UNIVERSITAT POLITÈCNICA DE CATALUNYA
DEPARTAMENT DE LLENGUATGES I SISTEMES INFORMÀTICS

Research report LSI-08-36-R
http://www.lsi.upc.edu/dept/techreps

Current Trends in Free Software Research

Ramon Navarro Bosch
ramon@epsem.upc.edu

Sebastià Vila-Marta
sebas@lsi.upc.edu

February 4, 2009

Abstract

This paper analyzes the research trends concerning free software. We explore publications about free software in scientific journals and conferences proceedings. The data thus obtained is analized and the most salient trends related to free software discovered. We also reviewed the main works published in each free software research area.

Contents

1 Introduction

This paper analyzes the cientific research related to free software. The main goals are to know which are the interests of the scientific community on this area, which are the topics that concentrate the research and, which are the characteristics of the scientific publications related to free software.

Free software is a phenomenon that has been growing up during the last twenty years. It is widely accepted that free software has achieved an important significance on many areas such as computer science and business. It is also accepted the influence of free software on some government policies. However, in the scientific community it seems not to be a broad consensus about whether this phenomenon is a matter of study. This article is focused on solving this question. Nonetheless, free software is a wide phenomenon. In this work we only consider free software from a computing science point of view.

We do not know of any publication with the same goal as this paper. Still, there are some surveys that report on comparable topics. Krogh and Hippel, in [157], report on free software research issues. They emphasize the following research areas: the motivation for contributing to a free software project; the governance, organization and innovation process in a free software project and the competitive dynamics in a free software project. They conclude that free software has widened the fields of interest of many researchers and practitioners. Scacchi, in [145], discusses about which patterns, practices and techniques are used on free software development projects. He concludes that free source software development offers new kinds of practices, processes and organizational forms. Scacchi notices that this opens an interesting research area. The goal is to discover, observe, analyze, model and simulate these new practices, processes and organizational forms. The paper by Feller and Fitzerald, [69], analyzes the free source development paradigm. They conclude that there are three free software related areas that deserve more research: cross-methodological comparative studies, psychological and sociological inquiries and investigation of economic models and business forms. However, Feller and Fitzerald, do not devise any computer science related area of interest.

In this paper we conclude that there is an increasing free software research activity that mainly embraces the software engineering knowledgment area. That research activity is mature, is published in journals and have well stablished research groups around the world.

This work conforms to an observational approach. We review an important number of published scientific works around free software that are meaninful from a computer science standpoint. Then, we collect a set of parameters related to each publication. This data set is analyzed using statistical procedures. The results allow us to sketch some conclusions. Finally, we describe the main research areas discovered and give some insight on the open problems in each of them.

The paper is organized in two parts. Section 2 offers an introduction to free software. Section 3, explains how this work is set out from a methodological point of

view. Section 5 describes the data set. Section 5 is devoted to the analysis of the data set. Section 6 reviews the main research topics on the most important areas discovered below. Finally, Section 7 summarizes this work and point out some future work.

2 Free Software

In this section we introduce what is free software. We begin by showing the legal foundations of free software. Then, a historical perspective follows. At the end, we sketch the connexions between free software and the main related areas aside from computer science. It is important to note that in the literature several terms are used instead of free software, say for instance open source. Althought it can be argued that they are not equivalent concepts, in this paper we will use the term free software to denote all of them.

2.1 Legal Framework

The difference between free software and propietary software is a legal issue. On most countries software protection is covered by copyright. This means that the creator of a software product is guaranteed some exclusive rights on his creation. These exclusive rights can be transferred or sold to a third person. A license is the legal way to transfer these rights. The difference between free and propietary software ends up in the precise rights transferred through the license.

Richard Stallman defined free software, see [76, 77], as the software that guarantees to the owner the following four rights:

1. The right to run the program for any purpose.
2. The right to study and modify the program.
3. The right to copy the program so you can help your neighbor.
4. The right to improve the program, and release your improvements to the public, so that the whole community benefits.

