

# Deriving Project Health Indicators of Open Source Software Projects using Social Network Analysis

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Wirtschaftsinformatik**

eingereicht von

**Andreas Kaltenecker**

Matrikelnummer 0325400

an der  
Fakultät für Informatik der Technischen Universität Wien

Betreuung

Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Mag.rer.soc.oec. Stefan Biffi

Mitwirkung: Univ.-Ass. Dipl.-Ing. Dr. Alexander Schatten

Wien, 06.05.2010

\_\_\_\_\_  
(Unterschrift Verfasser)

\_\_\_\_\_  
(Unterschrift Betreuer)



# Erklärung zur Verfassung der Arbeit

Andreas Kaltenecker, Brückelgasse 9, 7471 Rechnitz, Österreich

*“Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.”*

Wien, 06.05.2010

(Andreas Kaltenecker)





# Abstract

Free/Libre Open Source Software (FLOSS) differs from closed source software development in a variety of ways. There are different development processes, distinct motivations, toolsets and structures of a development team. In both cases it is possible to analyze the current quality of software. As a difference, Open Source Software provides the chance to look behind the scenes. Due to the open development process it is possible to analyze not only the product quality but also the structure of the community and the development process behind.

This diploma thesis takes up this advantages and uses publicly available data in a social context in order to extract information about how members of a community act and interact. The interpretation of this information and the analysis of change over time allows to draw conclusions about the structure of a community which is used in order to evaluate the current health status and future development of a FLOSS project.

The instruments introduced in this thesis which meet the requirements for evaluating the health status of FLOSS projects are provided within the concept of "Social Network Analysis". Social Network Analysis can be seen as a roof for measures describing and analyzing the composition, size and scope of social networks which consist out of actors and relations combining them.

The possibilities of the Social Network Analysis are discussed and health indicators which are applied to specific examples in the practical part of this thesis are introduced. In order to calculate relevant measures or illustrate the dependencies within a network an own tool called "SNAnalyzer" [Kal] was developed.



## Kurzfassung

Free/Libre Open Source Software (FLOSS) unterscheidet sich von Closed Source Software Entwicklung in einer Vielzahl an Dimensionen. Es gibt Unterschiede im Entwicklungsprozess, in der Motivation der Zusammenarbeit, in der Frage, welche Werkzeuge verwendet werden und auch im Aufbau des Entwicklerteams. In beiden Fällen ist es möglich, die Qualität eines Endproduktes zu erheben. Im Unterschied zu proprietärer Softwareentwicklung bietet FLOSS die Möglichkeit, hinter die Kulissen zu blicken. Der offene Entwicklungsprozess macht es möglich, nicht nur das Produkt am Ende zu analysieren, sondern auch die Art und Weise der Zusammensetzung einer Gemeinschaft welche hinter einem Projekt steht zu untersuchen.

Diese Diplomarbeit greift die genannten Vorteile auf und verwendet öffentlich verfügbare Daten aus einem sozialen Kontext heraus, um auf die zugrundeliegende Struktur einer Gemeinschaft zu schließen, aber auch um Informationen über die Art und Weise, wie Kommunikation stattfindet zu untersuchen. Die Interpretation dieser Informationen und die Analyse der Veränderung im Laufe der Zeit ermöglicht es, Rückschlüsse auf Abhängigkeiten innerhalb eines Projektteams zu ziehen.

Die Instrumente, welche in der vorliegenden Diplomarbeit die Ansprüche zur Erhebung der "Projektgesundheit" erfüllen, werden mit dem Konzept "soziale Netzwerkanalyse" (SNA) zur Verfügung gestellt. Dieses Konzept kann als Dach für eine Sammlung an Ansätzen und Kennzahlen gesehen werden, welche die Analyse der Zusammensetzung, Größe und Schwerpunkt eines sozialen Netzwerkes ermöglicht. Ein soziales Netzwerk besteht aus Akteuren (in diesem Fall die Mitglieder eines FLOSS Projektes) und Beziehungen, welche Akteure miteinander verbindet.

Das Instrument "soziale Netzwerkanalyse" wird in dieser Diplomarbeit vorgestellt und es wird beschrieben wie eine Umsetzung auf "Projektgesundheit" erfolgen kann. Konkret wird dies im praktischen Teil dieser Diplomarbeit anhand von vier Projekten angewendet. Um die Daten für eine weitere Interpretation aufzuarbeiten, wurde ein eigenes Projekt ins Leben gerufen, welches die notwendigen Indikatoren errechnet und die Abhängigkeiten zwischen den einzelnen Projektmitgliedern illustriert [Kal].



## Acknowledgements

My special thanks go to Stefan Biff and Alexander Schatten for guiding and supporting this Diploma Thesis. They improved the work with many hints and provided constructive feedback in various discussions.

Furthermore I want to thank my colleagues at ZZ Vermögensverwaltung. They made it possible to put studies and work under one manageable roof by providing understanding and support all the time.

My deepest gratitude goes to my friends for moral support and family for their unflagging love and support throughout my life. Especially I appreciate the help of my brother Franz for supporting this thesis in delicious questions about graphical illustrations in times of being up to his ears with work and training. Rock on Franz!

Above all I thank my girlfriend Elisabeth for giving moral support, motivation and inspiration whenever needed.



# Contents

<b>1</b>	<b>Motivation</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Related Work</b>	<b>7</b>
3.1	Social Network Analysis . . . . .	7
3.2	Project Health . . . . .	8
3.3	Overview . . . . .	9
<b>4</b>	<b>Open Source Software</b>	<b>11</b>
4.1	Institutional vs. Collaborative . . . . .	12
4.2	Market . . . . .	14
4.3	Characteristics . . . . .	17
4.3.1	Actors . . . . .	18
4.3.2	Motivation to Contribute . . . . .	19
4.3.3	Team Structure . . . . .	20
4.3.4	Ressources . . . . .	22
4.3.5	Project Management . . . . .	22
4.3.6	Development Process . . . . .	24
<b>5</b>	<b>Project Health</b>	<b>29</b>
5.1	Why Project Health? . . . . .	30
5.2	Project Lifecycle . . . . .	31
5.3	Health Indicators . . . . .	32
5.3.1	Defining the Shape . . . . .	33
5.3.2	The Cathedral and the Bazaar . . . . .	34
5.3.3	High Dependency - High Risk? . . . . .	34
<b>6</b>	<b>Research Questions</b>	<b>37</b>
<b>7</b>	<b>Social Network Analysis</b>	<b>39</b>

## Contents

7.1	Introduction . . . . .	39
7.2	Development of Social Network Analysis . . . . .	41
7.3	Key Concepts . . . . .	44
7.4	Data Management and Illustration . . . . .	45
7.4.1	Graphs . . . . .	45
7.4.2	Matrices . . . . .	46
7.5	Indicators measuring Centrality or Prestige . . . . .	48
7.5.1	Degree Centrality . . . . .	49
7.5.2	Closeness Centrality . . . . .	52
7.5.3	Betweenness Centrality . . . . .	55
7.5.4	Information Centrality . . . . .	58
7.5.5	Eigenvector Centrality . . . . .	59
7.5.6	Centrality Indicators for Directed Relations . . . . .	60
7.5.7	Prestige . . . . .	61
<b>8</b>	<b>Social Network Analysis in Practice</b>	<b>65</b>
8.1	Action and Interaction Data . . . . .	65
8.2	Availability of Social Data in FLOSS Projects . . . . .	67
8.3	Reference Tools and Libraries for Social Network Analysis . . . . .	68
8.3.1	Independent Tools . . . . .	69
8.3.2	Libraries . . . . .	70
<b>9</b>	<b>SNAlyzer</b>	<b>73</b>
9.1	Technical Solution . . . . .	73
9.2	Functionality . . . . .	74
9.3	Social Data . . . . .	75
<b>10</b>	<b>Evaluation</b>	<b>79</b>
10.1	Healthy Project . . . . .	80
10.2	Convalescent Project . . . . .	85
10.3	High Dependency - High Risk! . . . . .	91
10.4	Collateral Damage . . . . .	93
<b>11</b>	<b>Summary</b>	<b>97</b>
<b>12</b>	<b>Critical Discussion</b>	<b>101</b>
<b>13</b>	<b>Suggestions for Future Work</b>	<b>103</b>
<b>A</b>	<b>SNAlyzer</b>	<b>105</b>



**B How to use the statistic environment R within a Java application 113**

*Contents*

# 1 Motivation

Free and Libre Open Source Software (FLOSS) has become a real counterpart to closed source software and for nearly every proprietary product an equivalent out of the area of open source is available [Van09]. For stakeholders but also for the initiators of a project the question of stability and continuity is essential. In the area of software development, a lot of different indicators, measuring the quality of a certain project, are existing. Out of the view of stakeholders it is important to differentiate a variety of projects in order to create a basis for decision making. Project management although is interested in identifying critical situations to be able to react on time.

Qualitative indicators could rely on two different approaches. The first would deal with the product itself and evaluates solely the output of an underlying process (like a blackbox). Without knowing something about how the product is created or what demands have been previously defined the product itself can be analyzed using criteria like reaction times or release intervals. Another approach could furthermore put the focus towards the underlying process (whitebox). By assessing the environment and the factors which influence the final output another view can be introduced and used for evaluations but to be able to achieve, this insight into the underlying project needs to be available. Comparing open and closed source software development one particular aspect is eye-catching and essential for the mentioned approach. Due to the framework which defines FLOSS the development process is completely transparent.

Understanding the principles of FLOSS development also means that a researcher should not only see the chance to be able to evaluate the development process of open source software projects because of the mentioned transparency. Furthermore the way how a community is set up provides the potential or raises constrictions to become a successful project and provide a high quality product. Both situations define the future direction of a project [WSM<sup>+</sup>06].

## *1 Motivation*

Aim of this thesis is to introduce kind of a clinical thermometer in the first instance for open source software projects in order to identify the current health status of a project. The health status itself bases on the shape and the evolution of a underlying project team over time. This will be determined using social data which is stored during the development process and freely available. With the instruments introduced in this thesis it will be possible to differentiate between healthy and sick projects by identifying any variations in typical developments of healthy projects. How healthy projects develop will be previously defined and will be shown on specific examples. Furthermore in a second step by using health indicators it will be possible to make prognoses about the future development of a project by looking at specific indicators.

## 2 Introduction

This diploma thesis introduces a possibility to research the health condition of FLOSS projects. Health in this case stands for stability and continuity in the development process which is essential for stakeholders but also for members of a project. As basis for a practical implementation which is the content of the second part, a theoretical background is provided. This theoretical background consists of three chapters which can be summed up briefly as follows:

**Free and Libre Open Source Software** Chapter 4 introduces Free and Libre Open Source Software (FLOSS). FLOSS itself has left the area of being interesting for a niche group and has reached the open market. Reason for this change can be found in the following points [AAB<sup>+</sup>06]:

- Low cost and low barrier to entry
- Availability of customization and local support services
- Vendor independence and flexibility

In nearly every software group FLOSS alternatives are available and in some of them they are even dominating the market. Programs like Linux, Apache or Open Office are widely known and are often recognized as representatives for a whole ideology. In order to understand the reason for the focus on open source in this thesis it is important to know some basics about this kind of software development.

Before software became a commercial product it was just an add on until some people recognized the valuable asset. Besides that commercial approach, a parallel universe started to develop where people had strong ideologies containing different cornerstones. Primarily enabled by the internet, open source relies on the work of people from all around the world who participate freely out of different kinds of motivation. The result of this effort is put under the roof of

## 2 Introduction

certain licenses but not to protect knowledge in order to generate a business advantage.

In fact the freedom of information needs to be protected which means that projects agreeing to one of the various open source licenses usually depict the following freedoms:

- The underlying source code of a product has to be freely available
- Changes and redistributions are explicitly allowed
- Derived products have to guarantee the above mentioned freedoms

Although the approach of project health could also be useful for closed source projects it is focused on open source projects out of different reasons. Due to the infrastructure which allows people to participate in one project they leave footprints during the whole development. People communicate via mailinglists which are readable for public, coordinate via tracker tools and use revision control systems like the concurrent versioning systems (CVS) in order to manage the source code and all these systems are public. Walt Scacchi [Sca07] introduces for these mediums the term informalisms. These informalisms make this thesis possible.

**Project Health** In chapter 5 project health is introduced. Different health states of a FLOSS project are defined and indicators are introduced. Furthermore the evolution of healthy projects is shown which has a typical form and will be used as a reference for research.

The know-how about reading and interpreting footprints of FLOSS development is the topic of the next chapter: Social Network Analysis (SNA)

**Social Network Analysis** In order to detect the health status of a FLOSS project instruments available within Social Network Analysis (see chapter 7), will be used. The instrument Social Network Analysis is defined by Job-Sluder [JS06] as follows:

“Social Network Analysis (SNA) is another diverse set of methods for examining the relations between individuals and groups (Wasserman & Faust, 1994). A relation can be any form of social interac-

tion including knowing the other person by name, communication, or shared membership within a group. Hampton and Wellman (2003) examined Internet interactions, name recognition and residence visits. While Paolillo (2001) examined responses in chat interactions. SNA provides powerful methods for quantifying the size, shape and scope of a given network.”

The advantage of FLOSS which is a total transparency of the development process is used. Actors or in the case of this thesis participants in one project are connected to each other by a diverse set of social relations. This can be either a common contribution to a patch file or communication in mailing lists. Out of this relations a graph like in figure 2.1 can be generated which illustrates the complete social relationships of a development team in a specific period. In a next step the instruments, provided by SNA can be used to “[...] quantify the size, shape and scope of a given network.” [JS06] which is a basis for further interpretation.

2 Introduction

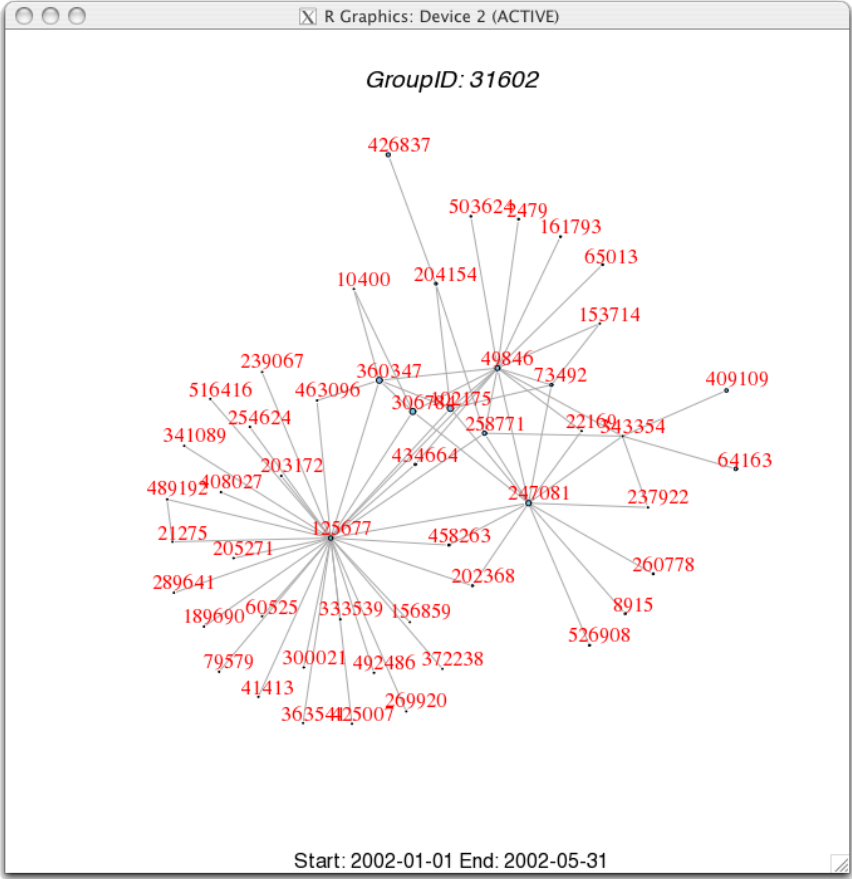


Figure 2.1: Social Network with common commits to a source file as linking relation



## 3 Related Work

The following chapter gives a brief overview about the contributions which had high impact on this thesis and furthermore tries to differentiate from existing contributions on this topic.

### 3.1 Social Network Analysis

Social Network Analysis has a very long history and parts of the instruments originate from an era when computers had not been invented yet. History of some parts in the concept can be traced back to the 1950s and descends from the area of sociology. The basic knowledge about the instrument of Social Network Analysis used in this thesis relies on work by Stanley Wasserman and Katherine Faust. With [WF94] they provide a book which covers a general introduction to the topic and ranges to very detailed explanations of specific instruments, relevant for this thesis. It is used as a reference for the bigger part of chapter 7.

Although the offspring of the next work was the year 1979, the very important contribution to Social Network Analysis by Linton C. Freeman from the Lehigh University in Bethlehe, U.S.A. needs to be mentioned. He sums up in his groundbreaking (he called it) conceptual clarification [Fre79] on basic concepts like the different approaches for centrality and has established a basis for the understanding of centrality in social networks. Out of this work knowledge about the calculation of different indicators measuring the dependencies within the members of FLOSS teams is used.

During the research phase the scientists Kevin Crowston and James Howison from Syracuse University were identified as big players in the area of Social Network Analysis in the recent time of this science. They have shown different ambitions concerning FLOSS which contain the following approaches: They

### 3 Related Work

are hosting a project called FLOSSMole [HCC06] where information of FLOSS projects are collected out of different data sources in order to provide material for scientific research in different manner. Besides hosting and providing data they are also using Social Network Analysis in order to research the development of FLOSS. The information provided in [CH04] shows some basic information about the social structure and the shape of virtual teams.

Hossain et al. [HZ08] are dealing with centrality measures and put it in relation to project quality. In their context project quality is evaluated by measuring:

- Number of defects fixed per software promotion
- Number of defects reported
- Time between bug report and bug fix

In their paper they have evaluated if the underlying structure has effect on their definition of project quality. What differs from the approach in this thesis lies in the fact, that the knowledge about structures and changes of an underlying network is used in order to analyze the health state and it is tried to predict the future development of a project. Project health is defined as a stability and continuity factor which has influence to the project by assuming that only a healthy project can produce high quality software over a longer period of time.

## 3.2 Project Health

Wahyudin et al. are dealing in their work with project health and introduce this approach to FLOSS [Wah08][WSM<sup>+</sup>06]. They defined the mature and immature state and introduced kind of a circuit of life for FLOSS projects which was picked up for this thesis. Unlike to this thesis, Wahyudin et al. do not use Social Network Analysis as a method to indicate the project health of the underlying project. Moreover they rely on indicators considering, how they call it, the developer contribution patterns which capture for example a ratio between email conversation and defect status changes. Another indicator relies on service delay which stands for the bug response time or time to resolve an issue/defect [Wah08].

Now to focus more on the shape of healthy projects it is referred to Eric

Ramond [Ray00]. He explained the idea of “The Cathedral and the Bazaar” in which he depicts that young FLOSS projects are usually managed like cathedrals with actors who are located very central in the project and decide what to do. The longer projects are existing and the more participants are contributing the more decentralized a project gets of which he speaks of a bazaar.

Crowston et al. have also researched the conditions of a healthy FLOSS project[CH06]. They assume that healthy projects are generally onion shaped and the different contributors have their specific position in the shells. Social Network Analysis is used in order to determine this fact. They also describe some standard patterns concerning organization within a project and point out typical problems.

## 3.3 Overview

This thesis is structured as follows: Chapter 1 depicts the motivation for the thesis and also describes the goals followed by chapter 2 which deals with a general introduction. Chapter 4 picks up an introduction to Free/Libre Open Source Software which contains information ranging from general basics to a description of structures and processes of FLOSS projects. The next chapter (chapter 5) deals with a description of the project health approach containing information about health indicators or typical developments within FLOSS projects. After the research questions in chapter 6 an overview about Social Network Analysis is provided in chapter 7. In a general introduction methods are shown which manage relational data and indicators are introduced measuring the dependencies within a project team. What follows next is a practical implementation of the knowledge given so far. Chapter 8 provides a bridge between the theoretical and the practical part of this thesis. A tool created in order to support the identification of project health is described in chapter 9. Chapter 10 contains concrete examples which have been researched. Followed by the final part of this thesis where the findings are summed up (chapter 11) and critically discussed in chapter 12. Further research possibilities are described in chapter 13.

### 3 *Related Work*

## 4 Open Source Software

**Software becomes commercial** When somebody thinks of buying a washing machine it is a matter of fact that the software which controls the machine is included. In the beginning of information technology it was quite the same with computers. When DEC or IBM have started their business operating systems have been a logical add on, also including source code and documentation of the operating system [The04]. Neal Stephenson emphasizes this fact very appropriate in the following anecdotes:

“A person who went into a coma before Microsoft was founded, and woke up now, could pick up this morning’s New York Times and understand everything in it—almost: Item: the richest man in the world made his fortune from-what? Railways? Shipping? Oil? No, operating systems. Item: the Department of Justice is tackling Microsoft’s supposed OS monopoly with legal tools that were invented to restrain the power of Nineteenth-Century robber barons. [...]” [Ste99]

Stephenson argues that it was not imaginable to that time that somebody might sometimes become the richest person on earth with earnings out of licenses and copyright based on intellectual property in a time where only hard facts were counting.

**FLOSS as answer to commercial software development** Out of a certain point of view, Bill Gates and Paul Allen are responsible for the gap between open and closed source software development. After the source code of software which was created by them was spreading rapidly, Gates became very angry because just a minority of the users paid for their copy. This has caused him to write “an open letter to hobbyists” which was first published on the 3rd of February 1976 in Computer Notes [Gat76]. The letter had high impact on the

## 4 Open Source Software

hacker community at that time because one out of their own rows was questioning the basis of their ideology. Gates claimed that it is a crime to steal software. He was the first one thinking of business models based on selling something abstract like software. He introduced intellectual property and broke the rules of the manifest. Whether it was his aim or not, this letter caused a paradigm change. Williams declares it in his work “Free as in Freedom” as follows:

“Software had become such a valuable asset that companies no longer felt the need to publicize source code, especially when publication meant giving potential competitors a chance to duplicate something cheaply.” [Wil02]

**The birth of free and open source software development** The development of FLOSS as a separate paradigm goes hand in hand with the story of some key actors and the appearance of some specific software projects. Richard Stallman was the first who picked up the aim to create a completely free software system he called GNU (which is a acronym for GNU’s Not Unix) and today, he is one of the biggest proponents of the free software movement. With Linux Linus Torvalds has created an operating system which became a representative for FLOSS. No other Open Source Software Projects became as famous and big as Linux. Step by step the movement achieved acknowledgment and today it is simply indispensable. FLOSS needs to deal with constrictions, commercial projects do not have but can also play out advantages closed source projects do not have and history has shown that success is possible (see section 4.2). In the following part of the thesis it is focused on the different approaches of commercial software development and Free/Libre open source software development. Furthermore it is pointed out what rank FLOSS has enveloped and information is provided in order to understand how organization happens.

### 4.1 Institutional vs. Collaborative

Unlike to Bill Gates’ position that good software will always rely on proprietary approaches, history meanwhile has shown something else (see section 4.2). According to this fact the interesting questions are: What made the success of Open Source Software Development in different areas possible and what are the

#### 4.1 Institutional vs. Collaborative

differences in the underlying value adding process of open and closed source software projects is the content of the following part of the thesis.

Clay Shirky introduces in his book “Here Comes Everybody” [Shi08] two structural approaches to follow the aim of producing valuable output which he names institutional and collaborative. Out of the view of this thesis it is about open and closed source software development but economically considered he writes about coordination costs. Coordination costs are that kind of costs that need to be invested in order to organize a bunch of people to produce coherent output with lasting value. The first and classical approach would be to coordinate people by starting an institution. You have to raise infrastructure and provide a framework in order to put everything relevant under one manageable roof. The problem which appears is that increasing complexity leads to raised communication costs which are a big contributor to coordination costs. To get rid of this Shirky presents a second approach which is not to institutionalize rather than to put the cooperation into an infrastructure. By providing systems which let people organize and coordinate as a byproduct, you get an alternative point of view which we have in the case of Open Source Software Development.

**“Given enough eyeballs, all bugs are shallow”** [Ray00] Out of the view of both approaches a tension logically arises. Institutionally we would ask, which people should be hired to get a specific job done and what is following next is creating restrictions. What needs to be done is to create an evaluation system to select a specific amount of people in order to get the best effort despite the constrictions you have to deal with. On the other hand, from coordination point of view we can ask, what is the value of a specific contribution because we do not have to tear down the resources.

According to Shirky, Steve Ballmer complained in an interview, that they have analyzed the community of Linux and it is a myth that thousands of people are participating because the majority has shown up only once [Shi08]. A corporation like Microsoft would never hire someone who is wasting time and money and makes minor input because the valuation is different. However FLOSS projects like Linux can deal with the question: Was it a good input of this specific person? What if this single contribution was the solution to a major security problem. These kinds of values will never be reachable for the classical approach and Ballmer has to be frightened of this single developer! According to Shirky, projects like Linux can draw on abundant resources.

## 4.2 Market

What has once started small in the rooms of enthusiastic individuals has spread over the whole world and has created a movement which is not imaginable to live without these days. An article in the New York Times refers to European Commission's competition chief Neelie Kroes who told that FLOSS has put huge pressure on commercial software development because in nearly every area, FLOSS provides an alternative [Van09].

Table 4.1: FLOSS usage by application in companies in the UK, Sweden, and Germany [AAB+06]

	UK		Sweden		Germany		Total
	small	large	small	large	small	large	
OSS as server operating system	8.1%	3.7%	9.8%	11.0%	30.7%	30.6%	15.7%
		6.4%		10.1%		30.7%	
OSS for databases	13.3%	4.6%	7.5%	8.2%	14.1%	20.8%	11.1%
		9.9%		7.6%		15.7%	
OSS on the desktop	7.6%	2.0%	3.4%	3.2%	13.7%	6.5%	6.9%
		5.4%		3.3%		12.0%	
OSS for websites	7.9%	4.3%	7.5%	8.7%	15.8%	17.3%	10.1%
		6.5%		7.8%		16.2%	
Source: Survey results (n=395).							

**Marketshare** According to a European study on the economic impact of open source software [AAB+06], FLOSS products are ranked under the top 3 in software areas like operating systems, desktop operating systems, web browsers, databases or e-mail clients. Table 4.1 shows the result of this study, based on an End-User survey with a sample size of 625 companies from Germany, UK and Sweden. Different areas have been researched and the results show that FLOSS has the highest acceptance as operating system on the server market with about 15.7% in total. Also noticeable is the fact, that out of these 3 countries, Germany has the highest FLOSS quote across all different categories.



