subject.

4.4 Investigated research areas

Introduction

The articles can be divided into a number of main areas based on the contents of the articles. The formulation of content areas was done without the explicit objective to adhere to the identified research questions defined in Section 3.

The categories were defined based on the contents of the articles. That is the whole articles were used and no specific common parts of the articles were derived with a data extraction template. Each article was sorted into the category where it belonged the most even if it could be argued that some articles to some extent were related to more than one category. However, there was no article where it was really hard to decide the category or where we thought that it was equally related to more than one category. This is probably natural since the categories were defined based on the contents of the articles, and the objective during this process was to define categories based on the articles. It should be noted that the classification

was first conducted in phase 1 based on the 19 articles that were identified in that phase. It was rather easy to classify the 4 new articles in phase 2 in the same way, which means that the same classification scheme (i.e. the four areas) was kept in phase 2.

The identified categories are listed below. For each category the articles related to it are listed.

- *Company participation in open source development communities:* [Bonaccorsi07], [Hauge07], [Henkel08], [Lundell06], [Robles07].
- *Business models with open source in commercial organizations:* [Bonaccorsi05], [Bonaccorsi06], [Harison10], [Hauge09], [Lindman09], [Munga09], [Westenholz06], [West03].
- *Open source as part of component based software engineering:* [Arhippainen03], [Ayala09], [Li05a], [Li05b], [Li06a], [Li06b], [Li09].
- *Using the open source process within a company:* [Gaughan09], [Gurbani05], [Lindman08].

The research conducted in each area is shortly summarized below.

Company participation in open source development communities

It is clear that there is company participation in many open source projects. For example [Bonaccorsi07] found that in one third of the most active projects on SourceForge there was some form of company participation. Companies can participate as project coordinator, collaborator in code development, and by providing code etc. In [Hauge07] one additional role, which is more concerned with integration of open source components, is identified.

Concerning the number of companies that participate in this kind of development, [Lundell06] suggests that a significant number of the companies marginally participate in open source community. However, the participation has increased especially in SME, compared to earlier conducted studies. Of the companies that use open source projects, 75% can be said to have ”symbiotic relationship” with the OS community. This can be compared to the investigation presented by [Rables07] that show that 6-7% of the code in Linux Debian GNU distribution over the period 1998-2004 has been contributed by corporations. That is, it is clear that a rather large part of the open source code has been provided by commercial organizations, and that those commercial organizations play crucial roles in open source projects. This is especially clear in the larger and more active projects.

It is also clear that if software should be provided to a community it is important to provide enough documentation and information to get the community members going (e.g. [Hauge07]).

One risk that could be seen by companies is that people working in the organization would reveal too much information to the outside of the organization if they work with an open source community. However, the revealing behavior of this kind of software engineers was investigated in [Henkel08] and it was found that even if the engineers identified with the community they were significantly less identified and ideological about open source than the control group of non-commercial developers. The conclusion from that research is that there is indicators of commercially harmful behavior in this kind of development.

[Bonaccorsi07] presents a list of important questions for further research, which is relevant with respect to these questions. For example, are companies participating in open source projects more successful than other companies, and what are the characteristics of companies participating in open source projects? It is worth noting that no identified paper presents much research about how companies' internal processes for collaborating with communities work. This could also be an area for further research.

Business models with open source in commercial organizations

Concerning the business models it is clear that companies involved in open source development besides developing open source products also offer customized software based on open source products. It is also common to offer consulting and training (e.g. [Bonaccorsi06]). It is also clear that business models include hybrid strategies, such as described by [West03], where the focus is on large software vendors. The paper presents an in-depth analysis on historical development of operating systems, computers, and business strategies adopted by vendors. It is also possible to base the business on being the link between an open source project and the enterprise customers by integrating the product in a commercial package [Munga09].

[Hauge09] presents a case study on a small Norwegian company that successfully established a business model around two open source products by establishing three specialized user communities. The paper also concludes that while it is important to attract developers to the community, it is as important to retain some control over the product for commercial benefits.

Bonaccorsi [Bonaccorsi05] investigates reasons why companies participate in open source communities. In particular, the paper analyzes discrepancies between attitudes and behaviors in relation to three primary research questions. The questions deal with motivation to set up an open source business, whether the firms' claims to uphold intrinsic, community-based values are aligned with the firms' actions, and finally, if there is discrepancy, are there any observable patterns. The conclusion points out that there is misalignment in attitudes and behavior of firms in open source market place, confirming earlier research that companies use intrinsic values to attract developers in order to fulfill their own extrinsic goals. In [Harison10] it is found that software companies with higher proportions of highly

educated personnel are more likely to adopt a business model based on supplying open source software.

[Westenholz06] offers an insight into challenges of creating a sustainable business around open source business model based on a case study. The study offers insight into shifting of business strategies conducted by the entrepreneur in order to make the business profitable around the combination of open source and proprietary software.

[Lindman09] asserts that business models can sometimes be too generic and undertake an exploratory case study of three different organizations in order to empirically identify different incentives companies have in releasing a product as open source beyond the revenue generating ones. The paper points to challenges in attracting and sustaining a community for a software product that is highly specialized.

Open source as part of component based software engineering

The article from Arhippainen [Arhippainen03] is a case study conducted at Nokia on the usage of the OTS components. The paper presents a detailed analysis on usage of third party components in general, and discusses advantages of using proprietary over open source components and vice versa. It also identifies issues related to software development methodology in terms of including third party components.

The research presented in [Ayala09] assesses the state of reusable components in the open source market based on survey conducted in Spanish and Norwegian companies. The results of the survey also assess the needs of OSS industrial users in component selection and identify challenges that can aid in maturing the open source components market.

The 5 articles [Li05a], [Li09], [Li05b], [Li06a], [Li06b] are based on the analysis of data collected through state-of-the practice survey conducted in Norway, Germany, and Italy. This research, conducted on over 100 projects that use proprietary or open source OTS components, looks into risks associated with use of such components, reasoning behind using OTS components, and impact on development process when using the OTS components. Some of the findings suggest that selection of OTS components is a very informal process, that OTS components are selected throughout the development process life cycle even though early selection yields benefits. It is also suggested that estimates on effort needed to integrate components is informal and dependent on experience, and as such, is often inaccurate. Furthermore, some general conclusions of the studies point that the OTS components rarely have negative impact on the system. Open source OTS components are used in the same manner as proprietary components, thus without modification. If a problem occurs with the OTS components, it takes substantial amount of effort to correct them.

It can be noticed that in [Li05a]–[Li09] there is no in-depth analysis of what kind of open source components as OTS components, were used by companies. For example, many mainstream proprietary IT workshops, sometimes very ”hostile” to the idea of using open source components, like the ones producing software for big financial houses, use PGP and other Unix based open source components as these over the time have become de-facto standard. Investigating into the diversity and type of open source OTS components that are used in projects can be a question for further research.

Using the open source process within a company

An interesting area that is investigated in [Gurbani05] and [Lindman08] is that of using an open source process within a company. That is, the product is not provided to any community outside the organization, but instead handled as an open source project within the company. One unit of the organizations owns the product and provides it to the rest of the organization. Everyone in the organizatin are allowed to use and modify the code, and changes are approved by the original owners as in any other open source project.

Gurbani et. al. [Gurbani05] presents a case study on transferring the open source development model for one software product at Lucent technology. In this case the approach was judged successful by the authors, for example because the product was needed in several products and the architecture was suitable for this. Lindman et. al. [Lindman08] also investigates the usage of an internal open source development methodology through a case study. This case study is conducted on usage of Nokia iSource portal for hosting projects. The portal became very popular for managing heterogeneous types of projects: SCM, distributed, agile, intercompany collaboration projects. The research results showed that implementation of open source project management tools can facilitate innovation within the company. In [Gaughan09] the area is also investigated in a set of studies. Positive aspects based on increased visibility in the organization such as as better code quality and pride in work are listed. However, visibility can also lead to other aspects, such as privacy and knowledge retention, and easier workplace monitoring. It should be noticed that all case studies are conducted at large companies, which probably is natural.

A number of further research questions can be identified. One concerns how contributions can be included in this kind of product when different developers have different needs for the developed product.

5 Discussion

In this review, 23 articles were identified. We do not think that this is a large number of articles compared to the importance of the field, and the general amount of discussion about how open source can be used by commercial organizations.

Many of the studies are in the form of surveys, which gives a broad and necessary understanding. Based on this it would probably be possible to conduct more studies investigating specific cases of implementation of methodologies for dealing with different aspects of open source in industry. More case studies could probably be conducted on all aspects of the research questions. More case studies could probably also provide more knowledge of research question 3 and research question 4. That is, research could be carried out to understand more about the cost and advantages of different approaches, and why different approaches are chosen. It is also worth noticing that there are no controlled experiments at all in the identified articles.

There is, of course, a risk that some articles have been missed in the search, either because the search string has not identified all relevant articles, or because the set of searched journals was not complete. The search string was developed through a ”trial and error” approach in order to find as many relevant articles as possible, but it is impossible to guarantee that all articles have been found. The same is true for the coverage of the search. It is not possible to guarantee that all relevant journals and conferences have been searched. Here the most severe risk is probably for articles not in the traditional software engineering literature, such as articles on business models, which is more general than traditional software engineering. Since there is a risk that all articles have not been found it is reasonable to discuss the effects of missing articles. Of course, the more complete the selection of articles is the better it is. However, in this case the objective is more to identify and summarize conducted research and experience than to carry out meta-analysis, which probably means that the effect of missing single articles is lower. Even if more articles would be found it is not unlikely that the major conclusions in terms of identified areas, and main conclusions in areas, would be the same.

6 Conclusions

Concerning research question 1 and 2, i.e. what approaches and process are used to introduce and provide open source components, it was possible to divide the the identified papers into different areas. The following areas were defined based on the articles: i) participation in open source development communities, ii) business models with open source, and iii) treating open source software as components in component based development. Besides this there are articles on iv) how open source processes can be used within a company.

In all areas experience in some form were presented in the articles, although the papers were on rather different areas. Some results were presented on motivation and incentives.

The areas are important for research and it is interesting to see that research is available in all these areas. The question of how to use open source practices within a closed company (iv) is for example an interesting area for further research.

Based on this review we also propose that further research is conducted on how companies can transform their proprietary software to open source and build a community on it. Further research related to all four research questions in Section 3.1 could involve more case studies on implementation of specific methodologies for dealing with different aspects of open source in industry.

Acknowledgment

This work was partly funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (`http://ease.cs.lth.se`).

References

[AB09] N. Abdelkafi and T. Blecker. "From open source in the digital to the physical world: a smooth transfer?" In: *Management Decisions* 47.10 (2009), pp. 1610–1632.

[Che+08] L. Chen, M. Ali Babar, and C. Cawley. "A status report on the evaluation of variability management approaches". In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2008.

[Die+07] O. Dieste, A. Griman, and N. Juristo. "Developing search strategies for detecting relevant experiments". In: *Empirical Software Engineering* 14 (5 2007), pp. 513–539.

[DD08] Tore Dybå and Torgeir Dingsøyr. "Strength of evidence in systematic reviews in software engineering". In: *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. ESEM'08. Kaiserslautern, Germany, 2008, pp. 178–187.

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

[Goo05] Sigi Goode. "Something for nothing: management rejection of open source software in Australia's top firms". In: *Information & Management* 42 (5 2005), pp. 669–681.

[HOA10] Martin Höst and Alma Oručević-Alagić. "A Systematic Review of Research on Open Source Software in Commercial Software Product Development". In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2010.

[KC07] Barbara Kitchenham and S. Carter. *Guidelines for performing systematic literature reviews in software engineering, v. 2.3*. Tech. rep. Keele University and University of Durham, 2007.

[Kri06] Sandeep Krishnamurthy. "On the Intrinsic and Extrinsic Motivation of Free/Libre/Open Source (FLOSS) Developers". In: *Knowledge, Technology & Policy* 18.4 (2006), pp. 17–40.

[Lea+02] G. H. Leavesley, S. L. Markstrom, P. J. Restrepo, and R. J. Viger. "A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling". In: *Hydrological Processes* 16 (2 2002), pp. 173–187.

[LT01] Josh Lerner and Jean Tirole. "The open source movement: Key research questions". In: *European Economic Review* 45.4-6 (2001), pp. 819 –826.

[Ray01b] Eric S. Raymond. *The Cathedral and the Bazaar*. O'Reilly Media, 2001.

[RH09b] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

[SAB09] K-J. Stol and M. Ali Babar. "Reporting empirical research in open source software: the state of practice". In: *Proceedings International Conference on Open Source Systems*. Skövde, Sweden, 2009, pp. 156–169.

[SAB10] K-J. Stol and M. Ali Babar. "Challenges in using open source software in product development: a review of the literature". In: *Proceedings FLOSS*. Cape Town, South Africa, 2010, pp. 17–22.

[Web04a] S. Weber. *The Success of Open Source*. Harvard University Press, 2004.

Appendix: Articles included in review

Arhippainen03 L. Arhippainen, Use and integration of third-party components in software development. Technical report, VTT Publubilcation 489:84, 2003. In this report a case study on component based development, including the use of open source components, at Nokia is presented.

Ayala09 C. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, and K. Sandanger Velle, Challenges of the Open Source Component Marketplace in the Industry, In proc. OSS, pp. 213-224, 2009. The paper analyzes the state of open source market place and how companies interact to reuse components that the market place offers.

Bonaccorsi05 A. Bonaccorsi and C. Rossi, Intrinsic Motivations and Profit-oriented Firms in Open Source Software. Do firms practise what they preach?. In proc. OSS, pp. 241-245, 2005. The articles investigates true motivation behind companies involvement in open source activities based on data gathered through a survey of 146 Italian companies supplying open source solutions.

Bonaccorsi06 A. Bonaccorsi, S. Giannangeli, and C. Rossi Entry strategies under competing standards: Hybrid business models in the open source software industry. Management Science, 52(7):1085-109, 2006. This is a further analysis of the same survey as presented in [Bonaccorsi05]. They have developed a regression model explaining the "friendliness" to open source based on a set of factors.

Bonaccorsi07 A. Bonaccorsi, D. Lorenzi, M. Merito, and C. Rossi Business firms' engagement in community projects, empirical evidence and further developments of the research, In First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS'07, 2007. This is a survey based on a sample of 300 projects from SourceForge. In 97 of the projects there was at least one company participating. The three main types of involvement were "project coordinator", "collaboration", and "provision of code".

Gaughan09 Gaughan, G., Fitzgerald, B., and M. Shaikh, An examination of the use of open source software processes as a global software development solution for commercial software engineering. In proc. Euromicro Software Engineering and Advanced Applications, pp. 20-27, 2009. This paper summarizes experiences from using open source processes within companies.

Gurbani06 V.K. Gurbani, A. Garvert, and J.D. Herbsleb, A case study of a corporate open source development model. International Conference on Software Engineering, pp. 472-481, 2006. This paper presents a case study on transferring the open source development model for one software product to a commercial environment at Lucent technology, keeping the software proprietary in the company.

Harison10 E. Harison and H. Koski, Applying open innovation in business strategies: evidence from Finnish software firms. Research Policy, Vol. 39, pp. 351-359, 2010. A survey of 170 Finnish software firms with respect to business strategies is presented.

Hauge07 Ø. Hauge, C.F. Sørensen, and A. Røsdal, Surveying Industrial Roles in Open Source Software Development. In proc. OSS, pp. 259-264, 2007. This paper defines different industrial roles in open source community: provider, integrator, participant, and inner source software participant. Through a survey, it investigates motivation, challenges, and development practices of the companies taking on these roles within the ITEA COSI project.

Hauge09 Ø. Hauge and S. Ziemer, Providing Commercial Open Source Software: Lessons Learned. In proc. OSS, pp. 70-82, 2009. This paper presents a study on a small Norwegian software company that has built its business around own OSS products and compares the findings to other cases reported in literature.

Henkel08 J. Henkel, Champions of revealing – the role of open source developers in commercial firms, Industrial Corporate Change, Vol. 18, No. 3, pp. 435-471, 2008. This paper discusses how company employed persons can cooperate in open source communities, with focus on how code is committed to the community.

Li05a J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, An empirical study on off-the-shelf component usage in industrial projects. In Product Focused Software Development and Process Improvement (Profes), pp. 54-68, 2005. The paper presents survey conducted a large number of companies from Norway, Germany, and Italy on the off-the shelf (OTC) components usage. It

focuses on factors that influence the choice in terms of whether the OTS component is open source or proprietary.

Li05b J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, Validation of new theses on off-the-shelf component based development, International Software Metrics Symposium, pp. 231-240, 2005. The paper focuses on validating six theses related to usage of off-the-shelf components within companies.

Li06a J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, An empirical study on decision making in off-the-shelf component-based development. In proc. International Conference on Software Engineering (ICSE), pp. 897-900, 2006. This article investigates research questions resembling those in [Li05a], but with a larger sample of projects.

Li06b J. Li, M. Torchiano, R. Conradi, O.P.N. Slyngstad, and C. Bunse, A state-of-the-practice survey of off-the-shelf component-based development processes. In Reuse of off-the-shelf Components, International Conference on Software Reuse, pp. 16-28, 2006. This paper focuses on the development process when OTS components are used.

Li09 J. Li, R. Conradi, C. Bunse, M. Torchiano, O.P.N. Slyngstad, and M. Morisio, Development with off-the-shelf components: 10 facts. IEEE Software, 26(2):80-87, 2009. This article basically summarizes the findings from the earlier articles by the same group of authors. The conclusions are presented in the form of 10 facts learned about development with OTS components.

Lindman08 J. Lindman, M. Rossi, P. Marttiin, Applying Open Source Development Practices Inside a Company. In proc. OSS, pp. 131-387, 2008. This paper investigates characteristics of using open source and agile development practices within a company. It argues that usage of such practices can unlock innovation potential within a company.

Lindman09 J. Lindman, J.P. Juutilainen, M. Rossi, Beyond the Business Model: Incentives for Organizations to Publish Software Source Code. In proc. OSS, pp. 47-56, 2009. The paper investigates incentives for companies to release software as open source through three exploratory case studies at different stages of code release.

Lundell06 B. Lundell, B. Lings, and E. Lindqvist, Perceptions and Uptake of Open Source in Swedish Organizations. In proc. OSS, pp. 155-163, 2006. This paper investigates usage of open source within Swedish companies from the perspective that goes beyond mere adoption of an open source software product. It focuses on participation of the companies within the open source communities in roles of code contributors on existing third party projects or its own products.

Munga09 N. Munga, T. Fogwill, and Q. Williams, The adoption of open software in business models: a Red Hat and IBM case study. In proc 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, pp. 112-121, 2009. This paper investigates the business models of open source related aspects of Red Hat and IBM.

Robles07 G. Robles, S. Dueñas, and J.M. Gonzalez-Barahona, Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time. In proc. OSS, pp.

121-132, 2007. This paper investigates corporate involvement in Linux Debian GNU distribution over the period from 1998-2004 based on copyright attributions in the source code. The results of the research show that 6-7% of the code has been contributed by corporations.

West03 J. West, How open is open enough? melding proprietary and open source platform strategies. Research Policy, 32(7):1259-1285, 2003. The authors present a timeline for what has happened with respect to ”hybrid” software systems, that is software systems that consists of a mixture of open source software and proprietary software, such as an Apple computer. Three case studies are presented.

Westenholz06 A. Westenholz, Institutional Entrepreneurs and the Bricolage of Intellectual Property Discourses. In proc. OSS, pp. 183-193, 2006. This paper is a case study on an institutional entrepreneur who builds his own company on a business model that mixes open and closed source software practices.

USAGE OF OPEN SOURCE IN COMMERCIAL SOFTWARE PRODUCT DEVELOPMENT

PAPER II

Abstract

Open source components can be used as one type of software component in development of commercial software. In development using this type of component, potential open source components must first be identified, then specific components must be selected, and after that selected components should maybe be adapted before they are included in the developed product. A company using open source components must also decide how they should participate in open source project from which they use software. These steps have been investigated in a focus group meeting with representatives from industry. Findings, in the form of recommendations to engineers in the field are summarized for all the mentioned phases. The findings have been compared to published literature, and no major differences or conflicting facts have been found.

1 Introduction

Open source software denotes software that is available with source code free of charge, according to an open source license [FF02]. Depending on the license type,

Martin Höst, Alma Oručević-Alagić, and Per Runeson
International Conference on Product Focused Software Development and Process Improvement (PROFES), pp. 143-155, 2011

there are possibilities to include open source components in products in the same way as other components are included. That is, in a large software development projects, open source software can be used as one type of component as an alternative to components developed in-house or components obtained from external companies.

There are companies that have experience from using well known open source projects. Munga et al. [Mun+09], for example, investigate business models for companies involved in open source development in two case studies (Red Hat and IBM) and concludes that "the key to their success was investing resources into the open source development community, while using this foundation to build stable, reliable and integrated solutions that were attractive to enterprise customers". This type of development, using open source software, is of interest for several companies. If open source components are used in product development there are a number steps that the company needs to go through, and there are a number of questions that need to be solved for each step.

First potential components must be identified, which can be done in several ways. That is the company must decide how to identify components. Then, when potential components have been identified, it must be decided which component to use. In this decision there are several factors to consider, and the company must decide how to make this decision. Using the components there may be reasons to change them, which gives rise to a number of questions on how this should be done and to what extent this can be recommended. A company working with open source components must also decide to what extent to get involved in the community of an open source project.

There is some research available in this area [HOA10], although there is still a need to collect and summarize experience from companies working in this way. In this paper, findings are presented from a workshop, in the form of a focus group meeting, where these topics were analyzed by industry representatives.

The outline of this paper is as follows. In Section 2 the methodology of the research is presented, and in Section 3 the results are presented. The results are compared to results presented in the literature in Section 4, and in Section 5 the main conclusions are presented.

2 Methodology

2.1 Focus group

The workshop was run as a focus group meeting [Rob02; Kon+08]. At the workshop, participants informally presented their experience from development with open source software, for example from using open source components in their product development, or from participating in open source communities. The intention was to give all participants both an insight into how others in similar situ-

ations work with these issues, and to get feedback on one's own work from other organizations. The result of a similar type of workshop was presented in [ER10].

Invitations to the workshop were sent to the network of the researchers. This includes earlier participants at a seminar on "research on open source in industry" where rather many (≈ 50) people attended, and mailing lists to companies in the region. This means that the participants cannot be seen as a representative sample of a population and generalizations cannot be made in traditional statistical terms. Instead analysis must be made according to a qualitative method, e.g. as described by Fink [Fin02, p. 61-78]. This is further discussed in Section 2.4.

2.2 Objectives and discussion questions

The main research questions for the study were:

- How should open source components for inclusion in products be selected? Is there a need to modify selected components, and if so, how should this be done?

- To what extent is code given back to the open source community, and what are the reasons behind doing so?

Discussion questions could be derived from the objectives in different ways. One possibility would be to let the participants focus on a specific project and discuss issues based on that. The advantage of this would be that it would probably be easy for the participants to know what actually happened since it concerns a specific project. The difficulties with this approach are that there is a risk that participants have valuable experience from more than one project and therefore cannot express all experiences they have since they should focus on one specific project. There is also a risk that data becomes more sensitive if it is about a specific project. Another alternative is to ask about more general experience from the participant and let them express this in the form of advice to someone working in the area. That is, the participants use all the experience and knowledge they have, without limiting it to a specific project or presenting details about projects, customers, etc. This was the approach that was taken in this research.

Based on the objectives of workshop, the following discussion questions were phrased:

1. How should one identify components that are useful, and how should one select which component to use?

2. How should one modify the selected component and include it in ones product?

3. How should one take care of updates from the community?

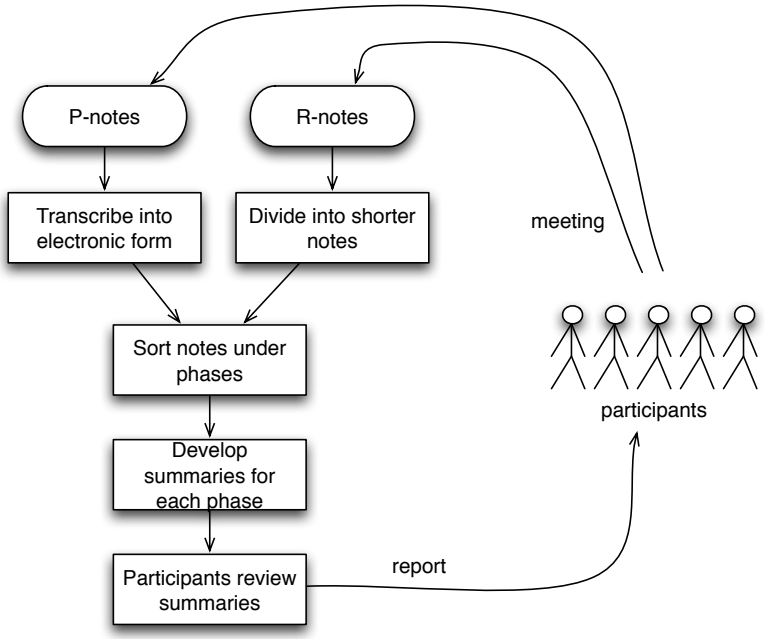


Figure 1: Main analysis steps

4. How should one handle own modifications/changes? What are the reasons for giving back code (or not giving back code)?

In order to get a good discussion, where as many relevant aspects as possible were covered, it was monitored in the following way. For each discussion question, the participants were given some time to individually formulate an answer, or several answers, on a Post-it note. When individual answers had been formulated each participant presented their answer to the others, and the notes were posted on the wall. During the discussions, the researchers also took notes.

2.3 Analysis procedure

The main data that was available for analysis were the notes formulated by the participants (”P-notes” below) and the notes taken by the researchers (”R-notes” below). The analysis was carried out in a number of steps, which are summarized in Figure 1 and explained below.

First all P-notes were transcribed into electronic form. In this step one note was transformed into one line of text. However, in some cases the participants wrote lists with more than one note at each piece of paper. In these cases this was clearly

marked in the transcript. When interpreting the notes, the researcher were helped by the fact that the participants had presented the notes at the meeting earlier.

The R-notes were derived by dividing a longer text into single notes. After this the P-notes and the R-notes were on the same form.

After this a set of phases were defined, based on the lifecycle phases in software development. These phases were based on the areas covered by the questions, but not exactly the same. Then, all notes could be sorted under the phases in which they are relevant.

Next, all notes were grouped in related themes within phases, and based on these summaries were developed. This means that one presentation summary was developed for each phase. The final version of these summaries are presented in Section 4.

Based on this, a report was developed with the summaries. The participants were given the possibility to review and adapt the summaries in the report. This resulted only in minor changes.

This procedure results in a summary, as presented in Section 4. The results were given back to the participants in the form of a technical report. This result is also compared to the literature in Section 4 of this article.

2.4 Validity

Since the collected data is analyzed qualitatively, the validity can be analyzed in the same way as in a typical case study, which in many cases also is analyzed qualitatively. Validity can for example be analyzed with respect to construct validity, internal validity, external validity, and reliability [Rob02; RH09b].

Construct validity reflects to what extent the factors that are studied really represent what the researcher have in mind and what is investigated according to the research questions.