Any software that is received through a license that surrends these four rights is free software. There is not a specific kind of free software license but a number of them. The most popular licenses are the GNU Public License (GPL), [78], the Berkeley Software Distribution (BSD), [74], the Masachussets Institute of Technology license (MIT), [161], and the Lesser GNU Public License (LGPL), [79].

It is common to classify the free software licenses into two groups: the virical and the non-virical licenses, [75]. A virical license forces to any derived work to adopt the same license. This is the case for the GPL license. A non-virical license is more liberal with the licenses of derived works and do not requires to apply the same license to derived

works. The BSD license is non-virical. There is an intense debate on which licenses family better boosts innovation, see [118]

Not all the licenses are legally compatible between them. Thus, building a free software product up by recombining existing free software pieces requires to carefully choose the right licenses. This is often seen as an important source of legal risk.

2.2 Historical considerations

Before the decade of seventies to sell software licenses was an inusual practice. Software was commonly shared by individuals who used computers and by hardware manufacturers who were glad that people were making software that made their hardware useful. In the seventies and early eighties, the software industry began to apply copyright law, and began using technical measures such as only distributing binary copies, to prevent computer users from being able to study and modify the software.

In 1983, Richard Stallman launched the GNU project, [138], after becoming frustrated with the effects of the change in culture of the computer industry and users. Software development for the GNU operating system began in January 1984, and the Free Software Foundation (FSF) was founded in October 1985. Stallman introduced the free software definition and the "copyleft" concept, designed to ensure software freedom for all.

In 1991, Linus Torvalds released the Linux kernel as freely modificable source code. The decision to use GPL on Linux kernel enabled to combine it with the almost-finished GNU operating system making the first complete free software operating system. Ian Murdock in the 1993 began Debian GNU/Linux, a distribution of the Linux kernel and the GNU software with a filosofy close to FSF.

In 1997, Eric Raymound published The Cathedral and the Bazaar, [137], a reflective analysis of the hacker community and free software principles. The paper received significant attention from comercial companies like Netscape Communications Corporation who released their popular Netscape Communicator Internet suite as free software, the base of the nowadays Mozilla Firefox.

During the year 1998, Tim O'Reilly, Linux Torvalds and Bruce Perens created the term *open source* to encourage business to share their code and open their products. This initiative has grown until today and has had a big impact on computer software enterprises. For instance, Sun Microsystems released in 1999 the StarOffice office suite, that was renamed to OpenOffice, as free software under the GNU Lesser General Public License and in 2007 the Java Development Kit as OpenJDK under the GNU General Public License.

Free software products are increasingly used. For example, the Apache web server market share is growing since 1995, [124] and Mozilla Firefox is known to have an continuous growing market share since the first version, [131].

2.3 Other Aspects of Free Software

Free software is an interesting phenomenon not only from a complex science perspective. In this section we overview other areas in which free software is a significant phenomenon.

The economic aspects related to free software have deserved a considerable atention during last years. The old question about the possibility of making money with free software has been answered by the market. Currently, the business related to free software is growing very fast as stated by Gartner. Traditional software industry such as IBM, or Sun Microsystems have becomed important players in free software arena. In those companies, free software plays an important estrategical role, [144]. Many companies whose core business is not in the information technologies sector are also choosing free software for critical missions as their internet information and sales sites. As a result of this activity free software has also raised interest of economy and management scientifics. Current problems include the business models related to free software, the management of the relationship between free software comunities processes and the more rigid industry processes, or the economical models that try to explain the market share evolution of free and privative software. Repositories such as IDEAS, [5], contain a number of interesting papers on these topics.

The legal framework of free software offers new oportunities for the development teams. The colaboration between different project teams is far more natural due to the unique freedom to manage the project code that free software guarantees. This fact, together with some historical circumstances, has encouraged a rather specific software development practices and methods. These practices, commonly known as community development, are characterized by distributed, self-organized and loosely coupled teams, agile development methodologies, easy and public distribution methods, and high degree of user involvement in the project. These organizational practices are prevailing in free software development. To further emphasize the importance of these managerial patterns, Fuggetta and Cerri, in [42], define free source as an aproach to manage the development and distribution of software. These organizational and managerial practices have attracted a lot of interest from the industry and the academy. The prevailing research topics include managerial best practices, development communities governance, development comunity success factors or free software quality assessment.