Another example comes out of the web server area. Looking at figure 4.1 you can see the historical market share of different top servers beginning in 1995 and reaching until 2010. Since 1996 the leading web server with a worldwide market share in 2010 of about 50% is the open source product Apache [A<sub>paa</sub>].

To sum up, FLOSS became irreplaceable in many kinds of businesses. In the public sector the highest usage of FLOSS can be found in Europe, followed by Asia and Latin America. However in the commercial software developing world, more and more medium and large scale enterprises decide towards FLOSS. Looking at table 4.2 you can see the biggest contributors which are the former company Sun Microsystems (now part of Oracle), IBM and Red Hat and the market values of their contributions.

Table 4.2: Cost estimate for FLOSS code contributed by firms. Refined from [AAB<sup>+</sup>06]

Number of firms	986		
Source lines of code	31.2 million		
Estimated effort	16 444 person years		
Estimated cost	1.2 billion Euro		
Top contributors			
Rank	Name	Person-months	Cost (mil euro)
1	sun microsystems inc.	51372	312
2	ibm corp.	14865	90
3	red hat corp.	9748	59
4	silicon graphics corp.	7736	47
5	sap ag	7493	46
6	mysql ab	5747	35
7	netscape communications corp.	5249	32
8	ximian inc.	4985	30
9	realnetworks inc	4412	27
10	At&t	4286	26

**Marketvalue** To answer the question how far FLOSS got up to now - out of a financial point of view - it is referred to some facts [AAB<sup>+</sup>06]:

- Market value of all FLOSS products with acceptable quality has an equivalent commercial value of about Euro 12 billion.

## 4 Open Source Software

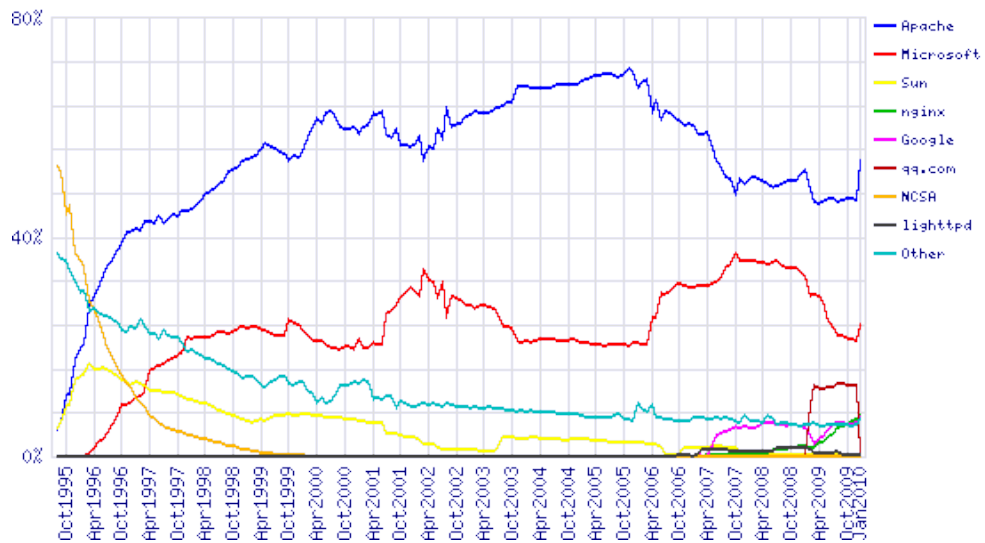


Figure 4.1: Market Share for Top Servers Across All Domains August 1995 - January 2010 [Net10]

- The lines of codes have been doubling every 18-24 months in the last 8 years.
- About 131.000 real person years is the relevant number to recreate the state of FLOSS software at the time, the study was created.

An article from the New York Times points out that the societal and strategic relevance of open source companies stand in no relation to their financial value [Van09]. To point out an example, the article uses MySQL (which is an open source database formally supported by Sun Microsystems<sup>1</sup> which has been taken over by Oracle<sup>2</sup> in January 2010) where the number of downloads (about 60.000 a day) strongly outperforms the profit, earned from support deals. The article mentions a statement by Simon Cosby, the chief technology officer at Citrix Systems<sup>3</sup>: “There’s only one company making real money out of open source, and that’s Red Hat”.

<sup>1</sup><http://at.sun.com/>

<sup>2</sup><http://www.oracle.com/index.html>

<sup>3</sup><http://www.citrix.com>

## 4.3 Characteristics

Two different initiatives stand behind the approach of developing free and open source software. Both originate from the same movement but have split up due to interest conflicts. The result of this is also expressed by the difference in the terms “open source software” and “free software” and the difference is quite difficult to see. Richard Stallmans initiative is following the idea that software should be free in the sense of “free speech not free beer” [Sta09]. His approach lies in the sense that knowledge should not be limited by barriers because only free knowledge can grow. Stallman compares this also with good science, where it is common to share knowledge. Therefore he has founded the Free Software Foundation (FSF) to follow his idea of producing free software with high quality. [Fou09] The fact that free in the sense of “for free” can lead to misunderstanding caused Eric Raymond to substitute “free software” with “open source software”. Out of this thought and also to focus more on economical points of view the OSI (Open Source Initiative) arose. Supporter of the open source initiative have always pushed the FSF into the corner of eccentricity but Stallman expresses the difference the following way: “Open source is a development methodology; free software is a social movement” [Sta09]. He also argues that the term “open source” is too weak for expressing his original idea. Now, to step over this gap, in this thesis, expressions will be used interchangeably and none of the approaches will be preferred. It is focused on the result of both initiatives: High quality software.

Although the details and approaches are different, the cornerstones of free software and most open source software can be summed up as follows [Sca07]:

- The source code has to be available for public access
- Modifications of the code have to be allowed
- Redistribution is explicitly permitted
- After a redistribution the above mentioned freedoms have to be guaranteed

The framework and the differences within FLOSS have been declared. What is even more interesting is the question, how FLOSS development differs closed source software development in respect to the question whether gained knowledge is also valid in proprietary software development. To get an introduction table 4.3 shows some concrete differences between open source and closed source software development. If specific ideas, gained within this thesis, are valid for commercial

## 4 Open Source Software

software development as well, will be handled separately in the empirical part of this thesis.

Table 4.3: Open vs. Closed Source. Refined from [Wah08]

Dimension	Closed source project	Open source project
Goal	Produce proprietary software	Produce FLOSS. Based on own needs a project arises; going public in order to get feedback and attract other developers
Actors [SSS07]	Commercial organizations, corporate developers	Volunteers, Universities, non profit organizations, commercial organizations
Customer	Internal or external	Usually at the beginning developer = customer[Sca07]
Developer	Staff	developer = customer [Sca07]
Trigger for project kick off	Needs of the customer, production plan	In most cases own needs
Physical separation of the project team	Usually small	Members spread all over the world and barely see each other face to face
Who has the administrative authority?	Project leader/central authority	No one has the authority to tell a member what to do. Members are free to decide where, when or how to participate [Sca07]
Success measure	Satisfaction, acceptance	Project health, high number of users, rankings
License	Copyright, patents for protecting knowledge and secrets of a business	Copyleft in order to protect the freedom for public access, modification and redistribution of the source code

### 4.3.1 Actors

People participating in FLOSS projects have a very distinct background. As it is also available in table 4.3 the range goes from single developer, universities,

non-profit organizations like FSF (see section 4.3) or Apache Software Foundation [Aaaa] to commercial organizations like IBM or RedHat [SSS07].

Different kind of actors can be identified but the basic differentiation deals about volunteers and corporate developers. Volunteers are those who spend their time and energy without getting paid for. They have different motivations as it is pointed out in section 4.3.2. On the other hand corporate developers do nothing else then working for FLOSS projects. They are hired by companies in order to promote or participate at projects [SSS07]. According to [AAB<sup>+</sup>06] almost 2/3 of FLOSS software is developed by volunteers, firms participate with about 15% and the 20% left come from other institutions.

#### 4.3.2 Motivation to Contribute

It was shown so far, what open source is and how the movement arose. Without the existence of a community Open Source Software Development would not work and the interesting question is, what keeps people spending their time and effort? What makes people contributing to FLOSS projects and what benefits do they have? Linus Torvald's answers this question in his book "Just for fun" [TD02] that in times where people can live quite save money is not everything and especially it is not a good motivator. Torvalds declares that different scientific studies have shown that people do their best job if they are driven by passion. They are contributing voluntarily and feel very lucky to work together with people all over the world, having the same interests and sharing respect to each other.

Logical to say, somebody contributes to something voluntarily like Open Source Software Development if the person has benefit. According to Hetmank [Het06] such reasons can be:

- Earnings for corporations
- Reputation
- For own usage
- Identification with the community
- Fun

Osterloh et al. [ORK02] introduce two categories named intrinsic and extrinsic

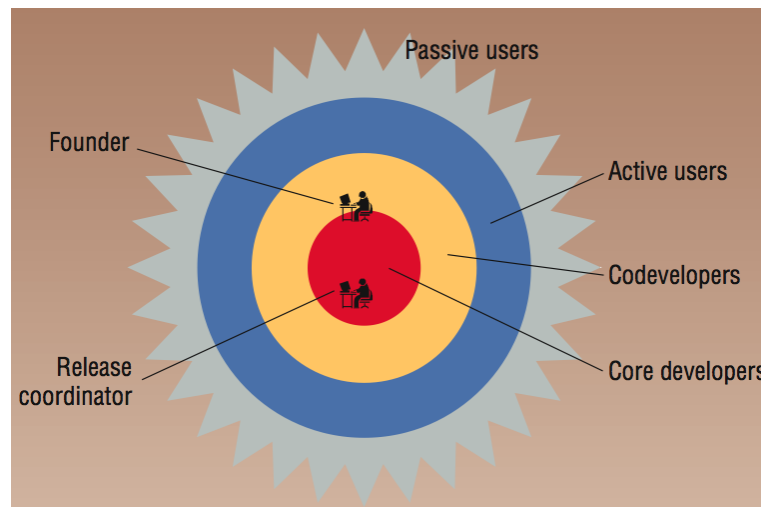


Figure 4.2: A healthy team is onion shaped with different roles [CH04]

motivators in which the named reasons above can be classified. Intrinsic motivators are those which emerge out the own inner decision. This can be own needs, identification with a community or simply the experience of fun. Is the contribution motivated by some external factors like monetary compensation or reputation we can speak of extrinsic motivators. In one aspect Torvalds’ “just for fun” is right. According to Hetmank the motivation of hobby software developer is primary intrinsic oriented but another thing is logical too: The more professional it becomes the more important extrinsic values become.

### 4.3.3 Team Structure

Open Source Software Development can be seen as an extreme form of distributed software development with the main indicator that the members barely see each other face to face. FLOSS projects have to deal with barriers in time, space and culture. To step over this gap specific development styles, techniques, communication behaviors and platforms for organization have found their place in FLOSS development.

Another difference to proprietary software development deals also with the kinds of members of a project team and the underlying structure of development

teams. A wide range of different actors participate in FLOSS [SSS07]. Crowston et al. [CH06] point out that healthy projects generally are onion shaped (see figure 4.2) with different roles as shells which define this form.

**Core developer** Positioned in the center of this shape are the core developer who have specific duties, responsibilities and privileges regarding the code repository. The founder of the project can also be found in this group but for others to get into this position, this needs to be earned. Usually this group has around 3 to 10 members. Crowston et al. [CH06] write about an example where less than 1 percent of the more than 100,000 projects managed within the platform sourceforge [sou] have more than 10 core developers which means: Projects like Linux and Apache [Aaaa] represent the minority in the FLOSS world. The core developers take the technical or organizational decisions and have full access to the code repository [Sca07]. To get into this group someone not only needs skill, time and effort but also the ability to communicate, to learn from others and needs to prove seriousness. According to Scacchi [Sca07] FLOSS developers tend to participate in more than one project at a time and a mentioned study [HO02] reports that 5% of developers work in 10 or more projects. This means, that developers sometimes know each other from earlier projects which can be helpful to get faster into the inner circle of the development team.

**Codevelopers** Following the core developers, codevelopers can be identified. Members of these group contribute by providing bug reports, are writing code, batches or bug fixes but do not have the rights to commit to the repository [Pod09]. They usually add their changes to patch files and give them to core developers who make reviews of the committed code. If the code seems to be free from defects they add it to the code base [CH06].

**Active users/Passive users** Healthy FLOSS projects have a large community of active users [CH06]. They run new software to do the testing, give feedback by writing bug reports and documentation. Active user also build a protecting shield around the developer by answering questions and giving support to passive users which are those who just use software without participating in the development process. Crowston et al. [CH06] tell that ideally this group consists of one or two long term participants and a high number of interchanging users.

### 4.3.4 Ressources

Walt Scacchi [Sca07] introduces the terms formalisms and informalisms. According to Scacchi informalisms are “the information resources and artifacts that participants use to describe, proscribe or prescribe whats happening in a FOSSD project” [Sca07]. Out of his view he puts informalisms on the opposite side of formalisms which are relevant entities like requirement specifications or design notations in the proprietary software engineering world. He mentions over 20 different types of informalisms which stand for all the information and communication channels in order to coordinate and inform what is going on in FLOSS projects. Some of this informalisms are mentioned as follows:

- Communication within Email lists
- Threaded messages like forums or blogs
- How-to-guides
- To-do-lists
- Wikis
- Architecture diagrams
- Multi-project websites like sourceforge [sou]
- Code repositories
- Bug-Issue management systems like bugzilla

Although informalisms stand in contrast to classical entities in commercial software development, out of the perspective of traceability it makes this thesis possible because in contrast to closed source projects, all this entities are available to public.

### 4.3.5 Project Management

Compared to proprietary software development the process of creating software is different and ironically expressed by Crowston et al. [CH06]: there are few projects which could apply for a ISO 9000 certificate soon. Scotto et al. [SSS07] argue that only corporate developers bring real standardization into the field of FLOSS development by applying predefined processes. While this group represents the minority, it is not said that anarchy has found the way into FLOSS



development. Different studies show that well accepted standards have spread in the FLOSS world [CH06] [SSS07] although other values which can be found in closed source software projects are at least as important as the mentioned standards.

**Conflict management** is very important for the development process of FLOSS. Experienced core developers set up their projects already with the aim of minimizing the occurrence, duration and effort for conflicts [Sca07] with the support of tools and project organizational forms. Revision systems pick up the challenge of recurring stress situations and support the total development process. By relying upon SVS systems a need for coordination has to be accepted but as a result, standardization and synchronization takes place. Scacchi also points out the potential to create an “online venue for mediating control[...]” [Sca07] over version and release decisions.

**Trust and social accountability** Crowston et al. [CH06] write that specific controlling mechanism can be found in the development process. For example, it is very common that only a limited group of developers (core developers - see section 4.3.3) have full write access to the repository. Developers from outside have to send their proposals to core developers in order to let them review (see section 4.3.6) their code. Besides that, trust and social accountability play a very important role as well [SSS07]. This soft facts are invisible resources of a project and can be found in a variety of different manner. It ranges from assuming of ownership or responsibility for a specific module to the acceptance of status or hierarchy of a core developer. A lack in this “social capital” [SSS07] would lead to conflicts like to find solutions for specific problems or more general to plan the future direction of the project.

**Organization for fun vs. organization for efficiency** According to Crowston et al. [CH06] a main task of proprietary software development is to use all resources as efficient as possible. Due to the inconsistencies in the number of project members and their inputs these aims have to be secondary in FLOSS because other values are more important. Crowsten et al. call this the tension between “organization for fun” and “organization for efficiency”. It is not possible to order voluntary members of a team, what to do or when and this fact sometimes causes redundancies or lacks in capacities. As well some duties soon

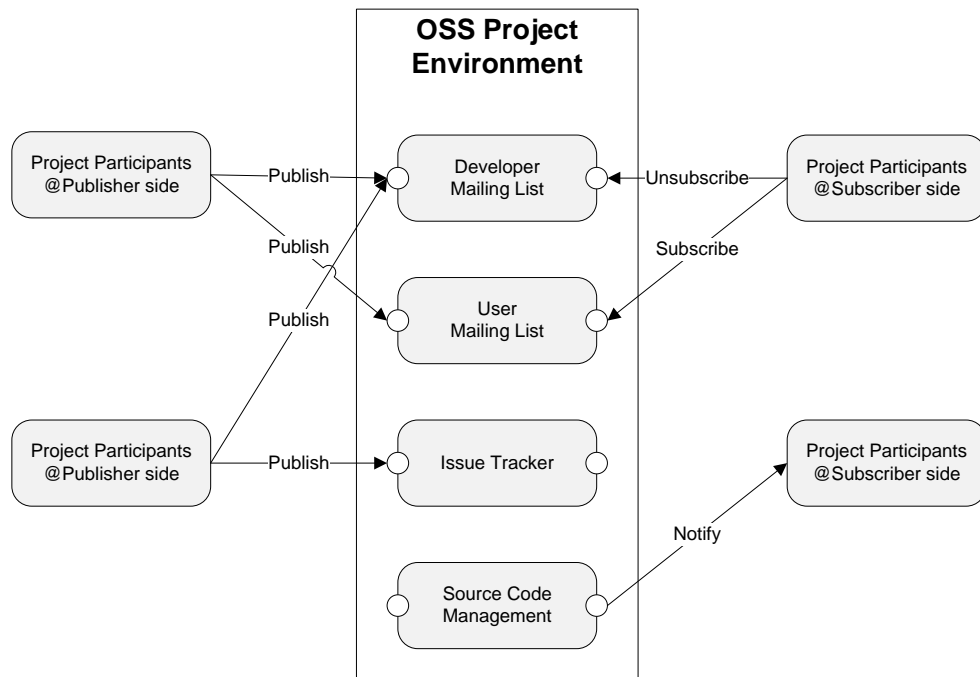


Figure 4.3: Publish-Subscribe mechanism as communication pattern in OSS projects [Wah08]

cause demotivation especially when members “[...]are driven by intellectual curiosity rather than service mentality” [CH06] because in a software development there are task to do which are not always driven by a lot of passion. According to [CH06] it can be a very big challenge to set up a release date and manage all resources that way to fulfill the aim which also sometimes means to stop being creative and do standard jobs like testing and bug fixing.

### 4.3.6 Development Process

Open source software development process can be defined by the dimensions time, place and resources. Due to the huge inconsistencies in that dimensions the development process itself has to be very loose coupled and usually uses publish/subscribe-like interaction schemas [Wah08] like it can be seen in figure 4.3. A simple process like a bug report could reveal the following

steps [Wah08]: Somebody is reporting a bug by entering the information on the publisher side into a defect tracker system (`defect_reporting_event`). This event is processed internally and initiates an instance which informs the people who are subscribed (`new_defect_notification_event`) through a specified information channel (which can be a mailing list). After the processing the responsible person fires another event by replying with feedback to the tracker system (`defect_diagnoses_event`). By tracing this communication behavior it is possible to get an overview about the composition of a development team which is used as approach in this thesis.

Wahyudin describes [Wah08] some standard patterns which can be found in FLOSS projects. Basically, the aim of all concepts is **continuous improvement** and how this process of each release cycle could look like can be seen in figure 4.4. The figure differentiates between three different views and illustrates five basic concepts in FLOSS development which read as follows:

**Design review** The beginning of this cycle can be seen in the design review subprocess and what happens in there is mostly decided on the basis of information out of the tracker tool. Such change requests in the tool are either requests for new features or requests for enhancements. Before inserting a new issue a developer usually creates some specification and tries to evaluate this with other developers (see circle 1 in figure 4.4). Accepted tasks are added to the tracker tool or the developer himself starts to implement a solution and publishes it in order to get the code reviewed.

**Coding and testing** After the design, the second sub circle (see circle 2 in figure 4.4) follows with coding and testing. Generally, the information what should be implemented comes again out of the issue tracker tool. If any troubles in the testing phase do appear, the issue can be set to status “open” and goes back to the tracker tool, together with information which have been collected during development. Out of the perspective of this thesis information about the underlying project team can be gained by analyzing mailing lists which are used for coordination but also extracting information out of revision systems like CVS or Subversion.

## 4 Open Source Software

**Code peer review** Is an issue ready for peer review it is marked as “resolved” [Wah08]. Other developers review the code and decide whether the changes can be added to the next release or have to go back to coding and testing in order to clarify left doubts (see circle 3 in figure 4.4).

**Release and defect validation** After the changes have been added to the new release, the user community takes up work. By using the new software they implicitly or explicitly test (whitebox testing - see circle 4 in figure 4.4) the new release and in the case of detecting problems they provide feedback to the developer by creating a defect report. The developer uses the information from the user community in order to understand the problem. With black box tests (see circle 5 in figure 4.4) he tries to detect the specific problem and if the problem has proven as valid the developer himself starts to correct the bug or opens a new issue in the tracker system.

### 4.3 Characteristics

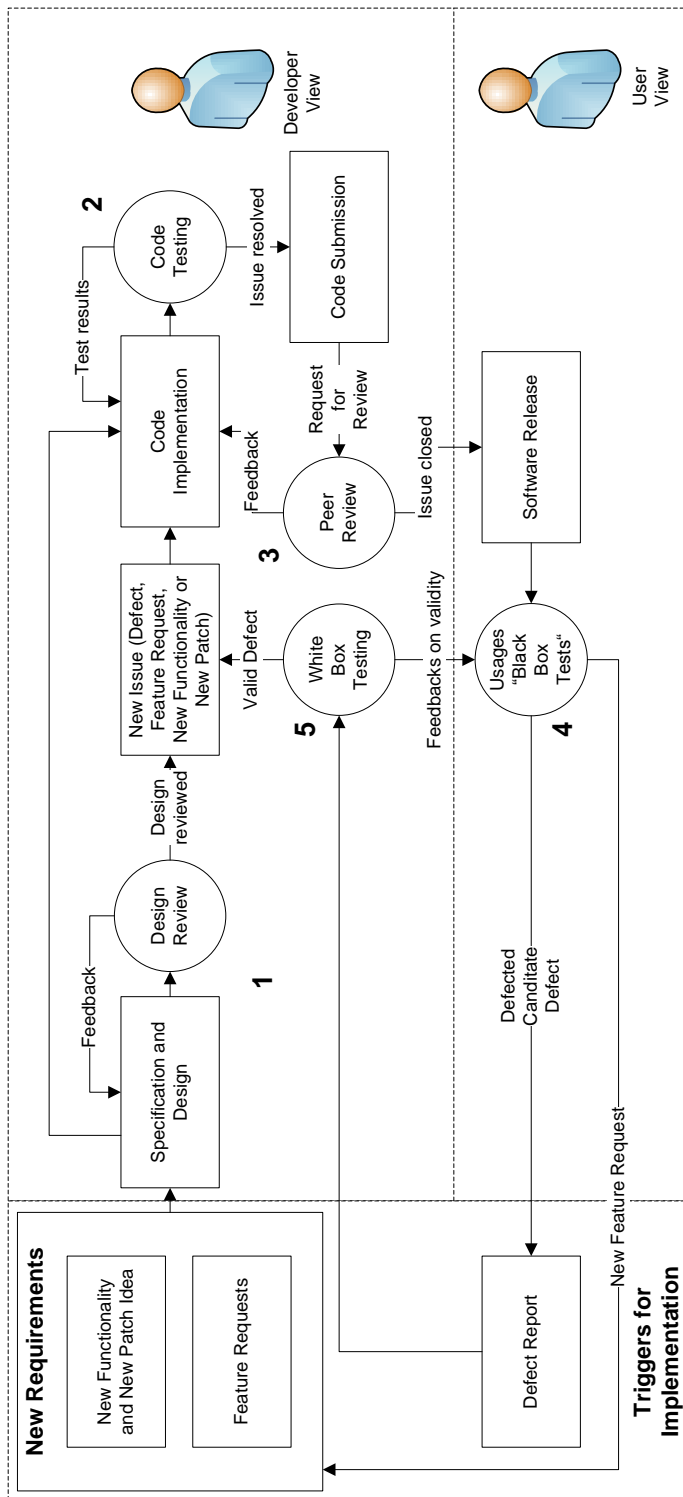


Figure 4.4: Continuous software product improvement within an OSS project refined from [Wah08]

## 4 *Open Source Software*

## 5 Project Health

The importance of FLOSS has already been shown in the previous part of this thesis. In some areas open source has become a real counterpart to proprietary software products. From an outside point of view, closed source projects provide different quality evaluation methods compared to open source projects. Furthermore closed source projects usually have the constriction to provide only a view to the product. It is not possible to look at the underlying process of software development which is a main difference to open software projects. It was shown so far, how FLOSS projects are usually set up and who are the different contributors in the development process. In this chapter, it is focused on the development process with the aim to provide indicators for a project's health status which furthermore guarantees the survival of the project.

**Project health** is defined in the context of this thesis as a qualitative characteristic for stability and continuity on the basis of the underlying development process and community (see also [Wah08], [WSM<sup>+</sup>06] and [CH06]).

Just because it is possible to look at the underlying process this is not reason enough to focus on the processes in this thesis. Due to the information given in the previous chapter it should be possible to understand, why the success of FLOSS projects strongly relies on the community. For the success of this thesis, it is furthermore hypothesized that only a healthy community can produce high quality software [WSM<sup>+</sup>06].

What keeps projects alive is the result of a variety of factors and due to the complexity this question cannot be easily answered. As it was already pointed out, FLOSS projects have similar approaches but how they are managed in detail differs from project to project and can change over time as well. A health analysis should support stakeholder to get an overview of the current status and future development of software projects.

## 5.1 Why Project Health?

This chapter will be opened with a quote by Michael Olson, CEO of Cloudera<sup>1</sup>, which is an open source startup: “A lot of open source firms are one-product companies, and it’s hard to build a long-term, successful business that way” [Van09]. For users or investors in open source projects it is essential to know about the health condition of a project in order to create a reliable basis for own efforts. To provide an instrument which is able to differentiate between healthy and sick projects the project health approach is introduced in this thesis.

Out of a specific view a software project can be compared with a sensitive organism. A lot of different factors influence the health status of this organism and if the “immune system” is not strong enough, a project can become sick what sometimes can even lead to death (see life cycle in figure 5.1). In the first instance it is important to be able to separate between healthy or sick projects. Furthermore health indicators should make it possible to make prognoses about the future conditions of a project by evaluating the current health status of a FLOSS project.