In this study we believe that the terms (like ”open source”, ”component”, etc.) that are used are commonly used terms and that the risk of not meaning the same thing is low. It was also the case that the participants formulated much of the notes themselves, which means that they used terms that they fully understood. Besides this, the researchers participated in the whole meeting, which means that it was possible for them to obtain clarifications when it was needed. Also, the report with the same material as in Chapter 4 of this paper was reviewed by the participants.

Internal validity is of concern when causal relations are examined. In this study no causal relations are investigated.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case.

The study was conducted with a limited set of participants from a limited set of organizations. This means, of course, that the results cannot automatically be generalized to other organizations. Instead it must be up to the reader to judge if it is reasonable to believe that the results are relevant also for another organization or project. The results are compared and validated to other literature and the type of results is not intended to be specific for a certain type of results.

It should also be noticed that the findings from the focus group are based on the opinions of the participants. There may be a risk that the opinions are very specific for one participant or for the organization he/she represents. The nature of a focus group meeting helps avoiding this problem. According to Robson there is a natural quality control and participants tend to provide checks and react to extreme views that they do not agree with, and group dynamics help in focusing on the most important topics [Rob02, Box 9.5].

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers.

In order to obtain higher validity with respect to this, more than one researcher were involved in the design and the analysis of the study. Also, as mentioned above, the report with the same material as in Chapter 4 of this paper was reviewed by the participants.

Another aspect that is relevant to this is how the questions were asked and what type of data the participants are asked to provide. In order to avoid problems with confidentiality, the participants were asked to formulate answers more as advice to someone who is working in the area than as concrete experiences from specific (and named) projects. We believe that this makes it easier to provide data for this type of participants.

3 Results from focus group meeting

3.1 Participants

At the workshop the following participants and organizations participated:

A. Four researchers in Software Engineering from Lund University, i.e. the authors of this paper and one more person

B. One researcher in Software Engineering from another university

C. Two persons from a company developing software and hardware for embedded systems.

D. One person from a company developing software and functionality based on an embedded system

E. One person from an organization developing software and hardware for embedded systems with more than 10 years tradition of using open source software

F. One person from an organization with the objective of supporting organizations in the region to improve in research, innovation and entrepreneurship in mobile communications

That is, in total 10 persons participated, including the authors of this paper.

3.2 Identification

Previously, companies were used to choose between making components themselves or to buying them. Now the choice is between making or buying, or using an open source component. That is, there is one more type of component to take into account in the identification process. It should also be pointed out that it is a strategic decision in terms of whether the product you are developing should be seen as a closed product with open source components or as an open source product.

When components are identified it is important that this is based on a need in the development and that it maps to the product requirements. When it comes to the criteria that are used when identifying components, they should preferably be identified in advance.

In the search process, the advice is to start with well-known components and investigate if they fulfill the requirements. There is also a lot of knowledge available among the members in the communities, so if there are engineers in the organization that are active in the community, they should be consulted. A further advice is to encourage engineers to participate in communities, in order to gain this kind of experience. However, the advice to consult engineers in the organization is not depending on that they are members of the communities. A general knowledge and awareness of existing communities is also valuable.

The next step is to search in open source forums like sourceforge and with general search engines like google. The advice here is to use technical terms for searching (algorithm, protocols, standards), instead of trying to express what you try to solve. For example, it is harder to find information on "architectural framework" than on specific techniques for this.

3.3 Selection

The more general advises concerning the selection process is to, again, use predefined criteria and recommendations from colleagues. It is also possible to conduct a basic SWOT-analysis in the analysis phase.

A more general aspect that is important to take into account is if any of the identified components can be seen as an ”ad hoc standard”, meaning that they are

used in many products of that kind and if it will increase interoperability and the ease communication with other components. One criterion that is important in this selection concerns the legal aspects. It is necessary to understand the constraints posed by already included components and, of course, other aspects of the licenses.

Other more technical criteria that are important include programming language, code quality, security, and maintainability and quality of documentation. It is necessary to understand how much effort is required to include the component in the architecture and it is necessary to understand how the currently used tool chain fits with the component. A set of test cases is one example of an artifact that is positive if it is available in the project.

A very important factor concerns the maturity of the community. It is necessary to investigate if the community is stable and if here is a ”backing organization” taking a long-term responsibility. It is also important to understand what type of participants in the community that are active. The roadmap of the open source project is important to understand in order to take a decision that is favorable for the future of the project.

3.4 Modification

First it should be emphasized that there are disadvantages of making changes to an own version of the components. The disadvantages are that the maintenance costs increase when updates to new versions of the components are made, and it is not possible to count on extensive support for specific updates from the community. So, a common recommendation is to do this only if it is really necessary.

There are some reasons why modifications must be made. Especially adaptation to specific hardware is needed, but also optimizations of different kind. When these changes are made it is in many cases favorable to give back to the community as discussed in the next section but if this is not possible an alternative is to develop ”glue software” and in that way keeping the API unchanged.

If changes should be made it is necessary to invest effort in getting a deep knowledge of the source code and architecture, even if a complete set of documentation is not available.

3.5 Giving back code

It is, as discussed in the previous section, in many cases an advantage to commit changes to the open source project instead of working with an own forked version. In this way it is easier to include updates of the open source component. In order to manage this it is in many cases an advantage to become an active member of the community, and maybe also take a leading role in it. When modifying an open source component it is, of course, an advantage if ones own changes can be aligned with the future development of the open source component.

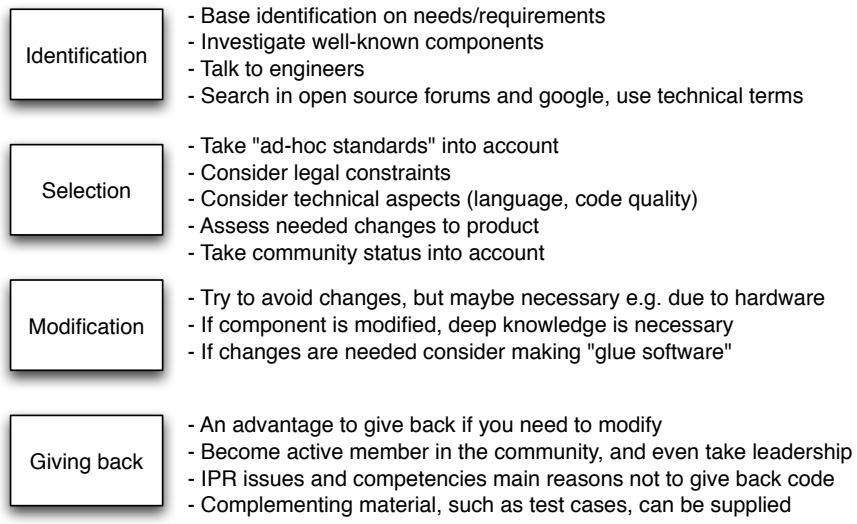


Figure 2: Main findings from workshop

However, there are some reasons not to give back changes too. The most important reason is probably that you want to protect essential IPR's and core competences in the organization. That is, key competence must in some situations be hidden from competitors. It should, however, be noticed that there may be requirements from the license to give back code. Also, after some time, all software will be seen as commodity, which means that this kind of decision must be reconsidered after a while. Another reason not to make changes public is that possible security holes can be made public. In some cases it is easier to get a change accepted if test cases are supplied.

3.6 Summary of results

The main findings from the workshop, in terms of recommendations for the four phases, are summarized in Figure 2.

4 Conclusions

We believe that many of the recommendations from the participants are important to take into account in research and in process improvement in other companies. The most important findings from the workshop are summarized below. The find-

ings are in line with presented research in literature as described in Section 4, although the details and formulations are specific to the results of this study.

In the identification phase it is important to take the needs and the requirements into account, and to investigate well-known components. It is also advised to discuss the needs with engineers in the organization, since they can have knowledge of different components and communities. When forums are searched, an advice is to use technical terms in the search string. When selecting which components to use it is important to, besides taking technical aspects, like programming language, into account, also consider legal constraints and ”ad-hoc standards”. It is important to investigate the status of the community of a project, and the future of the project, which for example depends on the community. In general it can be said that changing components should be avoided if possible. If it is possible to make adaptations with ”glue-code” this is in many cases better since less effort will be required in the future when components are updated by the community. However, there are situations when it is necessary to make changes in the components.

Even if there may be issues with property rights, it is in many cases an advantage to provide code to the community if changes have been made. In general it can be said that it is advised to become an active member in open source projects.

The findings from the focus group meeting were compared to published literature, and no conflicting facts were found.

Together with further research on the subject it will be possible to formulate guidelines for software project managers on how to work with open source software.

Acknowledgments

The authors would like to thank the participants for participating in the study.

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (`http://ease.cs.lth.se`).

References

[ER10] Emelie Engström and Per Runeson. “A Qualitative Survey of Regression Testing Practices”. In: *Proceedings of International Conference on Product-Focused Software Process Improvement (PROFES)*. 2010, pp. 3–16.

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

[Fin02] Arlene Fink. *The Survey Handbook*. 2:nd. Sage Publications, 2002.

[HOA10] Martin Höst and Alma Oručević-Alagić. “A Systematic Review of Research on Open Source Software in Commercial Software Product Development”. In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2010.

[Kon+08] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. “The Focus Group Method as an Empirical Tool in Software Engineering”. In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. Springer, 2008.

[Mun+09] Neeshal Munga, Thomas Fogwill, and Quentin Williams. “The adoption of open source software in business models: A Red Hat and IBM case study”. In: *Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists,* 2009, pp. 112 –121.

[Rob02] Colin Robson. *Real World Reserach*. 2:nd. Blackwell Publishing, 2002.

[RH09b] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

A Case Study on the Transformation From Proprietary to Open Source Software

PAPER III

Abstract

Many large companies, from the traditionally proprietary software industry, are opening up and embracing the open source software (OSS) development process model as a part of their business strategy. Despite the recognized potential the OSS community offers, there are still many questions and unknowns about the transition process. We present an extensive analysis of static software quality metrics changes for Ingres, an open source enterprise database management system (DBMS), as the software was moved from the proprietary into open source software development environment. The software quality metrics of special interest for the research are cyclomatic complexity, effective lines of code, the degree of system modularity, and the amount of comments in the code. The conducted research shows an overall improvement in the software quality metrics and significant increase of the source code base. The overall improvement is comprised of a decrease in software quality metrics for source files that were changed between the proprietary and the OSS version and an increase in software quality metrics for the source files added through Ingres OSS community development process.

Alma Oručević-Alagić, Martin Höst
Extended version of 6th International IFIP WG 2.13 Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, May 30 - June 2, 2010, Proceedings, pp. 367-372, 2010

1 Introduction

In the last few decades, a traditional software production has come to presume an ”in-house” or closed source development process, which means that all development is carried out by engineers employed within the same organization and the software source code that is not available for the general public. There are several different types of OSS licenses [FF02; Ray01a] with common characteristic of source code being available free of charge to the general public. This means that the traditional business models which are based on the sale of software licenses are not applicable for software produced through the OSS community process.

The traditional software development has been taking advantage of third party components, among which open source components have been playing an important role [Li+06b; Li+06a]. Thus, the open source components are perceived as just another third party component or an alternative to in-house solution and COTS. This approach has large similarities to traditional development, but a difference is that the company that uses the open source component must decide to what extent it will participate in the development of the component. It would, of course, be possible to use the available component as is, but for several reasons companies may want to participate in OSS development process, e.g. to increase their understanding of the software they are using, to be able to affect the software evolution, and to support it in order to secure its future existence. There is also an interest not only in participating in an open source community, but to provide a software product to an open source community, e.g. [Bon+07].

Since virtually no development starts from scratch, an important process to investigate is that of transforming a traditional proprietary software product to an open source product. In this process the company will probably have less control over the evolution of the software than in traditional development, and an important question concerns what impact the changes made by the community will have on the software quality metrics such as cyclomatic complexity of the code and modularity.

The purpose of this paper is to investigate one case of this type of transformation. The case that is chosen is the Ingres database management system [Ing09], which, according to many, has received a new breath of life after its release into the open source community. The software was for a long time proprietary and after that it was transformed to open source. Software metrics that show how the source code has evolved under the OSS development process, are identified and then analyzed for this product before the transition started and after the transition has occurred.This is one case of this type of transformation and the intention is to learn as much as posible from this case. It should however be seen as one case and the results are not by default representative for all other cases. Instead, other case studies can bring light on other cases and together they can form aggregated knowledge.

The outline of this paper is as follows. In Section 2, the background informa-

tion on the case software and related research studies is presented. In Section 3, the research method is further defined. Section 4.4 presents the obtained results, while Section 5 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 6.

2 Background and related work

The roots of the Ingres reach back to the 1970s and UC Barkley, when the initial development of the software was started as open source. The same code base was modified and spawned into Sybase and Microsoft SQL server in 1980s. In 1994, the software was acquired by CA (Computer Associates) from the ASK Group, the company that created a proprietary version of the Ingres code. By the year 2004, there were roughly around 5000 customers of the proprietary Ingres DBMS software which was rather small customer base compared to the customer base of some of its major competitors such as MySQL and Oracle. In order to increase the market share, CA decided to transform the product to open source in 2004. The company implemented loss-leader/market positioner business model [Ray01a]. To reaffirm its commitment and support to Linux development process, Computer Associates has contributed Kernel Generalized Event Model software to Linux. The software, incorporated into the Linux kernel, improves security of Linux and feeds performance information from Linux systems to management systems [O'G04].

In November of 2005, Computer Associates and Garnett and Helfirch capital created a new company, Ingres Corporation. The main role of Ingres Corporation is to oversee the open source development process, provide support and services for Ingres and OpenRoad. Today, Ingres customer base includes 10,000 enterprise customers, among which 136 belong to the Fortune 500 companies like 3M, Bea Systems, and Lufthansa [Ass09]. Hence, the positive turnaround Ingres has made since it went open source, make the analysis of the software code quality metrics even more interesting, especially when viewed from the historical perspective, i.e. comparing the code metrics of the last proprietary version of the software from 2004, and the most recent one released as open source in November of 2008.

The Open Source Report, released in 2008, is the product of a two years long effort by Coverity Software with support from the US Department of Homeland Security[Sof08]. Over 55 million lines of code over the two year period for more than 250 open source projects, totaling around 10 billion lines of code, was analyzed. One of the main purposes of this study was to provide developers with better understanding of the relationships between software defects and fundamental elements of coding such as function lengths and code complexity.

Bonaccorsi et. al. [Bon+07] have investigated business firms involvment in OSS projects. They found that in 97 out of 300 sampled projects, at least one business firm was involved. Three main kinds of involvement were found: project

coordination, which was the most frequent case, collaboration of software development, and provision of code. This confirms the need for investigation of the process of transforming proprietary software to OSS.

Stemlos [IS02] conducted code quality analysis in open source development for 100 applications written for Linux. It was determined that some open source products have lower quality of code produced in OSS environment then that which is expected as an industry standard. Unlike this research which compares software quality of the same product in its latest proprietary version and version created after 4 years of OSS development, the presented work in [IS02] analyses quality metrics for code produced by OSS community against industry standard.

Related work also includes the work of Gurbani et. al. [VKG06], who have conducted a case study of managing an internal project as an OSS project. The difference compared to this research is that the work presented in [VKG06] concerns a project that was kept within a company, and there was not the same focus on code quality metrics.

3 Research approach

3.1 Introduction

The study is conducted as a case study [RH08]. The investigated case is the transformation of the Ingres code, from proprietary to open source. The study is exploratory with the overall objective to understand what type changes that were made to the case software and how this affected some commonly use code metrics.

In this study a quantitive approach has been taken. Hence, code metrics such as cyclomatic complexity, effective lines of code, modularity or average file function count, have been measured and compared. For example, the study performed by Zhang [HZ07] on public NASA datasets shows that static code complexity measures can be useful indicators of component quality.

In order to analyze and compare code metrics of the most recent proprietary version, further referred as 2004v, and open source version, further referred as 2008v, of Ingres, the 2004v was obtained by directly contacting the Ingres Corporation. The 2008v was downloaded from the Ingres Open Source community web site[1] in November of 2008.

3.2 Research questions

The following research questions were investigated during the research:

1. What parts of the Ingres DBMS software components went through the most source code changes in terms of source files added, changed, and deleted?

[1] http://community.ingres.com/wiki/Ingres _DBMS _Downloads

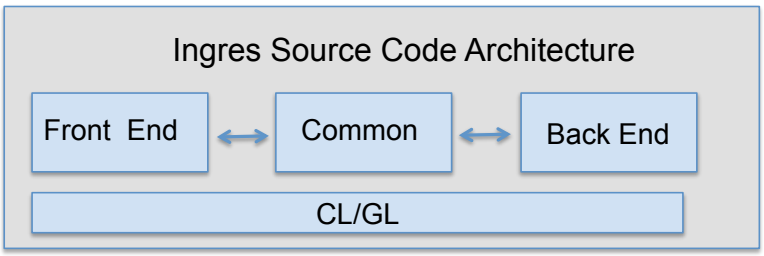


Figure 1: High Level View of Major Source Code Components of Ingres Source Code Architecture

2. How did Ingres DBMS code base change under the OS community process in terms of static source code metrics?

For research question 1, the focus is on architecture level changes. For research question 2, metrics with respect to quality attributes like size, complexity, and amount of comments in the code are of interest. It should be noted that there is no simple linear relationship between these metrics and how ”good” the software is. For example, with respect to complexity it is in general recommended not to have too high complexity, but when the complexity is below a certain value, required functionality cannot be implemented. For comments there is probably a similar relationship. If there are very few comments it is probably not as good as if there are more comments, but if there are very many comments it is probably not better than, or even as good as, if here a bit fewer comments. This makes it important to highlight the objective of this study, i.e. to understand what changes that have been carried out, and not to assess or compare the case software to any other software.

3.3 Investigated software

In order to ease the understanding of the approach for collecting and analyzing data the high level architecture of the case software is described here. The architecture is illustrated in Figure 1. It is grouped into four major components:

Front End: Functionality covers user interface facilities.

Back End: Functionality covers DBMS server functionality.

Common: Functionality covers connectivity and communications between the front end the back end.

Utility: Functionality covers utility libraries that interact with operating system.

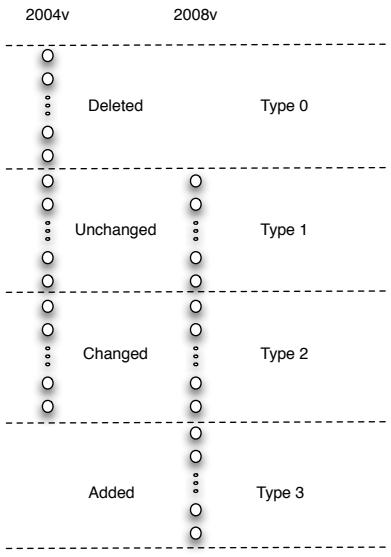


Figure 2: File types (files represented by circles)

A program that parses through the 2004v and 2008v code base was created, or more specifically, the files and subdirectories under the main `src` directory that contains all of the source files. The files were compared between the two code bases in order to determine which files exist only in 2004v, further referred as file type 0, which files are identical in 2004v and 2008v versions, further referred as file type 1, which files were modified between the two versions, further referred as file type 2, and finally, which files only exist, in 2008v, further referred as file type 3. The following list provides an overview of the changes, and the types are also shown in Figure 2.

- File type 0 : Source files that can be found only in 2004v
- File type 1 : Source files identical - unchanged between the 2004v and 2008v
- File type 2: Source files that were changed between the 2004v and 2008v
- File type 3: Source files that were added in 2008v

3.4 Metrics

The following metrics were measured in both versions:

- lines of code (LOC)
- effective lines of code ($ELOC$)
- comment lines (C)
- total cyclomatic complexity (TCC)
- file functions count (FFC)

All metrics are calculated on file level. For example, LOC denotes how many lines of code there are in each source file, and the sum of these values for all source files denotes the total number of lines of code per source code base. The LOC metric takes into consideration all lines of code but blank only or comment only lines. Hence, all lines in the source file except the blank lines or comment lines are taken into consideration by the LOC statistic.

For coding purposes developers often use braces or parenthesis to make code more readable, but this practice can inflate LOC metrics [RSM08]. The $ELOC$ metric takes into consideration all lines of code except blank only or comment only lines as well as the lines containing only standalone braces or parenthesis (`{`, `}`, `(`, `)`) Thus, lines counted by the $ELOC$ metric are a subset of the lines counted by the LOC metric.

C denotes the number of comment lines. The comment lines can appear by themselves on one physical line of code, or can be co-mingled.

The TCC or total cyclomatic complexity metric, also known as McCabe's cyclomatic complexity, is the degree of logical branching per source file. The cyclomatic complexity is calculated as

$$TCC = E - N + 2P$$

where E denotes the number of edges of the graph, N the number of nodes of the graph, and P is the number of connected components [FP98]. This value is calculated for each function in a file. For each file a value is calculated as the sum of the complexity of each function in the file.

FFC, or total number of file functions, within a source file determines the modularity of the file.

The FFC metric combined with $ELOC$ metric produces average number of effective lines of code $AELOC$ metric, calculated as:

$$AELOC = \frac{ELOC}{FFC}.$$

In the same way, the average cyclomatic complexity (ACC) can be calculated as:

$$ACC = \frac{TCC}{FFC}$$

In addition to the above metrics, the amount of comments are of interest. Therefore a metric describing the relative number of comments in each file RC is calculated:

$$RC = \frac{C}{ELOC + C}$$

Metrics for each file were derived with a metrics tool and stored in a database together with file type information for analysis.

3.5 Analysis procedure

Analysis with respect to research question 1 was conducted by determining the percentage changes in terms of file type 0, file type 1, file type 2, and file type 3 per major components of the source code. The components correspond to directories listed under "src" directory which houses all of the Ingres source code. In addition, more granular analysis were conducted on file level.

When analysing research question 2, the differences between the different versions of the case software were investigated with hypothesis tests. The null hypotheses state that the code changes made to 2004v, resulting in 2008v, had no impact on code metrics. The two-sided alternative hypotheses state that there was an impact.

Let $T = \{0, 1, 2, 3\}$ denote file types according to above and let $M = \{AELOC, ACC, RC\}$ denote the different metrics of interest, so that $\mu_m(v, T_s)$ represents the expected mean of metric $m \in M$ for all files of types $T_s \subseteq T$ in version v. Then the following null hypotheses and alternative hypotheses have been defined:

$$H0_{m,changed} : \mu_m(2004v, \{2\}) = \mu_m(2008v, \{2\})$$

$$Ha_{m,changed} : \mu_m(2004v, \{2\}) \neq \mu_m(2008v, \{2\})$$

and

$$H0_{m,new} : \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{3\})$$

$$Ha_{m,new} : \mu_m(2004v, \{0, 1, 2\}) \neq \mu_m(2008v, \{3\})$$

and

$$H0_{m,all} : \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{1, 2, 3\})$$

$$Ha_{m,all} : \mu_m(2004v, \{0, 1, 2\}) \neq \mu_m(2008v, \{1, 2, 3\})$$

That is, three null hypotheses have been formulated for each metric in M so that there is one concerning only the changed files ($H0_{m,changed}$), one concerning all files from 2004v and only newly added files to 2008v ($H0_{m,new}$), and one concerning all the files in 2004v and 2008v ($H0_{m,all}$). This means that $|M| \times 3 = 3 \times 3 = 9$ null hypotheses and equally many alternative hypotheses have been defined in total.

This means that hypothesis tests are conducted by treating file level measurements as independent samples. This gives the possibility to see if observed changes are of a true pattern (it is possible to reject the null hypothesis) or if they have occurred more by chance (it is not possible to reject the null hypothesis). Analysis of data for distributions of metrics results for version 2004v and 2008v were performed and it was determined that data for the metrics do not follow normal distribution. Hence in order to compare distribution of the metrics, non parametric tests, Mann-Whitney and Wilcoxon were performed. The Wilcoxon Signed-Rank Test for matched pairs was used in order to compare paired data sets (i.e., in analysis of $H0_{m,changed}$), and the Mann-Whitney U test was used to compare un-paired data (i.e., in analysis of $H0_{m,all}$ and $H0_{m,new}$).

3.6 Validity

In this section the validity of the research is analysed with respect to the types of validity threats presented, for example, in [Woh+00].

Construct validity: The construct validity is related to the relationship between the concepts and theories behind the experiment and what is measured and affected. Commonly accepted metrics for static software quality measurements such as cyclomatic complexity, lines of code, file function count, effective lines of code, were used. This means that the risk of using metrics that do not represent the concept of code quality is lowered.

Conclusion validity: The conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. All of the population distributions analyzed did not follow normal distributions, and thus, in order to analyze the distributions, tests of lower statistical power than the t-test had to be used. Hence, the statistical tests used to analyze the data were Wilcoxon Signed-Rank Test for a matched pairs experiment and the Mann-Whitney U Test for independent random samples. This means that the statistics were not dependent on a normal distribution. It should be noted that the number of data points can be considered high. This means that even if non-parametric tests were to be used, the chance of detecting differences in distributions can be seen as high due to the large amount of data.

Internal validity: The internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. Over some period of time, software quality will change, as the software goes through various maintenance processes. The software that is left unchanged for a longer period of time, will normally see its software quality metrics depreciated e.g. as stated by Lehman's laws [Som07]. On the other hand, a software that is maintained will see a change in its software quality metrics. It is unknown

what an average change in software quality metrics for similar products over four year period would be in proprietary environment. If the average change amount was established then it could be compared to the one introduced by the OSS maintenance process. However, the fact that the product has been transformed to OSS has been a major event for the product during these years, and it is not probable that the transformation have not had any affect on the quality.

External validity: The external validity is related to the ability to generalise the results of the experiments. While the case software is quite relevant in terms of its source code size, market, decades long life span, and impressive customer base, more research is needed to make general conclusion on whether the results of this study can be applied to other similar software systems. This is a case study, and the focus is on the case as such and not on generalization.

4 Results

4.1 Research question 1: Distribution of source code changes

Figure 3 shows the distribution of 2008v source files in percentages for each subdirectory under src directory, or i.e. directories: src/tst, src/tools, src/testtool, src/sig, src/ha, src/gl, src/front, src/dbutil, src/common, src/cl, src/back, and src/admin. Not all of the source code subdirectories will be analyzed in more detail, but only front, back, common, gl, and cl, since these directories contain almost 95% of the code. Hence, the most of the source files are located under /src/front directory or 54.7% of all 2008v. In the second place is src/cl directory housing 15.40% of source files in 2008v, followed by the src/back and src/common, housing 14.06% and 10.44% of all 2008v source files, respectively. Thus, these four directories contain 94.6% of 2008v source files. For this reason, in the following discussions, the types of changes made to these directories are analyzed in more detail (see section 3.3).

Under the src/front directory the components that belong to the front end layer of the software are stored. The front end functionality includes embedded SQL support, character based tools such as Application by Form (ABF), Query by Form (QBF), Report by Form (RBF) and Terminal Monitor (TM). Furthermore, the front end also includes Web Deployment Option that enables inclusion of data from Ingres data source into HTML page. Finally, the front end also houses the functionality related to replication that facilitates consistency of data sources located on different targets. From the data in Figure 3, it is clear that over 50% of all changes in the front end layer are due to the addition of the new source file components (type 3). Another 33% of changes are due to changes (file type 2).