Many hardware manufacturers use free software in their products. Network equipment, storage units or printing devices that embbed free software are very common. More interesting are manufacturers like Nokia, [7], OpenMoko, [9] or Arduino, [1]. These are involved in developing and selling electronical gadgets based in free software which are themselves open products in some sense. That is, the involvement with free software principles partially extends to their hardware and their software development processes. There are also microchips which are being developed with specifications released under free licenses. The OpenCores project, [8], is an example.

Finally, there is a significant influence of free software as an ethical and social issue

into other causes. The free culture movement, [2], tries to extrapolate free software seminal ideas to a broader field. For instance, the Creative Commons initiative, [44], is a sucessful suite of permisive licenses that can be applied to creations other than software, say music or literary works.

3 Data Set Acquisition

This work is based on the systematic study of the free software related publications. Therefore, the way to obtain these publications is of great interest. This Section describes the process that we followed to obtain the set of publications. We detail the steps followed during the work process particularly emphasizing some methodological issues.

In this set we consider the following types of publications:

- Articles in a journal.
- Full papers in a conference proceedings.
- PhD dissertations.
- Master thesis.

We used a two steps method to obtain the data set of publications:

1. We conduced a search using the main engines indexing scientific publications: The ACM Library Portal, [17], the IEEE portal, [97], Google Scholar, [4], ScienceDirect, [15], and the ISI Web of Knowledge, [6]. On these portals we applied the following search terms:
    - Free software
    - Open source
    - Floss (a usual acronym meaning "Free and Libre Open Source Software").
2. We carried on a new search for the citations in the publications obtained in the previous step.

The set of publications obtained from the procedure explained before contained a number of free software publications in which computer science is not the central topic. There are, for instance, some articles that mention the application of free software to medical problems, see [119, 133, 153, 162]. These publications are mainly from the human health knowledgment area.

We proceed by choosing only the publications whose central topic is related to computer science.

Year	Number of documents
1999	3
2000	1
2001	1
2002	4
2003	7
2004	7
2005	16
2006	11
2007	32
2008	9

Table 1: Number of documents in the data set published each year

4 Data Set Characteristics

In this section we describe the main characteristics of the publications data set that will be analized. This data set is build after the steps explained in the previous section.

The data set size is of 100 observations. An observation corresponds to a publication. For each observation we defined the following variables:

subject1 subject2 Every publication is classified into one or two subjects. We chose the ACM Computer Classification System 1998, [16], as the reference. This classication system is specific for the computer science field and thus well suited for our purposes. Indeed, a number of publications from the data set were previously classified according to the ACM system. The ACM classification is a three levels hierarchical system. The levels are coded acording to the format `X.N.M` where `X` is a letter denoting the first classification level, and `N` and `M` are numbers denoting the second and third levels respectively. Classification tags can be written also by adding the subject descriptions. For instance, H.2.4 [Systems]—Object-oriented databases.

The criteria to classify the publications is the following:

- If a publication contains a ACM classification tag, use it. Otherwise,
- If the article is in the ACM portal and it is classified, use it. Otherwise,
- Classify the publication following the principles explained in [18].

After classifying the observations we found K.6 [Management of computing and information systems], and D.2 [Software engineering] to be the most frequent subjects, see Table 4.

publication Observations of the data set come from 20 different journals, see Table 4, and 15 different conference proceedings, see Table 4, aside of master and PhD thesis. This variable encodes the observation's publication.