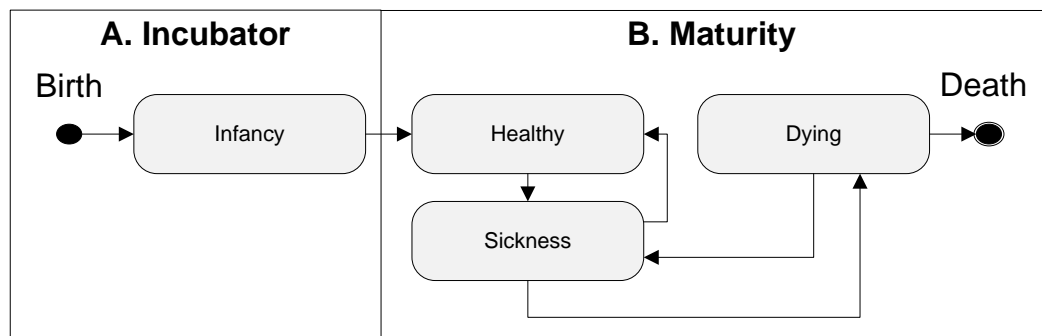


Figure 5.1: Open Source Project Lifecycle refined from [Wah08]

<sup>1</sup><http://www.cloudera.com/>



## 5.2 Project Lifecycle

Wahyudin [WSM+06] shows a classical project lifecycle (see figure 5.1) of open source software projects. Basically it can be differed between four states which are infancy, healthy, sickness and dying.

**Infancy** Before initially going public a code base is typically developed by a single developer or a very small group [CH06]. Platforms like Sourceforge [sou] have their own approaches which should support new projects in order to reach maturity. The Apache Software Foundation (ASF) [Aaaa] calls their environment Apache Incubator [Apab] which has the aim to support podlinds (according to [Aaaa] it is the codebase and it's community during the incubation process) on their way in becoming a member of the ASF family. The main aim in the incubation phase is to attract more and more participants and to create a healthy basis of the project (like in the case of Apache Incubator).

Croston et al. [CH06] mention, that projects with an atmosphere of intellectual engagement and exploration early in the lifecycle seem to be very interesting for a user community. This projects are most likely to grow up and to be able to make the step towards maturity.

**Healthy** After leaving nursery the project reaches it's maturity phase. The project has to compete against other projects and has to attract a user and developer base in order to stay healthy. This can be seen as a circuit because healthy projects usually attract publicity and due to publicity, other users and developers join. Nevertheless according to a study by Wahyudin [WSM+06] less then 2% of the projects listed in sourceforge reach their maturity phase (see diagram 5.2).

**Sickness and dying** According to figure 5.1, a project status can turn from healthy to sick or even death. Reasons for this developments can be the following:

- Decreasing motivation of a community due to rising conflicts [Sca07]
- Poor project management
- No or late feedback demotivates the user community

## 5 Project Health

- Loss of core developers in a very centralized project (see section 7.5)
- Lack of acceptance of project leaders within a project [Sca07]
- Technical revolutions which make the project redundant

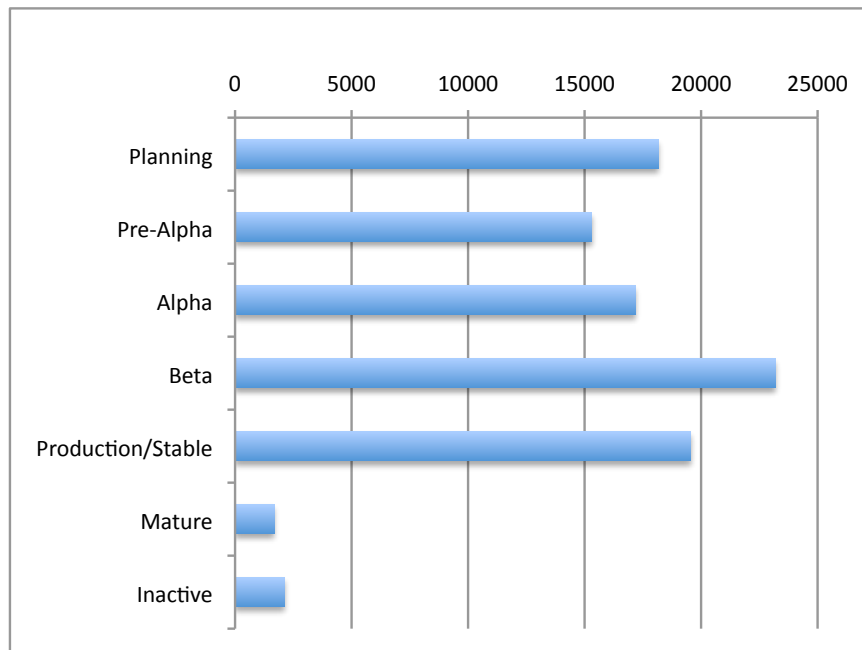


Figure 5.2: Sourceforge - projects' degree of maturity in relation to the number of projects refined from [Wah08]

## 5.3 Health Indicators

The points mentioned above are the reasons for change in health status. This part of the thesis deals with indicators, which should identify problems in time in order to make it possible for stakeholders but also for project management to get aware of a critical situation.

### 5.3.1 Defining the Shape

Experts, analyzing FLOSS projects take a variety of indicators into consideration. To get a general overview of an underlying project it is necessary to look for a framework. Such frameworks deal with the shape and the development process of the underlying FLOSS development team. Crowston et al. advise to consider the following points [CH04]:

**Size of the project team** In order to get the number of project members some criteria need to be defined because in FLOSS projects there is usually no official document available which describes all active participants. Due to transparency of the underlying processes, the question, who has contributed what and to which time can be traced back easily. With informalisms (see section 4.3.4) like mailinglists, defect tracker tools and versioning systems which are available to public and contain the historically output of the process, the relevant information can be extracted.

In section 4.3.3 different roles in a FLOSS project were described which contained core developers, codevelopers and active/passive users. Whereas in healthy projects, the core group should stay stable, the fluctuation in the groups codevelopers and active/passive users is usually very high.

**Tendency in change of size** Based on the last point, another indicator for project health is the tendency, in which direction the size of a project team is changing. To be able to see the change, data over a period of time need to be researched. Already mentioned in this thesis, it can be seen as a circuit: The more publicity a project produces, the more users and developers get attracted which can also be valid vice versa. Losing users can also be a signal for a dying project.

**Assignment of responsibilities** Already introduced in section 4.3.3 healthy projects are generally onion shaped with core developers (consisting of founder and roles like release coordinator) in the centre. According to Crowston et al. [CH04] it needs to be asked, how responsibilities within a project team are assigned. Are there subgroups who are specialized or is everybody doing everything? Are there gatekeepers who centralize communication channels or is

it a very diverse team with balanced, equal contributors? This questions deal with the social structure of the team. To provide a view to this structure Social Network Analysis (see chapter 7) will be used in this thesis.

### 5.3.2 The Cathedral and the Bazaar

One of the most famous models in FLOSS development was created by Eric Raymond and was published under the name “The Cathedral and the Bazaar” [Ray00]. Raymond identified two different approaches in FLOSS development which he called “Cathedral” and “Bazaar”. Before he started to research the issue of FLOSS he believed that most of the important projects had the structure of cathedrals: Step by step build out of the hand of some “individual wizards” or small groups of “mages working in splendid isolation, with no beta to be released before its time” [Ray00]. Linux although showed him a complete different approach which Raymond called “a great babbling bazaar of differing agendas and approaches” and out of his view it was a curiosity that the output of this development style could be that coherent and stable.

After exploring the bazaar style with the example of Linux (release early and often, delegate everything and be open to the point of promiscuity) he had to revise his opinion and started to look for reasons, why that type of project management can be successful. On the basis of a mail client called Fetchmail [BW] he started a research project whether it is possible to guide a project towards bazaar style. Out of this experiment he gained different insights but what is important for this thesis lies in the assumption, that FLOSS projects usually start as cathedral and then reach the status of a bazaar [CH06]. The developer core gets bigger and is surrounded by a diverse set of codevelopers and active/passive users. What the impact of this fact is or how it can be identified will be answered in chapter 7.

### 5.3.3 High Dependency - High Risk?

The work of Krishnamurthy [Kri02] shows, that from the top 100 mature projects available at sourceforge [sou] only a minority has reached the bazaar status. The median number of developers, participating in one project was 4 but does that mean, these projects are unhealthy?

### 5.3 Health Indicators

This question is rather difficult to answer because out of the information provided so far, it needs a certain period of time to reach maturity and also Krishnamurthy found in his study a positive correlation between time of existence and number of developers in the project. These projects can be on a good way towards maturity but the fact, which can not be ignored is a higher risk of sickness or death of the project by relying on a weak fundament. Speaking the language of Social Network Analysis (see chapter 7), these projects show a high centrality that finally results in a high dependency on a small set of actors. Appearing problems like a sudden exit of an actor can have high impact on a project. Summing up, maturity should also contain a decline of the risk to fail by reducing the dependency on certain actors and getting a broader core developer group. Importantly, as stated by Crowston et al. [CH06] certain actors like founders are always crucial for a project and even if they already dropped out of the project, their opinion still counts certain value on the project progress. Resulting of this, a transition of leadership from one actor (which can also be the founder of a project) to another one is always a challenge for a project. Successfully managed transitions are according to Crowston et al. a positive sign, but in the same breath they point out the fact that high dependency of certain leaders provides a risk for the whole project. Transitions in such a case are very critical for a project.

## 5 *Project Health*

## 6 Research Questions

In chapter 4 an introduction into the basics of FLOSS was given. It was shown, how a typical development process looks like and the different roles of a project team were introduced. The previous chapter contained the concept of project health and answers questions about how healthy projects develop and tells something about the different health conditions of a project. As a linking element between these two concepts Social Network Analysis will be introduced in the next chapter. Which questions should be answered in this diploma thesis can be described as follows:

**H1:** Healthy projects develop an onion shaped structure which can be identified using Social Network Analysis

The first question deals with the structure of an underlying project team. How healthy projects are usually set up was described in chapter 5. Information out of different software projects will be extracted and analyzed with the help of Social Network Analysis. It will be tried to identify this onion shape of healthy projects using the footprints left by developers during development. The different roles of a software project should tried to be identified with the instruments provided with Social Network Analysis in order to foresee any possibility of observing this mentioned structure in healthy projects.

**H2:** Centrality decreases with time of existing of a project

Passing time the centrality (see section 7.5) of healthy projects decreases. Using Social Network Analysis it will be tried to identify this pattern in healthy or sick projects. It will also be researched whether an increase of this indicator is a signal for sickness of a certain project.

**H3:** Relation between high centrality and project failure

## *6 Research Questions*

The last question contains kind of a risk analysis and tries to answer the question if a generally high centrality indicator stands in relation to a higher risk to fail. It will tried to research the drop out of a certain very central actor in order to see the effect on the underlying network. High centrality in this case would mean, that the rest of the network will not be possible to handle that situation.



# 7 Social Network Analysis

Social Network Analysis represents the core concept of this thesis. It is used as a technical method to research given Open Source Software Projects in order to gain the information which is necessary to get answers to the research questions in this thesis. The following chapter basically relies on the work of Stanley Wasserman and Katherine Faust [WF94].

Analyzing complex systems is a very challenging task and the words of the physicist Philip Anderson used by Clay Shirky in his book “Here comes everybody” [Shi08] illustrate the situation accurately: “more is different” [And72]. Anderson continues to argue, that it is not possible to predict developments of entities like atoms by aggregating their constituent parts. It is hardly possible to make rational interpretations about future development of any system by using information only based on individuals and simply adding up information. Anderson declares, that “chemistry isn’t just applied physics” because it is not possible to understand the properties of water by focusing on the characteristics of its isolated atoms. Neither “Sociology is just psychology applied to individuals” because the explanations of Shirky’s individuals within the group show behavior that no one could predict by researching the isolated person. A person is not extroverted as long as it is sitting alone in a room. According to Shirky a group is not just the sum of individuals.

Social Network Analysis allows to keep that view from above and provides instruments, methods and approaches, relevant for the needs of this thesis.

## 7.1 Introduction

A formal definition of a social network by Wasserman et al. [WF94] reads as follows:

## 7 Social Network Analysis

“The concept of a network emphasizes the fact that each individual has ties to other individuals, each of whom in turn is tied to a few, some, or many others, and so on. The phrase “social network” refers to the set of actors and the ties among them.”

As it is also defined by Jamali and Abolhassani [JA06], a social network is a set of entities which is connected by a given social relation. As an example, this relation can be friendship, relationship or communication behavior in the field of human social science and a possible entity can be an individual. The instrument Social Network Analysis itself is not limited to this research area. In fact it provides potential to be used in different fields like economics, marketing, and like in this thesis as well, information technology, to research relationship patterns between entities.

When social network analysts deal with a certain problem, they try to illustrate the dependencies of the given context in a model in order to detect the relationships. With the help of certain indicators which will be introduced in this chapter it is possible to research the composition of the underlying network. Such indicators can answer questions like to which extend a project depends on individual members (see centrality indicators in section 7.5). As Wasserman et al. [WF94] point out, the next logical step would be to study the impact of the individual in the group and/or to study the influence of the whole to each individual within the group. To illustrate this theory more visible Wasserman et al. draw the following example:

**Example** The World’s economic system with its different actors (states) represents an incredible large and complex social network with observable variables (relations) like trade, loans or foreign investments. Social network analysts can now try to find regularities or patterns in the system, with the aim to understand the characteristics of each nation, for example the economic development, by describing the location of a nation within the economic system.

**Factor time** Social Network Analysis is not only used as a possibility to analyze current states of given problematics. One other important approach is to look at changes in social network over a period of time. Adding the “factor time” into the “World’s economic system” example (see 7.1), makes it possible to research changes in the economic system by drawing the social network for each period

of time and comparing them.

## 7.2 Development of Social Network Analysis

Guy et al. [GJS<sup>+</sup>08] declare that in the beginning social science tools like surveys and interviews have been used as a fundament for Social Network Analysis. Due to progress and introduction of the internet, the virtual world has become a playground as well.

Scott [Sco00] writes, that different roots and strands have led to the methods we today know as Social Network Analysis. Passing time these strands have forged and spread, have influenced each other as well and because of this, it is difficult to draw exact lines in the lineage of Social Network Analysis. Today we can say, that there have been different motivations in different teams, place and also time, which have led to the development of Social Network Analysis and this is the reason for the interwoven lineage of Social Network Analysis.

The method itself is not that new. Parts of the theoretical fundament have been introduced already in the 1930s and basically we can say that there have been three main strands (see figure 7.1).

**Socioeconomic analysts** The Sociometric Analysts existed of small teams and made important progress in the field of graph theory. Originally this strand was dealing with Gestalt tradition<sup>1</sup> which was mostly contributed to Wolfgang Köhler [Köh30]. Jacob Moreno was highly influenced by Gestalt theory and primarily it is due to him using a sociogram (like in figure 7.2) as a way to illustrate social configurations. This approach had huge impact and was even reported in the New York Times in 1933. The vocabulary has already been existing before Jacob Moreno had his ground-breaking ideas. People were talking about webs, the social fabric and sometimes also of networks and relations, but it was him putting the information into a sociogram where for example edges stand for different humans and vertices illustrate the information flow between them (see figure 7.2). Another group around LLoyd Warner and Elton Mayo

---

<sup>1</sup>Gestalt tradition is a topic of psychology and deals with distinct perception and thoughts of entities and it's constituent parts [Sco00].

## 7 Social Network Analysis

was dealing with interpersonal relations. Theoretical research during the 1930s and 1940s was dealing with composition of networks by constituent sub-groups. They had the aim to use any relational data they could obtain in order to find techniques which makes it possible to take apart any social system into its parts. This goal could not be achieved totally.

**Manchester Anthropologists** Max Gluckman and later Clyde Mitchel were the central figures in the line of the Manchester Anthropologists and made eminent contribution to the development of Social Network Analysis. Gluckman was investigating social structures (“communities”) in different parts on earth. They were looking for an approach to determine the central element which is responsible for conflict and power in maintenance and transformation.

**Harvard researchers** It was in Harvard, where the real breakthrough was happening. In fact two parallel mathematical innovations have led to this success.

- The development of an algebraic model for describing kinship and other relations using set theory<sup>2</sup>
- A multidimensional scaling technique for mapping relationships to social “distances” in a social space

A group around Harrison White was using these ideas and Social Network Analysis as a method for structural analysis was born.

**Classical Social Network Analysis vs. Social Network Analysis in software development** As it was just pointed out, Social Network Analysis has a long history which goes back to an era where computers have not been invented yet. The offspring can be found in the area of social and behavioral science who were one of the first profiting from Social Network Analysis by using the new perspective in order to research specific patterns. Clever researchers soon noticed the potential of this concept in the field of software development in order to analyze the composition of software development teams [WF94] [Fre79].

---

<sup>2</sup>Set theory is a mathematical method, dealing with the infinite and studies properties of sets, which are abstract objects [Jec02].

7.2 Development of Social Network Analysis

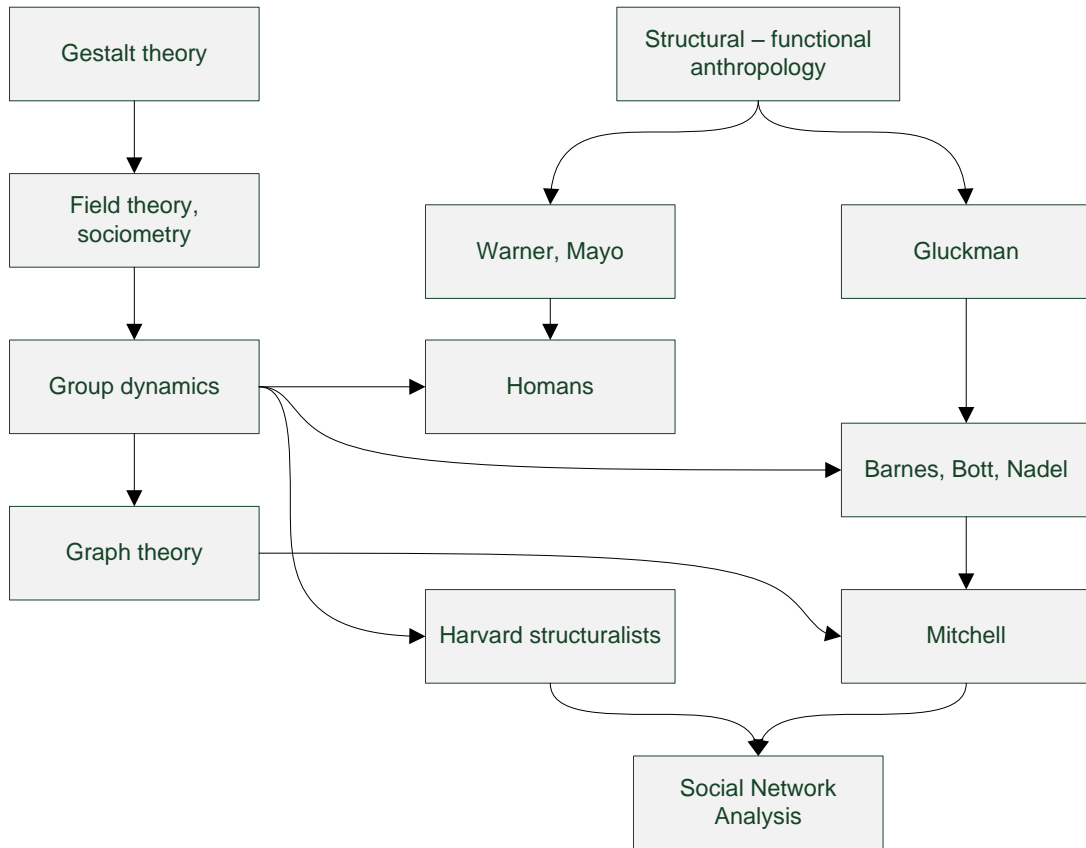


Figure 7.1: The lineage of social network analysis

## 7.3 Key Concepts

This section provides a brief introduction into the vocabulary which is used in Social Network Analysis. Basically it is orientated on the information given in the work of Stanley Wasserman and Katherine Faust [WF94] which can be seen as a reference book in the area of Social Network Analysis.

**Actor** Possible actors can be people in a group, departments within a corporation or nations in the world. We speak of one-mode networks when all actors are of the same type (e.g. people within a group) which is the common usage. Although Wasserman et al. also mention other scientific areas where it makes sense to accept actors of different kind (researching community members contacts with public service agencies by Doreian (1990)).

**Relational Tie** A set of connections of a specific kind is called a relation. Only the focus, how we look at the group, defines a specific relation because relations are not limited to a dimension. In a group of people we can identify examples like friendship, kinship or labour relations as different relations where not everybody has to be involved.

The idea of relational ties is, that they are representative for the bond between a pair of actors. Dimensions associated with relational ties can be quite different. Beginning with the example introduced in 7.1 where the relational tie stands for trade between nations, we can find a relational tie in the communication behaviour of an Open Source Software Projects what used in this diploma thesis. Relationships in families, physical connections (roads, bridges) which together with cities as actors describe a map are further example for this linking entity.

**Subgraph** A graph  $G_s$  is subgraph of Graph  $G$  if

- the set of nodes  $N_s$  of graph  $G_s$  is subset of the set of nodes  $N$  of graph  $G$
- the set of lines of graph  $G_s$  is a subset of the set of lines of graph  $G$
- mathematically expressed  $N_s \subseteq N$  and  $L_s \subseteq L$  is valid.

**Dyad** A dyad is a subgraph and consists of a pair of nodes connected by a tie or not. This means that an unordered pair of nodes can take up two possible dyadic states: Two nodes are adjacent or they are not adjacent.

**Group** Wasserman et al. [WF94] use a definition of a group that reads as follows: “For our purposes, a group is the collection of all actors on which ties are to be measured. One must be able to argue by theoretical, empirical, or conceptual criteria that the actors in the group belong together in a more or less bounded set.” They define a group simply with all members which are connected by relational ties.

**Social Network** With the formal definitions given so far, a social network can be seen as a finite set of actors and the relation defining their linkage.

## 7.4 Data Management and Illustration

Depending on the aim of how the gained data will be further used, it is possible to specify two different approaches for managing data, used for Social Network Analysis. For storage or further statistical research data can be organized in matrices. Another possibility is using graphs as a method for drawing the social network and illustrating the dependencies. In the following section a brief introduction into matrices and graphs is given, as they are used in Social Network Analysis.

### 7.4.1 Graphs

Wasserman et al. [WF94] describe graph theory with its vocabulary and methods for modeling social structure properties as a multifunctional and useful part of the methods of Social Network Analysis. Graph theory provides as well a mathematical approach for quantifying and measuring relations. As already mentioned in section 7.2 Jacob Moreno was one of the first using graphs as a method for detecting social properties. These graphs are called **sociograms** if they illustrate the social relationship between humans.

**Undirected dichotomous relations** Graphs provide the possibility to model social networks with relations that do not have a direction. As an example we can think of kinship or neighborhood as relation. If a person A is related to a person B, B is also related to person A and resulting of this, a relation between two actors exists or does not exist (there is no other possibility). Graphs consist of a set of nodes (are called vertices in graph theory - representing the actors)  $N = \{n_1, n_2, \dots, n_g\}$  and lines (are called edges in graph theory - representing the relation between two actors)  $L = \{l_1, l_2, \dots, l_L\}$  which connect the nodes. A line is represented by an unordered pair of distinct nodes (no self relation) and is annotated in the following form:  $l_k = (n_i, n_j)$ . Two actors  $(n_1, n_5)$  are adjacent if the set of nodes contains a line  $l_k = (n_1, n_5)$ .

**Directed relations** Relations can also have a direction. Wasserman et al. [WF94] present an example where friendship of 6 children are used to explain directional relations. When one kid considers another as a friend it is not given that the relation in the other direction is right as well.

These kinds of relations can be represented by directed graphs. Like undirected relations, we have as well two kinds of elements. Digraphs exist of a set of nodes  $N = \{n_1, n_2, \dots, n_g\}$  and a set of arcs  $L = \{l_1, l_2, \dots, l_L\}$  but as a difference to undirected graphs, arcs consist of an ordered pair of nodes and are annotated the following way:  $l_k = \langle n_i, n_j \rangle$ . Back to our “friend of” relationship an arc  $l_1 = \langle n_1, n_2 \rangle$  would mean, that out of the view of  $n_1, n_2$  would be considered as a friend but this relation is not valid vice versa. By changing the direction of the arc the knowledge base would change as well. Summing up, we have indegrees (incoming relations) and outdegrees (outgoing relations) that are connecting a pair of nodes.

### 7.4.2 Matrices

Basically matrices and graphs contain the same information but differentiate in the way how they organize it. A matrix is a compact method for managing information about relations in a social network. If the network, we are watching at is very large other graphical methods like graphs (see section 7.4.1) soon become very complex and the extraction of relevant information becomes very difficult. Managing social structural properties in matrices makes it more useful for fur-



## 7.4 Data Management and Illustration

X	$n_1$	$n_2$	$n_3$	$n_4$	$n_5$	$n_6$
$n_1$	-	0	0	0	1	1
$n_2$	0	-	1	0	0	0
$n_3$	0	1	-	0	0	0
$n_4$	0	0	0	-	1	1
$n_5$	1	0	0	1	-	1
$n_6$	1	0	0	1	1	-

Table 7.1: Example of a sociomatrix: relation “lives near” [WF94]

X	$l_1$	$l_2$	$l_3$	$l_4$	$l_5$	$l_6$
$n_1$	1	1	0	0	0	0
$n_2$	0	0	1	0	0	0
$n_3$	0	0	1	0	0	0
$n_4$	0	0	0	1	1	0
$n_5$	1	0	0	1	0	1
$n_6$	0	1	0	0	1	1

Table 7.2: Example of an incidence matrix: relation “lives near” [WF94]

ther calculations with computers where as well a lot of interfaces use matrices for exchanging information. The following two different kinds of matrices are used for Social Network Analysis [WF94].

**The Sociomatrix** The lines and columns of sociomatrices or adjacency matrices represent the different nodes of the social network. Each node has it’s own row and column and a binary system represents, if there is a connection between two nodes or not. As we have seen it in section 7.4.1 we have to differentiate between directional or nondirectional relations. The sociomatrix of a undirected relationship is symmetric, which means, that all elements or cells of a matrix are mirrored on the main diagonal (see table 7.1) There is a connection between node  $n_i$  and  $n_j$  if the cell  $x_{ij}$  or  $x_{ji} = 1$ . Cell  $x_{51}$  is representative for the available relation between node  $n_5$  and  $n_1$  because of the existence of a 1 in cell  $x_{51}$ .

**The Incidence Matrix** Adjacency matrices have for each node a column and a row and a binary system describes the social network. However incidence matrices have for each node a row and for each line a separate column, as it can be seen in table 7.2. The matrix has the size  $gXL$  for  $g$  nodes and  $L$  lines. Two nodes  $n_1, n_5$  are incident if they have a 1 in one column as it can be found in column  $l_1$ .

## 7.5 Indicators measuring Centrality or Prestige

Social Network Analysis not only provides methods for illustrating social relationships. In fact it is important to differentiate between more or less important members of a network or compare characteristics of different networks and therefore Social Network Analysis provides methods to calculate different indicators. For this use, the next section introduces the different indicators, used in this thesis.