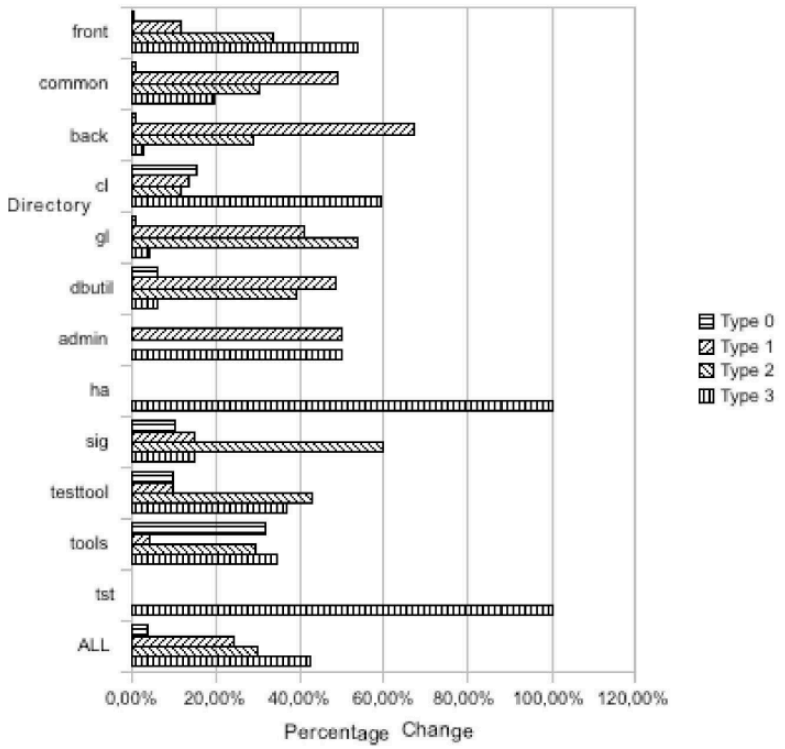


Figure 3: Distribution of file types (0, 1, 2, 3 according to Section 3.3) for source code directories

Therefore, around 88% of front end layer source files have been changed since the case software went open source. When this fact is combined with the fact that the front layer houses 54.7% of all source files, it can be said that some 48%, or almost a half of the source code was added and changed, with 44% of all 2008v file changes being contributed to addition of new source files (file type 3) to the front layer.

Under src/cl library source files for Ingres Compatibility Library are housed. The Ingres Compatibility Library provides interface to underlying operating system. This library provides the common interfaces for Memory, I/O, IPC and it may not call the higher levels of Ingres code. Referring to Figure 2.0 it can be observed that this library grew 69% between 2004v and 2008v, that is it contains 69% of file type 3 files. The src/back end components are deemed very important as the proper functioning of these components significantly affects database performance. The back end components are responsible for query storage, parsing, optimization and execution. In addition, the back end also facilitates logging, locking, archiving and

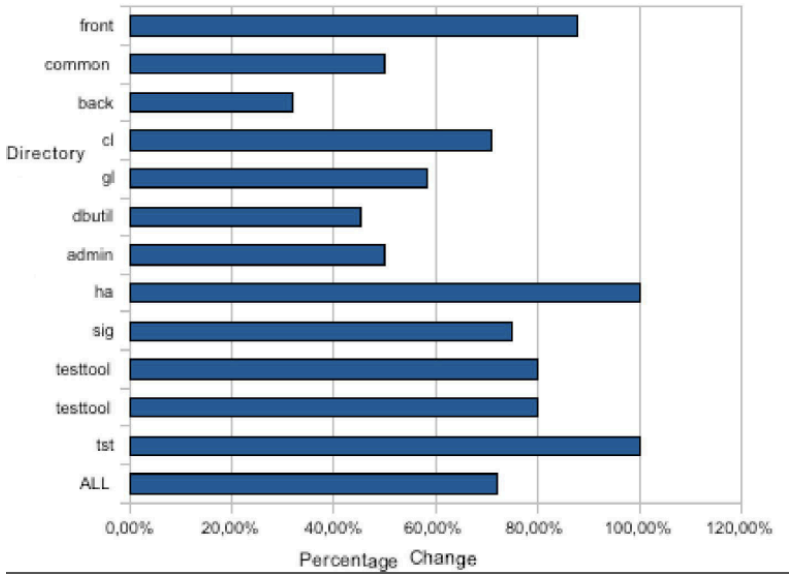


Figure 4: Total source code changes (File Type 2 and File Type 3) per each top level source code directory

recovery operations. The back end components went through the least amount of source code changes and additions, having 67.5% of code unchanged (file type 1) between the 2004v and 2008v. It also contains the least number of file additions (file type 3), thus only having 2.92% of the total number of the source files added (file type 3).

Finally the src/common contains components used by both, front and back end. These components include Abstract Datatype Facility (ADF), Common Utility Facility (CUF), General Communications Facility (GCF), Ingres .NET Data Provider, Java Database Connectivity (JDBC), Open Database Connectivity (ODBC) and Open Application Program Interface (Open API). The common components contain 49.05% of file type 1, or almost half of its components are same for 2004v and 2008v. It can be observed that 19.5% of its file were of file type 3, or newly added components.

4.2 Research question 2: Change in Static Code Quality Metrics

Table 2 displays code metric statistics summarized for the entire source code base of 2004v and 2008v. Hence, it can be observed that the number of file functions, lines of code and effective lines of code has increased. As one would expect,

Table 1: Summary of source code metrics for the whole system

Code Metric	2004v	2008v
Total LOC	840,502	1,442,225
Total $ELOC$	650,055	1,110,261
Total C	484,349	630,635
Total TCC	167,753	300,493
Total FFC	15,588	45,216

Table 2: Mean values and p-values

$H0$	mean 2004	mean 2008	p	reject $H0$
$H0_{AELOC,changed}$	41.35	41.69	< 0.001	yes
$H0_{ACC,changed}$	10.47	10.80	< 0.001	yes
$H0_{RC,changed}$	0.53	0.54	1	no
$H0_{AELOC,new}$	23.68	11.85	0.0042	yes
$H0_{ACC,new}$	6.12	2.80	0.01	yes
$H0_{RC,new}$	0.56	0.42	< 0.001	yes
$H0_{AELOC,all}$	23.68	19.02	0.1383	no
$H0_{ACC,all}$	6.12	4.85	0.1841	no
$H0_{RC,all}$	0.56	0.50	< 0.001	yes

the higher number of functions and lines of code produced higher values for total cyclomatic complexity of 2008v code compared to 2004v.

The results of hypothesis testing for the stated hypotheses are presented in Table 2. As significance level, 0.05 is chosen.

Concerning $AELOC$, this metric is somewhat increased for changed files, meaning that when files are changed the functions in the files have become somewhat larger. For new files the metric is much lower than for old files, meaning that functions in new files are smaller than in older files. In total, looking at all files, the metric is higher in the new version than in the older version. The differences are statistically different for changed code and new code compared to old code, but not for all code.

Concerning ACC the same type of observation as for $AELOC$ can be made. For changed files the complexity is slightly higher and for new files the complexity is much lower.

For RC there is no significant difference for changed code, but for new code there are significantly less comments. In total there is relatively less comments in the new version compared to the old version.

5 Discussion

The results of the conducted research indicate that in terms of number of files that were changed and updated, source files grouped under the front end component were most affected. The source files grouped under the components library (the src/cl directory) have seen the most of the 2004v source files deleted (file type 0) number-wise. The least amount of changes was seen in the back end component or src/back library. This means that more changes have been made to the "top level" components than the more "lower level" components. There can be many reasons for this, e.g. simply that more changes were needed in these components, but another reason may be that these are nearer to the interest of the new community that was formed during the open source transition process. This is a question that can be investigated in future research.

The metrics pertaining to file type 2 changes indicate that changes made to 2004v source files resulted in significant increase in average cyclomatic complexity ACC and increase in average effective lines of code $AELOC$. A simple answer to why this happened cannot be given, but one possible explanation may be that many times in practice, when maintenance of certain components is done, rather then doing complete re-factoring of the source code affected by the changes, chunks of code deemed too complicated to thoroughly re-factor are surpassed. This way, changes made to source code in terms of the affect on the rest of the system are minimal, but such actions can increase complexity. This too needs further research.

The results of comparison of code quality metrics between all files in 2004v and new files in 2008v show significant and large decrease in ACC and $AELOC$, that is, significant and large increase in quality metrics for code developed by the OSS community. The code quality decrease in metrics smaller than the increase of the changed files, and as a result the code quality metrics for 2008v are higher than those of the 2004v, but this increase in code quality is not significant.

At the same time the number of comments per effective lines of code (RC) has seen significant decrease between the 2004v and 2008v of source code base. Hence, while there was a small improvement in ACC and $AELOC$, the lower number of comments per effective lines of code suggests that code in OSS community was not documented as much as in closed source environment. This is also an input to further research. It cannot at this stage be said if the lower number of comments is an increase or decrease of quality, but it is clear that there has been a change in the way comments are made.

For the companies planning to go open source, this study can provide an example on how the OSS community can have a positive impact on software quality metrics in terms of files that are added to the source code base, but also the negative impact in terms of the files that were changed.

6 Conclusions

The conducted analysis have shown that over half of the changes made to the case source code were made in the front end group of source code components, while the least of the changes were seen in the back end components. The overall code quality metrics, in terms of average cyclomatic complexity and the average effective lines of code per function has increased somewhat for changed code, and decreased rather much for new code. This might be interpreted as an improvement for added code. The number of comment lines per effective lines of code ACC has decrased and there are significantly less comments in newly added code.

The transition of the software was also accompanied by 100% increase in customer base, out of which some 138 customers belong to the Fortune 500 group, and 32% revenue increase reported for the 2008. Hence, the example of Ingres DMBS software migration from proprietary to OSS environment provides one example on how software development can be transitioned from the proprietary environment to OSS community and what kind of impact community can have on the static software quality metrics. To be able to draw some more general conclusions or propose guidelines for improvement of the proprietary to OSS transition process and related software quality metrics, a more analysis of ongoing and completed transition processes should be done.

Based on the presented research it is possible to formulate a number of research questions for further research in the area. In the research it was found that the complexity and size of changed functions increased somewhat. Further research can be carried to understand more about the reasons for this, and to understand if it is an effect that is general for more systems.

It was also found that the length and complexity of functions that were added after the software was transformed to open source were lower. Further, it was found that the amount of comments were lower in code added after the open source transistion. These facts could also be further researched. More research is also needed to determine how these static quality metrics affect dynamic software quality metrics such as performance, reliability etc.

Acknowledgment

The authors would like to express their gratitude to the Ingres Corporation for providing us with a last proprietary version of the software.

References

[Ass09] Matt Assay. *February 2009 Web Server Survey*. `http://news.cnet.com/8301-13505_3-10156188-16.html`. 2009.

[Bon+07] Andrea Bonaccorsi, Dario Lorenzi, Monica Merito, and Cristina Rossi. "Business Firms' Engagement in Community Projects - Empirical Evidence and Further Developments of the Research". English. In: *Proc. International Workshop on Emerging Trends in FLOSS Research and Development*. 2007, pp. 57 –61.

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

[FP98] Norman E Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach, Revised*. PWS Publishing Company, ITP International Thomson Publishing Company, 1998.

[HZ07] Ming Gu Hongyu Zhang Xiuzhen Zhang. "Predicting Defective Software Components from Code Complexity Measures". In: *Dependable Computing, IEEE 13th Pacific Rim International Symposium* (2007), pp. 93–96.

[Ing09] IngresWebSite. *Official Web Site of Ingres Corporation*. `http://ingres.com/`. 2009.

[IS02] Apostolos Oikonomou Ioannis Stamelos Lefteris Angelis. "Code Quality Analysis in Open Source Development". In: *Information Systems Journal* 12.1 (2002), pp. 43–60.

[Li+06a] Jingyue Li, Marco Torchiano, Reidar Conradi, Odd Petter N. Slyngstad, and Christian Bunse. "A state-of-the-practice survey of off-the-shelf component-based development processes". In: 2006, pp. 16 –28.

[Li+06b] Jingyue Li, Reidar Conradi, Odd Petter N. Slyngstad, Christian Bunse, Marco Torchiano, and Maurizio Morisio. "An empirical study on decision making in off-the-shelf component-based development". In: *Proceedings - International Conference on Software Engineering*. 2006, pp. 897 –900.

[O'G04] Maureen O'Gara. *CA Exorcises Linux' Hooking Demons*. `http://maureenogara.sys-con.com/node/44941`. 2004.

[RSM08] RSM. *Effective Lines of Code eLOC Metrics for popular Open Source Software Linux Kernel 2.6.17, Firefox, Apache HPPD, MySQL, PHP using RSM*. `http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm`. 2008.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[RH08] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Sof08] Coverity Software. *Open Source Report*. `http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2008.pdf-`. 2008.

[Som07] Ian Sommeville. “Software Engineering”. In: Addison Wesley, 2007.

[VKG06] James D. Herbsleb Vijay K. Gurbani Anite Garvert. “A Case Study of a Corporate Open Source Development Model”. English. In: *Proc. International Conference on Software Engineering (ICSE)*. 2006, pp. 472 –81.

[Woh+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. The Kluwer International Series In Software Engineering. Kluwer, 2000.

A Prolonged Two Phase Case Study on Implementation of Open Source Development Practices within a Large Company Setting

PAPER IV

Abstract

It has ben seen that the implementation of open source development practices within commercial settings can bring benefits such as improved source code quality, lower maintenance costs, rapid team redeployment, and increased innovation. However, despite the benefits, a wide-spread in-house implementation of the practices has not been observed. The goal of this research is to understand factors which hinder the implementation. For the purpose, development practices of a large, global software and hardware organization that bases its products on open source software, and has over a decade long experience of contributing to various open source projects were studied. The results were validated through a set of structured interviews and a focus group meeting. It is found that the initial implementation of the process has not been carried out in a planned and systematic way

Alma Oručević-Alagić, Martin Höst, Submitted.

within the company, but rather in superficial manner, implementing some aspects of the process. The results of the follow-up focus group meeting show that while the company's practices acquired a higher degree of alignment over a two-year period, the change was necessitated by a need to have a more efficient development effort across new, globally distributed, development sites. This study shows that experience and a favorable view of open source development processes motivates the company to implement open source development practices in house, but when the implementation is not systematically planned and carried through, very little benefit is gained. It also shows that as the company grows, and development sites become globally distributed, a greater level of alignment evolves.

1 Introduction

Open source software (OSS) has influenced the way software is produced and distributed. Some companies use open source as another type of off-the-shelf software, while others integrate it into their software products and actively participate and contribute code to open source communities, see for example Höst and Oručević-Alagić [HOA11]. By participating in OSS development processes, companies gain experience in how online, distributed collaboration effort is organized, and how information and knowledge are managed in these projects. While the software built by OSS communities spans a wide range of domains, sizes, and maturity levels, an increased industry interest and involvement with OSS came with the emergence of a large, complex, and industry-grade software products, such as Linux [Fou15c], Android [Inc13], and Hadoop [Fou15b].

Most common profile of the developers contributing to OSS projects is that of unpaid, geographically distributed, and well integrated into highly organized and structured fabric of OSS projects [Ray01a]. However, with increased usage and participation of the industry in OSS projects, a greater participation of paid developers can be noted. As OSS gains mainstream acceptance, business models and industry involvement strategies mature as presented e.g. by Fitzgerald [Fit06a]. While it is important to understand the OSS impact on the software and related industries, understanding how OSS development practices could be applied in house to enhance proprietary software development process requires further investigation for several reasons. Scacchi [Sca10] argues that OSS development is an interesting alternative approach to development of large systems and suggests that further research, especially using empirical examination, is conducted in order to better understand OSS development practices (OSDP). A description on adopting open source development practices within the organizations, also known as inner source, was proposed by Stol and Fitzgerald [Fit06b]. They show three important aspects of software projects that are run as inner source: types of projects, practices and tools, and people and management. While there exist a number of case studies showing successful adoption of OSS in-house, HP [MM08], Lucent [Gur+06],

and Nokia [Lin+08b], still more evidence is needed to better understand how the companies apply OSDP internally and what common issues they encounter in the process.

In this case study we present alignment of software development practices and OSDP in a large, international, software and hardware company, referred to as the Case Company. The company has a long experience of working with mature communities, and has recognized the value of the OSDP. This study is a second part of the two phase study [OAH14a] which tracks the adoption of OSDP in commercial setting over a two year period.

The outline of this paper is as follows. In Section 2, the background information on OSDP is presented. In Section 3, the research approach is further defined, thus stating what the relevant research questions are, analysis approach and validity concerns. Section 4.4 presents the obtained results, while Section 5 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 6.

2 Background

Some of the studies that are used to demonstrate successful application of OSDP within commercial settings are the ones conducted in HP [MM08], Lucent [Gur+06], and Nokia [Lin+08b]. The studies analyzed, for example, development of a complex software product across departments [Gur+06] using OSDP, and transitioning of an entire development to adopt OSDP [MM08]. The software produced in such manner is called “inner source”, “progressive open source”, or “closed open source”.

A case study conducted by Stol et al. [Sto+11] focuses on the challenges of building and integrating software products developed as a shared asset. In this case study the focus is on challenges of developing and integrating software developed as a shared asset within the company setting, and comparing these challenges with the challenges of integrating an open source software product developed outside of the company. The Stol et al. [Sto+11] research concludes that organizations can benefit in adoption of OSDP, but that more research in this area is needed to further identify and address the challenges of OSDP within a company setting.

Melian and Mähring conducted a study in HP [MM08] observing the process of progressively transitioning HP’s development team to work under OSDP. The motivation for introduction of POS in HP is the business need to increase the development cost efficiency and shorten time to market by making software highly modular and reusable asset. The research has produced a comparative listing of open source and “progressive open source” development practices. Some of the biggest differences between the two practices lied in the aspects of organizational structure, time and budget to deliver, abundance of available human resources and reward system. They conclude that implementation of OSDP within a corporate

setting can bring long lasting benefits in terms of development efficiency and code quality, but they also state that more research is needed to address differences in reward system, and control and monitoring of individual participants.

A case study conducted by Gurbani and Gavert in Lucent [Gur+06] provides another relevant insight into what happens when a software product is developed within a company as a shared asset, and employees from other departments are involved in its development through development process compliant with OSDP. The lessons learned form that case study are that source code ownership and the "many eyeballs" contributing to a transparent development process facilitate efficient software development especially if the software product is shared and highly utilized across different departments as it was the case in that case study [Gur+06].

All the studies discussed above ([MM08], [Lin+08b], [Sto+11], [Gur+06]) identified the importance of having a common set of standard development tools, a single version control system, and an standardized change management system.

For the purpose of this study a set of Open Source development practices was identified based on the work by Fogel [Fog05] and outlined in Table 1. The work details the most important aspects and characteristics of free software development, based on experiences gained with the Apache Subversion project [Apa12], the OSS source code version control system with widespread use in open source and corporate setting. It provides a valuable insight on how the Subversion [Apa12] community has been built and sustained over a period of twelve years. Besides analyzing the infrastructure needed to support the project in an online environment, Fogel [Fog05] also elaborates on the importance of building a healthy environment culture, facilitating authority based on meritocracy and communication relying on standardized channels and formats.

While shared OSDP aspects, across different sizes and domains of mature OSS projects, include Infrastructure and Communication aspects, defined by IDs S1-S21, in the Table 1, they can differ in governance types, defined by IDs S22-S24. For example, the Linux project adheres to the "benevolent dictator" management practice, where lieutenants are assigned for different parts of the code, but the ultimate decisions are made by Linus Torvalds. The community source governance type for library and related fields software, popularized by Kuali Open Library Environment [KF16], enables institutions to share development resources and influence development of a software project in a closed source setting, provided that in the later phase the project is open sourced. The government type applied in the Apache Subversion project is base on meritocracy, also referred popularly as "do-ocracy", where roles, authority and promotion is based on the participants' demonstrated knowledge and contributions to a project. We argue that such governance model can be suitable also for a closed source industry setting.

While the adoption of open Source practices can benefit companies [Gur+10], there are also some issues it raises. Some of the issues include development of products across organizational boundaries, especially in the companies where the development process is highly hierarchical.

Table 1: Categories of Open Source development practices

Aspect	Category	Subcategory	I.D.
Infrastructure	Product Info	Features	S1
		Documentation	S2
		FAQ	S3
		News	S4
		Road Map	S5
		Security	S6
	Code Access	Download location	S7
		Binary package	S8
		Release Notes	S9
	Community Guide	Community Overview	S10
		Community Roles	S11
		Coding Conventions	S12
		Commit Conventions	S13
		Building and Testing	S14
		Debugging	S15
		Mailing Lists	S16
		Bugs/Issues	S17
		Releases	S18
Communication	Standardized	Message	S19
		Channel	S20
		Norm	S21
Management	Meritocracy	Role	S22
		Promotion	S23
		Authority	S24

3 Research approach

3.1 Introduction

The research presented in this study is conducted as focus group meeting [Rob02], [Kon+08] and it represents a second phase of the two phase prolonged study as depicted in Figure 1 which depicts the entire research process.

The Case Company is a global market leader in software and hardware production within its field, and its core products are based on an OSS product licensed under GPL. The company has over a thousand employees and, through own and partners' offices, it is present in over 170 world countries. The Case Company is also a significant contributor to a number of different OSS communities.

The participants of the focus group meeting were the same individuals that were involved in the execution of the first phase of the study and thus were well acquainted with purpose and details of the study. The meeting discussion was structured around predefined interview questions with overall goal of assessing current level of OSDP implementation within the company and identifying and understanding factors behind any changes in the implementation.

3.2 Research Questions

The main questions that are analyzed in the study are:

RQ1 What differences in the level of implementation of OSDP presented in Table 1 can be noted over the past two years in the Case Company?

RQ2 What are the underlying reasons for the change in the OSDP implementation levels?

The main research question, RQ2, tries to understand and answer why certain practices are introduced and other practices are not introduced. In order to understand this, RQ1 focuses on what has happened at the case company during the last two years.

In order to answer the questions it is necessary to understand the status before the two years started and what has happened during the two years, and why these changes have been made. Phase I (see Figure 1) of the study answers what practices were introduced two years before Phase II. Phase I, as presented in [OAH14a]), was conducted by studying the alignment of the case company and Open Source practices, and to what extent developers thought that practices would be feasible to introduce in their environment. This was done by observing the case company and conducting interviews six persons with roles covering project management, technical lead, architect, and developer. Phase I was carried out over a period of 2 months with the first author visiting the case company during this time. Some further details about the conducted research are presented when the research validity is discussed in Section 3.4.

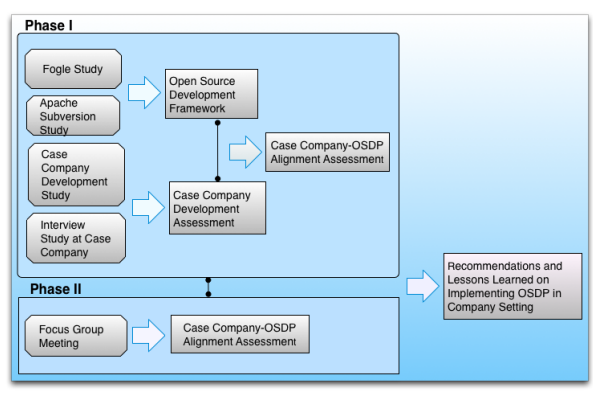


Figure 1: The two phase research process

Phase II was, as described above, conducted through as a focus group meeting. In order to understand what has happened after Phase I, and by that being able to answer RQ1 and R2, a number of discussion questions were phrased and used at the focus group meeting:

1. How often do you use intra company online resources to discuss or solve project related tasks(e.g. online communication, knowledgbase, ongoing projects)?
2. How are change requests handled?
3. Can developers or smaller teams independently choose task to work on?
4. Which aspects of OSDP need to be implemented at greater level and why?

In order to facilitate discussion in an efficient way, the following format was followed. For each discussion question participants were given some time to think about the question and write down their answers on an enclosed questionary. After formulating individual answers, each participant shared his answer with the group, which presented bases for a discussion. The individual answers were then collected by the researchers. During the discussion, one of the researchers also took notes.

3.3 Analysis Procedure

The participants answers were transcribed, systematized based on commonality of their responses, and further analyzed by comparing them to the notes taken by one of the researchers. Based on this, a report was developed with summaries for each of the interview questions. During the analysis the results were also compared to

the results from Phase I. The summaries are presented in Section 4.4 where they are also analyzed and compared to the results from the first phase of the study.

3.4 Validity

In this section the validity of the research is analyzed with respect to the types of validity threats typically present in qualitative studies: construct validity, internal validity, external validity, and reliability [Rob02], [RH08]. As this study is a continuation of the two phased study, the validity assessment includes both of the phases.

The **construct validity** is concerned with the relationship between the subject of the study and what is measured, in this case the alignment of OSDP of mature OSS communities and software development practices of the Case Company. In order to properly identify mature Open Source development practices, the work by Fogel [Fog05] was used and the result was verified by studying the Apache Subversion [Apa12] project. To assess the development practices within the Case Company the first researcher spent two months in the company, studying the company's processes and examining online communication trails and documentation.

Thus, a prolonged involvement [Run+11] was applied in order to improve the validity of the research. The results of the documentation study were discussed and validated with the Case Company senior employees through a set of six structured interviews, i.e. member checking [RH08]. The results were also reviewed by the second researcher who did not spend time in the company, i.e., peer debriefing [RH08]. This also reduces the possible bias that the first researcher might have developed with a prolonged involvement. It also means that research triangulation was applied which also increases validity of the research. The participants included in Phase II of the study were the same individuals as the ones that participated in Phase I of the study, and thus were well acquainted with the subject of research and previous work completed. The participants formulated the answers to the questions themselves, and participated in the discussion that followed which gave the opportunity to discuss and clarify their written answers even further. There exists possibility that participants were not representative sample, but the chances for this are very small.

The **internal validity** is concerned with causal relations. Since the nature of this study is to compare and analyze development practices, the causal relations are not seen as a threat of the study.

The **external validity** is related to the ability to generalize the results of the this study. The OSDP as defined in the paper can be relevant for future analysis. The Case Company studied is large software and hardware company, a world leader in its field, where the main products are built around OSS products. Hence, the participants are experienced in working with mature OSS communities. The competition in the market is typical. Hence, the findings of this research might be relevant to other large software companies that consider implementation of OSDP

internally. It provides a framework of characteristics present in OSDP and an insight on benefits and challenges on implementing OSDP within a company setting.

The **reliability** aspect of validity is concerned with the aspect of data and analysis dependence of the underlying research on the researchers. The study was conducted as a prolonged, two phase, structured case study, with the analysis, interviews, and focus group meeting conducted in a structured way.

4 Results

4.1 RQ1

Research Question RQ1 concerns the changes in alignment of OSDP and software development practices of the Case Company over the past two years. To answer research question 1, the results of the first phase and second phase of the study are presented and compared. The results are grouped under the three categories as outlined in Table 1.

Infrastructure

The web portal is well structured and contains documents with information on organization structure, administrative information, roles and responsibilities, information on development processes, methods, standards, past, and ongoing project related information, code repository, use-net groups, and training manuals. Development processes and the methodology are well defined, with a project management process which is best categorized as a set of sequential steps (also called tollgates), which need to be completed before the next step can be taken. The coding standards are clearly spelled out in the documentation. There exists ongoing project documentation mostly with information on project management plans, allocated resources, assigned tasks, and task completion.