Area	Name	Number of docuements
A.1	Introductory and survey	1
C.2	Computer-communication networks	4
C.5	Computer system implementation	1
D.1	Programming techniques	1
D.2	Software engineering	69
D.3	Programming languages	3
D.4	Operating systems	5
F.3	Logics and meanings of programs	1
H.1	Models and principles	2
H.2	Database management	3
H.3	Information storage and retrieval	3
H.4	Information systems applications	4
H.5	Information interfaces and presentation	10
I.2	Artificial intelligence	2
I.6	Simulation and modeling	2
J.1	Administrative data processing	3
J.m	Computer applications miscellaneous	1
K.1	The computer industry	4
K.2	History of computing	1
K.3	Computers and education	4
K.4	Computers and society	5
K.6	Management of computing and information systems	53
K.8	Personal computing	10

Table 2: ACM subjects found in the data set

Name	Acronym
ACM Transactions Software Engineering and Methodology	ACMTSEM
Advances in Computers	AC
Communication ACM	CACM
Computer	C
Computer and Education	CE
IEEE Proceedings Software	IEEEPS
IEEE Software	IEEES
IEEE Transactions Professional Communication	IEEETPC
IEEE Transactions on Software Engineering	IEEETSE
ITProfessional	ITP
Information and Management	IM
Information and Software Technology	IST
Interactingwith Computers	IC
Journal Systems Architecture EUROMICRO Journal	JSA
Journal of Software Maintenance and Evolution	JSME
Journal of Systems and Software	JSS
Management Science	MS
Queue	Q
SIGSOFT Soft Engineering Notes	SIGSOFT
Strategic Information Systems	SIS

Table 3: The the data set contains papers from these journals.

Name	Acronym
ACM workshop on Interdisciplinary software engineering research	WISER
Computer Supported Cooperative Work	CSCW
Conference on Software Engineering Education and Training	CSEET
Conference on Supporting Group Work	GROUP
ESEC FSE: International Workshop on Principles of Software Evolution	IWPSE
European Software Engineering Conference	ESEC-FSE
Free Libre Open Source Software Conference	FLOSS
Hawaii International Conference on System Sciences	HICSS
Human-Computer Interaction	HCI
ICSE: International Workshop on Mining Software Repositories	MSR
ICSE: Workshop on Open Source Software Engineering	WOSSE
IEEE International Conference on Software Maintenance	ICSM
International Conference on Information Systems	ICIS
International Conference on Software Engineering	ICSE
Technical Symposium on Computer Science Education	SIGCSE

Table 4: The data set contains full papers from these conferences.

Label	Impact factor
XXXXL	[4,3)
XXXL	[3,2)
XXL	[2,1.5)
XL	[1.5,1)
L	[1,0.7)
M	[0.7,0.5)
S	[0.5,0.2)
XS	[0.2, 0]

Table 5: Conversion table used to discretize impact factor.

Group	Institution
Dependability at School of Computing Science	Newcastle University
Institute for Software Research - Open Software Development	California University
Libre Software Group GSyC	Universidad Rey Juan Carlos
Software Engineering Group	Aristotle University of Thessaloniki
Software Engineering Group	University of Victoria
Syracurse FLOSS Research Group	Syracuse University
The interaction lab	University of Saskatchewan

Table 6: Research Groups

isjournal This variable that descriminates whether the observation has been published in a journal or not.

impact factor For the publications indexed in the ISI Web of Knowledge, we registered their impact factor. The value of this variable for observations with no impact factor has been set to zero. In order to discretize the value we adopted table 4

year This variable encodes the publication year, see Table 4

5 Data Analysis

5.1 Preliminar analisys

We tried to discover the research groups hidden behind the publications. For each publication we looked at the authors web pages when possible searching for any research group reference. The research groups discovered are shown in Table 5.1.

5.2 Principal Components Analysis

This Section explains the main analysis of the data set. The goal of this analysis is to discover the trends of free software research by analysing relationships between variables. The analysis is done by using the statistical tool known as *principal component analysis* (PCA), [11]. PCA is a family of techniques, mainly of descriptive nature, that is well suited the study of multivariated qualitative data sets. Roughly speaking, PCA thinks of observations as points in a $\mathbf{R}^n$ space where n is the number of variables of the data set. The tools of PCA allow to compute two-dimensional planes embedded in the $\mathbf{R}^n$ space such that, when observations are projected onto these planes, the relationships between observations become clear. The data set matrix can be transposed and the PCA applied to the new matrix. This allows to interchange the roles of variables and individuals. Therefore, relationships between variables can also be studied. From the diverse PCA techniques we use *multiple correspondence analysis* (MCA) because all the variables of the data set individuals are qualitative variables.