Generally we can differ between levels of aggregation. Analysis can be done on the level of actors. Furthermore, more aggregated, groups or networks can be researched. Some indicators base on undirected relationships (see section 7.5.1, 7.5.2 and 7.5.3), others need a special form of directed relationships as the notion of prestige (see section 7.5.7).

The Linton Freeman study [Fre79] about centrality and the difference between local and global centrality is considered to be fundamental in the area of Social Network Analysis. He adopted ideas which have their roots at Bavelas, who was the first, introducing centrality as a measure for importance [Bav48]. Bavelas was hypothesizing a relationship between centrality and the influence in group processes.

**Centrality vs. Prestige** Knoke and Burt [KB83] have introduced a differentiation of importance or prominence. They have defined centrality and prestige as subcategories. The concept of centrality is one representation of importance and differentiates between two approaches. The regular kind of centrality measure bases on nondirectional relations. However this measures can be adopted to use directed relations as well but to create a difference to prestige, directed centrality measures solely focus on outgoing degrees. A prestigious actor is defined by

## 7.5 Indicators measuring Centrality or Prestige

Wasserman et al. as an actor who “is the object of extensive ties, thus focusing solely on the actor as a recipient.” [WF94].

Freeman points out a potential for misunderstanding in his essay and introduces a consistent vocabulary. In the following part of this thesis in each centrality measure will be differentiated between point centrality (local) and graph centrality (global). For readers who are interested in more detailed information also including the derivations of the models in detail, it is referred to Linton Freeman’s groundbreaking study about centrality [Fre79].

### 7.5.1 Degree Centrality

Important or prominent actors are those who are highly connected (or highly visible) to other members of the social network [WF94]. In this case, there is no differentiation between seeing (being the source of a relation) or being seen (being the destination of a relation), only the “if” counts. A high actor centrality declares a high integration into the network and resulting of this, being a part of many relations. The indicator itself can reach a value between zero and one where one represents high centrality.

**Point centrality** Freeman describes the star or wheel sociogram as the origin of actor centrality. A central actor who is surrounded by other actors, as it is available in figure 7.2 has the highest possible degree (number of relations to other actors) and resulting of this approach we have at the same time the “most intuitively obvious conception” [Fre79] for centrality. Centrality is some function of the degree of an actor, or using Freeman’s vocabulary a point  $p_i$  is a count of all adjacent points  $p_j (i \neq j)$ .

The notation for degree centrality that was mentioned by Freeman [Fre79] and defined by Nieminen reads as follows:

$$C_D(p_k) = \sum_j^n a(p_i, p_k) \quad (7.1)$$

where

## 7 Social Network Analysis

$a(p_i, p_k) = 1$  if  $p_i$  and  $p_k$  are connected by a line, otherwise.

Different network sizes have an impact on this measure. To prevent inaccurate measures, this indicator needs to be normalized. Freeman describes the following notation as independent to the size of the network:

$$C'_D(p_k) = \frac{\sum_j^n a(p_i, p_k)}{(n-1)} \quad (7.2)$$

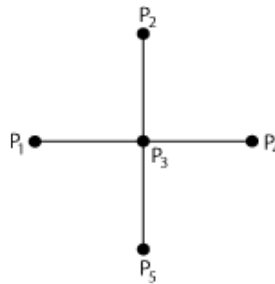


Figure 7.2: Sociogram: star or wheel with five points [Fre79]

**Graph centrality** Centrality is not consistently defined when we move on to total graphs. Freeman describes in his fundamental study [Fre79] that there has been a running controversy over 25 years concerning the sense of this phrase. Probably the parties have arranged this dispute meanwhile. Wasserman et al. [WF94] are already using another vocabulary though it will be continued with the notation of Freeman.

Freeman tries to illustrate that different scientists who have dealt with Social Network Analysis have applied at least the same approach however they have used other names for their indicators which represent the compactness of graphs. Summing up the basis of the different approaches, Freeman defines compactness the following way: “A graph is compact to the degree that the distances between pairs of its points are short.” [Fre79] Basically this indicator is built on the same idea as it was shown in the paragraph above (see 7.5.1) and the idea was only transferred to total graphs. Freeman points out some facts that graph centrality numbers should care about:

## 7.5 Indicators measuring Centrality or Prestige

- They should look at “[...] the degree to which the centrality of the most central point exceeds the centrality of all other points [...]”
- This should express “[...]a ratio of the excess to its maximum possible value for a graph containing the observed number of points.”

The general formula for calculating the centralization index reads as follows [Fre79]:

$$C_X = \frac{\sum_{i=1}^n [C_X(p^*) - C_X(p_i)]}{\max \sum_{i=1}^n [C_X(p^*) - C_X(p_i)]} \quad (7.3)$$

where

$n$  = number of points

$C_X(p_i)$  = centrality of point  $p_i$

$C_X(p^*)$  = highest value of point centrality in the whole network

$\max \sum_{i=1}^n [C_X(p^*) - C_X(p_i)]$  = the maximum possible sum of differences in the point centrality for a graph of  $n$  points

The graph centrality  $C_X$  can be interpreted as the ratio between the summated differences of the point with the highest degree centrality and all other nodes in the numerator, and the highest possible difference between the node with the highest centrality and all other nodes for a graph with  $n$  nodes in the denominator. To understand this we need to go back to the ideas, this theory bases on. The star or wheel sociogram has always been crucial to this approach because only this arrangement provides this requirement. The centrality of such a graph is used to normalize this indicator to an area between zero for low centralization and one for high centralization.

As a next step the denominator can be resolved and then be calculated directly. By doing this the indicator graph centrality  $C_D$  can adopt the following form [Fre79]:

$$C_D = \frac{\sum_{i=1}^n [C_D(p^*) - C_D(p_i)]}{n^2 - 3n + 2} \quad (7.4)$$

**Differentiation** According to it’s approach, degree centrality only focuses on local centralities which is relevant for studying the popularity or activity of actors [HWC06]. The problem degree centrality has to deal with lies in the fact, that only the neighbors of a node are considered for the calculation of this measure. If a certain group is separated from the rest of the network an actor

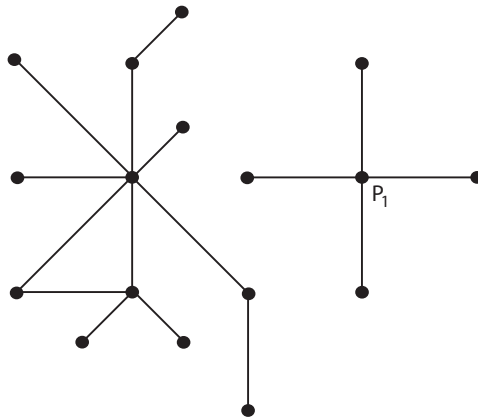


Figure 7.3: Degree centrality ignoring the total connectivity

positioned central within this group would have a high degree centrality like it can be seen in figure 7.3. The fact, that there is no connection to the rest of the network would be ignored. Out of this reason [HWC06] advise to use other indicators as well because of the mentioned lack of clarity.

### 7.5.2 Closeness Centrality

The second centrality measure looks at the degree of a point to which it is separated to all other points in the network. Freeman uses again the example of the star in order to explain closeness centrality [Fre79]. By looking at figure 7.2 we can see, that the distance of  $P_3$  to all other nodes is one, however all other points need at most two steps to reach all other points. Resulting of this  $P_3$  is closest to all other points in the graph. The idea of this approach is the question how dependent an actor is. When we look at communication as an example for a social network, information needs to pass different stations to reach an actor. The longer the way of one to another actor is the more an actor has to rely on other actors in the network [Fre79]. This approach was introduced by Bavelas [Bav50] and Leavitt [Lea51]. Wasserman et al. [WF94] also mention Beauchamp [Bea65] who found out, that a central actor in the sense of closeness centrality who is aware of the special position in the network can have a strong positive influence on information flows where on the other hand economic considerations become

## 7.5 Indicators measuring Centrality or Prestige

a topic. Again this indicator ranges from zero to one and a value closer to one represents high centrality.

**Point centrality** Freeman mentions different ways to calculate indicators but he proposes a measure by Sabidussi [Sab66]. Sabidussi describes that the centrality should be a function of geodesic distances<sup>3</sup> which means, if the geodesic distances of a specific actor increases the centrality should decrease. Using Freeman's notation Sabidussi's measure of closeness centrality is defined the following way:

If  $d(p_i, p_k)$  = number of lines in the geodesic linking for actor  $i$  and  $j$  then

$$C_C(p_k)^{-1} = \sum_{i=0}^n d(p_i, p_k)$$

This measure again is dependent on the size of the underlying network and therefore this factor needs to be normalized.

$$C'_C(p_k) = \frac{n-1}{\sum_{i=0}^n d(p_i, p_k)} = (n-1)C_C(p_k) \quad (7.5)$$

**Graph centrality** The next step will be to calculate closeness centrality for total graphs. Therefore it is necessary to sum up the differences between the most central actor (using the definition of centrality given above) and all other actors in the network. This equation reads as follows:

$$\sum_{i=0}^n [C'_C(p^*) - C'_C(p_i)] \quad (7.6)$$

where

$C'_C(p^*)$  is defined as the centrality value of the most central actor.

This equation 7.6 divided by the maximum value for the network (in order to normalize), which occurs when one point is separated from all other points only

---

<sup>3</sup>shortest possible connection between two points

## 7 Social Network Analysis

by one (again in the case of a star like figure 7.2), determines graph closeness centrality the following way:

$$C_C = \frac{\sum_{i=0}^n [C'_C(p^*) - C'_C(p_i)]}{(n^2 - 3n + 2)/(2n - 3)} \quad (7.7)$$

The derivation in detail can be found in Freeman's study [Fre79].  $C_C$  reaches the maximum if one actor (maximum geodesics of 1) has connections to all  $n - 1$  actors (maximum geodesics of 2) what is fulfilled by a star or a wheel graph and in this case the network is very central.

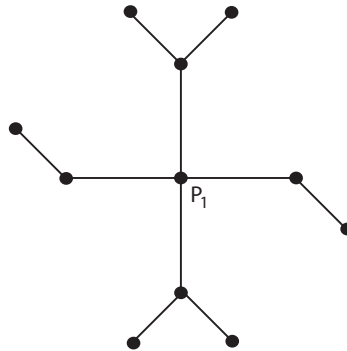


Figure 7.4: Closeness in node  $P_1$

**Differentiation** To sum up the approach of this indicator, closeness centrality points out the potential for independence and efficiency. It is a measure representing the autonomy of a member due to the potential in reaching all other members on the shortest path. Like it can be seen in figure 7.4 the node  $P_1$  is able to spread information with minimum time or costs [HWC06].



## 7.5 Indicators measuring Centrality or Prestige

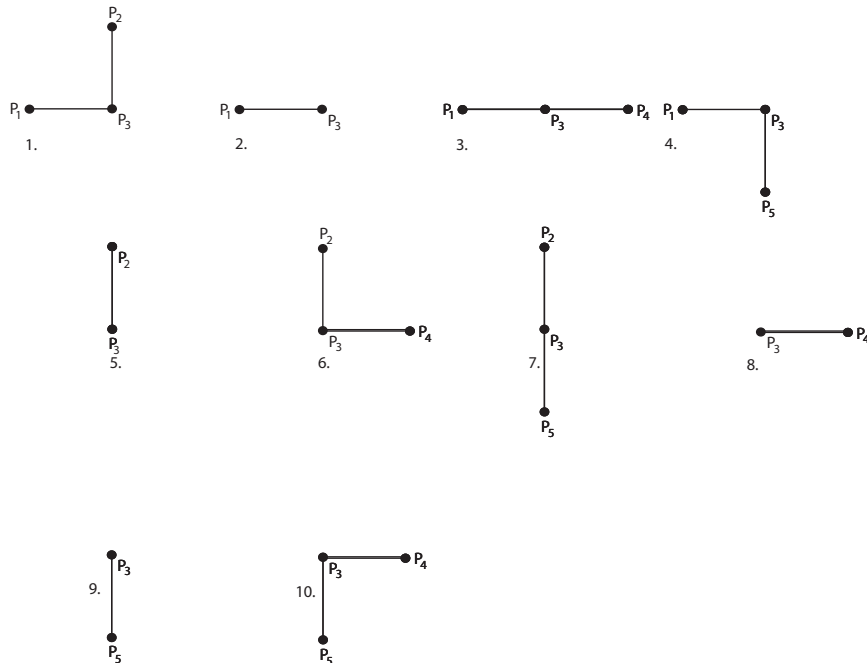


Figure 7.5: The ten geodesics from the graph of figure 7.2 [Fre79]

### 7.5.3 Betweenness Centrality

Betweenness is another construct we want to take a closer look at. This centrality factor focuses at the frequency of which a point lies between a pair of other points, defining a geodesic distance (a shortest connection). Using Freeman's example of a graph in star-form, figure 7.5 shows ten geodesics of a graph in star-form with five nodes. Four geodesics have a length of one and connect point  $p_3$  with all other nodes. The six other cases have a length of two and each time point  $p_3$  lies on the track which connect a pair of other points. The complete information flow in the graph now depends on point  $p_3$  and as a consequence  $p_3$  has total control.

What kind of influence has such an actor who is defined as central like it

## 7 Social Network Analysis

is the case of this approach? Freeman sums up the sense of this measure the following way: “It is this potential for control that defines the centrality of these points” [Fre79]. He also points out Bavelas [Bav48] and Shaw [Sha54] who argue that an actor who is a mediator by providing a linkage between other pairs of actors has huge influence on the communication process and information flow.

**Point centrality** The theoretical approach can be assigned to Shawn though he did not provide a way to calculate a measure. Anthonisse [Ant71] and Freeman [Fre77] were the first who have focused their work on creating a measure for betweenness centrality. Looking at the graph of figure 7.6, you can see that there are two geodesic connections from point  $p_1$  to  $p_3$ . The total dependency of one actor is resolved by the availability of a second shortest path, though  $p_2$  and  $p_3$  have a potential to influence the information flow. Now Freeman brings probabilities into account.  $g_{ij}$  is defined as the number of geodesics which connect two points  $p_i$  and  $p_j$ . All available geodesics have the same probability to be chosen and is defined as:

$$\frac{1}{g_{ij}}$$

The potential of a specific point  $p_k$  to be on the geodesic path and to be able to influence information flow is defined by Freeman [Fre79] as the probability that  $p_k$  is on the randomly chosen shortest path that connects point  $p_i$  and  $p_j$ . Again, one crucial fact is, that the likelihood of each geodesic to be chosen is equal. The equation of Freeman reads as follows:

$$b_{ij}(p_k) = \frac{1}{g_{ij}} \times g_{ij}(p_k) = \frac{g_{ij}(p_k)}{g_{ij}} \quad (7.8)$$

where

$g_{ij}(p_k)$  = number of geodesics linking  $p_i$  and  $p_j$  that contain  $p_k$

and  $b_{ij}(p_k)$  is the likelihood that the shortest connection is chosen, where  $p_k$  is involved

For calculating the centrality of point  $p_k$  the partial betweenness values for all unordered pairs of points where  $i \neq j \neq k$  have to be calculated and summed up.

### 7.5 Indicators measuring Centrality or Prestige

$$C_B(p_k) = \sum_{i < j} b_{ij}(p_k) = \sum_{i < j} \frac{g_{ij}(p_k)}{g_{ij}} \quad (7.9)$$

The minimum of  $C_B$  is zero and the maximum will be  $(g - 1)(g - 2)/2$  which is the maximum number of pairs not containing  $p_k$ . To make the measure  $C_B$  independent from network size it is necessary to normalize this equation by dividing with this maximum value. For calculating node betweenness centrality the following equation will be used:

$$C'_B(p_k) = \frac{2C_B(p_k)}{(n - 1)(n - 2)} = \frac{2C_B(p_k)}{n^2 - 3n + 2} \quad (7.10)$$

Betweenness centrality ranges between zero and one where a value of one stands for high centrality.

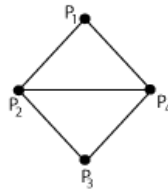


Figure 7.6: Graph with four points and five edges [Fre79]

**Graph centrality** Like it was the case in all other centrality measures the point centrality measure of betweenness is used as a basis for the graph centrality measure. Freeman mentions himself as the only one, providing a model for calculating this indicator [Fre79] and the idea of this model is the following: “It was defined as the average difference between the relative centrality of the most central point,  $C'_b(p^*)$ , and that of all other points.” Again, it is only focused on the final equation which reads as follows:

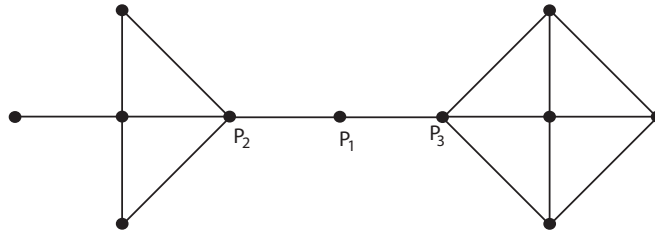


Figure 7.7: Gatekeeper with special care in betweenness centrality [HWC06]

$$C_B = \frac{\sum_{i=1}^n [C_B(p^*) - C_B(p_i)]}{(n-1)(n^2 - 2n + 2)} \quad (7.11)$$

where

$C_B(p^*)$  represents the relative centrality of the most central point and  $C_B(p_i)$  is equivalent to the centrality of all other points

**Differentiation** The idea, how to measure centrality differs to already mentioned closeness and degree centrality. Due to the special approach of this indicator, betweenness takes special care about structural bridges or gatekeepers. Like it can be seen in figure 7.7 node  $P_1$  has a very important role of being an intermediary between two subnets. In the case of this figure, node  $P_1$  would have the lowest degree centrality but the highest betweenness centrality.

### 7.5.4 Information Centrality

Wasserman et al. [WF94] mention that there is no other measure than betweenness centrality which is used more often and according to them, the reason lies in the generality of this measure. Information centrality relies on this measure and eliminates assumptions which have been introduced by Freeman in order to simplification. Freeman hypothesized that if there are more than one geodesic paths connecting a pair of actors, all paths have the same likelihood to be chosen. The probability to select a path where one specific actor is involved can

## 7.5 Indicators measuring Centrality or Prestige

simply be calculated by  $1/g_{ij}$  where  $g_{ij}$  represents the number of geodesic paths connecting the pair  $(i, j)$  (compare section 7.5.3).

Wasserman et al. then declare that this simplification is not appropriate all the time. If the graph is very regular where the nodes have constant degrees (number of connections to other nodes) the approach of Freeman may be valid but in practice only the minority of cases are regular. The key fact of their arguments is, that there is no equal likelihood for selecting paths. There are different influences and information centrality is one measure which tries to care about this influences. According to Wasserman et al. one solution is to try to consider if specific actors have higher degrees because as a result they are more likely to be chosen.

The next point deals with the assumption that geodesic paths are always the best selection which has not to be true all the time. Introducing an evaluation and scoring system cares also about other paths.

Stephenson and Zelen [SZ89] have dealt with this kind of problem and have defined a measure which takes this influences into account. For each node a weight of the “information” is calculated. Information is defined in a statistical manner by being the inverse of the variance of an estimator. If the variance is small the information contained by the estimator is high. Betweenness centrality is now extended by caring not only about geodesics but also about all available weighted paths.

**Differentiation** Basically, information centrality relies on the idea of betweenness centrality and has nothing in common with degree or closeness centrality. Betweenness and Information centrality differ in the way how the extend of being gatekeeper is considered.

### 7.5.5 Eigenvector Centrality

Eigenvector centrality introduced by Philip Bonacich in 1972 [Bon72] picks up the approach of closeness centrality but differentiates in one detail. Closeness centrality is evaluating the distance from one node to all others and treats all nodes equally. Eigenvector centrality introduces an additional view by considering the importance of nodes to which one node is connected. Generally spoken

one node has more power to influence if the nodes to which connections exist are themselves highly “influential” [New08]. According to Newman, Google’s page-rank algorithm bases on a variant of eigenvector centrality. The approach of Bonacich takes two factors into account which are the number of connections to other vertices and the importance of the connected vertices. Being highly integrated into a network means to have a lot of connections which is still important. Never the less a node with fewer but more connections to important nodes may outrank the situation of having more connections with little quality.

### 7.5.6 Centrality Indicators for Directed Relations

In the previous part of this chapter centrality measures only deal with undirected relations. There was no differentiation between like Wasserman et al. [WF94] call it choices made or choices received. In specific cases it makes sense to introduce such a differentiation and take a closer look at the behavior of the actors in the network in order to answer questions like: “Who are the most important distributors?” On this point it is important to mention again that per definition directed actor centrality just looks at outdegrees (or how Wasserman et al. use to call “choices made”) whereas prestige cares about indegrees or “choices received”.

Some of the centrality measures which have been introduced, can be adopted easily for directional relations, others have their problems.

**Degree Centrality** In the case of undirected relations, the degree (in and outgoing connections) is considered for this measure. By separating between in and outdegrees and just counting outdegrees in equations, the former undirected measure can be used for directed relations. Just by counting the specific degrees and adopting the denominator (which normalizes the indicators) to  $(g - 1)^2$  degree centrality can be further used for directed relations.

**Closeness Centrality** To recapitulate, closeness centrality cares about the distances of one node to all other nodes using geodesics. Also this indicator easily can be adopted for directional relations by differentiating between in and outdegrees. However this influences the geodesics because edges will not be available anymore due to the direction. The calculation of the indicator remains the same

## 7.5 Indicators measuring Centrality or Prestige

as it was already shown in section 7.5.2.

One problem arises with the requirement that the graph has to be strongly connected (each node has to be reachable). Also in the case of undirected relations this problem remains the same and it can only be solved by ignoring not reachable nodes. Otherwise some distances will be infinite and the measure can not be calculated.

**Others** The two left indicators (betweenness and information centrality) have been specially designed for undirected relations and for this reason adoption is a challenge. Wasserman et al. [WF94] present some approaches to adopt this indicators to make them usable for directed relations but also recommend using closeness and degree centrality for directed relations and leave the other two measures behind.

For this reason it is referred to Wasserman et al. in order to get further information about these two left indicators.

### 7.5.7 Prestige

Prestige does not differentiate between directed or undirected relations, meaning that the basis of this concept solely relies on directional relations. As it was already mentioned, prestige differs from directed centrality measures (see section 7.5.6) by only caring about indegrees (choices received) and uses this information for calculating representative indicators. Wasserman et al. [WF94] suggest that centrality and prestige measures should be computed together because compared to centrality, prestige captures slightly different structural properties.

**Degree Prestige** According to Wasserman et al. [WF94] degree prestige is the simplest measure and just counts indegrees of an actor and normalizes this value by dividing with the group size minus one.

$$P'_D(n_i) = \frac{x_{+i}}{g - 1} \quad (7.12)$$

where

## 7 Social Network Analysis

$x_{+i}$  represents the number of outdegrees of a node  $i$  and  $g$  stands for the size of the group

The interval of this value is a closed one between zero and one. A high value represents high prestige of an actor.

**Proximity Prestige** Proximity prestige uses the idea of degree prestige but extends this by counting not only direct adjacent nodes but also indirect adjacent nodes. In detail simply the distances of all nodes which can be reached from one specific node (prestige evaluates only indegrees) are summed up and the average is computed by dividing with the total accessible nodes. To make this measure ready for comparisons across different networks, this equation has to be normalized. Finally Wasserman et al. [WF94] propose the following equation:

$$P'_P(n_i) = \frac{I_i/(g-1)}{\sum d(n_j, n_i)/I_i} \quad (7.13)$$

where

$I_i/(g-1)$  represents the average number of actors who can be reached from one actor

$d(n_j, n_i)/I_i$  stands for the average distance between reachable actors and the specific node

On group level it is different to calculate a maximum value in the denominator. Therefore simpler methods like variances or averages of actor prestige values are used to calculate such indicators.

**Status or Rank Prestige** Wasserman et al. [WF94] notice that status or rank prestige is out of the mathematical point of view one of the most sophisticated measures for prestige. The theoretical approach is also different to those measures given so far. The prestige of an actor is calculated by looking at the ranking of the actors who are pointing to a specific actor. The idea of status or rank prestige can be interpreted the following way: an actor who is surrounded by a lot of other prestigious actors should as well have high prestige.

Thus this measure combines two different evaluations:



### 7.5 Indicators measuring Centrality or Prestige

- the distances to an actor
- combined with the rank of connected actors

With this idea a problem arises as well. To state the ranking of an actor, the ranks of the other actors are needed which leads into a lock out. To solve this problem Wasserman et al. [[WF94](#)] present several mathematical approaches.

## 7 *Social Network Analysis*

# 8 Social Network Analysis in Practice

The previous chapters contain a theoretical introduction to Social Network Analysis where the most important ideas of this concept were pointed out. Furthermore background about open source software development was provided and the reader was introduced to the concept of project health in other chapters. What follows next is the combination of the information given so far to one approach: Social Network Analysis will be used to evaluate the health status of FLOSS projects.

At the beginning of this chapter it will be tried to provide a bridge between the theoretical approach of Social Network Analysis and a practical usage of this concept. As a first step social information which is available in projects will be identified in order to tell something about the conditions and happenings in a FLOSS project.

## 8.1 Action and Interaction Data

Crowston and Howison [CH04] differ between two categories of data available in FLOSS projects which can be used in order to put actors or in this case members of the community into relation. This relations are necessary to evaluated the social structure with the methods Social Network Analysis provides.

During the software engineering process huge masses of data are stored which can be split up to the following categories:

- Information based on **interaction** can be directly used in order to do research on the social structure of a FLOSS project team. Every medium

which describes a relation and is reproducible can be used for these belongings. Out of conversation in mails, mailing lists, forums, instant messaging or bug-report systems the relevant information can be extracted. Usually the conversation contains a detailed follow up of messages which can be used in order to reconstruct the social composition of the FLOSS team with the help of a “has communicated with” relation.

- The second kind of measure is called **action** measure. They describe all side-information which accumulate during the development process. This kind of information refers to the active happening within the project like the collaborative writing and managing of code or documentation. Questions concerning the organization within the project team can be researched by looking at specific data which is stored as a byproduct of the development process. Reading out this side-information makes it possible to reconstruct the social composition of a FLOSS team with the additional constriction to define the linking entity previously. This linking entity can be the contribution to a single file or the contribution to a complete patch, it depends on the view of the observer.

For both of these measures assumptions need to be defined and it needs to be researched which actors are reached with the specific source. Before looking at the social data which is available in an explicit (interaction measure) or implicit (action measure) manner the researcher needs to know what should be proven by his work.