In the fist phase of the study, only two interview study participants indicated that they use the portal in their daily work, the architect and the senior project manager. The two interviewees use it for the purpose of updating project management plans or technical documents. At the same time developers, code block architects, and technical leads indicated that they use the portal very little in their daily project related tasks. They also agree that the documents on the portal were not well organized, were hard to search, and that much of the documentation they were interested in was out of date. In case there were multiple projects related to a product, each project had their own version of the documentation, which in itself was outdated.

In the second phase of the study, all focus group participants indicated that they use the company portals several times on daily bases. The portal is used for the following purposes: to get up-to-date information on current bugs, fixes, and releases, to prioritize backlogs, to review design and architecture, to engage in

communication about software projects, to follow mailing lists for different groups and projects, and as development wiki.

Communication

The internal portal also hosts infrastructure necessary to carry out discussions on various topics and create searchable archives.

In the fist phase of the study, the majority of the interviewees agreed that a great majority of inefficiencies and issues they encounter in their daily work are related to inadequate communication. Most thought that better communication would lead to more efficiency at work. They expressed that usage of electronic communication in a standardized form would be desirable, especially if it would create searchable archives which could later on be referenced for problem solving purposes, similarly to how they would search the internet to understand why programs produce certain error codes and how such issues could be resolved. On the other hand, the majority of the interviewees agreed that it is much more time-efficient and easier to "go and talk" to a person about a problem, recognizing that in this way no written trail on the problem would be left, and thus, no one could refer to it in the future. While there exist non-standardized means to communicate electronically, some interviewees said that the majority of developers refrain from using it, partly due to past experience, where questions and issues brought up through electronic discussions were not addressed in a manner that would facilitate such discussion.

In the second phase of the study, all of the participants indicated that they engage in online communication for the purposes of solving ongoing issues, discussing ongoing project work, and searching for information through communication logs, mailing lists, on daily bases. The participants also noted that the increase in online communication was necessitated by opening of a new, global distributed development sight. They pointed out that in the distributed development environment the "go and talk to a person" option was no longer viable and they also recognized the benefits of having the communication archives of the discussions available as they provided a searchable information trail. This has also encouraged different area knowledge experts to produce wiki documents on the most frequently asked questions, which reduced time inefficiencies previously manifested in repeating the same information in person-to-person communication, which also failed to produced any written trail one could refer to.

Management

In the fist phase of the study, it was found that the organizational structure and roles and responsibilities within the R&D resemble roles which can be found in the OSS communities. Hence, besides developers (code contributors), there are technical leads, code block maintainers, code block architects, and architects. The code block architects and code block maintainers can thus be seen as fulfilling

the role of gate keepers in the open source community. The majority of developers were clearly positive towards the idea of being able to select tasks they would work on from a pool of tasks in the similar manner as this is done in OSS communities. However, interviewees that were in manager positions indicated that this might not be feasible as much time would then need to be spent on managing conflicts for those developers that could not choose tasks or that were stuck with less interesting tasks. All interviewees agreed that task deadlines are needed, but sometimes too tight deadlines tend to negatively affect quality, as there then exists a tendency to put in as much functionality as possible, without properly testing it.

Five of the interviewees thought that the number of formal meetings held was excessive. They expressed that if more time was put in planning of the meeting and appropriate selection of the attendees, the meetings might be less frequent and more efficient. Interviewees at more advanced technical position believed that there was a tendency to involve them into projects too early or too late. This adversely affects efficiency on individual and project level.

Code block architects and code block maintainers noted that in practice their roles overlap with the role of technical lead. Such overlapping roles on the project are conflicting, as technical lead is perceived to be more of a project driver, while code block architects and maintainers are considered to be expert of a product or a part of it with a sole role of making sure that underlying product development is in line with overall architecture.

In the second phase of the study there were no changes noted with respect to the development roles which continued to be aligned with roles observed in mature OSS communities. The development teams are specialized in a part of the platform and do not have the opportunity to initiate changes independently without consulting the platform owner. However, in a case the changes are small, e.g. not affecting common APIs, not conflicting with customer requirements, and development resources are available, teams can independently implement the changes. The majority of the change requests are managed by platform owners who also distributes work to appropriate teams. An increased usage of online communication channels has positively affected the management aspect resulting in less unnecessary meetings, and greater information dissemination resulting in involving appropriate technical resources in early stages of project planning. However, the communication is still restricted to development resources, with platform owners serving as links to other project stakeholders or end customers. Hence, there is no direct feedback loop between the development teams and end customer, as is the case in OSS communities.

4.2 RQ2

Research question RQ2 concerns the underlying reasons for the changes in the OSDP implementation levels. Based on the results of the focus group meeting, the underlying reasons for change in level of the OSDP implementation practices is

opening of a new distributed development site. This necessitated greater level of online communication has taken up the same characteristics as the ones observed in OSS communities. The increase in online communication manifested itself in greater usage of mailing lists, project wiki pages, and other online resources, e.g. for project management, bug reports, etc., thus transferring some of the work from 'live' interactions to an online milieu. This results in the creation of searchable archives, which further increased the development efficiency.

5 Discussion

Based on the results of the prolonged study, it is evident that the level of the alignment of the Case Company development practices with OSDP has increased over a two year period, with the change being driven by the need to achieve greater development efficiency across distributed development sites.

Basing its core software and hardware products on OSS and with development teams highly experienced and versed in OSDP, the Case Company has tried to mimic major characteristics of OSDP as presented in Table 1. The company has replicated many of the roles present in OSS communities, such as the role of code block architects, technical leads, head architects, code maintainers, and they have implemented the online portal and setup the usenet groups. Development resources are highly aligned with OSDP in respect to the common understanding of technical issues and value of standardized practices in design, coding, testing, and development stages. Hence, in both stages of the study, it is evident that knowledge and experience gained on OSS roles and practices, assumed by primarily company employees in technical roles, through participation in the OSS community process were transferred back into the company.

Due to the lack of planned effort to encourage company-wide usage of an online communication milieu, in the first phase of the project it was observed that such resources are used scarcely. While a majority of developers preferred having searchable communication archives, a starting place where one could go to find out if there exists more information about an investigated issue, they perceive that communicating electronically instead of "face-to-face" is less efficient. The interviewees also indicated a reluctance to take part in open electronic discussions and such discussion are not formally encouraged or enforced to any degree. Face-to-face communication reduces the amount of time one needs to spend searching through archives and can also ask "on-demand" for further clarification of an issue. On the other hand, 'face-to-face' communication can be less efficient in case the resources one needs to talk to are not currently available. The second phase of the study showed significantly improved usage of company portals since they enabled more efficient communication and development between the existing and a new globally distributed sites. Transferring some of the 'face-to-face' communication to an online milieu created a searchable communication archives, and improved

the documentation process, e.g. creating more documentation on the system and "F.A.Q. lists". The more transparent, online, nature of discussions on ongoing projects helped involve relevant resources in early stages of project planning, thus reducing the number of unnecessary meetings and ensured that relevant inputs are acquired early on.

The greatest misalignment still is evident in the way work is assigned and projects are managed. There is still a closed feedback loop between developers and outside company partners and customers, and much of this communication is channelled through higher, management level roles, such as platform owners. Developers only make up-stream change request, without involving platform owners in cases where the changes are minor and not affecting common APIs, provided that there resources available.

In the studied case, it seems like the matter of adopting OSDP is highly motivated and carried out by technical personnel that has recognized its benefits through experience with OSS communities. The management structures need to be further educated on the OSDP so an appropriate adoption process can be found and implemented, as this was the case presented in earlier studies, e.g. at HP [MM08] and Lucent [Gur+06].

6 Conclusions

The results of the prolonged case study show that as a company expends and acquires geographically distributed development sites, adoption of OSDP is preferred and a natural way to engage all software product stakeholders in the most efficient way. This is a relevant finding, as with current need for highly skilled work force, many companies struggle with hiring appropriate human resources in one location, and are forced to either open new sites or outsource some part of development.

Unlike the previous studies which show how systematically planned OSDP implementation is carried out, we show what can happen when such process is carried out in a less planned way, driven by individuals in highly technical roles with extensive experience in working under the OSS process. Characteristics of such approach have shown that OSDP are implemented to a higher degree in a form of infrastructure, and less in a form of communication and management practices. Hence, there exist technical roles modeled around the software product, such as head architect and code block maintainer. There also exist standardized development practices and processes facilitating cross project work. The Case Company portal is created with the purpose of resembling an OSS community online milieu, but in practice, during the first phase of the study, the content of the portal was not well organized, complete, or up-to-date. However, with a new, geographically distributed development site, the second phase of the study observed that the online resources were used more, and thus more resembling OSDP. The greater

transparency brought through increased online communication and work has also ensured that appropriate resources are involved in earlier stages of project planning, thus also reducing unnecessary meetings. Repetitive, time consuming tasks, such as 'go-and-talk-to expert' were reduced by having online resources, such as up-to-date documentation and a project wiki.

The prolonged case study presented in this research shows benefits and challenges of implementing OSDP within a closed company setting, when the adoption is not carried out in a systematic and planned way, but rather driven by employees in highly technical roles with experience of working under the OSS communities. We believe that there are more case companies undergoing the same challenges, especially as the software industry increases usage of OSS products and related business models. For this reason, more studies of similar type are needed, not only to raise awareness to the possible problems, but primary to better plan the adoption process so its full benefits can be taken advantage of.

References

[Apa12] Apache. *Apache Subversion Open Source Project*. `http://subversion.apache.org/`. 2012.

[Fit06a] Brian Fitzgerald. "The Transformation of Open Source Software". In: *MIS Quarterly* 30.3 (2006), pp. 587–598.

[Fit06b] Brian Fitzgerald. "The Transformation of Open Source Software". In: *MIS Quarterly* 30.3 (2006), pp. 587–598.

[Fog05] Karl Fogel. *Producing open source software - how to run a successful free software project*. O'Reilly, 2005, pp. I–XX, 1–279.

[Fou15b] Apache Software Foundation. *Apache Hadoop*. `https://hadoop.apache.org/`. [Online; accessed 20-October-2015]. 2015.

[Fou15c] Linux Open Foundation. *Linux Operating System*. `http://www.linuxfoundation.org`. 2015.

[Gur+06] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. "A case study of a corporate open source development model". In: *ICSE*. 2006.

[Gur+10] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. "Managing a corporate open source software asset". In: *Commun. ACM* 53.2 (2010), pp. 155–159.

[HOA11] Martin Höst and Alma Oručević-Alagić. "A systematic review of research on open source software in commercial software product development". In: *Information & Software Technology* 53.6 (2011), pp. 616–624.

[Inc13] Google Inc. *Android Open Source Software Project*. `http://www.android.com/`. 2013.

[Kon+08] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. "The Focus Group Method as an Empirical Tool in Software Engineering". In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. Springer, 2008.

[KF16] Inc Kuali Foundation. *Open Library Environment*. http://www.kuali.org/ole. 2016.

[Lin+08b] Juho Lindman, Matti Rossi, and Pentti Marttiin. "Applying Open Source Development Practices Inside a Company". In: *International Conference on Open Source Systems, OSS* (2008), pp. 381–387.

[MM08] Catharina Melian and Magnus Mähring. "Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard". In: *OSS*. 2008, pp. 93–104.

[OAH14a] Alma Oručević-Alagić and Martin Höst. "A Case Study of Open Source Development Practices within a Large Company Setting". In: *ICSET 2014 International Conference on Software Engineering and Technology, Istanbul, Turkey, Sep 29-30, 2014* (2014).

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[Rob02] Colin Robson. *Real World Reserach*. 2:nd. Blackwell Publishing, 2002.

[RH08] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Run+11] Per Runeson, Martin Höst, Austin Rainer, and Björn Regnell. *Case Study Research in Software Engineering*. Wiley, 2011.

[Sca10] Walt Scacchi. "The future of research in free/open source software development". In: *Future of Software Engineering Research*. 2010, pp. 315–320.

[Sto+11] Klaas-Jan Stol, Muhammad Ali Babar, Paris Avgeriou, and Brian Fitzgerald. "A comparative study of challenges in integrating Open Source Software and Inner Source Software". In: *Information & Software Technology* 53.12 (2011), pp. 1319–1336.

NETWORK ANALYSIS OF A LARGE SCALE OPEN SOURCE PROJECT

Abstract

Industry involvement in open source software development has become a popular practice among companies which, e.g., share software development costs with other community participants or implement an open source based business model. An increased understanding of the underlying social structure and influences within an open source community, especially in a case where the community participants are composed of competing industry members, can be viewed as an important component of company's business strategy planning and management. One way to understand the social structure of an open source community is by applying social network analysis to source code repositories. We use methodology from social network analysis to study Android, an operating system for mobile devices.

The aim of this study is to understand how large and in many cases competing companies collaborate on a large scale company sponsored open source project. We propose a new approach for studying committers' networks as weighted digraph networks. To conduct the case study, change log records of all files bundled under the Android open source project stack were extracted and studied in the context of committers' networks.

Alma Oručević-Alagić, Martin Höst
Extended version of 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, August 27-29, 2014

The results obtained show that Android project development is highly influenced and led by development effort of Google.

This case study shows how a large, company sponsored, and industry backed open source project, i.e. open source project with the majority of community members affiliated with the industry, is structured. In particular, it shows that the involvement of an entire industry eco system within a company sponsored open source project does not imply more equal distribution of the participating community members' influences in terms of committers' social networks. This setup of an open source community by itself does not imply any particular, either positive or negative, connotations. Consequently, the results of the study should be interpreted on a case bases, within a context of a company's strategy to participate or base products around a company sponsored open source product.

1 Introduction

Open source software (OSS) has been growing in importance and affecting the way companies develop their products and services [HOA11], plan their business strategy and compete [Ray01a].

Many companies have already recognized that implementation of software product lines can facilitate software development in an assembly line like manner. A study by Linden et al. [Lin+08a] shows that software product lines can decrease software production cost. In a similar manner, an OSS product can be reused across different companies, e.g., Android phones produced by Samsung, Sony Mobile, HTC, etc. Another study by Linden et al. [Lin+09b] refers to software reused across an industry as commodity software. The study argues that a part of a software product over time loses commercial value, and thus becomes a good candidate for intra-industry or open source development. In this way, development costs of commodity software become shared among the industry members, enabling companies to focus their resources on development of commercial or differentiating parts of their software products.

There have been many studies conducted on open source projects by analyzing source code change logs and mailing list archives in order to understand the underlying structure and behavior of the community. The studies focused either on some individual open source projects [LF+06], or on an entire portal hosting tens of thousands of open source projects [How+06]. Different methodologies have been applied to analyze the obtained data including social network analysis. As open source communities are composed of geographically distributed participants that contribute on volunteer or paid basis (e.g. company participating in OSP), social network analysis has shown to be an effective methodology to analyze participants' affiliation networks, identify the most influential participants, and uncover cliques. The Android was initially developed as proprietary software by the Android corporation. In 2005, Google Inc. bought the Android [And05] and open

sourced the operating system in 2007 [And07] . At the same time, Google also founded an open handset alliance [And07]. The open handset alliance is a consortium of over eighty globally leading companies in market segments of mobile operators, handset manufacturers, semiconductors, and software and commercialization companies. The companies contribute to the development of the Android and deliver devices and services built around the Android operating system. Companies like Vodafone, Sprint, T-Mobile, Acer, HTC, Samsung, Sony Mobile, Arm, Intel, ST Ericsson, eBay, Accenture, are some of the members of the alliance.

Besides the core components open sourced by Google in 2007, the Android also includes over 150 other open source projects, the majority of which were in existence before the Android project. One such project that is included under the Android bundle is WebKit, which was initially developed as proprietary software and later open sourced by Apple. Hence, another interesting aspect of the study is assessment of cross collaboration among participants of various open source projects bundled under the Android stack.

The Android source code committer's network is analyzed through application of social network analysis. The committers are grouped based on affiliations, i.e., all contributors affiliated with a company or an organization are viewed as the same contributor, e.g., committers with a google.com email suffix are viewed as Google company committer. The affiliation data such as authorship information and contribution date and time are extracted from the source code repository logs.

The aim of this study is relevant given the importance of the Android in the mobile device industry and diversity and type of participants within the Android open source project. For any company that plans to either sponsor, lead, contribute to, or build products based on an OSS product, understanding of project's underlying community structure and behavior is an important factor in the company's strategic positioning. This paper shows one way to assess the structure of an open source community by applying social network analysis. Given that Android includes external open source projects, i.e. the projects not developed or open sourced by Google, it is also important to understand how the underlying community structure of the external projects can be affected when the projects are used by another company sponsored open source project with global industry support.

The outline of this paper is as follows. In Section 2, the background information on the case software and related research studies is presented. In Section 3, the research method is further defined. Section 4.4 presents the obtained results, while Section 5 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 6.

2 Background

The field of social network analysis (SNA) is based on network theory [Sal95] that provides tools for analyzing relationships between network nodes. A study

carried by Borgatti and Halgin [BH11] shows that papers indexed under Google Scholar have exhibited exponential growth in the past ten years in the number of articles which use the term social network across all fields of study. SNA has been used to study open source communities by analyzing publicly available data such as source code repositories and communication archives. Luis et al. [LF+06] proposes a methodology for studying committer and module networks. A committer network shows nodes of committers as connected or linked in case they have changed the same module. A module network consists of module nodes that are linked in case a module is changed by the same committers. Social networks where nodes are connected through some affiliation such as "committers modified the same module", or "modules changed by same committers" are also referred to as collaboration or affiliation networks. Research by Cleidson et al. [Cle+05] shows that the network structure of source code is highly related to the way a community is organized. Hence, source code produced by a community also has the community's organization structure inscribed in it. Cleidson also identifies three different forms of participation:

1. Centralized. The code is produced in a centralized way, where through one module, i.e., a central node, other modules are linked.

2. Densely Networked. Represents a network where contributions are equally divided and linked. Hence, there does not exist a node with significantly higher centrality from the rest of the network nodes.

3. Core and Periphery. Indicates that there exists a larger group of core developers that produces code that is highly connected and independent of code produced by another group of developers. Hence, there exist two or more distinct cliques within the network, that are very loosely linked.

A study by Jin Xu et al. [Xu+05] based on social network analysis of 39000 open source projects hosted under SourceForge argues that the open source projects have decentralized structure whose organization resembles that of self-organizing social networks. Further on, the study underlines the importance of understanding the social structure of the projects, especially from the industry's perspective. An ability to understand the evolution of collaborations in an open source project is important especially for the companies that plan to become involved in an OSP.

The aim of this study is to understand the underlying community structure and to identify the most influential participants and clusters through an application of social network analysis to the Android committer's network. The Android project is an important case to study due to the fact that it is maintained by one of the largest software companies of today, Google, as well as the fact that it has attracted the most of the leading companies across the entire mobile device eco system. While there are other open source projects that have a wide spread industry acceptance and usage, the Android stands out in terms of: being a large scale

company sponsored open source stack, having other large and competing companies basing their products around it, and including over 150 other, external open source projects within its core stack.

In particular, this study analyzes the involvement of all companies as source code contributors to the Android project since the project was initiated till today. Hence, the study has a two fold focus; firstly a study of contributions and interactions of contributors within the core Android components, and secondly, a study of contributions and collaboration with other, external open source projects included in Android project.

3 Research approach

3.1 Introduction

The study is conducted as a case study [RH08]. The investigated case is committers' network structure of Android OSP. The study is exploratory with the overall objective to understand how the community participants collaborate in development of the software through the Android OSS process. The data of the study was collected according to the process presented in the Figure 1. Thus, change log data was extracted for each file within the Android repository, and loaded into a database to simplify data manipulation process which identified all pairs of authors that modified the same file. For each identified co-authorship par, weight and direction of the relationship was calculated. Finally, this data was loaded into the Gephi[Org13], software for network visualization and analysis.

In this study a quantitive approach has been taken. A study by Luis et al. [LF+06] has shown how social network analysis methodologies can be used to study OSS projects in order to characterize the projects' evolution over time as well as the projects' structure. Affiliation networks are a special type of social network where two distinct sets of actors are related, e.g., a committer network relates a set of committers to a set of changed source code modules. Hence, there exists a link between two committers when they have changed a same module. An actor or a network node is referred to as a vertex and the links between the vertices are called edges as shown in Figure 2.

In this study we propose an approach for studying committers' networks. In Luis et al. Luis [LF+06] the proposed methodology establishes links between the committers, where the weight of the link or the edge is calculated as being the number of commits performed by committers to all common modules, i.e. the degree of relationship. The definition of the common module differs between projects, but usually corresponds to the top level directories of a source code repository.

According to Borgatti and Halgin [BH11] an important factor to consider when studying strength of the co-affiliation among an event's participants is the actual size of the event. The research suggests that one of the ways to normalize the strength of a co-affiliation between event participants is to weight participation

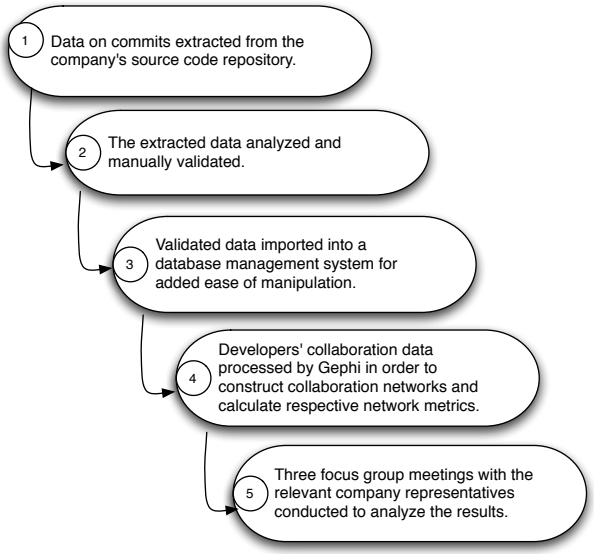


Figure 1: Analysis Process

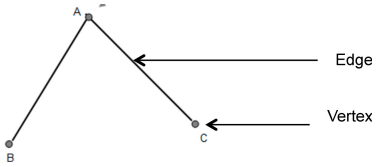


Figure 2: A Three Actor Network

relative to the size of event. In the studied context, the size of the event is the total number of changes made to same file. Then, the strength of co-affiliation among participants relative to the size of event can be expressed as the number of file changes performed by each participant relative to the total number of the changes performed on the file by all participants. For example, if two companies, A and B, make changes to same file, where company A makes only a few changes while company B makes a majority of the changes, then the influence of A over B is much smaller than the influence of B over A relative to the size of the event. A study by Hangal et.al [Han+M] also examines asymmetric influences of nodes through a friendship example and infers weights on friendship relationship. Hence, we propose a new approach to study committers networks as weighted digraphs as shown in Figure 3. The figure shows weights of the edges for committers associated with companies A, B, and C who have changed the same source file 5, 10, and 15 times, respectively. The edge weight is calculated as the number of the committers' changes on a file relative to the total number of changes for the file, which in this case is 30. Thus, the committer A infers a weighted influence of 1/6, B of 1/3, and C of 1/2 to the files's co-committers.

In Riitta Toivonen [Toi+07] argues the importance of strengths of edge ties when modeling social structure and dynamics of social networks. We argue that inferring the edge weight relative to the size of the event provides a more accurate social network structure from the one suggested by Luis et al. Luis [LF+06] which does not take into account relative size of event. For example, if only a degree of relationship is considered in the above example for the committers A, B, and C for, e.g., the total number of files they changed together, then the edge weights between the three committers would be the same. This would mean that the strength of co-affiliation between A, B, and C is the same relative to the source file change event, which is clearly not the case. While this is a simple and trivial example, in a context of a large network, with many committers, where, e.g, a subgroup of committers performs a large number of changes, computing edge weights relative to the number of all changes performed on a file is important in order to accurately assess relationship strength. This is more so as the data on committers, corresponding edges, and their weights are building elements of a network structure, based on which other network metrics are derived.

In this study, the weight of the edge between two participants is calculated on a file level. Affiliation networks link actors into a social network by virtue of participants attending a specific event. In the context of committer network analysis we define the event as performing modifications on a specific source code file. Hence, for a set of actors $V = \{v_1, v_2, ..., v_k\}$ and events $U = \{u_1, u_2, ..., u_m\}$ we define a weight W of an edge between an actor v_i and all other actors that participate in the event u_t as:

$W(v_i, u_t) = \frac{X(v_i,u_t)}{\sum_{c=1}^{k} X(v_c,u_t)}$

where $X(v_i, u_t)$ denotes the number of times an actor/committer v_i made changes to the file, i.e., participated in the event u_t.

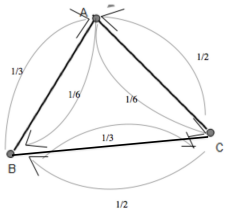


Figure 3: A Weighted Three Actor Network for Modification of One Source Code File

This means that the weight of the edge $W(v_i, v_j)$ for all events v_i and v_j attended together equals:
$W(v_i, v_j) = \sum_{t=1}^{m} W(v_i, v_j, u_t)$

In order to obtain committer data on the source code changes, Android project source code repository was downloaded in November 2012 from the Android project web site [Inc13]. Change log records, with information on authors and change dates for all Android source code files were extracted and loaded into a database. The social network data on network nodes/committers, edges, and associated edge weights and labels was analyzed using Gephi software for social network analysis [Org13]. The labels correspond to the main subdirectories under the Android source code tree, as displayed in table 1. The Gephi software was used to calculate relevant social network metrics which are discussed in more detail in the Section 2.4 as well as to generate a visual representation of the committers' networks. Besides analyzing committer network for the entire repository, we also analyze two additional distinct committer sub-networks. The tree committers' subnetworks are constructed:

1. External committers network which includes committers that changed files located under the external top subdirectory.

2. Core committers network which includes committers that changed files located under all top subdirectories excluding the external subdirectory.

3. The entire committers network which includes committers that changed files located under both, the core and the external subdirectories.

Since the external subdirectory contains source files for over 150 other open source projects, we believe that studying this diverse community separately from the core Android community can provide some additional insight. Committers that participate in the external open source projects do not necessarily use the Android OSS or participate directly in its community process. Distinguishing between the core and external committers can also provide an additional insight into committers that work under the Android OSS project. Finally, a combined network of all, the external and the core committers is studied.

Table 1: Android Subprojects or Modules

Top Level Subdirectory	Description
abi	Features
bionic	The C-runtime library for Android.
bootable	Boot and startup related code.
build	Utilities and scripts for building system implementation.
cts	Android compatibility testing framework.
dalvik	Android virtual machine.
development	Source code for SDK and NDK, and emulator.
device	Product specific code for different vendor devices.
docs	Tutorials, references, and miscellaneous information.
external	External open source projects (WebKit, SQLite, etc).
frameworks	Key Android framework library (JNI, services, phone and telephony components, etc)
gdk	Compiler infrastructure for the NDK based on LLVM
hardware	Libraries for basic hardware support.
libcore	The Harmony Java Virtual Machine used by Dalvik.
libnativehelper	Native development helper library.
ndk	Native Development Kit.
packages	Source code for default Android applications (e.g. calendar, contacts,etc).
pdk	Platform development kit provided to chipset vendors and OEMs before new platform is released.
prebuilts	Files distributed in binary form.
sdk	Android Software Development Kit.
system	Core Linux system libraries.
tools	Development Tools.

3.2 Research questions

The following research questions were investigated during the research:

1. What are the characteristics of the committers' networks for each set of the Android project source files: the core, the external, and combined core and external?