PCA is usually done with the help of statistical software. In this work we used the statistical package `R`, [13], together with the `FactoMineR` library, [114].

The analysis of the data set was done in two phases. In the first phase we run some preliminar analyses and observed the following issues:

- The tree levels classification for the subject was too fine grained. For many publications it is difficult to distinguish the subject precissely. We resolved to use only the two first levels of the classification system for the analysis. This results in a less informative but far more robust variable.

- Using only the first subject to conduce the analysis discarded too much information. There are many publications in the data set that should be classified in a pair of subjects to correctly describe them. In many publications, there is not a primary subject but two subjects of similar importance. This inconvenient was solved by duplicating every observation of the data set and asigning each copy one of the two subjects. Observations with only one subject were also duplicated and both copies were assigned the same subject. This rather unusual solution does not add any bias to the analysis.

- Some observations appeared to be outliers. We consider that these publications were individualy studied. Most of them were publications in a very specific subject. Turnu, [155], which were the only publication about "simulating and modeling", Rigby, [140], that writes about "artificial inteligence", Dittrich, [62], which is classified in A.1 Introductory and survey, and Samuelson, [144], that argues about "Microcomputers". All of them were deleted from the data set.

After cleaning the data set as explained before, we proceed by analyzing the relationships between the most interesting subsets of variables. The precise sets of variables were suggested by the process itself. For instance, if the result of an analysis sugests that it may exists a relationship between two variables then we explicitly investigate it. The result is a set of analyses that lets to discover some interesting results. Following we introduce these analyses.

5.2.1 Subject + Publication

This analysis studies the relationship between the variable `subject` and `publication`. The goal is to investigate whether there are some subjects that are more prone to be published in specific publications.

The analysis result is the graphic shown in Figure 1. In the resulting graphic it can be identified three distinct groups: the subjects and publications related to education, the subjects and publications related to applications and, in the central area, the most important group related to software engineering.