A concrete example: By looking at data which is available in revision control systems and describes the collaboration of a project team it needs to be evaluated who is allowed to use the system. Are there other people who contribute to the project as well and stand in relation to the developer checking in the source code? What if these external contributors just have no permissions to do this for their own? At least it needs to be considered if it is alright to leave this external contributors behind but in some cases it makes sense to search for additional sources which capture this kind of actors.

As long as it can be argued it is possible to combine various kinds of social relations. It is also possible to combine interaction and action measures in order to reach a wider circle of contributors to a project if it makes sense for the topic which is researched. That is, what this chapter is about. A tool was created in order to do Social Network Analysis based on various kinds of social relations with the aim to extract information about the potential of but also limitation in

FLOSS projects.

## 8.2 Availability of Social Data in FLOSS Projects

Depending on the field of research and the required information different sources have been used in the past to extract social information. Methods range from using simple questionnaires like it was done in the time before computers existed or in other areas where digital information is not available and goes to the extraction of information out of databases. Such databases for example are used by FLOSS hosting platforms and contain a huge amount of information. Almost everything what is happening in a project is stored there. Due to the focus on Open Source Software in this thesis relevant sources are limited to this context. In order to provide the possibility to research social aspects of FLOSS, different projects are available which have the aim to provide relevant data for research in this area. Some of these projects are used in the underlying thesis. The different kinds of data sources used within this projects are the following:

- Data dumps of hosting sides: The largest and most flexible kinds of data sources are databases which are hosting information created during the existence of a FLOSS project. This information is provided by hosting platforms which are used by FLOSS projects in order to publicize but also to manage the project. In these databases a huge amount of information is stored. Beginning with simple project description it ranges from a listing of all actors in a project to more specific information out of the bug reporting system (see [VAM08][GACM07][Mad]) or keeps the messages of a forum. Generally almost everything is stored what is happening within a project. Further information will be given in chapter 9 where SNAlyzer [Kal], the tool developed within this diploma thesis, is explained.
- Historical mailing lists: Another possible source to extract information about people contributing to a project are mailing lists. To have enough information for an analysis the history of mail exchanges for a longer period is relevant. Just register to a mailing list will not be enough because the history is not available. Different platforms close this lack of information and provide a history of mailing lists (see the project “gmane” [Ing02]). Contributions to this topic can be found in [HMS05].
- Revision control systems like Git [Cha05], CVS [PX06] or Apache Subver-

sion [The10]: In order to manage revisions such systems store the version history of a file and among others, information about the contributor and the time of contribution which is important for this thesis. FLOSS analysts have recognized this to be relevant for further research because the link “working together on a patch file” can be interpreted as a social relation within a FLOSS project. Extracting the information of a specific period allows making conclusions about the structure of a project (see also [GRMMORM03]).

- Expert finding: The last group contains individual applications which are harvesting relevant informations directly from project websites. It is also possible that different kinds of sources are combined [CHC05].

Collecting data for Social Network Analysis can be a hard job to do. Depending on the aim of the research topic, available data can be used or has to be gathered. Due to specific requirements in most cases there is no “take it and use it” solution and resulting of this, there is no other possibility than developing own software which fulfills the needs. In the case of this thesis it was decided to go this way and a software was developed which uses a combination of the mentioned data sources. Primarily data provided by [Mad] is used which contain monthly data dumps of the platform sourceforge [sou]. Furthermore data out of the revision control system of a project was extracted in order to get an overview of the project’s core team. More about the aim of his research and the information which is used can be found in chapter 9 where the tool “SNAnalyzer”, developed within this thesis, is introduced.

### 8.3 Reference Tools and Libraries for Social Network Analysis

Social Network Analysis has a long history and over time various tools have been established. Some of them are freely available whereas other ones are proprietary and need to be bought. In this section an overview of the most important approaches for conducting Social Network Analysis with FLOSS projects is given.

Basically it is possible to differentiate between independent programs which usually provide an interface in order to import or export data. Furthermore software libraries which equip own software with functionality needed in order

### 8.3 Reference Tools and Libraries for Social Network Analysis

to perform Social Network Analysis are available.

A lot of different tools are existing but not every solution is acceptable due to different constrictions. The tools, introduced in this chapter had to go through a selection process which had the following requirements as input:

- An interface to import data, describing the structure of the social network has to be available
- Centrality measures specified in section 7.5 have to be calculable
- It has to be possible to illustrate the social network with a sociogram for further graphical analysis
- Due to the topic of this thesis it was focused on FLOSS projects
- Java compatible solution is appreciated
- Compatibility to various operating systems

#### 8.3.1 Independent Tools

**Pajek** (Slovene word for spider) is a project from the University of Ljubljana (available at [BM08]). The program which runs on Windows is freely available for noncommercial use and especially supports the analysis of large networks. It is possible to factorize a large network into several smaller networks in order to provide the possibility to perform more sophisticated operations [BM98]. The program supports a widely used file format which is called “.net”. It is possible to use other software to extract the information about the structure of a network and store it in the mentioned format. Pajek can be used for further analysis.

**NetMiner** Although this software (available at [Cyr01]) does not agree to all mentioned requirements because of being a proprietary product, it needs to be mentioned. According to Crowston et al. [HC04] the big advantage of this program lies in it’s high capability for illustrating sociograms.

### 8.3.2 Libraries

**Java libraries** Due to the personal capabilities of the author and the fact, that other software for this thesis was already developed in Java he puts the focus on java libraries which can perform Social Network Analysis. A hint by a colleague who also uses Social Network Analysis (see [Omo08]) introduced the author to **JUNG** [OFN09] and **prefuse** [The09].

The Java Universal Network/Graph Framework (JUNG) is a Java library which supports modeling, analyzing and visualizing of data which can be illustrated with graphs or networks. JUNG implements not only concepts like importance measures (e.g. centrality - see section 7.5) or network distances out of the area of Social Network Analysis. Furthermore other algorithms originating from the research areas of graph theory and data mining are provided [OFN09].

Prefuse, also a Java library is developed at the UC Berkeley - Institute of Design [The09]. In comparison to JUNG calculating any measures, relevant for this thesis, is not possible because the focus of this library lies in providing methods for rich data visualization. Prefuse uses the Java 2D graphics library and provides methods to implement sophisticated interactive animations, dynamic queries and integrated search for different kinds of data. Due to the high flexibility the usage area of this library is very wide. To see what is possible it is referred to the tool SocialAction [PS08] which uses the Prefuse library and implements a Social Network Analysis tool.

**The open source statistic project R** The last solution for performing Social Network Analysis it is referred to is the open source project R [R D09]. R is a sophisticated statistic software, hosted by the Vienna University of Technology. Various libraries which are available to extent the functionality make this project as flexible as a Swiss Army knife. In combination with the Java library JRI which is part of the rJava [RFo] project a very strong instrument for performing Social Network Analysis is available. With the help of JRI R runs inside a Java application as a single thread. According to [RFo] it loads R dynamic libraries into Java and provides an interface for exchanging data between R and Java. JRI acts furthermore also as kind of a remote control to access to the functionality, provided in R. Like it can be seen in figure 2.1 a possibility to draw sociograms is available as well. According to Crowston et al. [HC04] the advantage of R can be found in it's high scriptability.



### 8.3 Reference Tools and Libraries for Social Network Analysis

Table 8.1: Reference Tools and Libraries for Social Network Analysis

Name	Type	FLOSS	Visualization	Operating System	Calculation of measures
Pajek	Tool	Freeware for non-commercial use	✓	Win	✓
NetMiner	Tool	–	✓	Win	✓
JUNG	Java Library	✓	✓	* <sub>1</sub>	✓
Prefuse	Java Library	✓	✓	* <sub>1</sub>	–
Statistic project R	Tool/Java Library	✓	✓	Unix, Win, MacOS	✓

\*<sub>1</sub> See Java supported systems [[Orac](#)]

8 *Social Network Analysis in Practice*

## 9 SNAalyzer

In order to work on the research questions an own tool fulfilling the specific needs of this thesis was developed. Which possible technical approaches could potentially fit to the requirements were already described in the previous chapter. Furthermore table 8.1 shows an overview of all possible solutions arranged according to all relevant criteria which influenced the decision making.

This chapter contains a description of the solution called “SNAalyzer” implemented within this thesis. For more information to the tool and the handling of the software it is referred to section A in the appendix.

### 9.1 Technical Solution

Based on the information out of reference work, personal experience and capabilities of the author the approach containing the open source statistic project R [R D09] was implemented. The advantage lies in it’s high flexibility because R is a very complex complete statistic environment. With the Java library JRI the complete functionality of R is available within a Java solution.

Figure 9.1 describes the technical concept of this project. SNAalyzer prepares information stored in a MySQL database [Oraa] for further analysis what happens within the statistic environment R and illustrates the results retrieved. For data exchange between SNAalyzer and R the Java library JRI is used. Another important feature in R furthermore provides a possibility to illustrate sociograms (e.g. figure 10.3). SNAalyzer also enables the possibility to export the results gained within the tool. Data generated within an analysis is stored in a text file in order to provide information for further analysis.

The solution of different technical problems concerning the setup of R in order

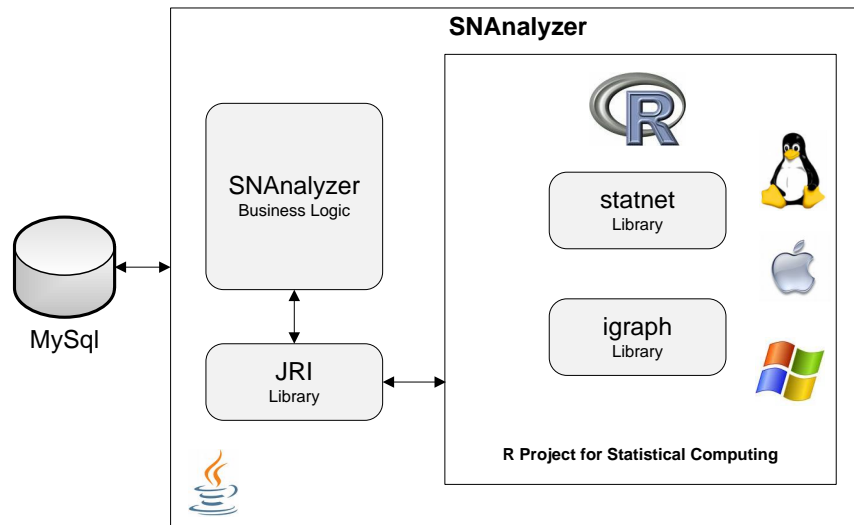


Figure 9.1: Technical concept of SNAlyzer

to be used within a Java application is described separately in appendix B.

## 9.2 Functionality

Social Network Analysis contains a sophisticated set of instruments but not all of them are relevant for this thesis. In order to provide information for health analysis some features were implemented which contain the illustration of a social network in a sociogram (e.g. figure 10.3) and the calculation of:

- Degree centrality (see section 7.5.1)
- Closeness centrality (see section 7.5.2)
- Betweenness centrality (see section 7.5.3)

The reason for the limitation towards the mentioned indicators lies in the fact, that the data sources selected, restrict the possibilities to extract social relations. Information out of revision systems and bug reports can only be interpreted as

undirected linkages. More about the kinds of social relations which are used in this thesis is mentioned in the next section.

## 9.3 Social Data

One part of this thesis deals with the extraction of relevant social information in order to perform Social Network Analysis. In the theoretical part of this thesis different possible data sources for extracting relevant social data have been discussed (see section 8.2). Out of an analysis which information could be interesting and what kind of information is available the bug reporting system and the revision system were selected as data sources for the health analysis. The advantage of these sources lies in the fact that they are matching to each other by using an identical identifier for a user.

**Bug reporting system** Crowston et al. [CH04] mention that the bug report process (see section 4.3.6) is essential for healthy FLOSS projects. Besides a wish list bug reports are the input for the task list of a project (see continuous improvement in section 4.3.6). Bugs influence what to do within the project and attract many people. Not always bug reports are clear to everybody from the beginning on what generates interaction in order to reach clarity. Figure 9.2 illustrates this case. A person opens a bug report and a discussion follows. The analysis of this information contains the assumption that people who participate very often in such projects are also very important for a project. The social relation is defined that way that people commenting a report have a social relation if the comments are within the observation period. The linking entity in this case is the comment to a bug report.

Sourceforge [sou] which is a hosting platform for FLOSS projects provides data for scientific research. The data is hosted within the project “The SourceForge Research Data Archive” [Mad] (SRDA) where the author had access in order to do research. Monthly data dumps can be read out with SQL-queries and the results can be stored in XML formatted files. A parser reading out these files and storing the information into a local MySQL database was developed.

## 9 SNAalyzer

**jEdit** [Share](#) [Donate](#)

[Summary](#) | [Files](#) | [Support](#) | [Develop](#) | [Hosted Apps](#) | **Tracker** | [Mailing Lists](#) | [Code](#)

[Add new](#) [Browse](#)

**Tracker: Bugs** [Monitor](#)

1 **Unclosable buffer - ID: 2958459** Last Update: Comment added ( [vampire0](#) )

**Details:** I got again an unclosable buffer with 4.3.1. I'm not sure what cause it, but the effect was that the buffer didn't have a mode set and this caused jEdit:1900 to throw a NPE. If it is appropriate for a buffer not to have a mode set, then there should be a null check, but I guess the problem is not the missing check but the missing mode. It could be that there was an IO Error when opening the file, I'm not sure about that.

**Submitted:** Björn Kautler ( [vampire0](#) ) - 2010-02-25 01:44:52 UTC **Assigned:** Matthieu Casanova

**Priority:** 1 **Category:** editor core

**Status:** Open **Group:** None

**Resolution:** None **Visibility:** Public

**Comments ( 4 )** [+](#)

Date: 2010-03-17 23:33:17 UTC  
Sender: [vampire0](#) NPE = NullPointerException if that is your question

Date: 2010-03-16 23:14:06 UTC  
Sender: [kpouer](#) No the mode is not set to null if you close the buffer, what was the exception ?

Date: 2010-02-26 23:23:36 UTC  
Sender: [vampire0](#) As I'm not able to reproduce it and already closed that file I cannot test your BS snippet currently. But I think I didn't even had it open completely. As I said I think there was an IO error on opening it. As the mode was null this would also fit, wouldn't it? The mode is not set to null when closing the buffer, is it?

Date: 2010-02-26 10:04:32 UTC  
Sender: [kpouer](#) When it happens, could you try to evaluate this beanshell snippet ?  
`buffer.isClosed()`  
It may be possible that the buffer was closed but not removed to the buffer list

Figure 9.2: Discussion based on a bug report

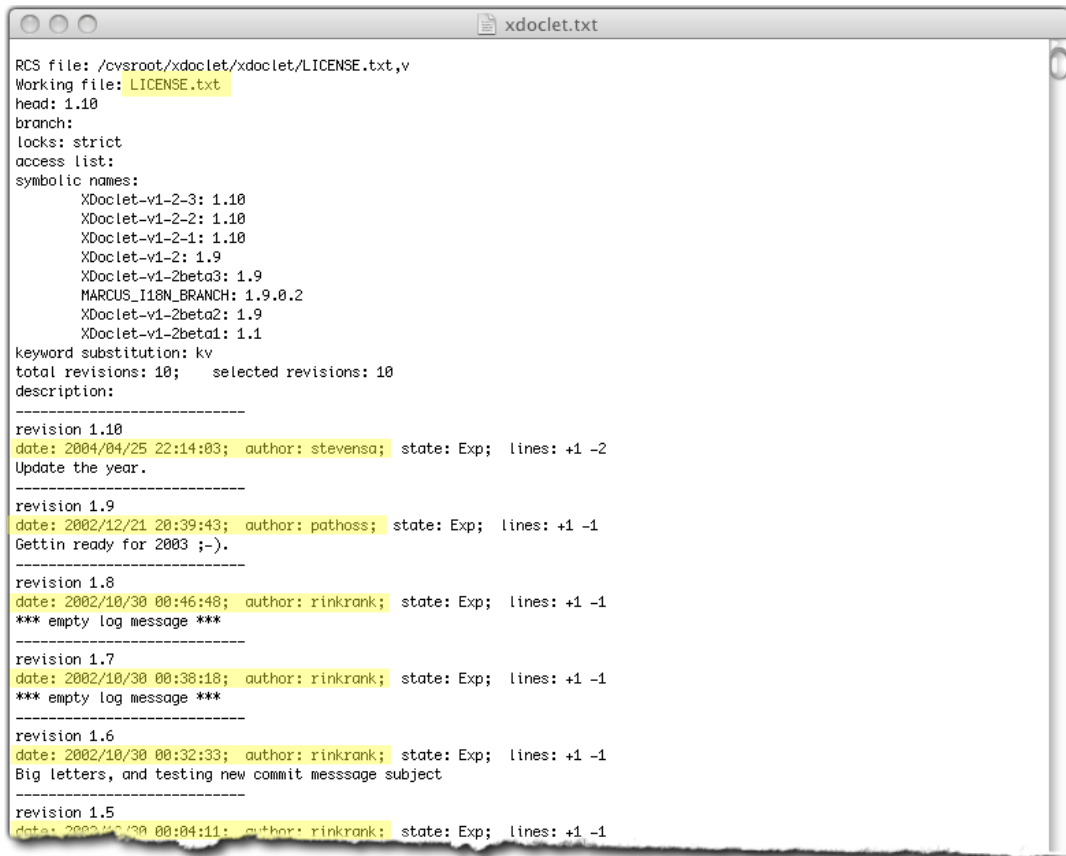
**Revision Control Systems** The second kind of data source describing a social relation deals with information which is available out of revision systems. In detail interfaces to read out the log of CVS [PX06] and Subversion [The10] were developed. These logs contain a history of commits to all files within a directory and the information is stored in a text file like it is available in figure 9.3. A parser reads out the following items and stores them in a local database:

- Name of a committer (what is matching to the identifier in the bug reporting system)
- Date and time of the commit
- Name of the file

The aim of this approach is to get information about the core project team for further analysis. Also the fact that only the core team of a project has permissions to commit to the revision system is considered. By definition actors have a social relationship if they are committing to the same file within the observation period.

If an actor has committed just one time within the total period of observation, he will also be marked as developer in all other periods even if he has only replied to a bug report in that specific period. This implies that a mixed sociogram containing social information out of bug reports and revision systems let conclude the offspring of the specific relation. By labeling the ID's of the actors with the colors blue for bug reporters and red for code committers the source of the relation can be traced back. A mixed relation between a red and a blue actor describes a core developer who has committed to a bug report. In at least one other time frame there has to be a relation between two red colored actors.

## 9 SNAlyzer



```
RCS file: /cvsroot/xdoclet/xdoclet/LICENSE.txt,v
Working file: LICENSE.txt
head: 1.10
branch:
locks: strict
access list:
symbolic names:
    XDoclet-v1-2-3: 1.10
    XDoclet-v1-2-2: 1.10
    XDoclet-v1-2-1: 1.10
    XDoclet-v1-2: 1.9
    XDoclet-v1-2beta3: 1.9
    MARCUS_I18N_BRANCH: 1.9.0.2
    XDoclet-v1-2beta2: 1.9
    XDoclet-v1-2beta1: 1.1
keyword substitution: kv
total revisions: 10;   selected revisions: 10
description:
-----
revision 1.10
date: 2004/04/25 22:14:03; author: stevensa; state: Exp; lines: +1 -2
Update the year.
-----
revision 1.9
date: 2002/12/21 20:39:43; author: pathoss; state: Exp; lines: +1 -1
Gettin ready for 2003 ;-).
-----
revision 1.8
date: 2002/10/30 00:46:48; author: rinkrank; state: Exp; lines: +1 -1
*** empty log message ***
-----
revision 1.7
date: 2002/10/30 00:38:18; author: rinkrank; state: Exp; lines: +1 -1
*** empty log message ***
-----
revision 1.6
date: 2002/10/30 00:32:33; author: rinkrank; state: Exp; lines: +1 -1
Big letters, and testing new commit message subject
-----
revision 1.5
date: 2002/10/30 00:04:11; author: rinkrank; state: Exp; lines: +1 -1
```

Figure 9.3: CVS Log used to extract information about social structure of a FLOSS project team



# 10 Evaluation

This chapter contains the practical implementation of the concepts, described within this diploma thesis. In order to validate the hypotheses in chapter 6 different show cases are used. Out of different criteria, examples were selected which should fulfill the needs in order to proof the validity of his assumptions. For each case the preconditions are described and the differentiation to the other cases is pointed out.

Different kinds of FLOSS projects were selected and each of them has a specific focus in order to declare the logic of project health. The first project should illustrate the typical development of a healthy project out of the knowledge which is available so far. A validation is not possible but this example is used in order to point out some specialities. The aim of this approach lies in the fact to declare the information provided by the different indicators.

The next example deals with change within the structure of a development team and puts the health status initially in the background. This project is used to introduce an additional view in order to extract some information about the roles within the project.

Project number three puts the focus towards stability and the relation between stability and project health. The relation between these two factors was already discussed in chapter 5.3.3. It will be tried to give an answer to this question with a detailed example.

The chapter is finalized with an approach which should point out the limitations of project health. More generally spoken the limitations such technical approaches have will be critically analyzed in order to provide preview about future developments.

It was decided to use a time frame of 10 years (starting from 2001-01-01 or earlier if the project does not exist that long) with periods of a half year.

During one period all available data is collected in order to reconstruct the social relationships within this timeframe. The constructed social network out of the respective period is researched to its change. Challenging is the aim to clear questions like why specific developments have happened or not.

### 10.1 Healthy Project

In order to find a project which potentially fits into this group, it was relevant to focus on the hypotheses defined in chapter 6 which depicts the requirements to a healthy FLOSS project:

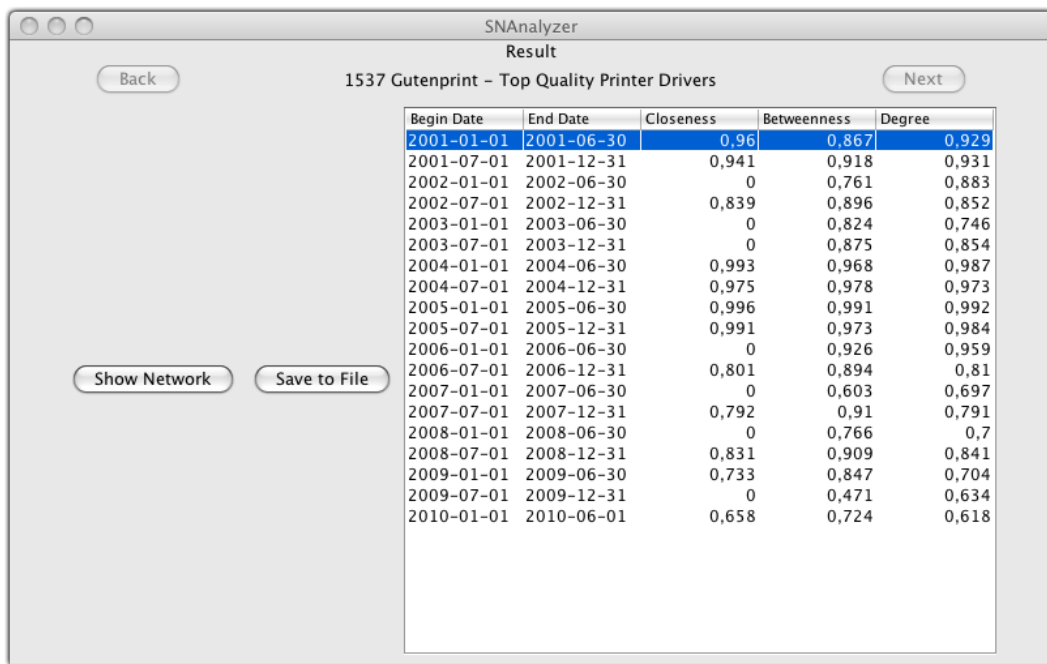
- Diverse set of core developers leads to higher decentrality within the core group. This can be verified by analyzing the data source revision control system separately.
- Decline in centrality indicators

Such kind of a project should be representative for having a low risk to become sick out of social reasons or the drop out of a certain actor. To show the details an additional view was introduced which differentiates between data extracted out of the data source revision system solely and data source connected to the information out of the bug tracking system. The first case makes it possible to put the focus towards the software development team and research change solely within the core team. The basic assumption contains a low centrality in the core team which is a main indicator for a healthy project.

The project which fits to the requirements and can be used as example for this category is called “Gutenprint - Top Quality Printer Drivers” although some constrictions dealing with the timeframe of observation have to be raised.

**Project description** Gutenprint [[mK](#)] provides a package of free printer drivers for various POSIX-compliant operating systems like Linux or Max OS. Due to it’s initial focus the highest usage can be found in open source systems like Linux. According to [[Wik10](#)] Gutenprint supports about 700 different printers.

## 10.1 Healthy Project



The screenshot shows a window titled "SNAlyzer" with a subtitle "Result". The main content is a table titled "1537 Gutenprint - Top Quality Printer Drivers". The table has five columns: "Begin Date", "End Date", "Closeness", "Betweenness", and "Degree". The first row is highlighted in blue. Below the table are two buttons: "Show Network" and "Save to File".