2. How can a company utilize network analysis to study the Android development community?

For research question 1, the focus is an assessment of the three distinct network structures, the core components committer network structure, the external committer network structure, and the combined core and the external committer network structure. Metrics on network influence, clustering, centrality, existence of sub-communities, and network density are presented and discussed.

For research question 2, we analyze results of research question 1 from the perspective of a company planning to develop software through Android or similar OSP.

3.3 Investigated software

A program that parses trough all source files located under the Android OSP subdirectories (table 1) was created and run in order to collect information on all the changes made to all the source files in terms of authors and change dates. The extracted data was loaded into a relational database in order to perform a thorough data validation and provide flexible way to create different file input formats for the Gephi [Org13] social network analysis software. All authors were grouped based on a company affiliation. The affiliation is determined based on committer's e-mail domain suffix. In case email data was not provided, authors individual names are used and no company affiliation is implied. All of the contributions made by the author named "Initial Android Open Source Project Contribution" was excluded from the analysis, as these contributions were not developed under the Android OSS community process, but internally by Google before the project was initially open sourced. The Gephi data records are of the form "source, target, edge weight, edge label". If we consider the earlier example depicted in Figure 3, a sample record would look like "A, B, 1/6, the changed source file's top subdirectory".

3.4 Metrics

The following metrics were measured for the three committer networks:

- Weighted average in-degree $WAID$ and weighted average out-degree $WAOD$ of a vertex.

- Betweenness centrality BC, closeness centrality CC, and eigenvector centrality EVC of a vertex.
- Average Clustering Coefficient ACC of a vertex.
- Modularity MC of a network.
- Number of MCN of a network.
- Graph density GD of a network.

textitWeighted degree of a vertex denotes degree of relationship of the vertex with its direct neighborhood. It is calculated as the sum of weights of all edges connected to the vertex. Since the analyzed network is weighted digraph, there exist two types of edges; the edges originating from a vertex, or the out degree ($WAOD$), and the edges pointing to a vertex, or the in-degree ($WAID$). Hence, the weighted average in-degree of a vertex denotes degree of relationship of the vertex to its direct neighborhood for the edges pointing to the vertex. By the same analogy, the weighted average out degree of a vertex denotes degree of relationship of the vertex to its direct neighborhood for the edges originating from that vertex. In the context of committer network analysis, the out degree can be interpreted as the measure of collaboration strength or influence of the committer on committers in its direct neighbourhood. The in degree can be interpreted as the measure strength of influence of committers in direct neighborhood of the committer on the committer.

textitBetweenness centrality index (BC) is the number of shortest paths that traverse through a vertex and it can be interpreted as a measure of importance of the vertex in a graph. The higher betweenness centrality index of a vertex, the more important the vertex is. In the context of this study, the betweenness centrality index indicates the number shortest distance paths between any two committers which traverse through a committer.

textitCloseness centrality CC indicates how close on average a vertex is to all other vertices. A high value of the distance centrality index identifies vertices that are well related.

textitEigenvector centrality EVC metric measures the influence of a vertex on a network by assigning scores to all vertices in the network. The scores are assigned so that an edge to a higher scoring vertices is valued more than the same edge to a lower scoring vertex. The eigenvector represents the most accurate metric for influence of a vertex on other vertexes in the network. Gephi uses an algorithm by Brandes [Bra01a] to calculate the centrality network measures for weighted graphs.

textitAverage clustering coefficient ACC of a vertex shows the tendency of the network to form cliques or isolated groups. The average clustering coefficient is calculated based on sum of individual clustering coefficients. The individual clustering coefficient is calculated as the number of edges from a vertex to its

direct neighborhood relative to the number of links that could exist between them [WS98a].

textitModularity of a network MC identifies the sub-communities within the network with densely connected vertices. The value of modularity is calculated as a difference in fraction of edges that fall into the sub-communities and a fraction of edges that could be found in the sub-communities if the edges were distributed at random per Blondel et. al[Blo+08]. In the context of the committer network study, the modularity class is used to identify committer sub-networks with higher degree of collaboration. The modularity value falls between the values of -1/2 and 1, where negative number indicates that a random distribution the edges is more likely to form sub-communities than the actually identified sub-communities. The modularity class number MCN indicates the number of identified sub-communities for a given modularity class MC.

textitGraph density index GD measures how close the network is to being complete, i.e., that there exist edges between all the vertexes in the network. A value of 1 for the graph density index indicates a fully complete or connected network.

3.5 Analysis procedure

Analysis with respect to research question 1 was conducted by calculating the $WAID$, $WAOD$, BC, CC, EVC, ACC, MC, GD on the core, external, and combined core and external Android OSP source code tree. For research question 2, the presented network structure data in question 1 is analyzed from a business/-company perspective.

3.6 Validity

In this section the validity of the research is analyzed with respect to the types of validity threats presented, for example, in [RH08].

Construct validity: The construct validity is related to the relationship between the concepts and theories behind the experiment and what is measured and affected. The subset of metrics from the network theory used in this research has been accepted and validated in other studies within the filed of OSP repositories and mailing archive studies. This means that the risk of using metrics that do not represent the concept of social network structure is lowered.

Conclusion validity: The conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. The interpretation of the metrics is grounded in the widely accepted network theory and the field of social network analysis.

Internal validity: The internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. The data extracted from the repositories was examined and validated manually through sampling. The

approach used in constructing committers network is grounded on network theory concepts applied in other disciplines.

External validity: The external validity is related to the ability to generalize the results of the experiments. The studied software is relevant example of a successful industry led OSP as the project includes leading global companies from the mobile eco system.

4 Results

4.1 Research question 1: An assessment of the three distinct network structures, the core components committers' network structure, the external OSPs committers network structure, and the combined committer's and external network structure.

The core committers' network has a total of 250 vertices and 3606 edges, which in this case means that committers have 250 distinct affiliations and there are 1803 distinct committer co-authorship pairs. Since the network is modeled as a weighted digraph, the edges are bi-directional. The external committers' network has 329 vertices and 11196 edges, while the combined core and external committers' network has 513 vertices and 14484 edges.

Table 2 shows ACC, MC, MCN, and GD for the three studied committer network structures. The average clustering coefficients for the three networks show high tendency of the networks to form cliques.

The identified number of closely related sub-communities MCN for the core committer network is 4. However, the MC value of 0,0009 indicates that a probability of such sub-communities occurring at random is very high. Hence, the identified potential sub-communities for the core committer network should be disregarded since their existence is not statistically significant. The number of sub-communities identified within the external committer network is 6 with the MC value of 0,356 indicating that existence of the 6 subnetworks is statistically significant. The number of identified sub communities for the combined, external and internal committers' networks is 7, with the MC value of 0,43 indicating that the existence of the sub-communities is statistically relevant.

The graph density metric GD for the core, external, and combined core and external committer networks is 0,058, 0,104, and 0,055, respectively. The value of 1 for GD indicates that all the components within the network are highly connected. Hence, all three types of the committers' network showing low graph density values indicate that the committers' networks are weakly connected. The high clustering coefficient shows that even though many edges between the committers are absent, committers in a direct neighborhood of a committer are well linked.

Table 2: Summary of the committers' networks measures

Metric	Core	External	Core and External
ACC	0,782	0.791	0,799
MC	0,0009	0,356	0,43
MCN	4	6	7
GD	0,058	0,104	0,055

Figure 4 displays the weighted average in-degree and out-degree for the top 16 committers in core committers' network. As noted earlier, the out-degree in the weighted directed network indicates influence of a vertex over other vertices in its direct neighborhood. Hence, committers affiliated with Google have the highest $WAOD$ value. The value is also more than two times higher than the $WAOD$ value for the second most influential group of committers, that is the group of committers are affiliated with Android OSP community. The $WAOD$ metric decreases tenfold for the third highest rated committer group affiliated with the Gmail.com address as compared to the Google. The weighted average in-degree $WAID$ metric for Google representing the influence of all other committers in Google's direct neighborhood on Google is 50% of $WAOD$ value for Google. Hence, the collaboration strength or the influence of the Google on the committers in its direct neighborhood is twice as high as compared to the influence of all committers in Google's direct neighborhood on Google.

Figure 5 displays the weighted average in-degree and out-degree for the top 20 committers in external committers' network. Committers affiliated with the apple.com email address have the highest value for $WAOD$ metric. Some 30% lower value for the $WAOD$ metric have committers associated with gmail and google. The fourth and fifth highest value of $WAOD$ metric have committers affiliated with nondot.org an zuster.org. The highest $WAID$ value has Google, followed by committer associated with gmail.com email address.

Figure 6 shows the weighted average in-degree and out-degree for the top 20 committers in the combined, core and external committers' networks. Committers affiliated with the Google email address have highest value of $WAOD$. Some 30% lower value of $WAOD$ has Apple, followed by committers associated with gmail.com, nondot.org, and android.com.

The $WAOD$ and $WAID$ metrics indicate that for the entire Android source code base Google has the highest strength of co-affiliation with members in its direct neighborhood.

Figure 7 shows network centrality metrics values BC,CC, and EVC, for the core committers network. Committers with Google.com and Android.com have highest values for EVC, followed by committers associated with Gmail, Sony Ericsson, and Motorla. The value of BC is highest for committers associated with Google.com and Android.com, and decreasing sharply for committers associated

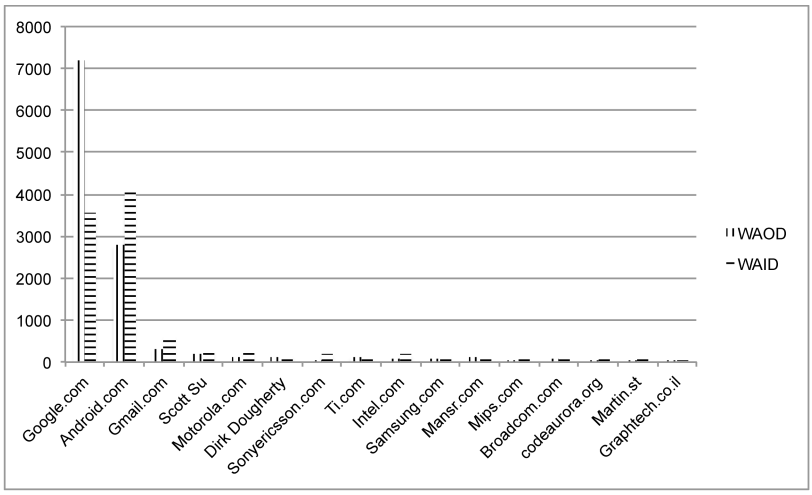


Figure 4: Weighted average in-degree and out-degree for top 16 committers in core committers' network

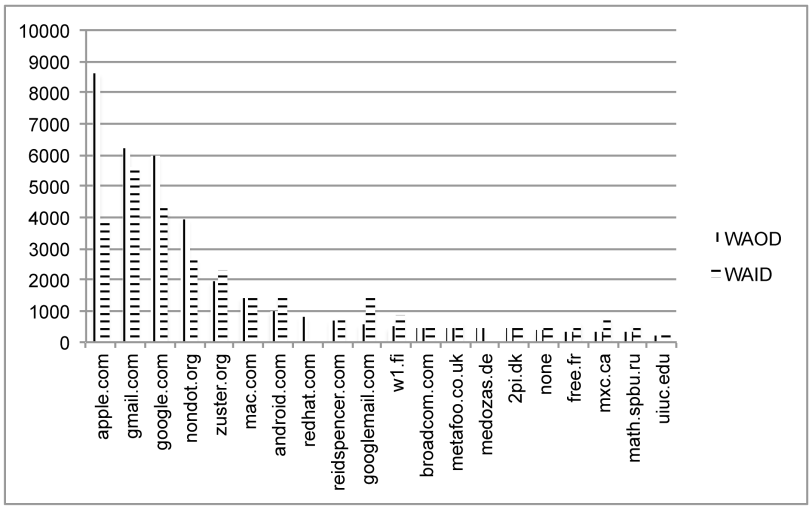


Figure 5: Weighted in and out degree for top 20 committers in external project committers' network

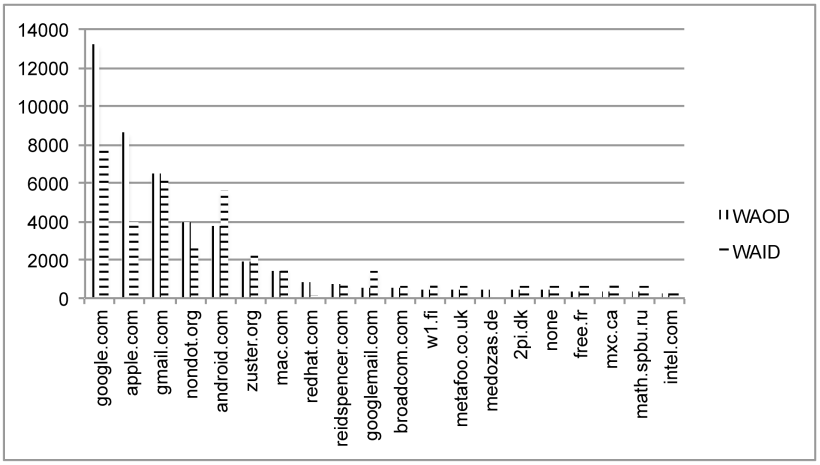


Figure 6: Weighted in and out degree for top 20 committers in core and external project committers' network

with Sonyericsson.com and Motorola.com. Thus, the majority, i.e., 40% and 50% of shortest paths between two committers within the core committer network pass trough committers associated with Google.com and Android.com email addresses, respectively. The third and fourth highest values for BC metric have committers associated with Gmail.com and Sonyericsson.com email addresses, with the metric values indicating that only some 2% and 1% of shortest paths traverse through these committers, respectively.

Figure 8 shows network centrality metrics values BC,CC, and EVC, for the external committers network. The highest values for the EVC metric have committers associate with gmail.com, google.com, debian.org, non dot.org, apple.com, etc. Unlike EVC values for the core committers' network, the EVC values for external committers' network is more balanced and does not indicate as high of a differences among the top 30 committers. The BC value is highest for the committers associated with the google.com address, followed by the commuters associated with the gmail.com and debian.org address.

Figure 9 shows network centrality metrics values BC,CC,EVC for combined core and external components. Unlike EVC values for the core external committers network, the EVC values for the combined core and external committers' network show highest values for committers affiliated with google.com, followed by committers associated with gmail.com, intel.com, debian.org, codeaurora.org, etc address. The combined core and external committers' network shows more balanced values for CC and EVC, while the BC values indicated that some 40% of the shortest paths traverse through committers associated with a google.com address, followed by gmail.com with some 20% of the shortest paths, and intel.com

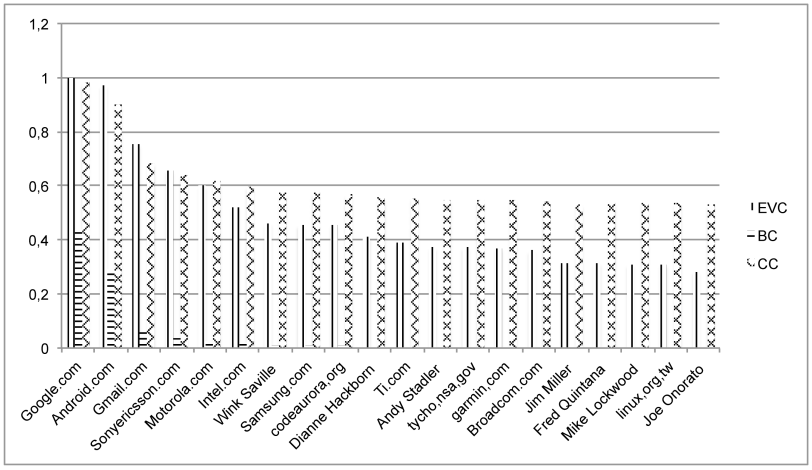


Figure 7: Network centrality measures for top 20 committers in the core committers' network

with some 2% of the shortest paths.

Hence the metrics presented above in summery show:

Android core committers network The high average clustering coefficient and low graph density indicate that committers in a direct neighborhood of a committer are well linked. The identified potential sub-communities for the core committer network should be disregarded since their existence is not statistically significant. The collaboration strength or the influence of the Google on the committers in its direct neighborhood is twice as high compared to the influence of all committers in Google's direct neighborhood on Google. The majority, i.e., 40% and 50% of shortest paths between two committers within the core committer network pass trough committers associated with Google.com and Android.com email addresses, respectively.

Android external committers network The number of sub-communities identified within the external committer network is 6 with the MC value of 0,356 indicating that existence of the 6 subnetworks is statistically significant. The EVC values for external committers' network is balanced among the top 30 committers. The BC value is highest for the committers associated with the google.com address, followed by the commuters associated with the gmail.com and debian.org address.

Android core and external network The number of identified sub communities for the combined, external and internal committers' networks is 7, with the MC value of 0,43 indicating that the existence of the sub-communities is

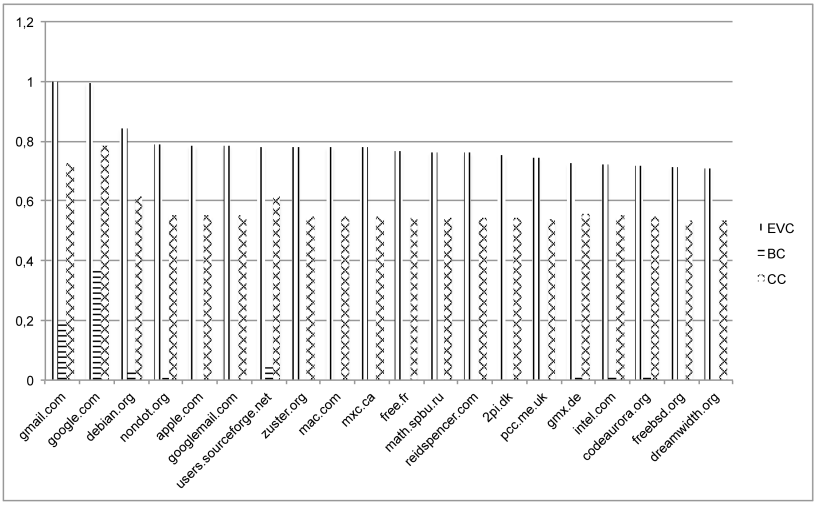


Figure 8: Network centrality measures for top 20 committers in external committers' network

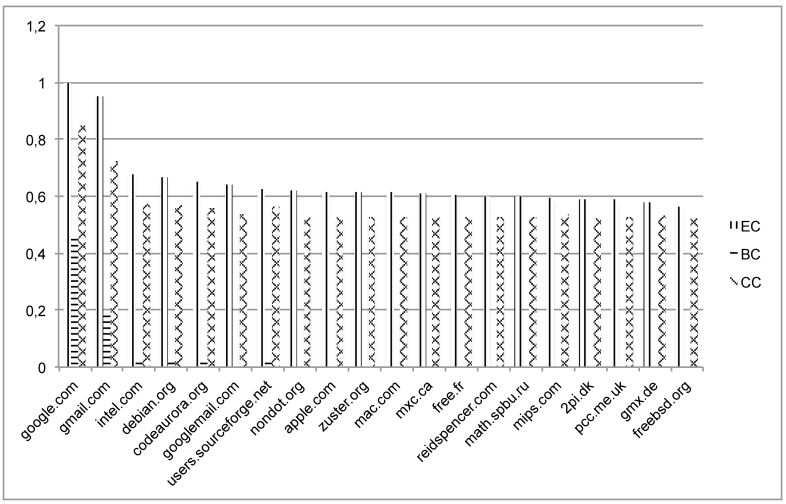


Figure 9: Network centrality measures for top 20 committers in combined core and external committers' network

statistically relevant. Committers affiliated with the Google email address have highest value of $WAOD$. Some 30% lower value of $WAOD$ has Apple, followed by committers associated with gmail.com, nondot.org, and android.com. Values for CC and EVC are balanced betweeing the top 20 committers, while the BC values indicated that some 40% of the shortest paths traverse through committers associated with a google.com address, followed by gmail.com with some 20% of the shortest paths, and intel.com with some 2% of the shortest paths.

Based on the results, the three committers networks show characteristics of highly centralized network structure, with committers affiliated with google.com, gmail.com, android.com, and apple.com being central to linking other committers. The four committers' affiliations also have the highest influence on other committers. Graph density metrics show that the core committers network and combined core and external committers network are not well connected with GD values of 0,058 and 0,055 respectively. The external committers network, composed of over 150 different OSPs has a twice as high value for the GD metric as for the Android OSPs core components network. As noted above, the highest value of GD metric is 1 indicating that all vertexes are connected.

The Figure 10, Figure 11, Figure 12 show graphical structures of core, external, combined core and external commiters' networks. The absence of subnetworks in the core committers network discussed above as well as low graph density can be seen in the Figure 10. The external network depicted in Figure 11 shows existence of subnetworks, which is expected for a source code base composed of different open source projects.

4.2 Research question 2: What type of concerns should a company take into consideration when planning to become a contributor to the Android or a similar type OSP

Based on the results presented for research question 1, Android OSP exhibits characteristics of a highly centralized OSP, where committers with affiliations to google.com, gmail.com, android.com, and apple.com have the highest level of influence. The external open source projects that Android OSP hosts in its source code repository under the external top subdirectory shows committers affiliated with Google.com as being third most influential. The committers affiliated with Google have the highest BC value for the external committer's network, indicating that 40% of committers in the external committer network have shortest paths to other committers transversing through committers affiliated with Google. This is a compelling evidence that Google has been the most central, and the most influential in the Android OSP development not only for the core source components, but also for the external open source projects. The Android committers network

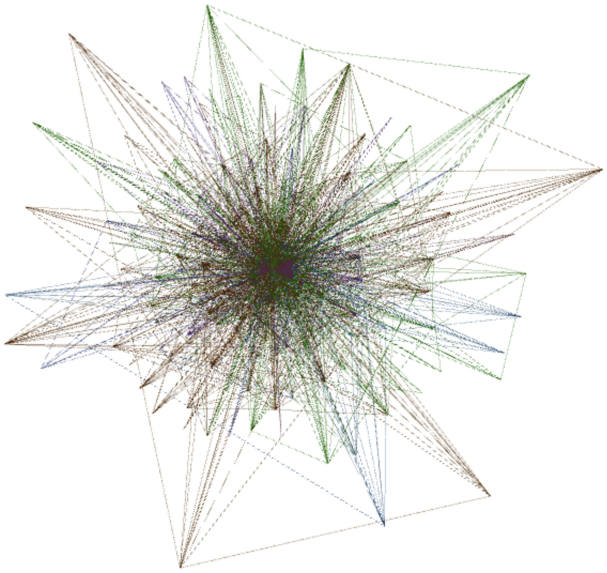


Figure 10: Core committers' network

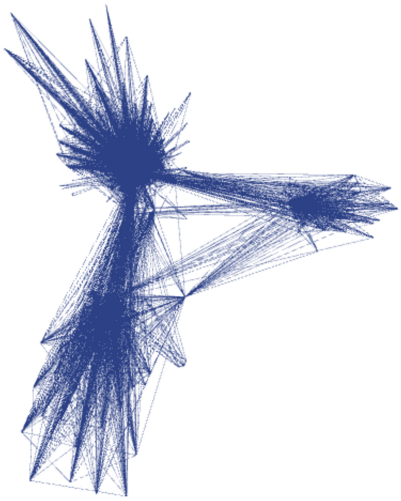


Figure 11: External committers' network

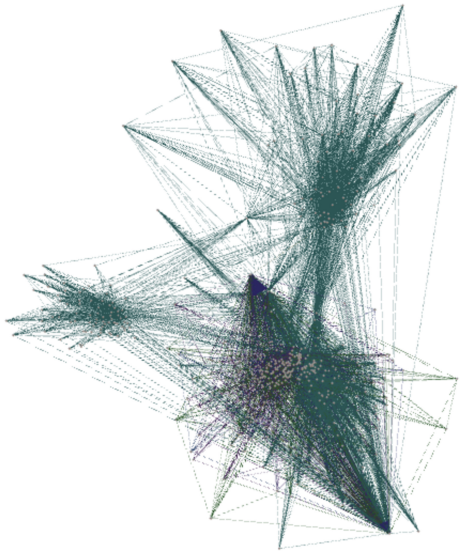


Figure 12: Combined core and external committers' network

has low graph density, i.e. low connectedness of committers, indicating low co-affiliation among committers.

From a perspective of a company that is planning to participate or participates in Android or a similar OSP this means that it should take into consideration that OSS product development tends to be highly influenced by one company. This might indicate that the company planning to incorporate the Android into its product will need to work closely with Google to ensure that the changes it needs to see implemented in the source code base are included in a future OSS product releases. Google has built different sales models around the Android, primarily the GooglePlay store, the application market for Android devices, AdMob platform, and Web search. Hence, it is in the Google's interest to have the Android used and distributed on as many mobile devices as possible since this would mean higher revenues from its GooglePlay store, AdMob, and Search engine. However, the company should be aware that sales and marketing models change, and different alliances form. In order to influence and lead a large open source project, a company controlling the project development usually has a large development effort dedicated to the project. In case a company is no longer able to support the development is is possible that some other company takes the lead. Hence, it is possible for a company with highest control over the open source project to take the project in a direction not favored by some other project participants.

5 Discussion

The Android OSS project is an open source stack of software, i.e., a mix of OSS from over 150 OSS projects and components which Google initially purchased from Android corporation and later open sourced. Hence, different OSS projects have been used as a reusable software components to build a mobile device operating system used by companies from entire mobile ecosystem. While different open source solution stacks are not in itself a novel idea, e.g. [LAM], the Android OSS project stack is unique for several reasons. Firstly, it combines over 150 OSS projects of highly diverse nature into one integrated OSS product. Secondly, the integrated OSS product is of large scale, mostly maintained and developed by one company. Finally, this product has become the most used OSS product for mobile devices in 2012.

Based on the social network structure analysis results for Android committers's networks, it is evident that Google has the highest degree of influence and centrality on both, the core, and the external components. This shows how a large company with significant resources can create a large scale software products using other OSS components. In a company sponsored open source project the company invests a large development effort into the OSS product and there exists a possibility that the company might not be able to maintain the high level of development commitment. This possibility would also mean uncertainty for the future of the OSS product development, and, if realized, it can bring shifts in committers' influence on the project. This can create uncertainty on the future of the OSS product development, an important factor that should be considered by companies planning to join similar company sponsored projects. A company might decide to also closed source and license the open source product. Such situation can then create a vendor lock-in effect, which contradicts a generally accepted notion of an OSS software product being free from vendor lock-in.

6 Conclusions

The conducted analysis have shown that Google has the major influence on the Android OSP. While it is favorable to use an OSS product as a commodity software, and thus decrease development costs by focusing available resources on developing differentiating parts of a product, at the same time this can raise many uncertainties. The future of OSS product whose development is highly sponsored and influenced by one company can come under the influence of market conditions the company finds itself in. This seems to go against the nature of OSS, which among other characteristics includes protection from vendor lock-in, i.e., high dependance of companies using Android on the Google.

More research is needed to understand and properly categorize OSPs in a way that would help the industry better understand own strategic position in a context

of using an OSP to build business model. The research approach proposed in this study can be used as one way of studying a committer's network structure of a software development community.

Acknowledgment

This work was funded by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se)

References

[And05] AndroidCorp. *Google Buys Android for Its Mobile Arsenal*. `http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal`. 2005.