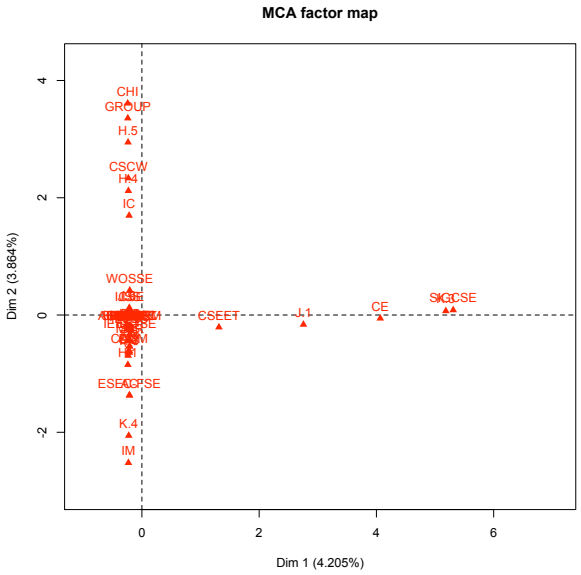

Figure 1: MCA analysys results for variables `subject` and `publication`

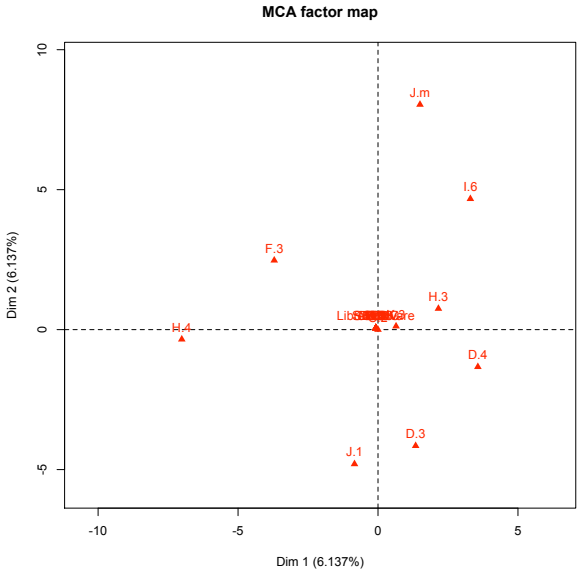

Figure 2: MCA analysys results for variables `subject` and `group`

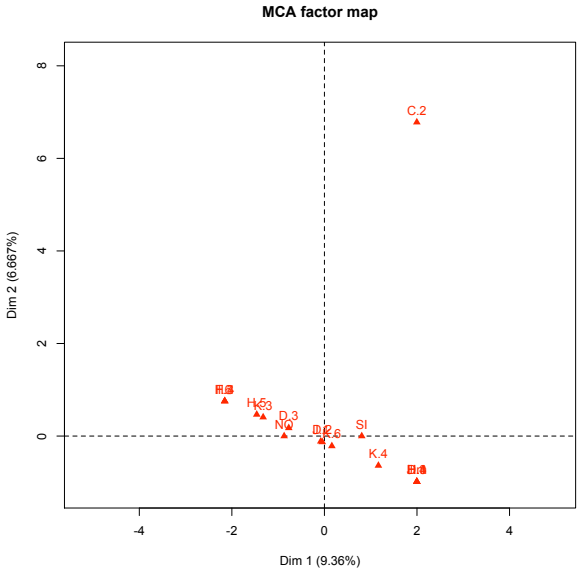

Figure 3: MCA analysys results for variables `subject` and `isjournal`

5.2.2 Subject + Research Group

This analysis studies the relationship between the variable `subject` and `group`. The goal is to investigate which are the target subjects of known research groups.

The analysis result is the graphic shown in Figure 2. In this graphic it can be identified a central group where all the known research groups are located. From the tabular of the results of the analysis we can identify five different subjects: groups that work on society relation with computers, groups that work on management of computing and information systems, groups that work on software engineering, groups that work on models and principles and groups that work on information interfaces and presentation.

5.2.3 Subject + Isjournal

This analysis studies the relationship between the variable `subject` and `isjournal`. The goal is to investigate which are the preferred subjects in journal publications. If we assume that journals publish more mature subjects, this result can help to identify the more mature free software research subjects.

The result of the analysis is shown in Figure 3. In the graphic two groups can be distinguished:

- subjects that are more common on journals:

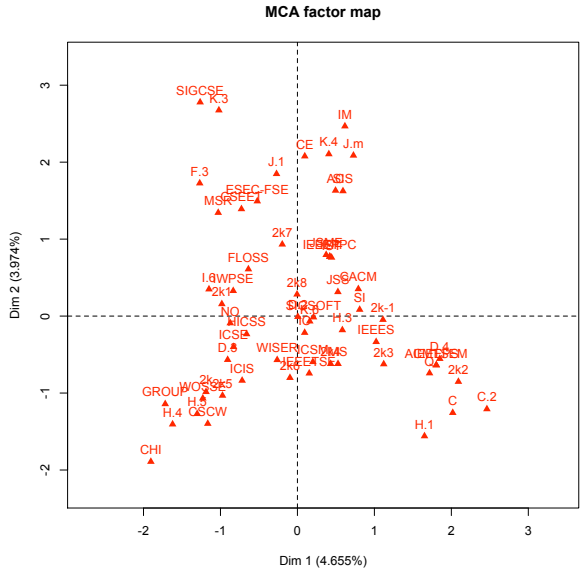

Figure 4: MCA analysys results for variables `subject`, `publication`, `year` and `isjournal`

K.4[computers and society], D.2[software engineering] and C.2[computer communication networks] and

- subjects more common on other kind of publications:

  K.3[computers and education], H.5[information interfaces and presentation], F.3[logics and meanings of programs], H.4[information systems applications] and I.6[simulation and modeling].