Begin Date	End Date	Closeness	Betweenness	Degree
2001-01-01	2001-06-30	0,96	0,867	0,929
2001-07-01	2001-12-31	0,941	0,918	0,931
2002-01-01	2002-06-30	0	0,761	0,883
2002-07-01	2002-12-31	0,839	0,896	0,852
2003-01-01	2003-06-30	0	0,824	0,746
2003-07-01	2003-12-31	0	0,875	0,854
2004-01-01	2004-06-30	0,993	0,968	0,987
2004-07-01	2004-12-31	0,975	0,978	0,973
2005-01-01	2005-06-30	0,996	0,991	0,992
2005-07-01	2005-12-31	0,991	0,973	0,984
2006-01-01	2006-06-30	0	0,926	0,959
2006-07-01	2006-12-31	0,801	0,894	0,81
2007-01-01	2007-06-30	0	0,603	0,697
2007-07-01	2007-12-31	0,792	0,91	0,791
2008-01-01	2008-06-30	0	0,766	0,7
2008-07-01	2008-12-31	0,831	0,909	0,841
2009-01-01	2009-06-30	0,733	0,847	0,704
2009-07-01	2009-12-31	0	0,471	0,634
2010-01-01	2010-06-01	0,658	0,724	0,618

Figure 10.1: Centralities of the project “Gutenprint” [BW] with data sources: Bug response and revision system checkins

## 10 Evaluation

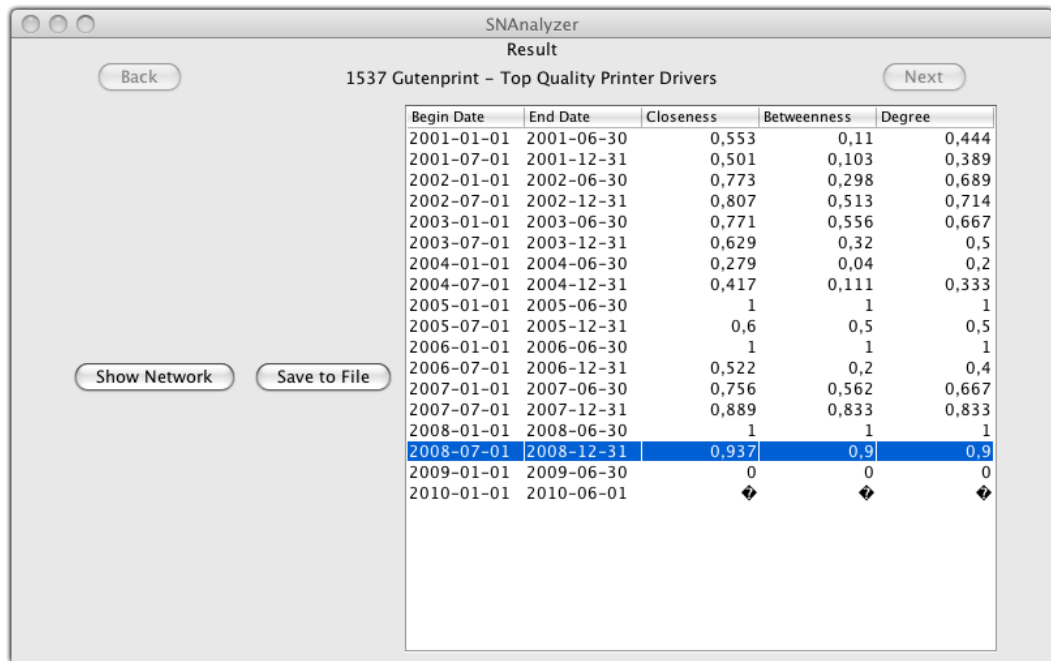


Figure 10.2: Centralities of the project “Gutenprint” [BW] with data source: Revision system checkins

**Datasource** As basis for the analysis bug reports and commits to the revision system were used which is in this case CVS. Although this example has a strong focus to the revision system as a separate data source in order to point out specific information.

**Results** Like it was already declared, the focus of this example lies on the period from the beginning of the data row until the end of 2003 because only this period shows the characteristic of a healthy project. What was initially very surprising were the centrality scores available in figure 10.1. The author has compared that information with the centrality calculated on the basis of the core team (see figure 10.2) which was evaluated by using information just out of the revision control system and came to the assumption that a high centrality in that case makes sense and should not be interpreted as a sign of sickness. It is assumed that a specific developer (see figure 10.4) named “5436” who is right in the middle of the network could have the special job to deal with bug requests

## 10.1 Healthy Project

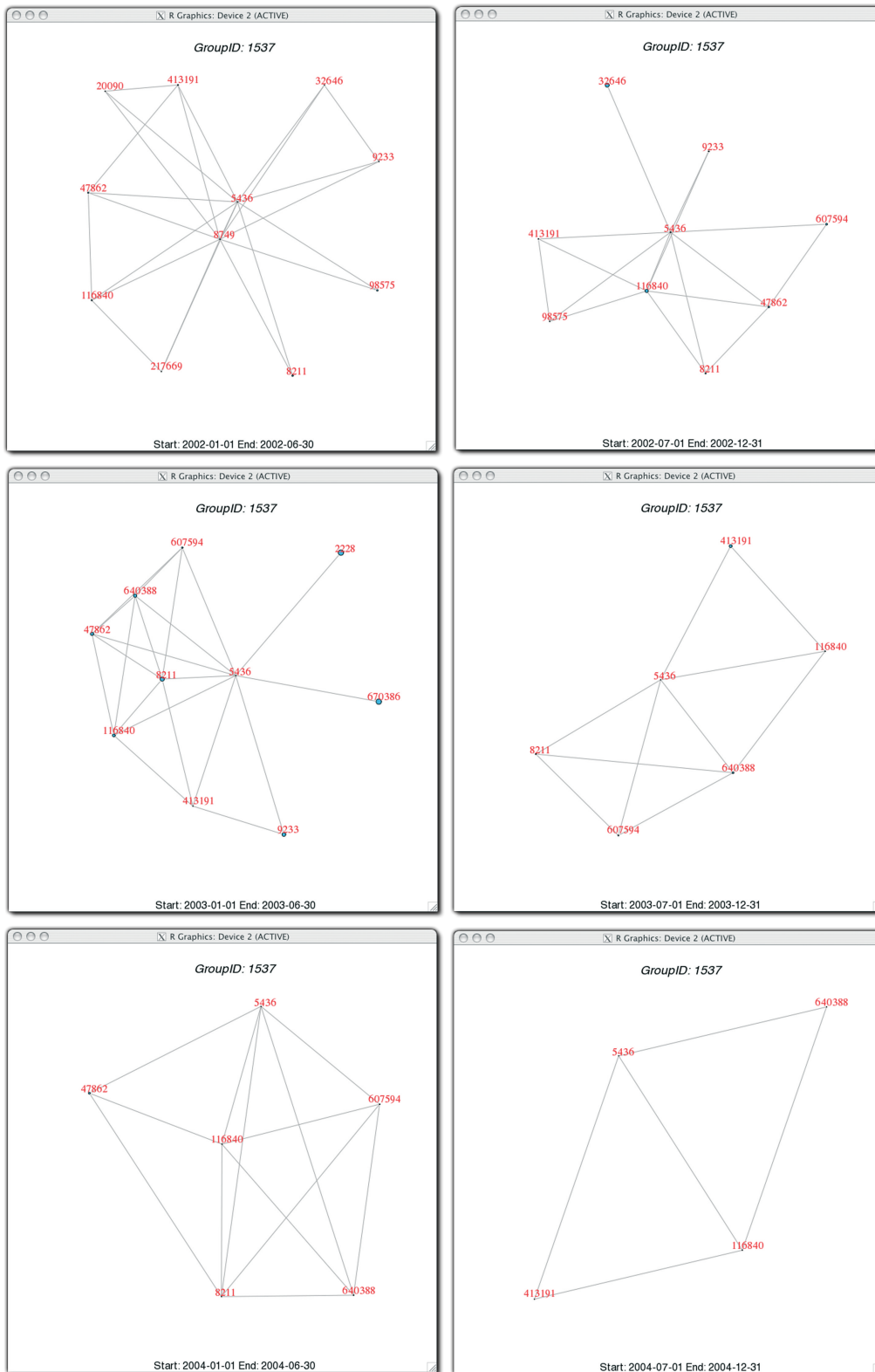


Figure 10.3: Sociograms “Gutenprint” [BW] with revision system data

10 Evaluation

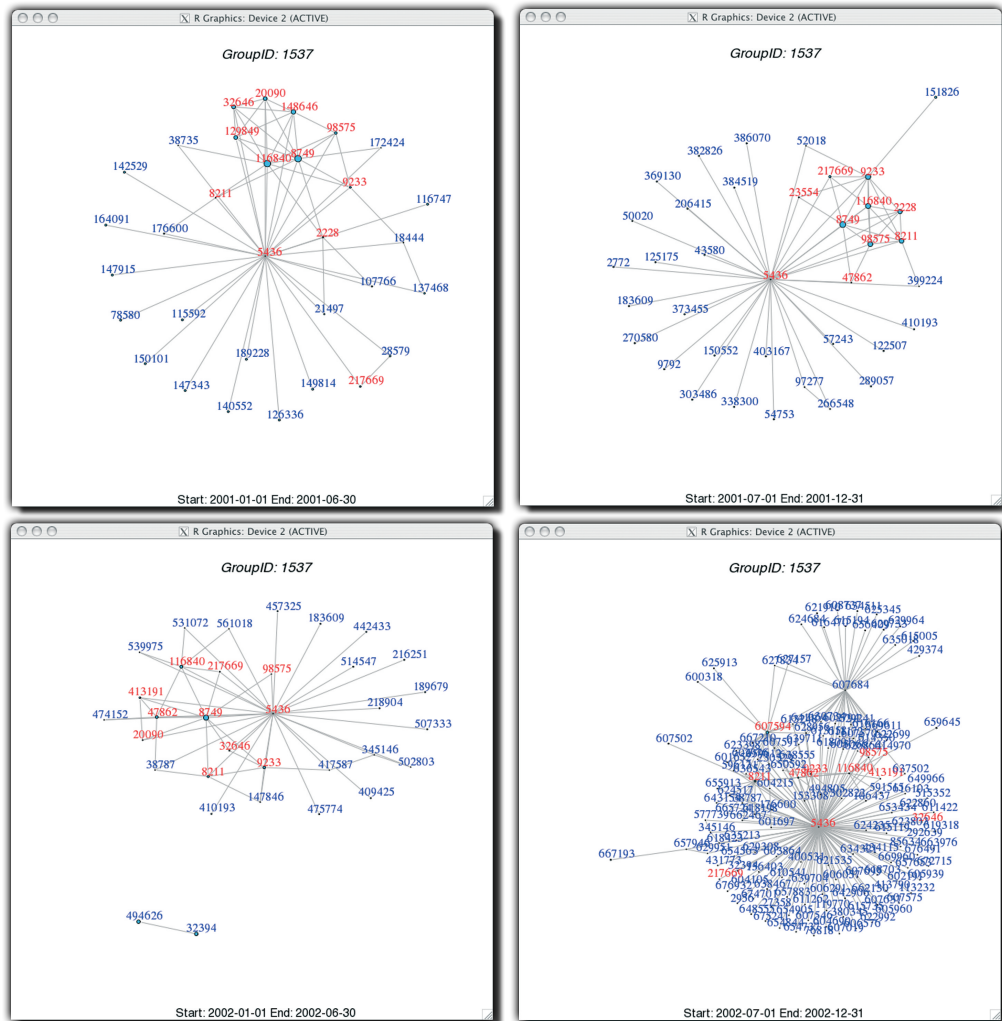


Figure 10.4: Sociograms “Gutenprint” [BW] with data sources: bug response and revision system checkins

and that is why he attracts that much attention.

Putting the focus back to the relations gained out of revision system enables the following conclusions: Comparing the sociograms in figure 10.4 with sociograms in figure 10.3 lets furthermore assume that the core group is very diverse with low centrality (see figure 10.2) which could indicate a very healthy project. The indicator is not decreasing but is staying relatively stable. The time after 2003 is very hard to be interpreted with the information available. It is quite striking that the centrality in the core team raises. Reason for this development could be a more and more specialization of the developer or an indication of sickness in the project. Specialization could lead to this development with the effect that developers tend to have individual areas with own responsibilities.

**Conclusion** On the basis of this example the possibility to reach different roles in the project team using the two available data sources separately was introduced. It was argued that although the graph centrality indicators were surprisingly high it was not necessarily a sign of sickness in this project. It could also be representative for a special role allocation in the project by having a few actors who are responsible for managing bug responses. Those people automatically attract a lot of attention. A kind of a proof of principle of this thesis can be seen in figure 10.4 which shows the red (developer) and blue (bug reporter) actor. In every period one actor is very central and surrounded by a lot of blue actors. This fact also leads to high graph centrality. Putting the information in relation to figure 10.4 which shows the core developer solely could be a validation for this assumption.

## 10.2 Convalescent Project

The aim of this category is to identify changes of a development team and research the influence to project health. According to Crowston et al. [CH06] changes within the core team can be very difficult to proceed if the dependencies within a project are very high (see section 5.3.3) . An example with the project “SquirreL SQL Client” [BW] was found where such a transition was successfully managed. What is essential for this example is the fact, that this project shows the typical developments of such a transition within the core team. It is important for the author to mention, that he was not aware of this fact when

## 10 Evaluation

he was initially looking at the project. He has analyzed the project due to find inconsistencies.

**Project description** According to [BW] Squirrel is a graphical SQL client developed in Java. It is possible to view the structure of a JDVC compliant database, browse the data in tables and use SQL commands to perform queries. The project was initially registered on 2001-5-31 at sourceforge [sou] by the founder of the project Colin Bell. The project has about 25 registered developers with commit privilege to the code repository.

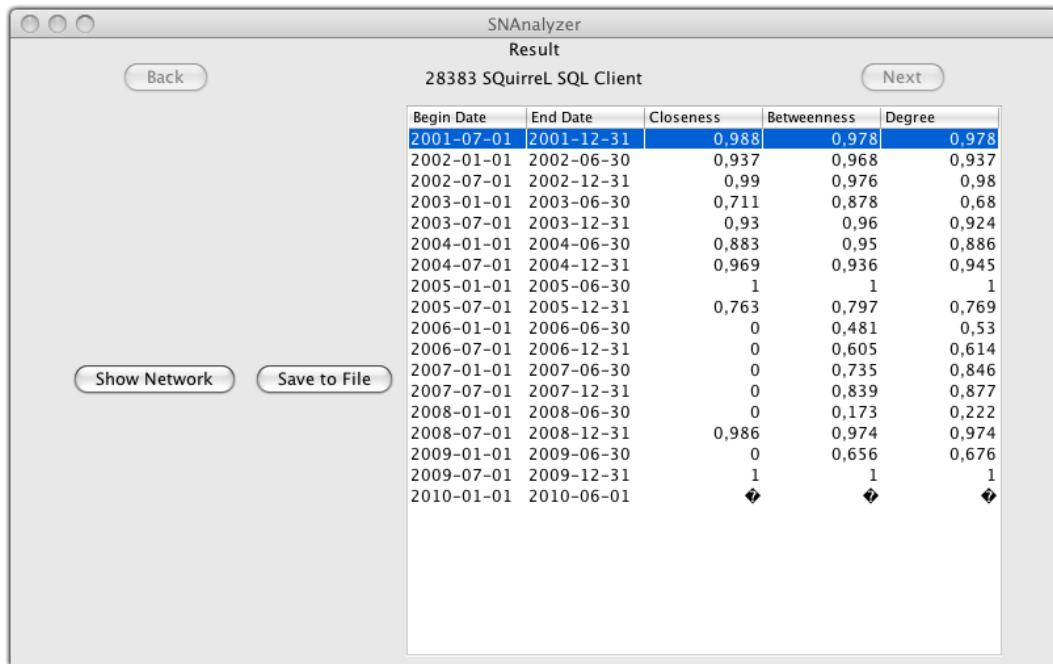
**Datasource** Basically two different data sources have been used for the analysis. Like it was the case with all other projects bug reports out of subversion were used. Due to the fact, that passing time a transition from CVS to Subversion has happened, it was necessary to merge the information out of the CVS and Subversion logs in order to get the relevant information. Very interesting is that the results of the analysis could indicate a relation between committers and core developers although one example can not be representative for such a conclusion.

**Results** The challenge of this specific example was to understand and declare two different events in the development of this project. Looking at figure 10.5 it is possible to see that the period at the beginning of 2005 is distinct to the time before. Figure 10.6 depicts this in a more detailed view where the change is illustrated in a slideshow of sociograms. The sociogram in the corner down left shows very low activity. Out of the view of this initial analysis it seems that a certain very important developer has left the project and kind of a vacuum had established. The same phenomenon can be found in the period from 2008-1-1 to 2008-6-30.

It was tried to find the reasons for this development and after contacting the community of Squirrel it was possible to get an answer. According to Robert Manning, who is core developer in the project Squirrel, the project was founded by Colin Bell. Bell dropped out of the project in late 2004 which is the reason for drastic change in the indicators. Manning also supported the conviction that the project went to sleep for a while until new leaders have taken over the job.



## 10.2 Convalescent Project



The screenshot shows a window titled "SNAAnalyzer" with a "Result" section for "28383 Squirrel SQL Client". The table displays centralities for various time periods from 2001 to 2010. The first row is highlighted in blue. The table has columns for Begin Date, End Date, Closeness, Betweenness, and Degree. There are also buttons for "Back", "Next", "Show Network", and "Save to File".

Begin Date	End Date	Closeness	Betweenness	Degree
2001-07-01	2001-12-31	0,988	0,978	0,978
2002-01-01	2002-06-30	0,937	0,968	0,937
2002-07-01	2002-12-31	0,99	0,976	0,98
2003-01-01	2003-06-30	0,711	0,878	0,68
2003-07-01	2003-12-31	0,93	0,96	0,924
2004-01-01	2004-06-30	0,883	0,95	0,886
2004-07-01	2004-12-31	0,969	0,936	0,945
2005-01-01	2005-06-30	1	1	1
2005-07-01	2005-12-31	0,763	0,797	0,769
2006-01-01	2006-06-30	0	0,481	0,53
2006-07-01	2006-12-31	0	0,605	0,614
2007-01-01	2007-06-30	0	0,735	0,846
2007-07-01	2007-12-31	0	0,839	0,877
2008-01-01	2008-06-30	0	0,173	0,222
2008-07-01	2008-12-31	0,986	0,974	0,974
2009-01-01	2009-06-30	0	0,656	0,676
2009-07-01	2009-12-31	1	1	1
2010-01-01	2010-06-01	◆	◆	◆

Figure 10.5: Centralities of the project “Squirrel SQL Client” [BW]

10 Evaluation

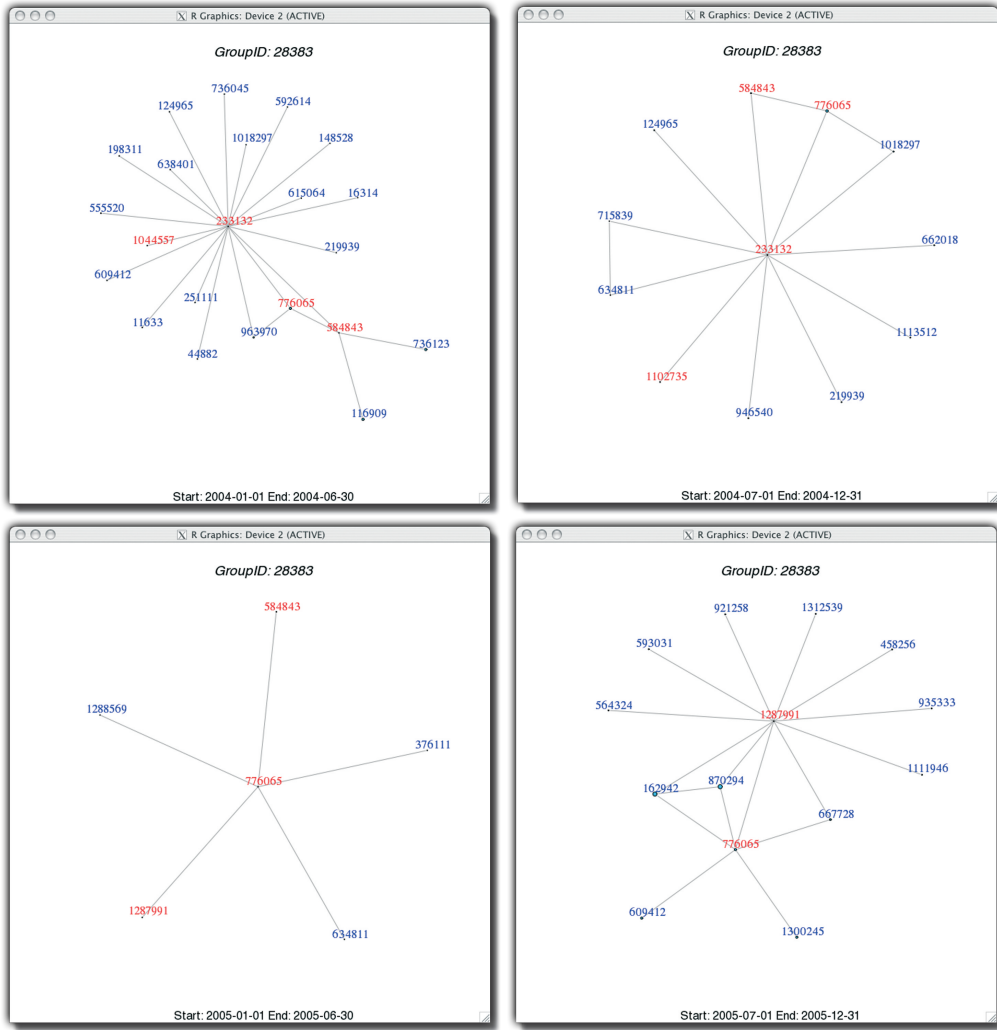


Figure 10.6: Sociograms of “Squirrel SQL Client” [BW] - drop out of an actor

Concerning the event in 2008 Manning told, that he went off the radar out of different reasons.

To provide a better possibility for researching this event an additional view will be introduced. Until this point of the thesis it was worked with indicators on basis of a whole network. A total network relies on some certain actors if the centrality indicator is high. Like it was described in chapter (see point vs. graph centrality 7.5.1) these indicators can also be used on actor level which means that there is one parameter which represents the importance of a specific actor in a network. If the indicator of a certain actor is high it indicates that this specific actor is important for the project. Figure 10.7 illustrates centralities on the basis of actors. For each actor within a network a separate centrality indicator for each time period can be calculated. Putting this information into a diagram the following information can be seen:

- The importance of each actor with the change over time is illustrated
- The characteristic of each line says something about the continuity of the participation of each contributor
- With low significance something about role allocation can be seen
- Relations between specific actors can be identified. If somebody reduces work temporarily, who takes over the work meanwhile?

The usage for degree centrality in the case of actor centrality lies in the fact that because of the small size of the network important actors are characterized by having most of the relations. They are simply doing most of the job. Out of the view of this thesis approaches like a gatekeeper in the case of betweenness would not work with this example.

**Conclusion** The example of SQuirreL was used to introduce the difference between actor and graph centrality. This approach had the aim to identify the effects of the drop out of a very central member of a project who was in this example also the founder of the project. Due to prominent position of Colin Bell (see figure 10.7 point *a*)) in the social network the establishment of a vacuum in project leadership (point *b*)) took place until Robert Manning together with Gerd Wagner have taken over the leadership (point *c*)).

Very interesting is also the fact that when Bell dropped out of the project it

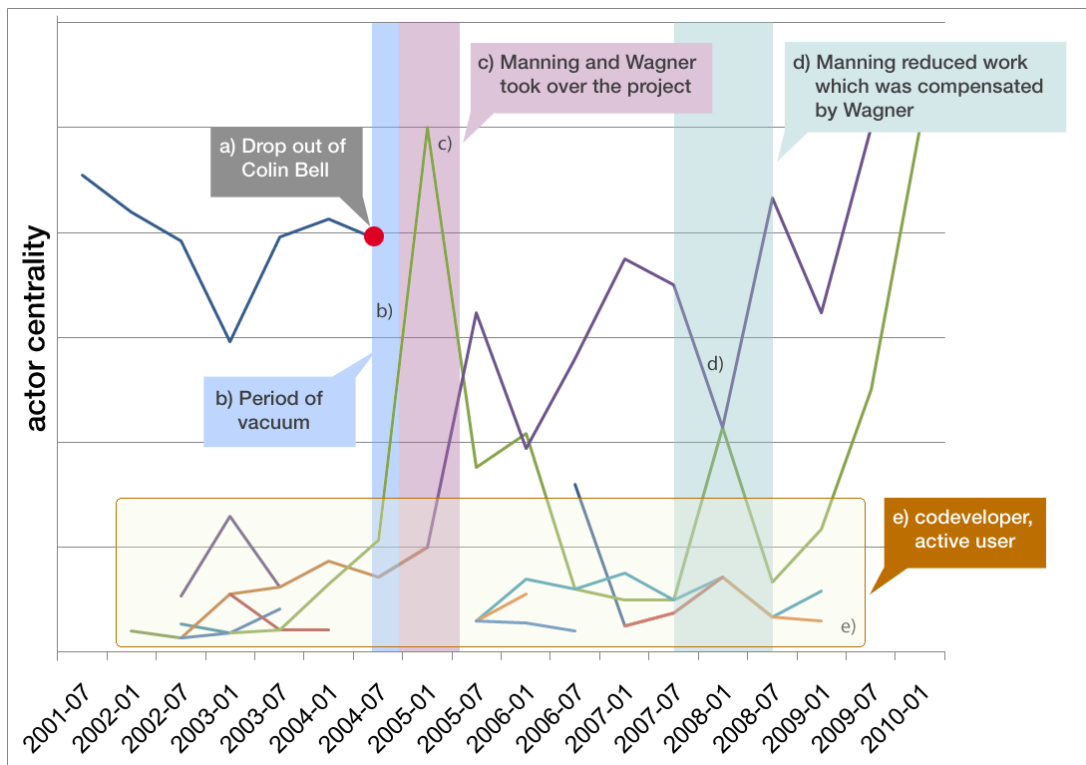


Figure 10.7: Degree actor centrality of the project “Squirrel SQL Client” [BW]

### 10.3 High Dependency - High Risk!

resulted in a status close to dying (compare figure 5.1) whereas the project was set up already that stable that the temporary retirement of Manning was partly absorbed by Wagner (point *d*) in figure 10.7) which can be seen in figure 10.7. At the bottom of the figure the activity of codevelopers and active users is illustrated and it provides information about the continuity of actors out of this group.

## 10.3 High Dependency - High Risk!

Content of this example is the aim to illustrate the risk which comes with high dependencies on a few specific actors. Although the majority of all projects hosted at sourceforge tend to base on a small developer group (see section 5.3.3) it is not said, that such projects are sick per definition. HSQLDB proves this by a continuous development progress.

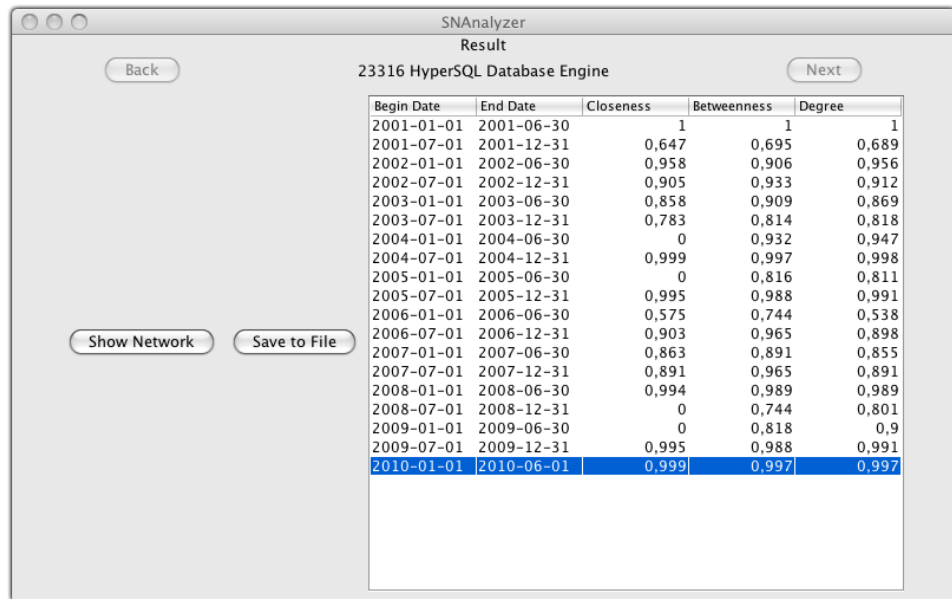
**Project description** According to the project description at sourceforge, HSQLDB [TS] is a relational database engine developed in Java. Advantages are described as being a fast and multithreaded engine and server with features like transaction isolation and encryption.