[And07] AndroidOS. *Breaking: Google Announces Android and Open Handset Alliance*. `http://techcrunch.com/2007/11/05/breaking-google-announces-android-and-open-handset-alliance/`. 2007.

[Blo+08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. "Fast unfolding of communities in large network". In: *Journal of Statistical Mechanics: Theory and Experiment* 10 (2008), P100.

[BH11] Stephen P. Borgatti and Daniel S. Halgin. "On Network Theory". In: *Organization Science* 22.5 (2011), pp. 1168–1181.

[Bra01a] Ulrik Brandes. "A Faster Algorithm for Betweenness Centrality". In: *Journal of Mathematical Sociology* 25 (2001), pp. 163–177.

[Cle+05] R. B. de Souza Cleidson, Jon Froehlich, and Paul Dourish. "Seeking the source: software source code as a social and technical artifact". In: *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*. 2005, pp. 197–206.

[Han+M] Sudheendra Hangal, Diana MacLean, Monica S. Lam, and Jeffrey Heer. "All friends are not equal: Using weights in social graphs to improve search". In: *SNAKDD-2010: 4th SIGKDD Workshop on Social Network Mining and Analysis* (ACM, 2010).

[HOA11] Martin Höst and Alma Oručević-Alagić. "A systematic review of research on open source software in commercial software product development". In: *Information & Software Technology* 53.6 (2011), pp. 616–624.

[How+06] James Howison, Keisuke Inoue, and Kevin Crowston. "Social dynamics of free and open source team communications". In: *Open Source Systems, IFIP Working Group 2.13 Foundation on Open Source Software*. 2006, pp. 319–330.

[Inc13] Google Inc. *Android Open Source Software Project*. `http://www.android.com/`. 2013.

[LAM] Open Source Community LAMP. *Linux, Apache, MySql, Python Open Stack*. `http://onlamp.com/`.

[Lin+08a] Frank Van der Linden, Björn Lundell, and Gary J. Chastek Chastek. "Open Source Software Product Lines". In: *International Software Product Line Conference* (2008), p. 387.

[Lin+09b] Frank Van der Linden, Björn Lundell, and Pentti Marttiin. "Commodification of Industrial Software: A Case for Open Source". In: *IEEE Software* 26.4 (2009), pp. 77–83.

[LF+06] Luis López-Fernández, Gregorio Robles Robles, Jesús M. González-Barahona, and Israel Herraiz. "Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects". In: *International Journal of Information Technology and Web Engineering* 1.3 (2006), pp. 27–48.

[Org13] Gephi Organization. *Open Source Software for Exploring and Manipulating Networks*. `https://gephi.org`. 2013.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[RH08] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Sal95] Gerald R. Salancik. "WANTED: A good network theory of organization". In: *Administrative Science Quarterly* 40 (1995), pp. 345–349.

[Toi+07] Riitta Toivonen, Jussi M. Kumpula, Jari Sarmaki, Jukka Pekka Onella, Janos Kertesz, and Kimmo Kaski. "The role of edge weights in social networks: modeling structure and dynamics". In: *Proc. International Society for Optics and Photonics, SPIE* 6601 (2007).

[WS98a] Duncan J. Watts and Steven H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), pp. 440–442.

[Xu+05] Jin Xu, Yongqin Gao, Scott Christley, and Gregory R. Madey. "A Topological Analysis of the Open Souce Software Development Community". In: *HICSS*. 2005.

DEVELOPMENT PROCESS MONITORING THROUGH APPLICATION OF NETWORK ANALYSIS ON SOURCE CODE REPOSITORY DATA

Abstract

Context: The emergence of new development practices, e.g. agile development, has prompted many companies to implement extensive changes to the way software development effort is organized and managed. Understanding if and how the implemented changes affect the way the teams collaborate to produce software products is crucial in assessing the effects of the implemented changes. One way to analyze the teams' collaborations is by studying developers' collaboration networks.

Objective: The goal of the research is to understand if and how the developers' network metrics can be used to assess and monitor the effectiveness of the newly introduced changes.

Method: In this work, network analysis of data extracted from a source code repository of a large unit within Ericsson, a Swedish multinational company with focus on communication technology, is performed. The unit has undergone major organization and development process changes in the recent years. For this

Alma Oručević-Alagić, Martin Höst, Submitted

purpose the source code repository data was mined in order to calculate relevant network metrics. The resulting network metrics were validated through three focus group meetings with participants from the company.

Results: The presented network analysis based approach has shown to be effective and useful in the assessment of software development dynamics. In addition, based on the feedback received in the focus group meetings, the underlying metrics can be useful to correctly assess software development structures and monitor how they evolve over the time.

Conclusion: Developers' network metrics can be used to identify and track changes in specialized development subgroups and degree of collaboration. More studies are needed to gain a better understanding on the best usage of the metrics.

1 Introduction

Effectively monitoring large software development efforts is an ongoing and complex task, especially in large software organizations. A number of methodologies with associated benchmarks have been proposed and implemented with the goal of improving software product quality and development efficiency, e.g. as described by Bohnet and Döllner [BD11]. The post-mortem meeting groups have been conducted in order to assess pros and cons of the implemented methodologies and suggest improvements, e.g. Kupiainen et al.[Kup+14].

The emergence of large, complex and industry-grade open source software products has prompted an increased research effort in the study of the open source software (OSS) development communities. Furthermore, the transparency of open source communities has offered an open access to developers' communication archives, source code repositories, as well as insights into the communities' organizational structure and participants' roles according to Crowston and Howison [CH05]. Some of the conducted research has utilized network analysis methods based on the social network analysis approach e.g. Wasserman and Faust [WF94a] to study communication archives and source code repositories. The results of the research provide an increased understanding of the communities' dynamics, especially from the perspectives of the participants' roles, communication channels and developers' collaborations, e.g. Crowston and Howison [CH05]. In the past decade network analysis has gained popularity in the software field and has been increasingly used to study open source community data López-Fern and Robles [LF+06]. In contrast, software development efforts carried out in large companies have not been studied as extensively which is probably due to the very nature of the proprietary software development, i.e. closed source code archives and intellectual property rights.

As Basili et al. [Bas+02] noted in their experiences gained in over twenty-five years long research effort on process improvement in NASA, the data collection, analysis, application of obtained results, and reevaluation of the applied changes is

an ongoing effort. Thus, there are many ways to evaluate efficiency and effectiveness of software development. Hence, the conducted research described in this paper should be viewed as an effort to collect data and analyze it within the scope of one large software company setting that has introduced extensive changes to development process and organization structure. In order to further the understanding on how the proposed methodology can be used to assess and monitor software development efforts, more studies over a longer time periods are required. However, the research presented here provides concrete results of one such study conducted within the closed setting, discusses issues encountered during data collection as well as evaluation process carried out via three focus group meetings, and offers possible direction for the future research.

In this paper the source code repository of a large development unit within Ericsson is mined, source code committers' networks constructed, and relevant network metrics calculated and validated through three focus group meetings. The goal of the study is to understand if the effects of the organization and process level changes implemented in the company can be observed in the developers' collaboration network metrics, and if so, if the monitoring of the metrics could be useful in management of software development effort. It is important to note that the goal of this research is not to evaluate the effectiveness of the changes introduced in the company, but rather to investigate if the changes could be observed in network metrics based on data extracted from the source code repository. Furthermore, we argue that if the changes could be observed on the source code level, then the network metrics could be used to monitor development effort and changes in development collaboration dynamics. In particular, one can observe if the organization level changes leave 'foot-prints' in the source code, or if some other software quality or development efficiency metrics are related to the topology of developers networks and corresponding metrics. A variety of network metrics is measured and evaluated, e.g., developer's centrality, influence, or change in number of distinct subgroups of developers which collaborate more closely.

The outline of this paper is as follows. Section 2 presents the background information about the case study and related research studies. In Section 3 the research method is presented in more detail, while in Section 4 the results of the study are presented. Section 5 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 6 .

2 Background and related work

In this section related research on application of network theory to study source code repository data is discussed. A brief description of the case company in which this research was conducted is also provided.

2.1 Related Research

Many studies have been conducted in order to examine the relation between the team organization and the underlying software being produced. The famous Conway law [Con68] coined in 1968 states that design of an information system is a copy of communication structure of the organization that produced the system. Relating an organization structure to the work produced has remained a relevant and interesting subject examined in a number of studies. In this paper, the organization structure is related to collaboration networks which are a type of social networks where relationships between the actors are formed though different events, such as, co-authorship on the same paper, Newman [New13]. In the work by Ning and Kumar [NK13] team structure and architecture of open source projects have been examined arguing that they moderate each other and affect development performance. Research by Cleidson et al. [Cle+05] shows that the network structure of the source code is highly related to the way a community is organized. In [Lia+06] communication archives were extracted and analyzed and it was shown that actors exhibiting high level of network centrality are correlated with the ability to coordinate actions of others in the team.

In work by Roach and Menezes [RM10] network analysis was used to construct undirected projection of bipartite developers' collaboration network where edges between the developers or the nodes of the network were formed if two developers changed the same file. The edges weights were then calculated based on how many different files developers changed together. In the fore mentioned research, authors argued that usage of network metrics to understand the most influential developers was superior to static network metrics such as source lines of code (SLOC), number of commits, or number of issues found and demonstrated this by analyzing Python OSS project.

The field of social network analysis is based on network theory e.g. Salancik [Sal95] that provides tools for analyzing relationships between network nodes. The study by López-Fernández et al. [LF+06] proposes a methodology based on social network analysis field that can be used to study developer collaboration. The methodology constructs weighted and undirected developers' collaboration networks, while in this study directed and weighted networks are constructed. The weighted and directed networks facilitate calculation of weighted out degree metric, which can be used to identify the most influential contributors. This research builds on the study by Oručević-Alagić and Höst [OAH14b] which proposes a methodology for constructing weighted and directed developers' networks based on source code commits data, arguing that networks built in this manner offer a more accurate developer network structures.

2.2 Case Company Setting

The company is a relevant case to study, not only due to its large software base, global presence, and distributed development sites but also because it has intro-

duced some major changes to the way work is organized, moving from a hierarchical to a vertical organization as well as introducing Scrum development framework, e.g. Schwaber and Beedle [SB01]. Thus, the development teams were reorganized so that each team is composed of members with expertise in all different software layers, rather than one specialized software layer. The decision to reorganize teams was motivated by the need for developers to broaden their system expertise, thus shifting from a specific architecture layer to a system wide view. By organizing teams in this way, the case company hopes to achieve greater knowledge dissemination, which in turn should lead to higher software product quality and increased development efficiency. The company was motivated to participate in the study since it could offer an increased understanding on how the implemented changes are reflected in developers' collaboration dynamics as observed in corresponding collaboration networks derived from the source code files changed.

3 Research approach

The study is conducted as a case study based on the case study guidelines by Runeson and Höst [RH09b]. The study is exploratory with the overall objective to understand if the organization and the process level changes can be observed in the developers' collaboration network metrics, and if so, if monitoring the changes in metrics would be useful for the company to track and assess the effects and effectiveness of the applied changes.

3.1 Research questions

The following research questions were investigated in this study:

1. How do committers' collaboration network metrics reflect development process changes within the case company?

2. How can the metrics be used in order to aid planning, assessment and monitoring of the development process?

For research question 1, the calculated network metrics are analyzed from the context of the company's organizational changes. For this purpose, discussion feedback and input obtained from the focus group meetings is analyzed. For research question 2, the results from research questions 1 were synthesized and discussed in order to understand if the proposed committers network data can be used to assess and monitor software development organization and process changes.

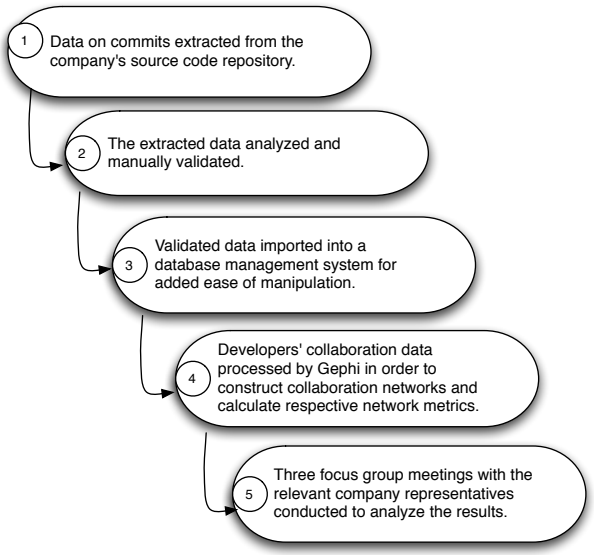


Figure 1: Research Process

3.2 Research Steps

The data used in the study was collected and analyzed according to the process presented in Figure 1.

The first step of the research was to obtain access to the Git source code repository of the case company, and run a custom built software to extract repository information such as developer unique ID, date and time of the source code commit, and the name and the location of the source code file the commit was performed on. The first author was seated at the company with a team dedicated to the integration and building of the software product. Thus, the integration team had a complete picture of all development efforts carried out throughout the company as well as know-how related to building the software product.

In step two, an experienced member of the integration team was assigned to help the first author validate the extracted data. Initially, developers having multiple IDs on the system were identified and consolidated under one unique id. When building a new software release, integrators tend to make changes to sets of files, very often the file headers, updating them with build or release related information. Thus, source code commits by the integrators were excluded from the extracted data sets as the changes they make are not considered relevant in the study and could possibly corrupt the results. Including data from the integrators' commits could skew the relevant study data, i.e. data related to actual developer level

commits. The resource from the integration team was also able to relate different teams to the source code subdirectories that the teams normally work on.

In the third step, the validated source code committers data was loaded into a database for ease of manipulation, i.e. transformation into a format that can be analyzed using the Gephi [Org13] software for network analysis. According to the methodology from Oručević-Alagić and Höst [OAH14b], Figure 2 shows an example, for illustration purpose, of the data transformation using three developers A, B, and C, each responsible for 3, 4, and 5 changes (source code commits) made to the same source file, respectively. The developers A, B, and C are nodes or vertices of the presented network, while the directed and weighted links between the nodes are referred to as the edges representing developers weights from one to another with respect to all changes made to the file. Thus, for each source file on the system all developers changing the file were identified as well as the number of changes each of the developers made to the file counted. Then, based on the number of changes per developer divided by the total number of changes made on the file, their relative edge weights are calculated. While this is a simple and trivial example, in a context of a large network, with many committers, where, e.g, a subgroup of committers performs a large number of changes, computing edge weights relative to the number of all changes performed on a file is important in order to accurately assess the relationship strength. This is more so as the data on committers, corresponding edges, and their weights are building elements of a network structure, based on which other network metrics are derived.

Using a more general notation to represent the procedure we identify a set of developers $V = \{v_1, v_2, ..., v_k\}$ and set of changes made on a file $U = \{u_1, u_2, ..., u_m\}$ and define a weight W of an edge between an actor v_i and all other actors that participate in changing the file u_t as:

$W(v_i, u_t) = \frac{X(v_i, u_t)}{\sum_{c=1}^{k} X(v_c, u_t)}$

where $X(v_i, u_t)$ denotes the number of times a developer v_i made changes to the file u_t.

Based on this, the weight of edge $W(v_i, v_j)$ for all changes v_i and v_j performed together equals:

$W(v_i, v_j) = \sum_{t=1}^{m} W(v_i, v_j, u_t)$

In step four, all the developers' pairs that changed the same file, their corresponding weights and the file name with its corresponding folder location were loaded into the Gephi. Hence, Gephi was used to create graphs as well as to calculate relevant network metrics. Figure 3 shows a network graph on developers' collaborations for the entire source code repository for the case company. The nodes of the graph represent developers, while the edges relate together two developers who made changes to the same file. The edges are assigned the different unique color label representing the subfolder the source file is located in. Figure 4 shows a sample folder structure under which source code files are stored. Hence, Figrue 3 shows developers' collaboration graphs at level 1 subfolder level. In the Figure 5, the developers' collaboration network graph is shown at level 2 for a sub-

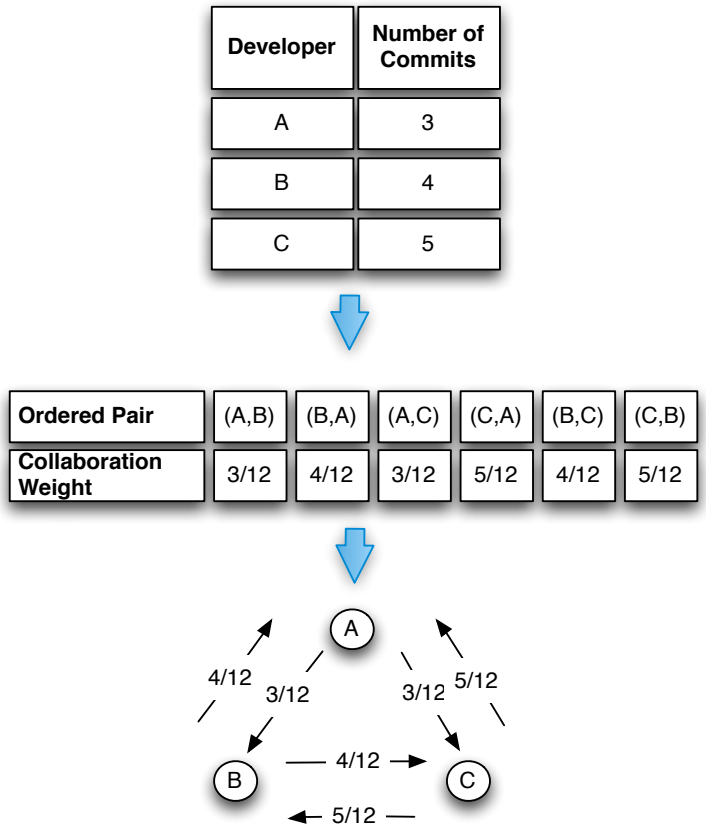


Figure 2: Transformation of the Extracted Data to the Weighted and Directed Pairs, Three Developers Changing the Same File Example

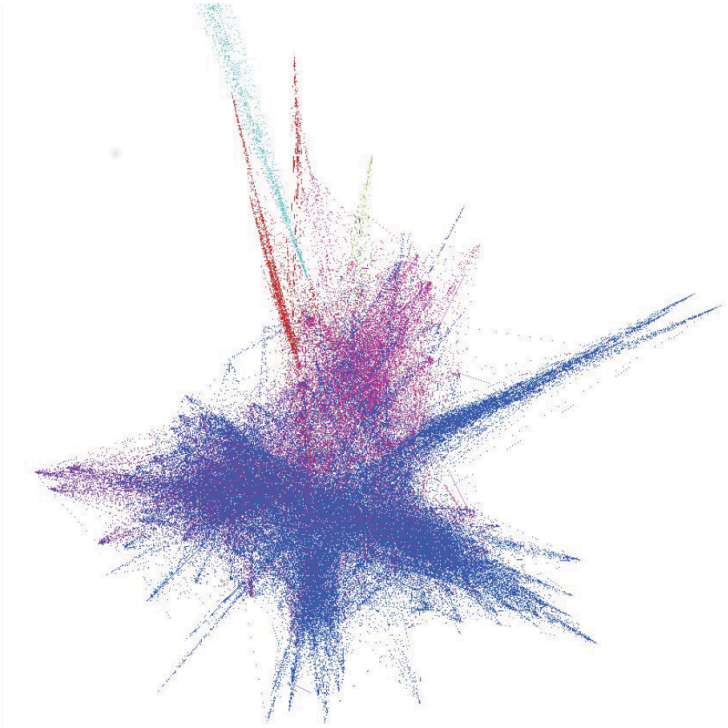


Figure 3: Developers' Collaboration Network at Level of the Entire Source Code Repository

folder A2, while Figure 6 demonstrates developers' collaboration network at level 3 for a subfolder A3. Each first level subfolder normally includes source files associated with a specific software function, e.g., all work related to modems would be located under modem subfolder. Through the Gephi interface one can than visually inspect developers' collaboration networks related to a particular subfolder, as well as obtain network metrics for the inspected subfolder. The goal of the presented graphs on developers' collaboration is to have a visual which can provide a high level insight into the structure of developers' collaboration, and relate different parts of the system though use of colored labels for the edges. While the graphs provide an interesting representation of developers' collaborations for a given folder which can also be further analyzed by zooming in on a subfolder one wishes to explore, in order to gain a better understanding on the network structure and dynamics, an in-depth examination of associated network metrics is needed.

The following network metrics explained in more detail in Wasserman and Faust [WF94a] were calculated and discussed in the focus group meetings:

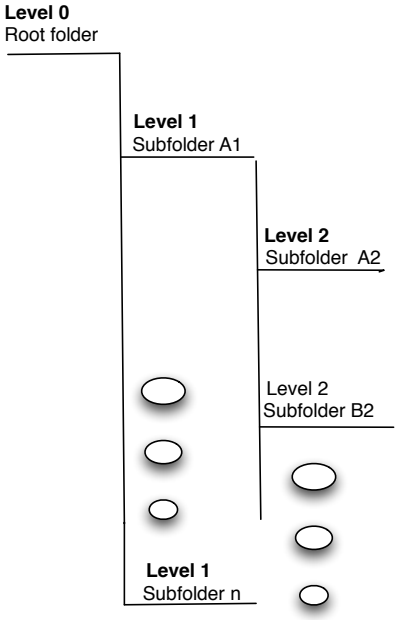


Figure 4: An Example of Folders/Subfolders Layout

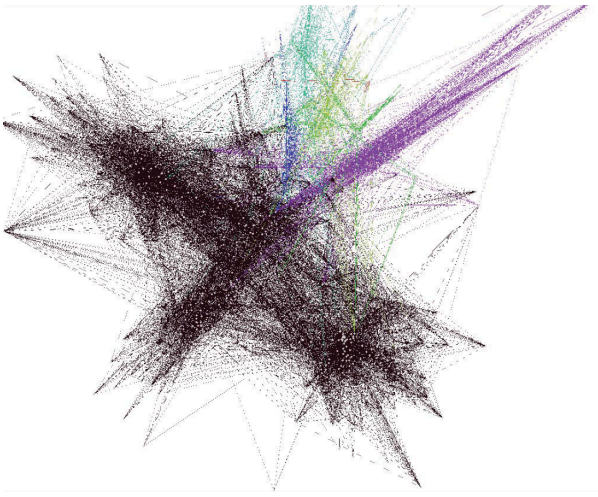


Figure 5: Developers' Collaboration Network at Level of the First Level of Subfolder

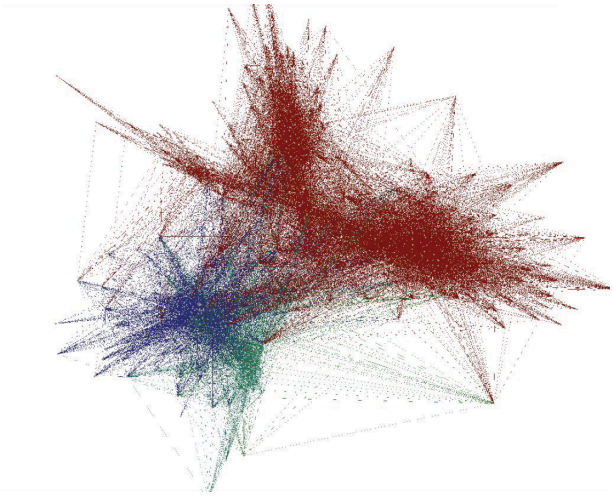


Figure 6: Developers' Collaboration Network at Level of a Second Level Subfolder

1. Weighted average degree (WAD). The WAD represents the average of the sum of weights of the edges of nodes. In case of developers' collaboration networks the nodes represents developers and a high WAD metric points to a developer with high level of collaboration with other developers.

2. Graph density (GD). The GD metrics shows how well the developers are connected and it is calculated as ratio of existing edges (links between developers) to all possible edges. Hence, GD of 1 means that edges exist between all vertices.

3. Average path length (APL). The APL metric is calculated as average of all shortest paths between nodes on the network. Thus, the APL of 1 means that every two developers within the network have changed all the files at least once.

4. Network diameter (ND). The ND represents the longest shortest path between two nodes.

5. Network modularity (NM). The NM score uncovers existence of sub-networks of developers that tend to work more closely on a set of files. A high network modularity score normally points to communities that are organized as opposed to random or unorganized.

6. Connected components (CC). CC indicates the number of subnetworks-components that exist within the network and which are not connected.

7. Clustering coefficient (CCF). CCF metric shows how well nodes in the neighborhood of some node are connected. Average CFF is calculated over all nodes of the network. Hence, e.g., if all developers connected to a developer are connected as well with the rest of the developers, and the APL is small, this can indicate a 'small-world effect' or rather cohesive development environment.

The metrics above are calculated for the directed network with weighted edges, which influences the calculation of the WAD metric. While there have been algorithms proposed for the calculation of metrics such as GD based on weighted edges, the metric has not been implemented as of yet in Gephi.

The network metrics explained above were collected for a year long period, and for the analysis purposes calculated for the four three-months long periods.

Finally, in step five, the obtained results were presented to three different groups composed of five to six actors, whose roles within the company were either senior technical roles such as software architects and code guardians, or middle management. The focus group meetings were divided equally into three parts over the two hour time period. At the beginning of the focus group meetings, the first fifteen minutes were used by the authors of the paper to present the goals and the methodology of the study. The main objective of the focus group meetings was to :

1. Identify organization and development process changes that have been implemented in the company in order to gain a better understanding of the changes introduced in the development organization of the case company.
2. Understand if and how the results of the study can be related to the above identified changes.
3. Understand if and to what extent the calculated network metrics could be used to assess and monitor organization and developer level changes.

In order to achieve the objective of the focus group meeting the following questions were discussed:

1. Name a couple of major changes you have seen in the period of the last year with respect to development teamÕs organization and collaboration.
2. To what degree the results of the study match participantÕs experience?
3. How can the presented metrics be used in planning and monitoring of the changes related to software development organization and collaboration?

Each of the above listed questions were discussed in a format where researchers posed the questions, participants would take five minutes to reflect on the questions and write down their answers on post-it notes. Then, an open discussion session

lasting approximately twenty minutes was held, where the focus group participants shared and discussed their answers. Both authors participated in the meeting, with first author serving the role of presenter and moderator, while the second author took notes on the discussion. At the end of the focus group meeting, the participants' post-it notes were collected and categorized according to the discussed subject. The categorized notes were analyzed for the purpose of this study along with the notes the second researcher took on the discussion.

3.3 Validity

In this section the validity of the research is analyzed with respect to the types of validity threats presented [RH09b].

Construct validity: The construct validity is related to the relationship between the concepts and theories behind the case study and what is measured and affected. The information on source code commits was pulled and analyzed using standard and tested software libraries. The subset of metrics from the network analysis used in this research has been accepted and validated in other studies within the domain of open source project repositories and mailing archive studies as discussed in the Section 2. This means that the risk of using metrics that do not represent the concept of social network structure is lowered. During the focus group meetings, the metrics used were explained to the participants, and through discussion sessions, the participants demonstrated understanding of the metrics.

Conclusion validity: Conclusion validity is concerned with drawing correct conclusions from the study results. The analyzed networks and related metrics were collected for a year-long period, and divided into four three months long periods. This way the focus group participants were able to relate each set of metrics for a given period to projects that were completed during the period. Analyzing obtained data through discrete periods and noting the changes in them can provide a better analysis results. However, there is no assurance that a one year period is long enough to account for possible result variabilities, and thus be sufficient representative to test the hypothesis.

Internal validity: The internal validity is concerned with factors that may affect the observations without the researcher's knowledge. The data extracted from the repositories was examined and validated manually through sampling. A highly experienced internal company resource was consulted during data extraction and validation process. The expert was able to identify source code committers that should be excluded from the data sets, such as software integrators that make changes to a large number of files normally updating file headers with build information. In addition, the extracted data was manually validated to ensure that developer identities are consistent.

External validity: The external validity is related to the ability to generalize the results of the study. The studied software and organization structure is a relevant example of a global telecommunication software based company with a large

and distributed development effort. Hence, given the company and development profile, the likelihood of the results being specific and unique to this company is lowered.

4 Results

Table 1 shows calculated values of the network metrics.

Metric	Period 1	Period 2	Period 3	All
WAD	12.5	12.55	9.385	27.691
GD	0.023	0.025	0.02	0.032
APL	2.915	2.597	3.018	2.609
ND	11	10	10	10
NM	0.772	0.732	0.783	0.69
CC	2	8	2	1
CCF	0.366	0.346	0.322	0.363

Table 1: Network Metrics Results

The decrease in average developers' strength or the WAD metric indicates that developer collaboration on average for the three consecutive periods has been decreasing. For the entire year it was at 27.6, over two times higher than for each of the three month periods. The WAD metric can be affected by the nature of software projects being worked on, i.e., for different projects different parts of the source code base need to be modified. However, when WAD of developers' is examined during the year-long period, thus taking into account the collaborations' over multiple projects, this metric seems to increase since multiple projects are more likely to require modification of greater number of source files and, thus, increase developers' collaboration or the WAD metric. Thus, to properly investigate the WAD metric it is important to understand the nature of the implemented software projects.

The graph density metrics, GD, with values lower than 0.033 indicate that the developers' collaboration network is scarcely connected. As noted earlier, the GD metric is computed as ratio of all existing edges to all possible edges. If the GD was 1, this would mean that each developer has changed all the source files at least once which is, of course, an unrealistic expectation for a global software intensive company with large source code base. However, the GD does seems to fluctuate less than the WAD metric when comparing the four-month and yearly values. This can be explained by the fact that unlike the WAD metric, the GD metric is not affected by the number of changes developers have performed together, as even one common file change would create an edge between the two developers. For the WAD metric the weights of the developers with respect to the file changed are based on the number of changes performed by the developers on the file. However,

the invariability of the GD metric during the year can indicate that implemented changes did not result in increased knowledge dissemination among the developers. Thus, one would expect the GD to increase as developers start working on code and hence changing other source files than the ones they are specialized in.

The average shortest path length is 2.69 for the entire network indicating that on average each developer is at least 3 degrees away from the other developers. Again, we should note that this metric is not fluctuating significantly between the individual periods and the cumulative year-long period. Like the GD metric, the APL metric is not affected by the number of changes developers have performed on a file, i.e. even a single common change on the file would create an edge between the two developers. As in the case of the GD metric, one can argue that formation of some new links between developers should be seen if the developers start working on the source code they have not worked on before the implemented organization changes.

The network diameter, ND, or the longest shortest network path for the entire system fluctuates between 10 and 11, indicating that there exist developers that work on quite unconnected parts of the system. This metric has insignificantly decreased for periods 2 and 3.

High network modularity, NM, indicates that development teams are highly organized into subgroups. While the modularity for the whole period is lower than the one for the each period, the difference is not significant.

Wile CC, connected components for the individual periods have values 2, 8, and 2, the network for the entire period seems to be connected, having CC value of 1. This means that during the individual periods, there existed networks of closely related developers that were disconnected from each other.

The clustering coefficient CCF can take values in the range from 0 to 1, thus clustering coefficient of 0.3 is on the lower end indicating that for the entire network on average developers in the neighborhood of a developer are not well connected.

The results of the focus group meeting are presented in the Figure 7. The focus group participants have identified the major changes in the development organization as restructuring of the teams so the members include developers with expertise in all software layers as opposed to one software layer, usage of a new software management system, adoption of SCRUM methodology, and introduction of distributed code ownership. It can be expected that such changes would make significant impact on network metrics, such as an increase in the weighted average degree of developers, greater graph density, i.e. developer connectedness, and a decrease in average path length. Greater collaboration would also imply an increase in the network density as well as create shifts in network modularity and clustering coefficients.

When presented with the impact of the changes on the collaboration of developers per calculated network metrics, the focus group participants agreed that they reflect what they have observed in their day to day work. Hence, while the team

organization and development process have went through significant changes, the effect of the changes on developers' collaboration is insignificant. The main reason for this was identified as time pressure to deliver, so the developers do not have time to learn other parts of the system or work in the cross-functionally structured team in manner that the knowledge would be disseminated over the time between the team members. Instead, the individuals with expertise in particular functionality were involved in the projects irrespective of the new team organization. The participants have also commented on the change in CC metric from 2 in the first period to 8 in the second, suggesting that such clustering coefficient was due to the nature and heavy load of the projects being worked on during the second period.

Finally, all of the focus group participants agreed that being able to monitor the network metrics presented in this paper would help them gain a better picture on the effect of the introduced changes as well as gain an insight into the dynamics of developers' collaboration. The participants were especially interested in the network metrics as a tool that would provide 'hard data' as opposed to subjective opinions, stating that the 'hard data' is less prone to open discussions. The importance of tracking the metrics for a longer time period was expressed as this would help in improved analysis of the data, such as establishment of benchmarks, and being able to identify trends that should prompt an action. Due to the scope and time of the focus group meetings only the presented high level metrics for the entire source code base were analyzed and discussed. The participants have expressed that the same analysis applied on finer grade, e.g. developers collaboration on a team level, for a particular project, or across geographically different development sites, would be helpful in understanding how to best use the metrics in order to improve management of the development effort.

5 Discussion

labelsection:discussion The presented developers' collaboration network metrics in the results section show that the changes implemented in the global software centric company have insignificant effect on the actual developers' collaboration. The findings from the focus group meetings are in line with the results of the analysis of the developers' collaboration metrics. As is often the case in a commercial development organization, the need to deliver products to the market tends to create a situation where the work is assigned in a manner that the task is completed fastest, thus not leaving space for the implemented changes to truly take the effect. Thus, while a work on a project under the new organization was supposed to be carried out by team members with less experience so they could learn from the new members on the team with required experience, in practice the work was allocated to the person with previous experience so it could be completed fastest. The low CCF value combined with the shortest average path length of 2.6, graph density of 0.03 and NM 0.69, indicates that the development teams at the case

What major changes were introduced to software development organization?	- Switch from GIT to Clear Case - Adoption of Scrum methodology - Teams restructured to cover a wider range of expertise - Distributed code ownership
How do the results of the study match experience?	- The results of the study reflect experience observed in the development organization. - The organization and process changes did not increase cross-collaboration - Due to time pressure to deliver, developers do not get an opportunity to work on different parts of the system. - Period 2 identified 8 distinct network components which is in line with projects being worked on during the period.
How can the network metrics be used to monitor development process changes?	- Calculated metrics are 'hard data' as opposed to, many times, subjective qualitative assessment of developers' collaboration. - Monitor metrics over sufficiently long time periods to establish benchmarks. and possibly discover action prompting trends. - Analyze metrics for different types of granularities, e.g, of geographically distributed development sites and compare results and trends.

Figure 7: Focus Group Meeting Results

company are highly specialized and that no significant difference in developers' collaboration can be observed during the one year period. Discussion on the values of the individual metrics is quite difficult, given that there are no benchmarks to compare them against. However, as focus group participants have confirmed, monitoring metrics over a prolonged period of time can help the organization to establish benchmarks, get an insight into changes in development dynamics on different levels such as for a particular team or development site. The focus group participants indicated that it would be relevant to derive and analyze the network metrics on individual developer level as opposed for all the developers, as this would provide an insight into shifts in developers influence and centrality by observing the corresponding WAD metric.

6 Conclusions

The results of the conducted study show that the presented methodology and related metrics can be useful in understanding dynamics of developers' collaboration especially with respect to the organization level and process level changes. Thus, one can use the metrics to identify and track changes in specialized development subgroups and degree of collaboration. Based on the presented metrics, focus group participants related project work to collaboration subgroups, as well as network density. Furthermore, the focus group participants expressed that the prolonged monitoring and analysis of the metrics would provide benchmarks that can improve the usage of the metrics. More studies over a prolonged period of time are needed to understand the full potential of using the proposed methodology and the metrics to manage software development effort. One way to better understand the results of the research is to create benchmarks either by prolonged monitoring of the changes in the network metrics, or by applying the same methodology on open source software projects of similar type and size. Relating the results of the study to some other software quality measurements, like number of bugs, time to delivery, etc can provide an insight on the network topology characteristics that are more correlated to production of higher quality software.

References

[Bas+02] V. R. Basili, F. E. McGarry, R. Pajerski, and M. V. Zelkowitz. "Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory". In: *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*. 2002, pp. 69–79.

[BD11] Johannes Bohnet and Jürgen Döllner. “Monitoring Code Quality and Development Activity by Software Maps”. In: *Proceedings of the 2Nd Workshop on Managing Technical Debt*. MTD ’11. Honolulu, USA: ACM, 2011, pp. 9–16.

[Cle+05] R. B. de Souza Cleidson, Jon Froehlich, and Paul Dourish. “Seeking the source: software source code as a social and technical artifact”. In: *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*. 2005, pp. 197–206.

[Con68] Melvin E. Conway. “How Do Committees Invent?” In: F. D. Thompson Publications, Inc., 1968.

[CH05] Kevin Crowston and James Howison. “The social structure of free and open source software development”. In: *First Monday* 10.2 (2005).

[Kup+14] Eetu Kupiainen, Mika Mäntylä, and Juha Itkonen. “Why are industrial agile teams using metrics and how do they use them?” In: *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics, WETSoM 2014, Hyderabad, India, June 3, 2014*. 2014, pp. 23–29.

[Lia+06] Hossain Liaquat, Andrè Wu, and Kon Shing Kenneth Chung. “Actor centrality correlates to project based coordination”. In: *Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work, CSCW 2006, Banff, Alberta, Canada, November 4-8, 2006*. 2006, pp. 363–372.

[LF+06] Luis López-Fernández, Gregorio Robles Robles, Jesús M. González-Barahona, and Israel Herraiz. “Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects”. In: *International Journal of Information Technology and Web Engineering* 1.3 (2006), pp. 27–48.

[NK13] Ning Nan and Sanjeev Kumar. “Joint Effect of Team Structure and Software Architecture in Open Source Software Development”. In: *IEEE Trans. Engineering Management* 60.3 (2013), pp. 592–603.

[New13] Mark Newman. *Networks*. Oxford University Press, 2013.

[Org13] Gephi Organization. *Open Source Software for Exploring and Manipulating Networks*. `https://gephi.org`. 2013.

[OAH14b] Alma Oručević-Alagić and Martin Höst. “Network Analysis of a Large Scale Open Source Project”. In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, August 27-29, 2014*. 2014, pp. 25–29.

[RM10] Christopher Roach and Ronaldo Menezes. "Using Networks to Understand the Dynamics of Software Development". In: *Complex Networks - Second International Workshop, CompleNet 2010, Rio de Janeiro, Brazil, October 13-15, 2010, Revised Selected Papers*. 2010, pp. 119–129.

[RH09b] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

[Sal95] Gerald R. Salancik. "WANTED: A good network theory of organization". In: *Administrative Science Quarterly* 40 (1995), pp. 345–349.

[SB01] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[WF94a] Stanley Wasserman and Katherine Faust. *Social Network Analysis. Methods and Applications*. Cambridge University Press, 1994.

BENCHMARKING APACHE SOFTWARE FOUNDATION PROJECTS: NETWORK ANALYSIS OF THE CONTRIBUTORS' COLLABORATION NETWORKS

Abstract

Software development is a collaborative effort that can be studied from perspective of changes made by developers to the source code repository. Network analysis has been applied in prior research to construct and study developers' networks. But to the best of our knowledge, similar analysis has not been performed on a larger scale with goal of understanding if there exist patterns in developers' collaboration network metrics in a set of unrelated, mature, and wide industry used software products.

Hence, the aim of this study is to understand if such patterns can be observed

Alma Oručević-Alagić, Nicklas Johansson, Christian Tenggren, Martin Höst,
Submitted.

across 258 software project hosted under the Apache Software Foundation. We argue that natural occurrence of the patterns in the network metrics can be indicative of healthy, preferable network topology for developers' collaboration networks, the one that possibly maximizes development efficiency.

For the purpose of this study, 258 source code repositories of projects hosted under Apache Software Foundation have been mined, storing all the historical data on commits into a database for the easiness of network and statistical analysis. We show that several interesting trends can be found especially relating the size of the project with developers betweens and closeness centralities, indicating that high specialization of majority of developers on a particular, smaller part, of the system with very few developers or experts that bind the larger parts of the system together might be the most efficient way to develop software in the open source software context. As with any preliminary findings, more work is needed to explore the patterns and understand their potential applicability within the closed source context.

1 Introduction

Software development is a collaborative effort, ultimately projected onto the final software product, i.e. the source code, where changes made to the source code base can be viewed as footprints left by the developers. The footprints can be studied to understand which distinct developers have crossed paths on how many occasions. If several projects are studied it is possible to study if there exist features common across different development efforts.

Until the recent couple of decades, studying source code repositories was not feasible due to the prevalence of closed-source projects, as described by e.g. Lerner [LT00]. The emergence and wide-spread usage of open source software provide access to mature, industry-grade open source projects. The public availability of source code repositories enables us to extract and study how developers' networks in distributed, online environment form and evolve.

As the production of software is a complex endeavor with a large number of software projects exceeding their initial budget, understanding if there exist patterns, or common features in developers' collaboration networks across mature, industry grade software products can potentially offer some guidelines for more efficient organization of the development effort. Bloch et al. [Blo+12] have shown that budget overruns of software projects are partially due to ineffectiveness, caused by inappropriate organization of development effort. Hence, companies are actively seeking and applying new practices that have potential to improve the developers' collaboration.

One way to formally study developers' collaboration networks is through the application of network analysis as presented, e.g., in work by Wassermann [WF94b]. Network analysis is a way of measuring specific relationships between different

entities in a graph. Each node or vertex in a network represents an entity, such as a developer, and the links or edges between vertices represent some kind of relationship. Edges can be either directed or undirected. Undirected edges indicate a mutual relationship between the vertices, while directed edges can be used to represent either a one-sided relationship or a mutual relationship with different weights. A weight attached to an edge can demonstrate how strong of a relationship it represents or how much information that flows through the edge as shown in Figure 1.

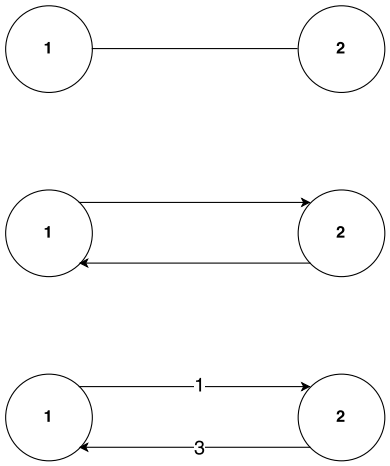


Figure 1: Example of undirected, directed and weighted directed graphs

There is a wide range of metrics that one can apply when performing a network analysis. Each metric looks at a specific property of the network. This study focuses on centrality indices and the clustering coefficient, described in detail in Section 3.3. The clustering coefficient can give an indication if there exist subgroups in the committer networks and the centrality metrics can show the influence the developers have over each other. The analysis performed is based on data extracted from 258 mature OSS projects hosted under the Apache Software Foundation (ASF) at the time this study was conducted.

This paper is organized as follows. Section 2 presents related work and Section 3 presents the design of the study, including the research questions. Section 4 presents the results and analysis of the data, while the section 5 discusses the results of the study in more detail. Finally, Section 6 presents the conclusions and future work.

2 Background and related work

Applying network analysis to study interactions on software projects is not a novel idea, as it has been used in earlier research, e.g. to study the collaboration between authors of scientific papers as reported by Newman [New01] and on neural networks in study by Latora [LM03]. Lopez-Fernandez et al. [LF+08] analyzed several open source projects, among which 'Apache' was one project, treating it as a one project and not as a collection of individual projects. This is due to the fact that the size and content of the Apache Software Foundation (ASF) has changed considerably since 2006. The research showed that even very large projects exhibit small-world effects, i.e. the ability to reach most other vertices from every vertex in the network with only a small number of hops.

Crowston [Cro03] analyzed the social structure of several open source software (OSS) projects using social network analysis. The analysis was based on projects hosted on SourceForge, which is a free web-based system that provides tools for OSS development and distribution. The focus of the study was on the developer interactions within the bug reporting process where the edges were defined as interactions between different developers. Madey et al. [Mad+02] performed a similar study on SourceForge projects with the focus on developer collaboration in and between different projects [Mad+02]. The research shows that there are individuals that serve as links between many projects.

A study by Oručević-Alagić and Höst [OAH14c] examined the committers' network of the Android open source project [OAH14c]. The aim for that study was to understand how one can utilize network analysis to study development communities where a majority of the community contributors are affiliated with commercial organizations. The results showed that the approach proposed in the study can be used to accurately study developers' collaboration in software development communities.

Bird et al. [Bir+08] studied the reply-to relationship on mailing lists and developers working on the same files. The study was carried out on five mature OSS projects using social network analysis. The results of the study show existence of sub-groups, i.e. that the networks were modular, as well as that the developers who performed changes to the same files were also more likely to interact with each other on the mailing lists.

Godfrey and Tu [GT00b] studied the growth of the large scale OSS projects over time. The results showed that the project kept a linear growth pattern even after reaching a huge size, several millions lines of code. This contradicts the commonly accepted belief that with an increase in size the growth declines. Mockus et al. [Moc+00] studied the Apache Web Server and showed that the developers' networks in Apache projects may have interesting properties, e.g. that the 15 most active developers stood for more than 85% of the lines of codes produced in the Apache web server.

In 1999 an initiative to create the ASF was taken when an already established group of developers decided that a more legal structure for their software development efforts was needed [Sev12]. Several of the projects hosted under the ASF have become leaders in their domains, e.g. the Apache Web Server [Fou15a], Apache OpenOffice [Ope], and the Hadoop distributed computing engine [Fou15b]. As the ASF over the time has taken more projects under its umbrella, the projects hosted under the ASF can now be divided into two groups; those originally started under the ASF and those that were added later in the the projects' life cycles.

This study aims to expand on earlier studies on open-source projects, by not focusing on one or few projects, but rather exploring a plethora of over 250 mature and industry used OSS projects hosted under the ASF. The goal is to understand if these distinct software projects share common features in terms of developers' collaboration network metrics for the purpose of establishing benchmarks that could potentially be used as guidelines for developers' collaboration networks on any software project.

3 Research approach

The research done in this study can be classified as case study as per Runeson and Höst [RH08]. The study is exploratory with the overall objective to understand if there exist patterns in network metrics of the developers' collaborations across OSS projects hosted under the ASF foundation. The results could be potentially used as benchmarks for assessing other software development projects.

3.1 Research Questions

The research conducted in this paper aims to answer the following research questions:

- Which commonalities can be observed in collaboration metrics for the projects hosted under the Apache Software Foundation?
- What (if any) benchmarks can be proposed based on the analyzed network metrics?

For research question 1, the calculated network metrics are analyzed for the presence of commonalities and any patterns are noted. Research question 2 investigates whether the results of research question 1 can be used as benchmarks, which software development projects in general can be compared to.

3.2 Research Steps

The data used in the study was collected and analyzed according to the process presented in Figure 2.

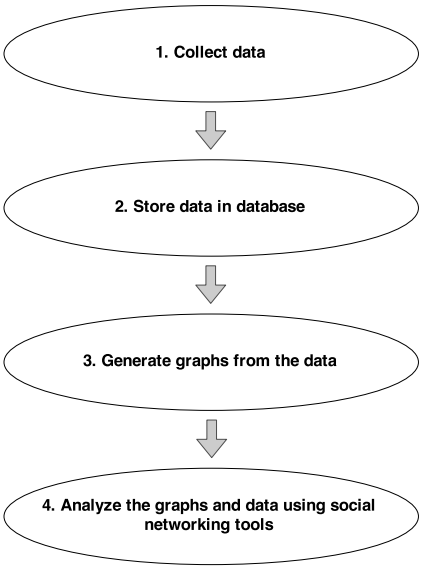


Figure 2: Overview over the work process in this project

In the first step, the location of the projects' source code repositories was collected. Apache hosts an alphabetical list of the projects [Apa] with links to pages with information about the projects, including the location of the repositories and which categories the projects belong to. Using the Curl library [Lib] to fetch data over HTTP, a program to collect the relevant information was written. The commit data was obtained in late February and early March 2015 for 249 different projects hosted under the ASF. The direct links to git projects were often wrong and needed to be manually corrected. A small part of them lead to repositories hosted on other services, primarily GitHub. The projects under ASF use different revision control systems to track changes in the software and provide backups so that it is possible to restore earlier versions of files. Such systems track when a file has been changed and by whom, as well as store commit messages describing and motivating the changes. By comparing different versions of files it is possible to see what changes have been made to the files between the versions. The ASF uses two different systems for revision control, Subversion (SVN) and Git, which are also the two most widely used solutions according to a survey made by Eclipse [Ske14].

In the second step, all the commit histories extracted from the repository containing data such as contributor's name, email, source file modified, time of modification, project name, and project category were loaded into a custom built SQL database. A program was written in order to parse the XML-files containing the commit histories and insert the data into the database. After this step, 228 projects remained from the original 249. Some project repositories had been unreachable at the time of data collection and some were discarded because the number of committers were below a threshold value needed to perform the analysis. The database schema presented in Figure 3 displays entities and corresponding attributes that were used in the analysis process.

In the third step, the commit history data was formatted into an appropriate network analysis format, thus creating edges between all developers who had worked on the same file in a project. The weight for an edge from developer d_i to developer d_j was defined as

$$\frac{\sum\limits_{f \in F_{ij}} e_{if}}{\sum\limits_{f \in F_{ij}} \sum\limits_{k} e_{kf}} \tag{1}$$

where F_{ij} is the set of all files modified by both developer i and j and e_{if} is the number of commits modifying file f by developer i.

Finally, in the forth step, network analysis of developers' collaboration networks was performed and appropriate per project network metrics calculated. The strength of the collaboration is calculated as the number of changes done by one developer to a file relative to the total number of changes for the file by all developers. An example is presented in Table1 and Figure 4 where a file has received commits from three different developers with in total 10 commits to that specific

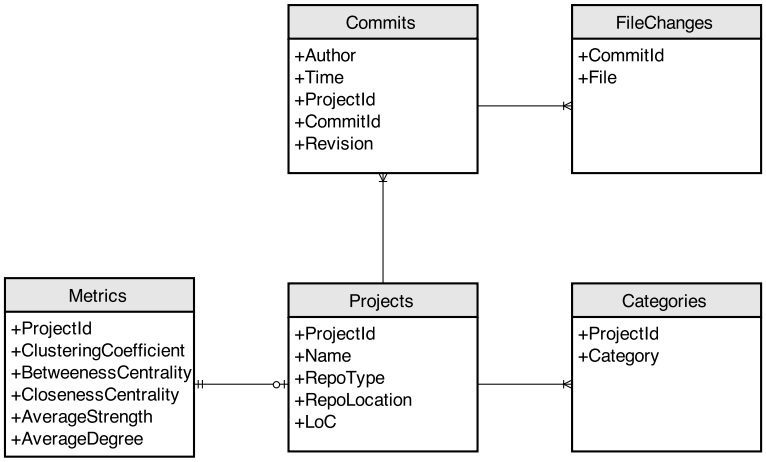


Figure 3: Database design

file. The three developers are called X, Y and Z. Developer X has committed once to the file while developer Y has done three commits and developer Z has done six changes to the file. The weight on the outwards edges from a developer will then be the number of commits that the developer has made divided by the total number of commits. For the edges originating from Developer Y the weight would then be $3/10$. The graph has two edges between each pair of vertices (developers) since all the developers have changed the file once. This process is repeated for all files in the projects and the resulting weights are summed up.

Developer	Number of Commits
X	1
Y	3
Z	6

Table 1: Number of commits per developer

3.3 Metrics

This section defines network metrics used in this study, and discusses their meaning within the context of the constructed developers' collaboration metrics. This study analyzes distribution of metrics with respect to size and type of the project in order to understand if there exists a correlation between the two.

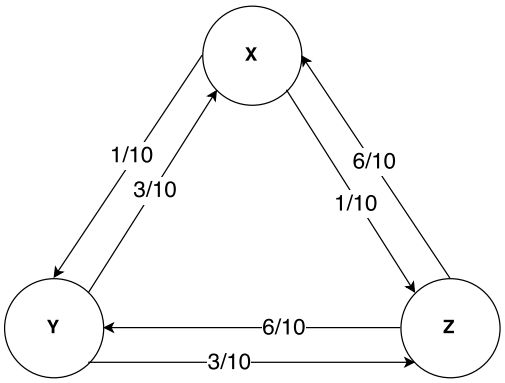


Figure 4: Network structure with 3 developers modifying the same file

Vertex Strength

The strength of a vertex is a representation of a developer's influence over the ones he/she has worked with. A high vertex strength indicates that the developer has a significant influence in the network. Vertex strength is a modified version of vertex degree where the weight of the edges are taken into account. In this study the definition of strength of a vertex v from Barrat et al. [Bar+04] is used:

$$s_v = \sum_{i \in N(v)} w_{vi} \tag{2}$$

where $N(v)$ is the set of all neighbours to vertex v and w_{vi} is the weight of the edge from vertex v to i. In unweighted networks (i.e. all weights are equal to 1) vertex strength and degree are equal. In directed networks each vertex will have an in-degree and an out-degree. The in-degree is the sum of the weights of all edges coming into the vertex and the out-degree is the the sum for all outgoing edges from that vertex.

Clustering Coefficient

The clustering coefficient metric was first introduced in 1998 by Watts and Strogatz[WS98b] as an indicator of how well connected the neighborhood of a vertex is. It can be applied both to a single vertex or to a complete graph, where it is defined as the average clustering coefficient of all vertices. For the network that is studied in this research, the metric indicates how well developers, that have all collaborated with a developer, are connected among themselves, if they have also made changes to same files. The clustering coefficient for a directed graph is defined as:

$$C_v = \frac{n_v}{k_v(k_v - 1)} \tag{3}$$

where n_v is the number of edges between the neighbors of vertex v, and k_v is the degree of v. In other words, the clustering coefficient is the number of edges between a vertex's neighbors divided by the maximum possible number of edges between its neighbors.

The average clustering coefficient is simply defined as the arithmetic mean of the clustering coefficients:

$$C = \frac{1}{|V|} \sum_{v \in V} C_v \tag{4}$$

where V is the set of all vertices in the graph and C_v is the clustering coefficient of vertex v.

For weighted networks, Barrat et al. [Bar+04] presented the following definition

$$C_v^w = \frac{1}{s_v(k_v - 1)} \sum_{i,j} \frac{w_{vi} + w_{vj}}{2} a_{vi} a_{vj} a_{ij} \tag{5}$$

where k_v once again is the degree of vertex v, $a_{vi} = 1$ if there exists an edge from v to i and 0 otherwise, w_{vi} is the weight of the edge from v to i and s_v is the strength of vertex v as defined in Equation 2.