**Datasource** HSQLDB uses Subversion as revision control system which was loaded into the database of SNAlyzer. Furthermore the bug reports were used as datasource in this example.

**Results** Although figure 10.8 again shows high centrality in all indicators, it nothing says about the structure of the underlying project team. High centrality in this situation just declares, that there has to be a specific person who is responsible for bug responses. When looking at the revision system data let soon one recognize that the project basically relies on the work of a few people containing Fred Toussi and Blaine Simpson.

It was already declared that such projects necessarily do not have to be sick rather the majority of all projects within sourceforge have a small developer base. Although a stakeholder has to be aware of the risk, high dependencies provide which was already shown in section 5.3.3.

## 10 Evaluation



The screenshot shows the SNAAnalyzer application window. The title bar reads 'SNAAnalyzer'. Below the title bar, there are 'Back' and 'Next' buttons. The main content area is titled 'Result' and '23316 HyperSQL Database Engine'. It contains a table with the following data:

Begin Date	End Date	Closeness	Betweenness	Degree
2001-01-01	2001-06-30	1	1	1
2001-07-01	2001-12-31	0,647	0,695	0,689
2002-01-01	2002-06-30	0,958	0,906	0,956
2002-07-01	2002-12-31	0,905	0,933	0,912
2003-01-01	2003-06-30	0,858	0,909	0,869
2003-07-01	2003-12-31	0,783	0,814	0,818
2004-01-01	2004-06-30	0	0,932	0,947
2004-07-01	2004-12-31	0,999	0,997	0,998
2005-01-01	2005-06-30	0	0,816	0,811
2005-07-01	2005-12-31	0,995	0,988	0,991
2006-01-01	2006-06-30	0,575	0,744	0,538
2006-07-01	2006-12-31	0,903	0,965	0,898
2007-01-01	2007-06-30	0,863	0,891	0,855
2007-07-01	2007-12-31	0,891	0,965	0,891
2008-01-01	2008-06-30	0,994	0,989	0,989
2008-07-01	2008-12-31	0	0,744	0,801
2009-01-01	2009-06-30	0	0,818	0,9
2009-07-01	2009-12-31	0,995	0,988	0,991
2010-01-01	2010-06-01	0,999	0,997	0,997

Below the table, there are 'Show Network' and 'Save to File' buttons.

Figure 10.8: Centralities of the project “HSQLDB” [BW]

Figure 10.9 furthermore provides other very interesting information. Basically just actors are considered who have participated in more than two periods. What is left provides information about actors who have a position within the network which should be further analyzed. The two core developer were already identified. Out of the view of the information provided it looks like that the roles within the projects were assigned that way that Toussi is responsible for answering to bug reports what declares the high centrality. The lack of capacity by Toussi in the period at the beginning of 2006 was neutralized by Simpson (see point *a*) in figure 10.9) All other participants at the bottom of the project could be co-developer who don't have a strong position within the network but provide more or less constant participation.

**Conclusion** Although the majority of FLOSS projects have this small fundament it is not obligatory that they produce bad software only because of this conditions. What can be learned out of this project deals with risk awareness regarding high dependencies in FLOSS projects.

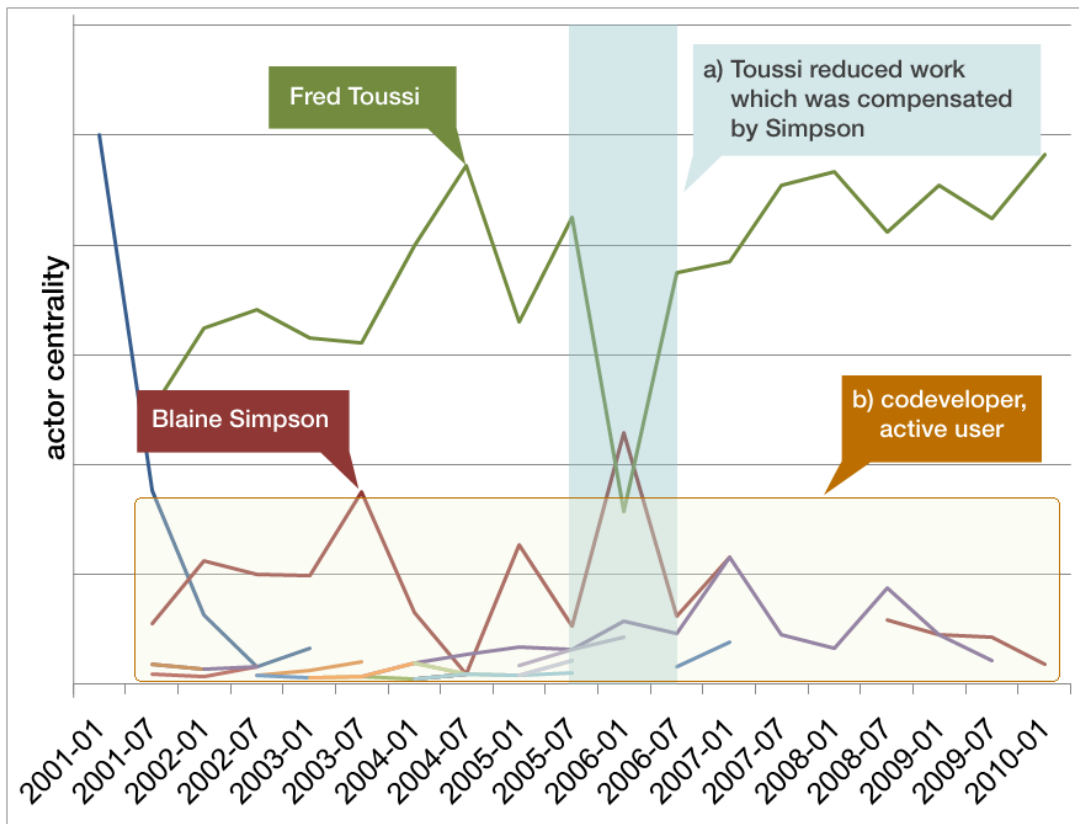


Figure 10.9: Degree actor centrality of the project “HSQLDB” [TS]

Out of a technical view again the approach to plot historic actor centralities was used in order to identify the kind of participation and relevance of each actor.

## 10.4 Collateral Damage

As a last example it is referred to a project where Social Network Analysis would not really have helped. XDoclet enabled attribute-oriented programming in Java but with the introduction of annotations in Java version 5.0 this project became obsolete. The reason for this example can be found in the effort to sensitize the awareness of project failing out of other groundbreaking influence.

## 10 Evaluation

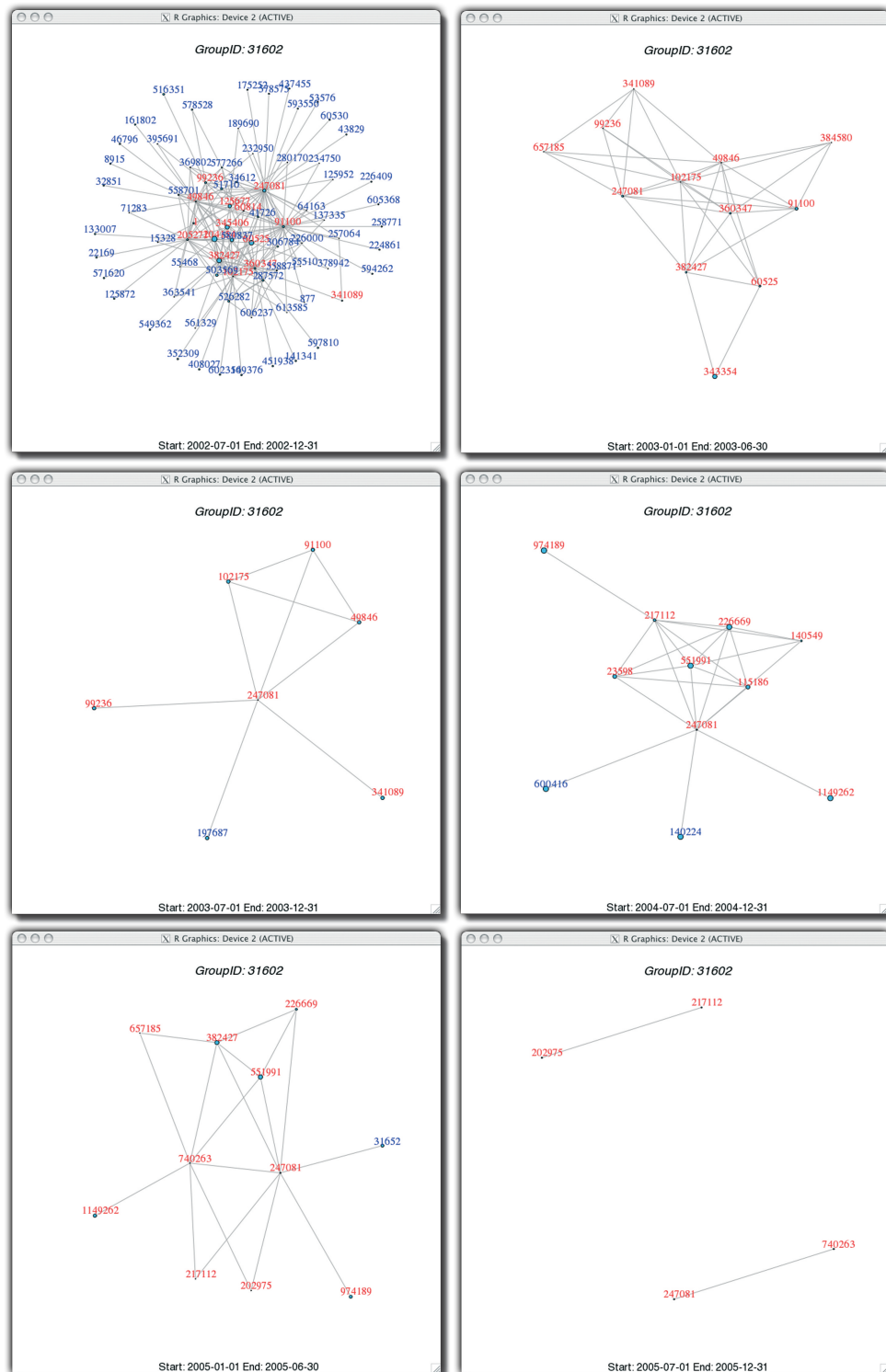


Figure 10.10: Sociograms of “XDoclet” [TS] with data sources: bug response and revision system checkins



**Datasource** Initially revision control and bug report information have been used for analyzation of the project status. After the project changed the bug tracker no more information was available. It was necessary to deal with that constrictions and to focus on the information provided by the project.

**Results** As it was already declared, the information out of bug reports ends with the period of 2002 which can be seen in figure 10.10. At that point the project changed its bug tracker tool and. Out of the information available at the beginning of the project's life cycle nothing special can be identified. The project seems to be quite healthy showing the typical structure with a diverse set of core developers and a lot of active users committing bugs.

In the period ranging from June 2005 to December 2005 it is possible to identify the break down of the project. What was happening was that Java 5.0 which was released at the end of September 2004 [Orab] removed the basic reason for the existence of HSQLDB. It was not possible to see the if the developers of HSQLDB have known about the actions within Java 5.0. What can be said is that the death of HSQLDB would have been hardly predictable with the information provided by Social Network Analysis. Generally this example should be representative for all kinds of influences which put the further existence of the project at risk. Not everything is predictable what is happening within a project.

**Conclusion** The last example should point out that blind faith in technical indicators is not always adequate. It is always necessary to keep eyes open and consider possibilities which could influence the development of FLOSS projects.

## 10 *Evaluation*

# 11 Summary

The aim of this thesis was to evaluate the possibility the instruments of Social Network Analysis provide in order to draw conclusions describing the health status of FLOSS projects. Therefore the reader was introduced into the relevant information of open source software (see chapter 4) containing basics about the development process and the structure of FLOSS projects.

The very complex instruments of Social Network Analysis (see chapter 7) were described as a next step. Different indicators measuring the dependencies within a social network and the importance of specific actors were introduced which prepared for the practical implementation containing the fusion of the concept project health and Social Network Analysis. Questions like which data source would be necessary to be able to research project health and what kind of information is available were discussed. The tool “SNAnalyzer” which was developed within this thesis was equipped with information out of two different kinds of data sources: information out of the bug reporting process and information out of revision systems. Each of the projects which were analyzed was selected out of a specific reason. The aim was to introduce additional views for interpretation and to give answers to the research questions raised in chapter 6.

**Role identification** The first example named “Gutenprint - Top Quality Printer Drivers” was used in order to show how the ideal development of a healthy project could look like. This project was used to introduce an additional view using information out of the revision system and the bug reports separately in order to identify different roles with the help of Social Network Analysis. Like it was the case in this example, some members were responsible for the bug reports and that was why their centrality indicator was higher than the rest. It was also possible to identify members of the core group by analyzing the continuity of their participation.

## 11 Summary

**Structural changes** Convalescence in the sense of changes in the health status was the topic of the second example. “SQuirreL SQL Client” was researched with the aim to identify structural changes within the core team. Two phases in the development of this project were observed which showed obvious abnormality. It was assumed that the reason for this effect to the indicator resulted out of change within the community. In order to verify the assumptions the project team was contacted. With the help of Robert Manning who is a project member it was possible to validate the hypothesis about transitions in the project leadership. For the first time it was possible to use Social Network Analysis in order to positively identify such a development.

Besides that this example was used to introduce an additional view dealing with the difference between point and graph centrality (see chapter 7.5.1). By plotting the historic development of actor centrality the information which was needed in order to identify the above mentioned changes can be provided.

**Dependency and Risk** The relation between dependency and risk for project failing was the topic of the third example. HSQLDB which is a successful project already for a longer time has the obvious weakness of being too dependent on specific members of the project. If one person would drop out of the project it would hardly be possible for the other developers to compensate this loss. Fact is that the majority of all projects within sourceforge have to fight against the same problem of having high dependencies within a project. If a transition in the core community takes place it could happen the way like it was shown in the example of SQuirreL. The community could fall into a vacuum and it could need time until new leaders emerge. Such changes could also lead to the death of a project if nobody takes over the project.

**Environmental influences** The practical part was finalized with an example which should illustrate that blind faith into technical approaches is not the best solution. The black swan theory [Tal07] which suggests to consider the inconsiderable deals with the handling of such risks. The author does not like to loose the focus he just suggests to keep a rest of sensitiveness and a sense for the kind of information which is analyzed.

**Reference for closed source projects?** As it was announced in the introduction of this thesis a short comment on the usability of gained knowledge for proprietary software development will be given. Basically Social Network Analysis is not limited to any kind of information. Everything which describes a social relationship can be analyzed hence closed source software development is not excluded. Project health analysis using Social Network Analysis from outside a company simply has to deal with the problem that information which will be necessary is not or just little available out of reasons which were described in the introduction of this thesis (see chapter 2). FLOSS software development provides some ideals (which were used in order to get to the results of this thesis) closed source projects do not have. The development process mentioned in this thesis bases on the content of a tracker tool and follows the aim of continuous improvement. Proprietary software projects base on other values and use other development processes (see also table 4.3 which describes the main differences between FLOSS and proprietary software development). In such a case the instrument needs to be adopted to the specific requirements. If a company decides to research dependencies with Social Network Analysis out of an internal point of view other constrictions have to be considered. Fact is, that the gained knowledge within this thesis does not have to be valid in the case of proprietary software development due to the following reasons:

- Closed source software projects have other development standards
- Closed source software projects have other development processes
- Communication is not only constricted to electronic media which can be read out like in the case of FLOSS (e.g. coffee talk) which makes it more important to evaluate the quality of the data source
- The structures of FLOSS project teams are different compared to those in closed source software projects

## *11 Summary*

## 12 Critical Discussion

This thesis introduced new possibilities to evaluate project health in FLOSS projects. It was shown how to identify certain developments of a project and it was researched how the status of project health could be influenced. Due to the promising results it is needed to continue with a more comprehensive research in order to validate the conclusions drawn within this thesis.

Furthermore two different questions have to be raised when thinking critically about the methods and the results of this thesis. The first question deals with the representativeness of the result out of a quantitative consideration. On the other hand it is necessary to ask if the data sources used in this thesis represent adequate information.

**Data quantity** On a scientific level it was not possible to validate or reject the assumptions raised in this thesis because of the focus on some special projects. That is why only findings or evidences based on profound knowledge described in the reference work were pointed out. For more precise conclusions on project health it would be necessary to research a larger set of projects in order to evaluate the ideas introduced in this thesis.

With respect to the research questions it was necessary to adjust some ideas. Out of the gained knowledge an empiric value for centrality which indicates project health or sickness was not possible to determine. The reason for this could lie in the fact that there seems to be too much changes in activity within the projects which influences the indicators but can not be further resolved. Also the data amount was not stable enough to evaluate a constant decline in order to positively evaluate project health with the underlying assumption.

What can be gained out of this assumptions is the fact that a significant change of this parameter can admit valuable information about the project. The example SQuirreL shows that this is a very important kind of information which

## *12 Critical Discussion*

needs to be traced back because it could represent an evidence for structural changes in the community.

**Data quality** Sometimes it it was hard to get the information needed in order to perform Social Network Analysis in an educated manner. It was necessary to deal with information drawbacks or other times was simply not enough activity in the underlying data source. The question has to be raised if the data sources used within this thesis really represents the total activity which is necessary to reconstruct the happenings within the project. What if the project has shifted towards other technology and this information was not available? Conclusion on the basis of such sources are very risky. It is suggested to look for reference measures in order to validate the informative value of the indicators within the project health approach.

To avoid such problems the specific projects have to be researched and the gained knowledge has to be critically analyzed. It was tried to act that way when the project community in the example of SQuirreL was contacted in order to discuss the findings.



## 13 Suggestions for Future Work

This thesis shows the basic potential of Social Network Analysis as an instrument in order to identify the health status of a project. Basically it would be important to discuss the findings on a wider basis. More projects should be investigated in order to try to validate the concepts identified in this thesis. The question has to be resolved if Social Network Analysis could cope with the requirements to be a standard instrument for the analysis of the health status of FLOSS projects. Like it has already happened within the example of “Squirrel SQL Client” it is necessary to do more validation on the findings presented in this thesis by contacting the specific community.

Further studies can go in two different directions: From a technical point of view possibilities can be identified in how the instruments within Social Network Analysis can be adopted in order to get a higher informative value. At least it has to be cleared if more detailed information can be extracted by the usage of directed relations. Probably they could indicate a more detailed view to the happenings within the project because it is possible to differentiate between indegrees and outdegrees. It has to be evaluated with practical examples if the usage of directed relations provide a better basis in order to identify roles and hierarchies within a community. The problem which comes along with this idea deals with more constrictions to possible data sources because the data sources have to be directed as well like it is the case in a forum or in mailing lists.

Another possibility which can be suggested would be to add weights (like the number of exchanged messages) to relations. Like it was the case in this thesis not the number of active happenings was relevant because the focus was different. The reason for this simplification in this thesis was the aim of detecting basic changes in the social structure of a project team over a longer period of time. Adding information about the activity would provide an additional kind of information.

*13 Suggestions for Future Work*

# A SNAalyzer

To generate the relevant measures out of the data which is available through the revision system or the bug reports an own tool called “SNAalyzer” [Kal] was developed. The basic functions of this application contain the extraction of the relevant data out of the local database, the preparation for calculating the numbers and the data exchange with the statistic environment R and Java.

The application dialog consists of four steps which have to be passed in order to retrieve a result. It is possible to switch between all steps until the button “Calculate Centrality Measures” is pressed.

## A SNAlyzer

### Step 1

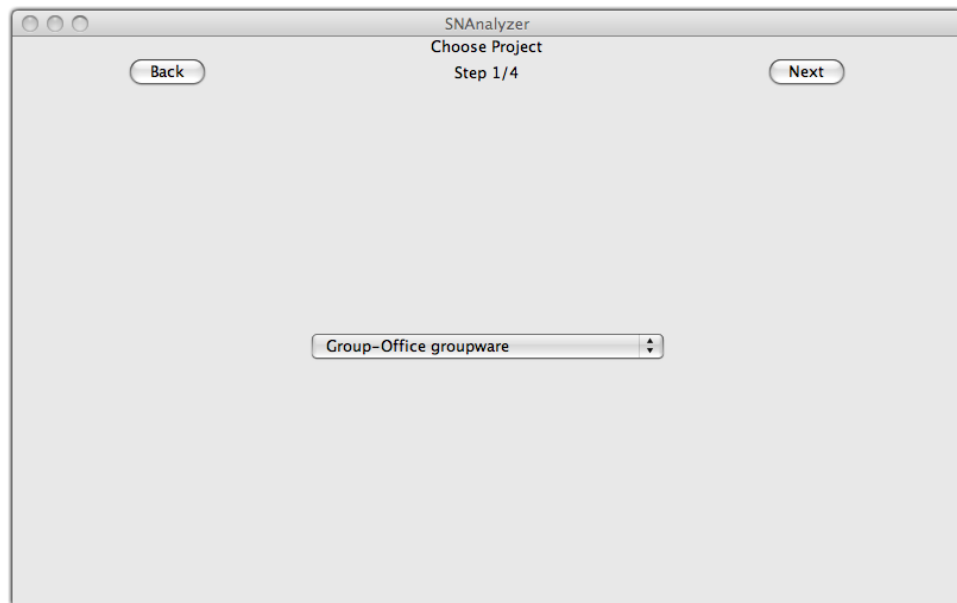


Figure A.1: SNAlyzer - Step 1

As it can be seen in figure [A.1](#) the first step provides a selection of all projects which are currently loaded into the system. In order to load a project into the system, the data has to be extracted out of the datadump and the revision system's log files. A parser extracts the information within this files and writes it into the database.

## Step 2

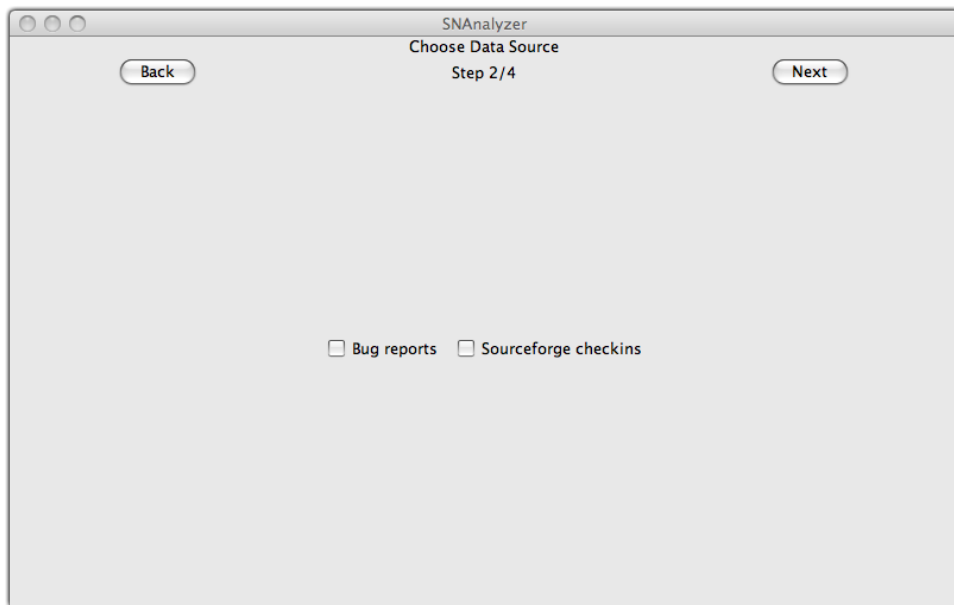


Figure A.2: SNAlyzer - Step 2

Figure A.2 shows a selection of the possible data sources for an analysis. This is necessary in order to use both sources separately. The differentiation provides further possibilities in order to research the structure of FLOSS teams.

### Step 3

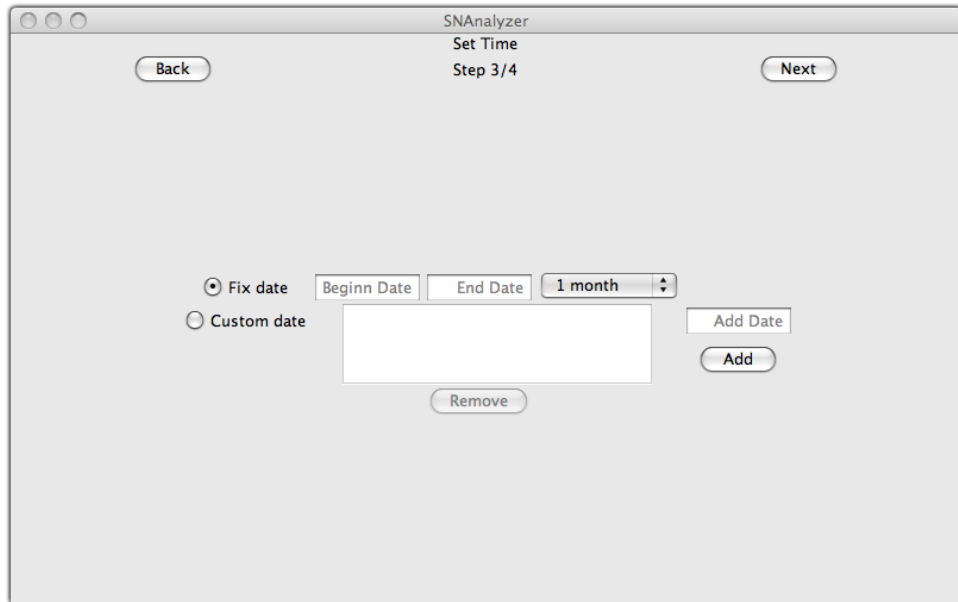


Figure A.3: SNAalyzer - Step 3

Figure A.3 provides the possibility to specify time frames in order to perform the analysis. Basically dates have to be entered in the following form: YYYY-MM-TT

The interface provides two different approaches. It is possible to enter the beginning and the end of the time frame. By defining the length of a period the system calculates the beginning and the end of each period automatically.