For directed networks, the standard definition of the global clustering coefficient is not applicable. Instead, a modified version of the global clustering coefficient that uses transitivity may be used.

$$C_G = \frac{T_C}{T} \tag{6}$$

Equation 6 shows the definition of the global clustering coefficient, where T_C is the number of triangles in the graph and T is the number of connected triples. When transitivity is taken into account, only triples where one of the included vertices has both an inwards edge and one outwards. In order for the triple to be included in the connected triangles, there has to be an edge from the start of the chain to the last (third) vertex in the chain. Optsahl and Panzarasa [OP09] build upon this version of the global clustering coefficient to adapt it to weighted networks. Note that each triangle is counted three times, once for each vertex.

Centrality

There are several different measures of centrality in a network. The centrality measures use different forms of criteria to indicate the importance of each vertex in a network. The importance of a vertex is not an individual attribute but is instead a measure of the level of influence that the vertex has over the other vertices [HR05]. There is no guarantee that a vertex that is considered important by one criteria is equally important for another criteria. Degree centrality uses the number of adjacent edges as criteria to rank

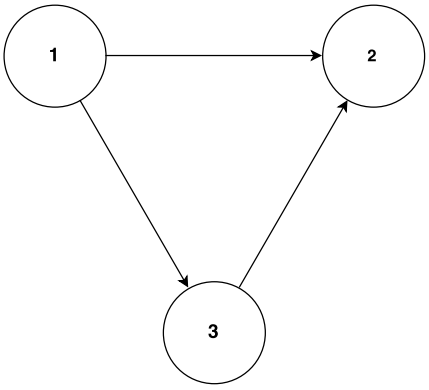


Figure 5: A transitive triangle when centered around vertex 3, but not when centered around vertex 1 or 2

the importance of the vertices. A high number of adjacent edges indicates that a lot of information passes through the vertex. The degree centrality for directed graphs can be expanded to an in-degree and an out-degree, where the in-degree is the sum of weights on the edges coming into the vertex and the out-degree is the total weight of edges leaving the vertex.

That is, a measure of degree centrality can be defined as

$$C_d(v) = d(v) \tag{7}$$

where $d(v)$ is the numbers of edges to or from vertex v and d is the direction, i.e. in-degree or out-degree.

Betweenness centrality is a measure of the centrality of an individual vertex in a network. It is defined as how many of all the shortest paths in the network that passes through a vertex. In this study, a high betweenness centrality of a developer indicates that a the developer is an expert on the system, since he/she has made a large number of changes on different parts of the source code base. A high betweenness centrality indicates that the vertex has a large influence in the flow of data in the network. A developer with very high betweenness centrality can be a risk in a project, since if that vertex for some reason disappears from the network then the communication in the network has to change drastically.

A proposed algorithm for calculating betweenness centrality on weighted networks is presented by Brandes [Bra01b]. Brandes suggests that communication might be quicker along a path that is a little longer than the shortest, but where the weights of the edges are larger. A larger weight indicates more frequent communication which can be beneficial.

The sum of the distance of the shortest paths from a vertex to every other vertex in the network is measured as closeness centrality. The idea is that it is easier for a vertex with a large closeness centrality index to spread information to the rest of the network, than for a vertex with a small index. Hence, in the context of this study, a developer with high closeness centrality is not necessarily the one who performed the highest number of the changes on the different parts of the system as is the case with betweenness centrality, but the one who has performed at least one change on the highest number of different source code files. The definition of the closeness centrality for vertex v is as follows:

$$C_C(v) = \frac{1}{\sum\limits_{s \neq v \in V} d_G(v,t)} \tag{8}$$

where V is the set of all vertices in the network and $d_G(v,t)$ is the length of the shortest path from vertex v to vertex t [Fre79].

3.4 Validity

Internal validity is concerned with influences that can affect the results without the researchers' knowledge. The ASF organizational structure allows a community participant to act in several different roles, which can influence the results of the study. Thus, there can be cases where the committer that performs a commit has not actually made the changes in the commit. The initiator of a change in the code could be a developer that does not have write access to the source code repository. In this case the developer proposes the changes to a committer which then approves them if the changes are considered a valuable improvement to the project. Hence, there exists a possibility that a committer has performed several commits that he/she is not the author of. On the other hand, the committer has to critically review the proposed changes and probably discuss with the developer the purpose of the changes. This would make the committer knowledgable of the changes so that he/she will be able to explain to other developers why the changes were made and what purpose they have.

Threats to external validity pertain to factors that limit generalization of the results. The case study conducted deals with developers' collaboration networks of OSS projects hosted under the ASF. While 258 different projects that were analyzed originate from and outside of the ASF and are developed by different groups of developers, there is no guarantee that the selected projects are good representative of the OSS projects.

Threats to construct validity stem from incorrect measurements being taken via tools used. The information on source code commits was pulled using standard and tested software libraries such as Curl, as well as software management tools for Git and SVN. Data samples were randomly chosen for manual validation as well.

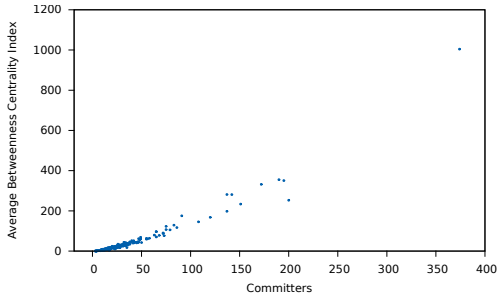
The software used to create and validate network metrics is commonly used for weighted networks and in line with the description of metrics used in this paper.

Conclusion validity is concerned with drawing correct conclusions from the study results. The calculated metrics are complementing in nature and are based on a field of network theory. The conclusions of the study are based on the previously established norms for the interpretation of the metrics used.

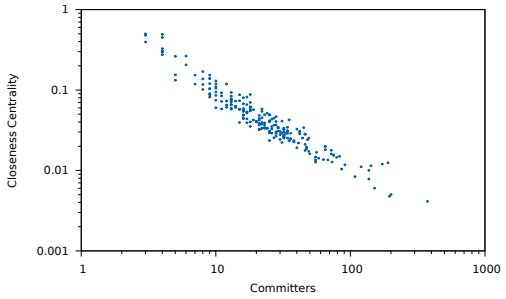
4 Results

In this section the results from the performed analysis are presented. In order to gain a better insight into distribution of the network metrics, majority of the figures presented in this section show metrics for the projects sized by number of committers (up to 30, 31-50, 51-10, 101-200, and more than 200).

4.1 Network Metrics in relation to number of committers



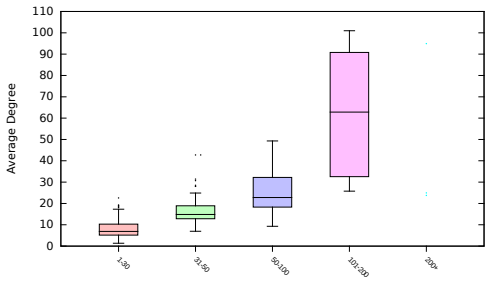
a) Betweenness Centrality



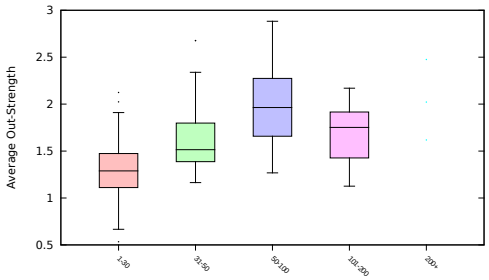
b) Closeness Centrality

Figure 6: Centrality metrics over number of committers per project. Each data point reapersents one project

The correlation coefficient for the average betweenness centrality shown in Figure 6(a) is 0.958, which indicates strong positive correlation of average betweenness centrality with the number of committers on the project. The increase in average betweenness centrality with respect to increase in number of committers on the project is almost linear which is in line with research by Godfrey and Tu [GT00b] discussed in Section 2. The average closeness centrality over the number of committers per project is shown in Figure 6(b). There is a -0.960 correlation coefficient, thus indicating a strong negative correlation. Hence, as the number of developers on the project increases, the average closeness centrality decreases. This indicates that as the number of developers on a project increases, so does their specialization in particular system functionality.



a) Average Degree



b) Average Out-Degree

Figure 7: Average Degree and Average Out-Degree over number of committers on the project

Figure 7(a), showing distribution of average degree over number of committers per project with correlation coefficient of 0.811, indicates a strong correlation between the number of committers and their average degree. Figure 7(b) also indicates low positive correlation between average strength of committers and the number of committers per project with correlation coefficient of 0.512.

4.2 Distribution of centrality indices

Figure 8 shows the measured distribution of individual developer betweenness centrality for different sizes of projects. It can be seen that the betweenness centrality tends to increase rapidly as the number of developers on the project increases, but only for a rather small percentage of developers. The vast majority of developers continues to have betweenness centrality values close to 0. This is indicative of highly specialized software development effort, with very few central developers having worked on all different parts of the system. When compared to the average betweenness centrality displayed in figure 6(a), a better perspective is gained on the high positive correlation between the betweenness centrality and number of committers on the project. The high positive correlation coefficient is due to central developers rapidly increasing values in betweenness centrality as more people become specialized in particular parts of the system.

Figure 9 shows the distribution of developers' individual closeness centrality in projects of different different sizes. The closeness centrality tends to decrease as the number of developers on the project increases and shows a more balanced distribution of values when compared to betweenness centrality.

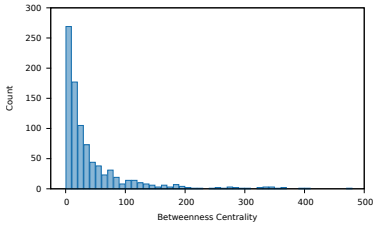
4.3 Clustering Coefficient

Figure 10 shows box-plots of the clustering coefficients for different sizes of the projects, indicating that the majority of the values fall between 0.5 and 0.8. As clustering coefficient with value 1 indicates a fully connected developers network, values in the range 0.5-0.8 can be interpreted as moderately to highly connected developers networks.

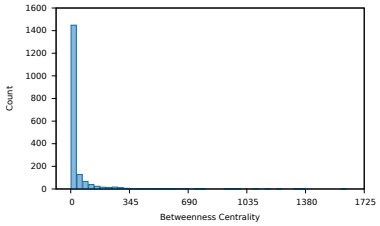
According to these measurements there is no large difference between projects of different size when it comes to the clustering coefficient.

4.4 Metrics for different project-categories

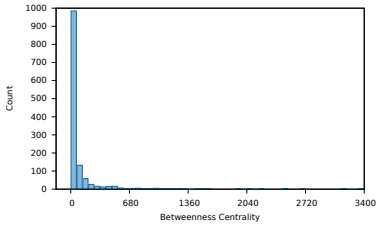
Table 2 shows average degree metrics of all projects based on a category they fall in. The category is based on a type or a domain of the software solution, e.g. all projects that pertain to content management are listed under the category content. While the average degree metrics representing sum of in and out degrees seems to vary between values 9-30, average out-degree or average developers strength seems to show lower variability in range of 1.2-1.9. Table 3 presents centrality metrics and clustering coefficients for different project categories. It is interesting to note here that clustering coefficient that can take values from 0-1, seems to fall into range 0.62-0.8 across all different project sizes and categories.



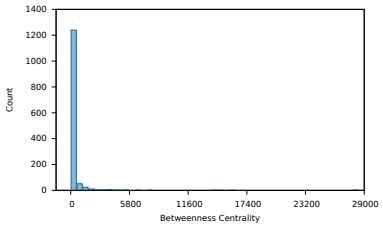
a) 30 or less committers



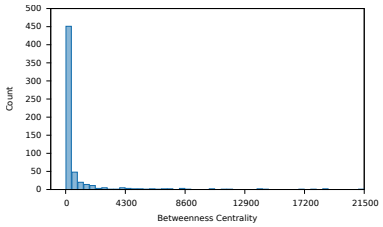
b) 31 to 50 committers



c) 51 to 100 committers

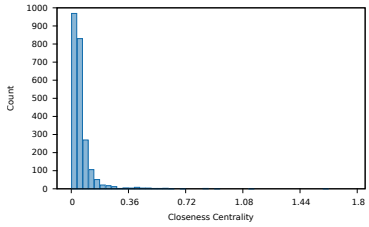


d) 101 to 200 committers

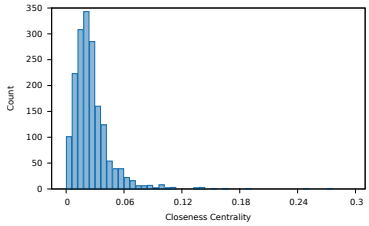


e) more than 200 committers

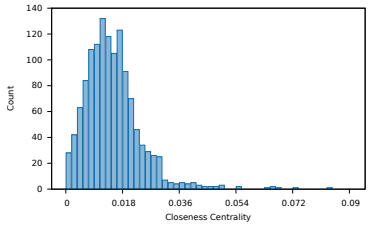
Figure 8: Betweenness centrality for projects with different numbers of committers



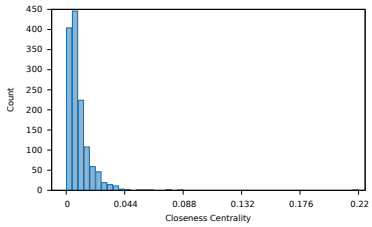
a) 30 or less committers



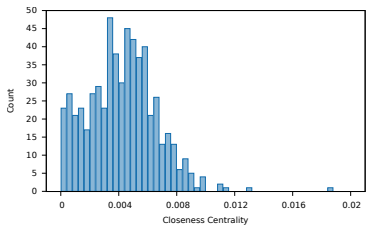
b) 31 to 50 committers



c) 51 to 100 committers



d) 101 to 200 committers



e) more than 200 committers

Figure 9: Closeness centrality for projects with different numbers of committers

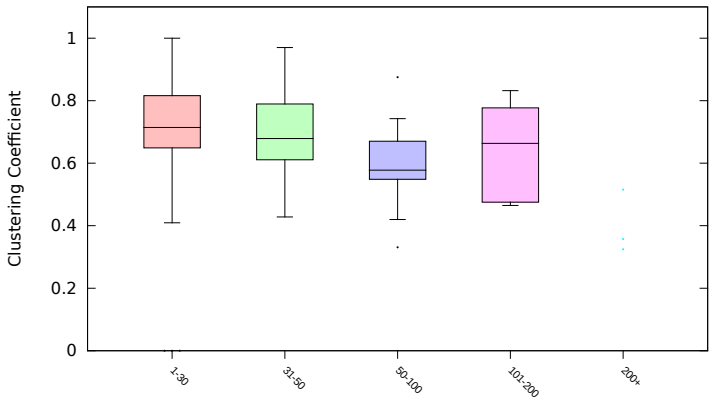


Figure 10: Clustering coefficient over number of committers per project.

Table 2: Average Degree and Strength for different project categories

Category	Count	Average Degree	Average Out-Strength
all	228	13.8456	1.4385
big-data	27	15.7087	1.4860
build-management	17	16.9502	1.1942
cloud	10	22.4263	1.5872
content	14	9.8305	1.5362
database	22	23.6451	1.5136
graphics	5	21.0555	1.6763
http	14	29.7259	1.7260
httpd-module	4	24.2663	1.2070
javaee	9	19.0324	1.8808
library	82	10.0269	1.3700
network-client	18	13.9835	1.5277
network-server	35	16.6735	1.5966
retired	9	9.4677	1.5785
testing	4	13.8425	1.2042
web-framework	25	14.8778	1.5341
xml	28	13.7537	1.5609

Table 3: Average centrality metrics and clustering coefficients for different project categories

Category	Count	Clustering	Closeness	Betweenness
all	228	0.6949	0.0659	39.6982
big-data	27	0.6794	0.0472	54.8562
build-management	17	0.6522	0.1318	49.1389
cloud	10	0.6261	0.0487	156.5869
content	14	0.6493	0.0515	21.3229
database	22	0.7433	0.0379	57.1528
graphics	5	0.7929	0.0419	37.8886
http	14	0.6872	0.0288	89.6320
httpd-module	4	0.7726	0.1703	46.5317
javaee	9	0.6850	0.0304	63.0450
library	82	0.6851	0.0578	24.0122
network-client	18	0.6627	0.0443	38.3626
network-server	35	0.6707	0.0494	42.5121
retired	9	0.7191	0.0575	16.3592
testing	4	0.7400	0.0829	53.5187
web-framework	25	0.7338	0.0734	33.7100
xml	28	0.7430	0.0705	26.8053

5 Discussion

In this section the results of the study are discussed in more detail in order to answer the research questions of the study.

5.1 Centrality

Both centrality indices, betweenness and closeness as depicted in Figure 6(a) and Figure 6(b), have a clear correlation to the number of committers in a project, with the correlation coefficients, 0.96 for betweenness centrality and -0.96, respectively. Thus, as the number of committers on the project increases, so does betweenness centrality, while at the same time closeness centrality decreases. One way to interpret such results is to consider the impact of adding new developers to the project. Those developers with high values of betweenness centrality, will see an increase in the metric as newly added members contribute source code there will be more shortest paths passing through them. On the other hand, closeness centrality will decrease, as newly added developers will form new edges, thus further decreasing distance from the central to peripheral nodes. If trends were different, e.g. betweenness centrality decreasing, and closeness centrality increasing, this could be indicative of shifts in the developers influence, with the strongest developers losing their standing within the project. This could possibly be attributed to addition of large chunks of new code by newly added developers. However, in the analyzed OSS project, both centrality metrics show high probability of linear correlation with the number of committers for the project indicating that central and most influential developers tend to be involved in the development. In complex networks, like networks modeling real world phenomena, it is common that a majority of the nodes have a few links to other nodes, while a small percentage of nodes are highly connected to other nodes.

The projects analyzed in this paper show that the 20.3% most active developers contributed more than 75% of the commits on average. The software projects analyzed in this study, often have a small core of developers that are responsible for most of the commits in a project. Consequently, this creates a network structure with a small amount of developers in the centre and the rest in the periphery. As discussed above, in such network topology, all of the shortest paths pass through a small percentage of vertices (developers) giving these vertices a very high betweenness centrality.

In a network with N committers, when a new committer is added to the developer network there will be N new shortest paths since there is a shortest path from all the ”old” vertices to the ”new” vertex. Almost all of these shortest paths will go through the core developers increasing significantly their betweenness centrality while the ”new” vertex will have a very low betweenness centrality. Thus, several vertices will have an increased betweenness centrality increasing the average for the whole network. The same reasoning can be applied to explain the

decrease in closeness centrality with an increase in number of committers. The number of developers in the periphery compared to the number of developers in the core increases with the total number of developers giving the network a lower average closeness centrality.

Distribution of centrality

The distributions of betweenness and closeness centrality presented in Figure 8 and Figure 9 for projects with different number of developers show a very large number of the developers with a betweenness index relatively small. It is only a very small amount of developers that have a high betweenness centrality but on the other hand their betweenness centrality are in some cases extremely large. This further implies a network structure where a few developers in the core are responsible for a large number of commits while most developers in the periphery of the developer networks only does relatively few commits. Closeness centrality has a similar tendency where the majority of the developers are in the lower half of the observed interval.

5.2 Clustering

Distribution of individual clustering coefficients presented in Figure 10 shows that as the number of committers on the project increases (1-30, 31-50, 51-100), the median clustering coefficient decreases. For the projects with over 100 developers, the second quartile of the distribution is larger, indicating more even spread of clustering coefficients when compared to the projects with less than 100 developers.

Extreme values for the clustering coefficient can be found in smaller projects (projects with a low number of committers) where the clustering coefficient is either 1, 0 or very close to them. This can not be seen in projects with more than about 20 committers. No correlation was found between the clustering coefficient and any other metric, such as project age, lines of code, number of committers, average degree or average strength. The clustering coefficients vary on average by some 0.2 points for data falling into the distributions' second quartiles, 0.65-0.8, 0.6-0.78, 0.56-0.75, and 0.45–0.78 for project sizes 1–30, 31–50, 51–100, and 101–200, respectively. When all three quartiles are taken into consideration, the clustering coefficients of the data varies from 0.3 to 0.9.

Average clustering coefficients for the projects shown in Table 3 fall into range 0.62-0.8 and indicate low variability of this metric across all different projects' sizes and categories.

5.3 Average Degree

As can be seen in Figure 7(a) the median degree and second quartile range increases as the size of the project increases. Thus in larger projects one can expect

to see higher and more diversely distributed values of the average degree. The average degree is calculated based on in and out edges from a developers node, while the average strength displayed in Figure 7(b) is based only on the developers out degree.

Taking into account that a developer's strength is an indicator of her/his collaboration strength measured as the relative number of changes performed by the developer, from Figure 7(b) we can see that average developer's strength increases at slower pace than the average degree. The average strength for projects with over 100 contributors, starts decreasing which can be interpreted that an increase in project size brings more even distribution of developers' strengths, as more developers are being specialized in different parts of the system.

5.4 Synthesis of Results

Per research question 1, the results of the study indicate that sustainable OSS projects of different types and sizes show certain commonalities that can be seen in linear growth of the centrality metrics based on a project size or the number of committers on a project, a medium to high average clustering coefficient and predictable developers' strength.

Per research question 2, benchmarks that can be proposed from the results include average clustering coefficient falling into range 0.62-0.8, average developers' strength metric falling into range 1.2-1.9, and high linear correlation of betweenness and closeness centrality metrics throughout a project's growth phase.

6 Conclusions

The work in this study investigated the developer collaboration in OSS projects hosted under ASF by applying network analysis with a goal of finding common collaboration metrics among the projects. The results show that the average betweenness centrality and average closeness centrality is correlated with the number of committers, which might be attributed to the structure of the developer networks where the core consists of a few developers that are responsible for a large proportion of the total number of commits. The distribution of centrality indices in the projects also seems to support this network structure. This information can be used as a baseline to which any software project effort could be compared and monitored, especially as new developers are added to the project. A large project with a very low average betweenness centrality compared to the ASF projects may have an ineffective development process because a low betweenness centrality indicates that most developers seem to do changes on almost all parts of the projects. The study results could indicate that the most effective way of developing quality software in OSS context, is to have the majority of the developers specialized on a specific part of the project and only a small percentage of the developers or experts that bind different parts of the system together into the final software product.

The individual clustering coefficients show more variability, an increase in median and lower second quartile distribution variability for projects of up to 100 developers, while for the projects with over 100 committers, the median clustering coefficient does not follow the increasing trend. However, trends were observed for the average clustering coefficient across different types and sizes of the project with value range from 0.62-0.8, so this range could potentially serve as baseline against any development effort as well. Very low clustering coefficients, that can be observed in networks that are randomly generated, were rare and only found in projects with less than 20 contributors.

Hence, the results of this study indicate a possibility that some common collaboration metrics in OSS projects exist, and more work is needed to validate results in OSS context by analyzing other widely used OSS products as well as in the closed source context. The results could be potentially used to assess effectiveness of development within the closed source environment by carrying out the similar analysis on closed source code, and comparing the network metrics to the one found in the OSS projects. We believe that the results of the study are promising enough to motivate additional analysis of other OSS communities that host mature and widely used open source projects with the purpose of establishing the benchmarks.

References

[Ope] *Apache Open Office*. `https://www.openoffice.org/`. [Online; accessed 20-October-2015]. 2015.

[Apa] *Apache Projects Alphabetical Index*. `http://projects.apache.org/indexes/alpha.html`. [Online; accessed 20-October-2015]. 2015.

[Bar+04] A. Barrat, M. Bartholemy, R. Pastor-Satorras, and A. Vespignani. "The architecture of complex weighted networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 101.11 (2004), pp. 3747–3752.

[Bir+08] Christian Bird, David Pattison, Raissa D'Souza, Vladimir Filkov, and Premkumar Devanbu. "Latent social structure in open source projects". In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM. 2008, pp. 24–35.

[Blo+12] Michael Bloch, Sven Blumberg, and Jurgen Laartz. "Delivering large-scale IT projects on time, on budget, and on value." In: *Mckinsey Quarterly* (2012).

[Bra01b] Ulrik Brandes. "A faster algorithm for betweenness centrality". In: *Journal of Mathematical Sociology* 25.2 (2001), pp. 163–177.

[Cro03] Kevin Crowston. "The social structure of open source software development teams". PhD thesis. Syracuse University, 2003.

[Lib] *Curl Open Source Project*. `http://projects.apache.org/indexes/alpha.html`. [Online; accessed 20-October-2015]. 2015.

[Fou15a] Apache Software Foundation. *Apache HTTP Server*. `http://httpd.apache.org/`. [Online; accessed 20-October-2015]. 2015.

[Fou15b] Apache Software Foundation. *Apache Hadoop*. `https://hadoop.apache.org/`. [Online; accessed 20-October-2015]. 2015.

[Fre79] Linton C. Freeman. "Centrality in social networks conceptual clarification". In: *Social networks* 1.3 (1979), pp. 215–239.

[GT00b] Michael W. Godfrey and Qiang Tu. "Evolution in open source software: A case study". In: *Proceedings of the International Conference on Software Maintenance, 2000*. IEEE. 2000, pp. 131–142.

[HR05] Robert A. Hanneman and Mark Riddle. *Introduction to social network methods*. University of California Riverside, 2005.

[LM03] Vito Latora and Massimo Marchiori. "Economic small-world behavior in weighted networks". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 32.2 (2003), pp. 249–263.

[LT00] Josh Lerner and Jean Triole. *The Simple Economics of Open Source*. Working Paper 7600. National Bureau of Economic Research, 2000.

[LF+08] L. Lopez-Fernandez, G. Robles, J. Gonzalez-Barahona, and I. Herraiz. "Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects". In: *International Journal of Information Technology and Web Engineering* 1 (2008), pp. 28–50.

[Mad+02] Gregory Madey, Vincent Freeh, and Renee Tynan. "The open source software development phenomenon: An analysis based on social network theory". In: *Proceedings of the American Conference of Information Systems*. 2002, p. 247.

[Moc+00] Audris Mockus, Roy T. Fielding, and James Herbsleb. "A case study of open source software development: the Apache server". In: *Proceedings of the 22nd international conference on Software engineering*. ACM. 2000, pp. 263–272.

[New01] Mark EJ. Newman. "Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality". In: *Physical review E* 64.1 (2001), p. 016132.

[OP09] Tore Opsahl and Pietro Panzarasa. “Clustering in weighted networks”. In: *Social networks* 31.2 (2009), pp. 155–163.

[OAH14c] Alma Oručević-Alagić and Martin Höst. “Network analysis of a large scale open source project”. In: *Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE. 2014, pp. 25–29.

[RH08] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Sev12] Charles Severance. “The Apache Software Foundation: Brian Behlendorf”. In: *Computer* 45.10 (2012), pp. 8–9.

[Ske14] Ian Skerrett. *Eclipse Community Survey 2014*. `https://ianskerrett.wordpress.com/2014/06/23/eclipse-community-survey-2014-results/"`. [Online; accessed 20-October-2015]. 2014.

[WF94b] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Vol. 506. Cambridge University Press, 1994.

[WS98b] Duncan J. Watts and Steven H. Strogatz. “Collective dynamics of small-world networks”. In: *nature* 393.6684 (1998), pp. 440–442.