The second possibility is to specify periods manually. The difference in two consecutive dates out of the list describes the beginning and the end of a observation period. This approach enables to set up a custom time frame with custom period lengths.

## Step 4

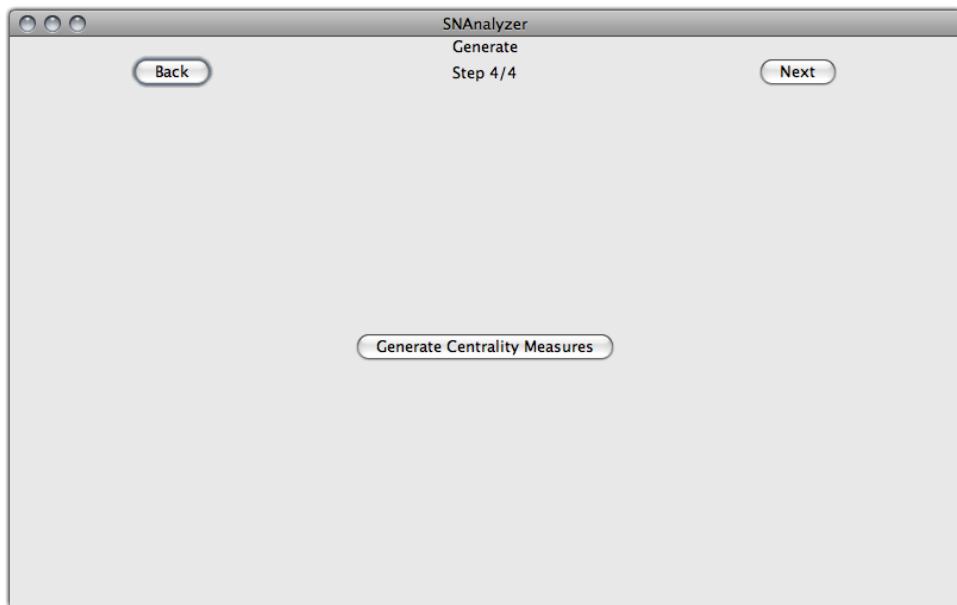
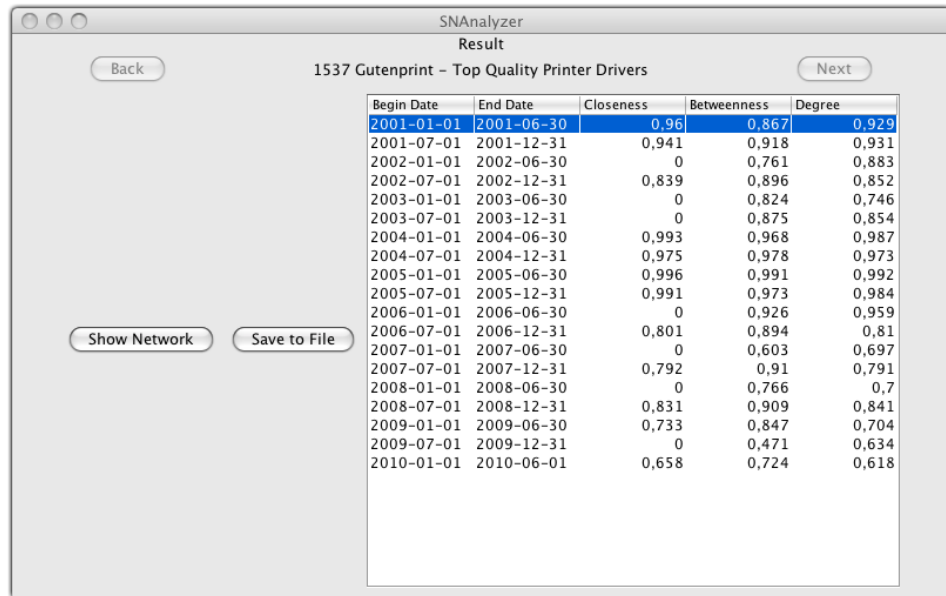


Figure A.4: SNAlyzer - Step 4

After clicking the “Generate centrality measures” the application starts to collect relevant data in order to calculate the different centrality measures with the help of the statistical environment R.

## A SNAlyzer

### Result



The screenshot shows the SNAlyzer application window with the title "SNAlyzer" and "Result" displayed. Below the title, it says "1537 Gutenprint - Top Quality Printer Drivers". There are "Back" and "Next" buttons at the top. A table of results is shown with the following data:

Begin Date	End Date	Closeness	Betweenness	Degree
2001-01-01	2001-06-30	0,96	0,867	0,929
2001-07-01	2001-12-31	0,941	0,918	0,931
2002-01-01	2002-06-30	0	0,761	0,883
2002-07-01	2002-12-31	0,839	0,896	0,852
2003-01-01	2003-06-30	0	0,824	0,746
2003-07-01	2003-12-31	0	0,875	0,854
2004-01-01	2004-06-30	0,993	0,968	0,987
2004-07-01	2004-12-31	0,975	0,978	0,973
2005-01-01	2005-06-30	0,996	0,991	0,992
2005-07-01	2005-12-31	0,991	0,973	0,984
2006-01-01	2006-06-30	0	0,926	0,959
2006-07-01	2006-12-31	0,801	0,894	0,81
2007-01-01	2007-06-30	0	0,603	0,697
2007-07-01	2007-12-31	0,792	0,91	0,791
2008-01-01	2008-06-30	0	0,766	0,7
2008-07-01	2008-12-31	0,831	0,909	0,841
2009-01-01	2009-06-30	0,733	0,847	0,704
2009-07-01	2009-12-31	0	0,471	0,634
2010-01-01	2010-06-01	0,658	0,724	0,618

At the bottom of the window, there are "Show Network" and "Save to File" buttons.

Figure A.5: SNAlyzer - Results

The last figure (figure A.5) represents the result of the specific example. The listing contains the periods and the value for the centrality measure. The button “Show network” exports the relevant information to R and draws a sociogram of the selected period. Actors who origin from the revision system data source will be labeled with a red color in the sociogram whereas actors aggregated out of bug reports are marked with a blue color (like it can be seen in figure A.6). Furthermore an export function was implemented which makes it possible to save the generated information into a text file in order to provide data for a later analysis.



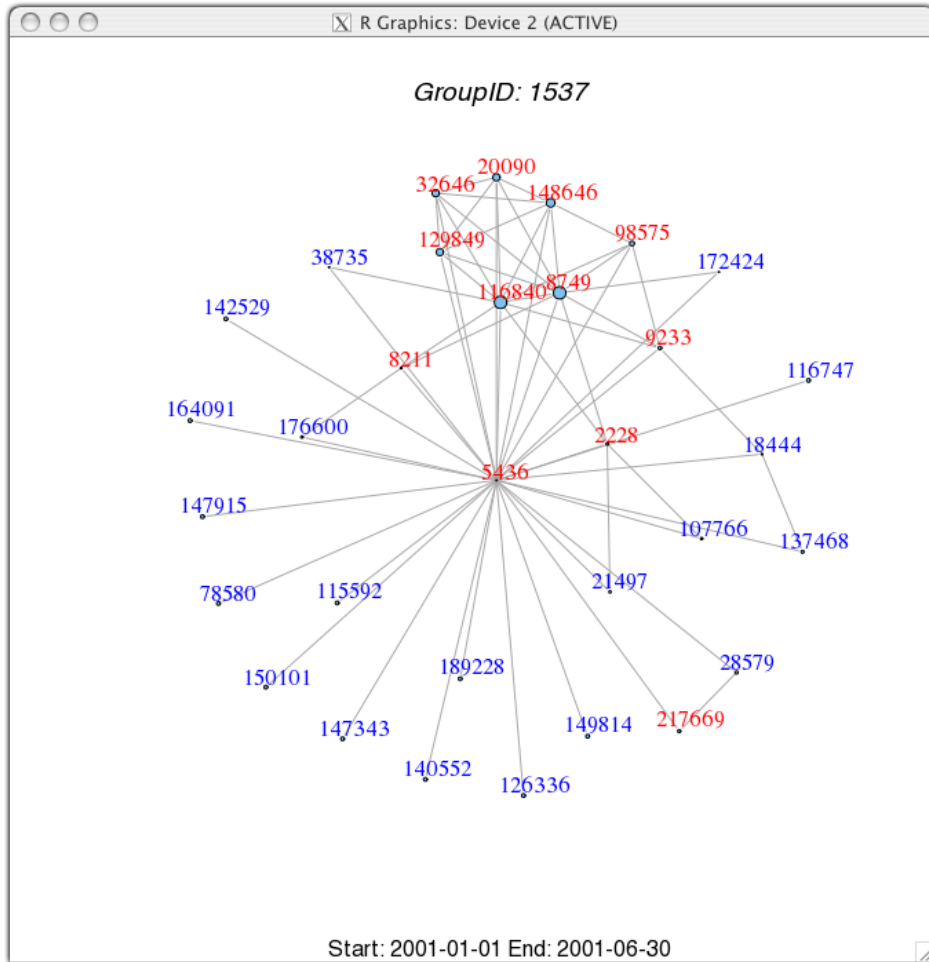


Figure A.6: Sociogram Gutenprint

*A SNAlyzer*

## B How to use the statistic environment R within a Java application

Due to the experiences the author has gained within this thesis he likes to provide this information to other people dealing with the same issue. It was quite striking to set up R in order to use it within a Java application. Due to the fact that little information for troubleshooting was available this knowledge is provided within the appendix of this thesis.

**Preconditions** Eclipse<sup>1</sup> was used as development environment. Before dealing with the setup of the development environment, R needs to be installed and the following relevant packages have to be added:

- JRI which is now part of rJava enables the communication between Java and R. The release version is available from CRAN (The Comprehensive R Archive Network) and can usually be installed using the command:

```
install.packages("rJava")
```

Further information is available at [RFo].

- Statnet which is a suit of packages for Social Network Analysis is used in order to calculate the required centrality measures. More information is available at [GHH<sup>+</sup>08]. For installing the package simply enter:

```
install.packages("statnet")
```

- igraph is used in order to draw the sociograms describing the relations between actors. Information is available at [sou]. A installation can easily be initiated with the following command:

---

<sup>1</sup>Eclipse is a software development environment available at <http://www.eclipse.org/>

## B How to use the statistic environment R within a Java application

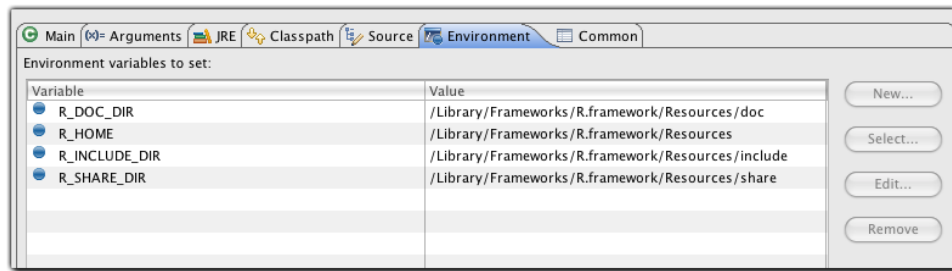


Figure B.1: Eclipse - Run Configurations

```
install.packages("igraph")
```

**Eclipse Setup** The challenge in setting up of the framework was to specify the right directories which contain the libraries relevant to access to R. Mac OS was used as operating system. After adding a run configuration of the type “Java Application” with the environment variables set to the values, available in figure B.1 it was possible to run the software. Critical was the fact, that R and the libraries had to be installed that way that the installation was accessible for every user in the system.

Further tests made it necessary to set up the testing environment on a windows based computer where the configurations looked like a bit different. It was necessary to add a variable to the environment leading to the directory where R is installed (see figure B.3). Furthermore it was necessary to specify the directory, containing the libraries of jri through the build path (see figure B.2).

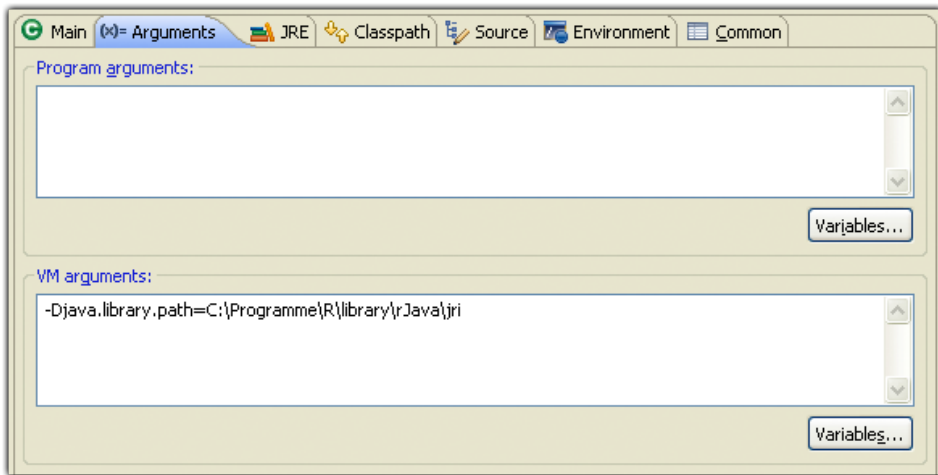


Figure B.2: Eclipse - Arguments Windows

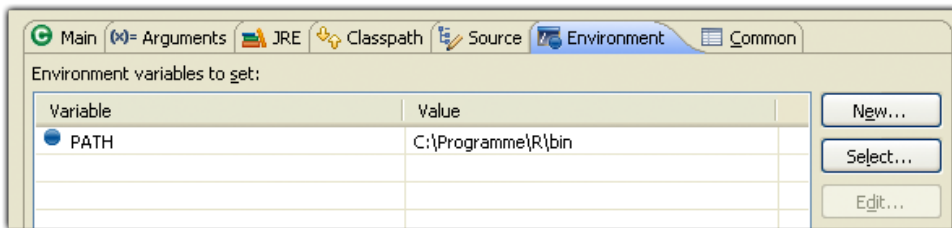


Figure B.3: Eclipse - Environment Windows

*B How to use the statistic environment R within a Java application*

# Bibliography

- [AAB<sup>+</sup>06] Philippe Aigrain, Roberto Andradas, Raphaël Badin, Renaud Bernard, Luis Cañas Díaz, Paul David, Santiago Dueñas, Theo Dunnewijk, Rishab Aiyer Ghosh, Ruediger Glott, Jesus Gonzalez-Barahona, Kirsten Haaland, Bronwyn Hall, Wendy Hansen, Juan Jose Amor, Huub Meijers, Alvaro Navarro, Francesco Rentocchini, Gregorio Robles, Barbara Russo, Giancarlo Succi, and Adriaan van Zon. Study on the: Economic impact of open source software on innovation and the competitiveness of the information and communication technologies (ict) sector in the eu, November 2006.
- [And72] P. W. Anderson. More is different. *Science*, 177(4047):393–396, August 1972.
- [Ant71] J. M. Anthonisse. The rush in a graph. Amsterdam: Mathematical Centre, 1971.
- [Apa0] Apache Software Foundation. <http://www.apache.org/> (2010-02-28).
- [Apab] Apache Software Foundation. Apache incubator. <http://incubator.apache.org/> (2010-02-28).
- [Bav48] A. Bavelas. A mathematical model of group structure. *Human Organizations*, 7:16–30, 1948.
- [Bav50] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.
- [Bea65] M. A. Beauchamp. An improved index of centrality. In *Behavioral Science*, volume 10, pages 161–163, 1965.
- [BM98] Vladimir Batagelj and Andrej Mrvar. Pajek - program for large network analysis. *Connections*, 21:47–57, 1998.

## Bibliography

- [BM08] Vladimir Batagelj and Andrej Mrvar. Pajek. <http://pajek.imfm.si/doku.php> (2010-03-02), May 2008.
- [Bon72] Philip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.
- [BW] Colin Bell and Gerd Wagner. Squirrel sql client. <https://sourceforge.net/projects/squirrel-sql/> (2010-03-03).
- [CH04] Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10, 2004.
- [CH06] Kevin Crowston and James Howison. Assessing the health of open source communities. *Computer*, 39(5):89–91, 2006.
- [Cha05] Scott Chacon. Git. <http://git-scm.com/> (2010-02-02), December 2005.
- [CHC05] Megan Conklin, James Howison, and Kevin Crowston. Collaboration using ossmole: a repository of floss data and analyses. *SIGSOFT Softw. Eng. Notes*, 30(4):1–5, 2005.
- [Cyr01] Cyram Co., Ltd. Netminer. <http://www.netminer.com> (2010-03-02), December 2001.
- [Fou09] Free Software Foundation. The free software definition. <http://www.gnu.org/philosophy/free-sw.html> (2009-07-14), June 2009.
- [Fre77] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, March 1977.
- [Fre79] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1979.
- [GACM07] Yongqin Gao, Matthew Van Antwerp, Scott Christley, and Greg Madey. A research collaboratory for open source software research. In *Proceedings of the 29th International Conference on Software Engineering + Workshops (ICSE-ICSE Workshops 2007)*, Minneapolis, MN, May 2007.
- [Gat76] Bill Gates. An open letter to hobbyists. *Computer Notes*, 1(9):1, 1976.



- [GHH<sup>+</sup>08] Steven M. Goodreau, Mark S. Handcock, David R. Hunter, Carter T. Butts, and Martina Morris. R - statnet package. In *Journal of Statistical Software*, volume 24, 2008.
- [GJS<sup>+</sup>08] Ido Guy, Michal Jacovi, Elad Shahar, Noga Meshulam, Vladimir Soroka, and Stephen Farrell. Harvesting with sonar: the value of aggregating social network information. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1017–1026, New York, NY, USA, 2008. ACM.
- [GRMMORM03] José Centeno-González Gregorio Robles-Martínez, Jesús M. González-Barahona, Vicente Matellán-Olivera, and Luis Rodero-Merino. Studying the evolution of libre software projects using publicly available data. In *Proceedings of the 3rd Workshop on Open Source Software Engineering, 25th International Conference on Software Engineering*, volume 3, pages 111–115. opensource.ucc.ie, 2003.
- [HC04] James Howison and Kevin Crowston. The perils and pitfalls of mining sourceforge. In *In Proceedings of the International Workshop on Mining Software Repositories (MSR 2004)*, pages 7–11, 2004.
- [HCC06] J. Howison, M. Conklin, and K. Crowston. Flossmole: A collaborative repository for floss research data and analyses. *International Journal of Information Technology and Web Engineering*, 1(3):17–26, 2006.
- [Het06] Maik Hetmank. *Open-Source-Software - Motivation der Entwickler und ökonomischer Hintergrund*. VDM Verlag Dr. Müller, Saarbrücken, 2006.
- [HMS05] Ralf Hölzer, Bradley Malin, and Latanya Sweeney. Email alias detection using social network analysis. In *LinkKDD '05: Proceedings of the 3rd international workshop on Link discovery*, pages 52–57, New York, NY, USA, 2005. ACM.
- [HO02] Alexander Hars and Shaosong Ou. Working for free? motivations for participating in open-source projects. *Int. J. Electron. Commerce*, 6(3):25–39, 2002.
- [HWC06] Liaquat Hossain, Andrè Wu, and Kenneth K S Chung. Actor centrality correlates to project based coordination. In

## Bibliography

- CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 363–372, New York, NY, USA, 2006. ACM.
- [HZ08] Liaquat Hossain and David Zhou. Measuring oss quality through centrality. *ACM*, 2008.
- [Ing02] Lars Magne Ingebrigtsen. Gmane. <http://gmane.org/> (2010-02-11), 2002.
- [JA06] Mohsen Jamali and Hassan Abolhassani. Different aspects of social network analysis. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 66–72, Washington, DC, USA, 2006. IEEE Computer Society.
- [Jec02] Thomas Jech. Set theory. <http://plato.stanford.edu/entries/set-theory/#2> (2009-04-14), 2002.
- [JS06] Kirk Job-Sluder. Automated social network analysis as a tool for evaluating sociability. In *ICLS '06: Proceedings of the 7th international conference on Learning sciences*, pages 940–941. International Society of the Learning Sciences, 2006.
- [Kal] Andreas Kaltenecker. SNAAnalyzer - Performing Social Network Analysis on sourceforge projects. <https://sourceforge.net/projects/snanalyzersf/> (2010-04-20).
- [KB83] D. Knoke and R.S. Burt. *Applied Network Analysis - A Methodological Introduction*. Sage Publications, 1983.
- [Köh30] Wolfgang Köhler. *Gestalt Psychology*. G. Bell and Sons Ltd., London, 1 edition, 1930.
- [Kri02] Sandeep Krishnamurthy. Cave or community? an empirical investigation of 100 mature Open Source projects. *First Monday*, 7(6), June 2002.
- [Lea51] H. J. Leavitt. Some effects of certain communication patterns on group performance. *J Abnorm Psychol*, 46(1):38–50, January 1951.
- [Mad] Greg Madey. The sourceforge research data archive (srda). <http://zerlot.cse.nd.edu/> (2009-07-07).

- [mK] mbroughtn and Robert Krawitz. Gutenprint - top quality printer drivers. <https://sourceforge.net/projects/gimp-print/> (2010-04-03).
- [Net10] Netcraft Ltd. Webserver marketshare. <http://news.netcraft.com/> (2010-01-07), January 2010.
- [New08] M. E. J. Newman. mathematics of networks. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, Basingstoke, 2008.
- [OFN09] Joshua O'Madadhain, Danyel Fisher, and Tom Nelson. Jung. <http://jung.sourceforge.net/> (2010-04-02), April 2009.
- [Omo08] Inah Omoronya. *Sharing awareness during distributed collaborative software development*. PhD thesis, University of Strathclyde, November 2008.
- [Oraa] Oracle Corporation. <http://www.mysql.com/> (2010-03-28).
- [Orab] Oracle Corporation. Five reasons to move to the j2se 5 platform. <http://java.sun.com/developer/technicalArticles/J2SE/5reasons.html> (2010-04-02).
- [Orac] Oracle Corporation. Java™ se 6 release notes supported system configurations. <http://java.sun.com/javase/6/webnotes/install/system-configurations.html> (2010-04-02).
- [ORK02] Margit Osterloh, Sandra Rota, and Bernhard Kuster. Open source software production: Climbing on the shoulders of giants. Arbeitspapier der Universität Zürich, 2002.
- [Pod09] Giacomo Poderi. Legitimate peripheral participation in free and open source software communities of practice — even non developers enter the community. In Walt Scacchi, Kris Ven, and Jan Verelst, editors, *Proceedings of the OSS 2009*, pages 73–82, Skövde, Sweden, June 2009.
- [PS08] Adam Perer and Ben Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 265–274, New York, NY, USA, 2008. ACM.

## Bibliography

- [PX06] Derek Robert Price and Ximbiot. Cvs. <http://www.nongnu.org/cvs/> (2010-02-02), December 2006.
- [R D09] R Development Core Team. R: A language and environment for statistical computing. <http://www.R-project.org> (2009-04-14), 2009.
- [Ray00] Eric S. Raymond. The cathedral and the bazaar. <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/> (2009-06-17), 2000.
- [RFo] RForge. Jri - java/r interface. <http://www.rforge.net/JRI/> (2010-04-03).
- [Sab66] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, December 1966.
- [Sca07] Walt Scacchi. Free/open source software development: recent research results and emerging opportunities. In *ESEC-FSE companion '07: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering*, pages 459–468, New York, NY, USA, 2007. ACM.
- [Sco00] John P. Scott. *Social Network Analysis: A Handbook*. SAGE Publications, January 2000.
- [Sha54] M. E. Shaw. Group structure and the behavior of individuals in small groups. In *Journal of Psychology*, volume 38, pages 139–149, 1954.
- [Shi08] Clay Shirky. *Here Comes Everybody: The Power of Organizing Without Organizations*. Penguin Press HC, The, February 2008.
- [sou] sourceforge.net. <http://sourceforge.net/> (2010-02-28).
- [SSS07] Marco Scotto, Alberto Sillitti, and Giancarlo Succi. An empirical analysis of the open source development process based on mining of source code repositories. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):231–247, 2007.
- [Sta09] Richard Stallman. Why “open source” misses the point of free software. <http://www.gnu.org/philosophy/>

- [open-source-misses-the-point.html](#) (2009-07-14), June 2009.
- [Ste99] Neal Stephenson. *In the Beginning... Was the Command Line*. William Morrow & Co., Inc., New York, NY, USA, 1999.
- [SZ89] K. Stephenson and M. Zelen. Rethinking centrality: Methods and examples. *Social Networks*, 11(1):1–37, 1989.
- [Tal07] Nassim N. Taleb. *The Black Swan: The Impact of the Highly Improbable*. Random House, April 2007.
- [TD02] Linus Torvalds and David Diamond. *Just for Fun*. Dtv, Dezember 2002.
- [The04] Markos Themelidis. *Open Source Die Freiheitsvision der Hacker*. Books on Demand GmbH, Norderstedt, 2004.
- [The09] The Berkeley Institute of Design. The r project for statistical computing. <http://www.r-project.org/> (2010-04-02), January 2009.
- [The10] The Apache Software Foundation. Apache subversion. <http://subversion.apache.org/> (2010-04-02), February 2010.
- [TS] Fred Toussi and Blaine Simpson. Hypersql database engine. <https://sourceforge.net/projects/hsqldb/> (2010-04-03).
- [VAM08] M. Van Antwerp and G. Madey. Advances in the sourceforge research data archive (srda). In *Fourth International Conference on Open Source Systems, IFIP 2.13 (WoPDaSD 2008)*, Milan, Italy, September 2008.
- [Van09] Ashlee Vance. Open source as a model for business is elusive. <http://www.nytimes.com/2009/11/30/technology/business-computing/30open.html> (2010-02-02), November 2009.
- [Wah08] Dindin Sjahril Fadjar Wahyudin. *Quality Prediction and Evaluation Models for Products and Processes in Distributed Software Development*. PhD thesis, Vienna University of Technology, 1030 Wien, Baumgasse 58/43, 11 2008.
- [WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis : Methods and Applications*. Cambridge University Press, November 1994.

## Bibliography

- [Wik10] Wikipedia. Gutenprint — wikipedia, die freie enzyklopedie. <http://de.wikipedia.org/w/index.php?title=Gutenprint&oldid=71850228> (2010-04-03), 2010. [Online; Stand 11. April 2010].
- [Wil02] Sam Williams. *Free as in Freedom: Richard Stallman's Crusade for Free Software*, chapter 13. O'Reilly, 2002.
- [WSM<sup>+</sup>06] Dindin Wahyudin, Alexander Schatten, Khabib Mustofa, Stefan Biffel, and A. Min Tjoa. Introducing "health" perspective in open source web-engineering software projects based on project data analysis. In *iiWAS*, pages 269–278, 2006.