5.2.4 Subject + Publication + Year + Isjournal

This analysis studies the relationship between the variables `subject`, `publication`, `year` and `isjournal`. The goal is to investigate the evolution of subjects along the years.

The analysis result is the graphic shown in Figure 4. In the graphic it can be identified that on 2002 the main topics were C.2[computer-communication networks] and H.1[models and principles] and on 2000 and 2005 H.4[information systems applications] and H.5[information interfaces and presentation]. It can be observed that in the center of the graphic we have the big group of articles that talk about software engineering, that there is a clear relation of subjects with journals and conferences and that on 2000 and 2005 the mayority of publications are on group topic conferences (GROUP, WOSSE, CSCW, CHI).

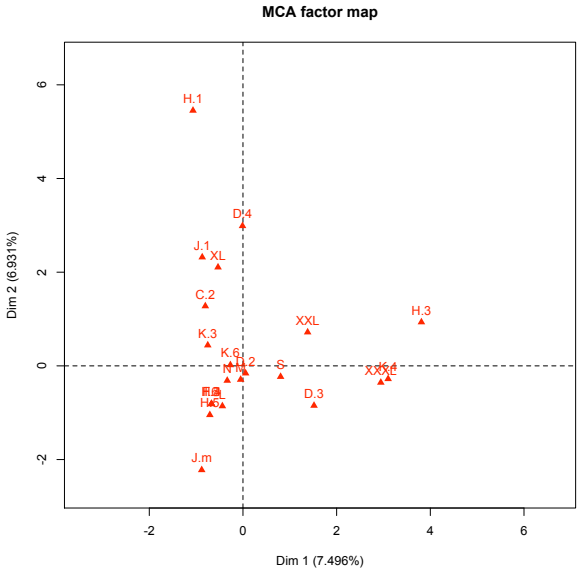

Figure 5: MCA analysys results for variables `subject` and `impact factor`

5.2.5 Subject + Impact Factor

This analysis studies the relationship between the variables `subject` and `impact factor`. The goal is to investigate which subjects have a greater impact factor.

The analysis result is the graphic shown in Figure 5. In the graphic, the right side has the subjects with the highest impact factor that is : computers and society (K.4) and information storage and retrieval (H.3).

5.2.6 MCA analysis D.2

This analysis studies the relationship between the variables `subject` and `impact factor`. The goal is to investigate the relation of different subareas on D.2 topic with the impact factor of their publications. It only considers subareas with more than 3 publications in order to remove outliers.

The analysis result is the graphic shown in Figure 6. In the graphic it can be identified that D.2.8[metrics] and D.2.13[reusable software] followed by D.2.9[management] are the topics with a highest impact factor.

5.2.7 MCA analysis of K.6

This analysis studies the relationship between the variables `subject` and `impact factor`. The goal is to investigate the relation of different subareas on K.6 topic with

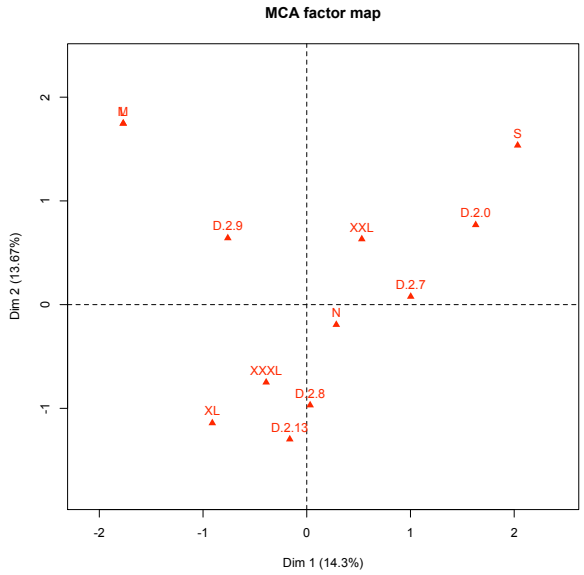

Figure 6: MCA analysys results for variables `subject` and `impact factor`

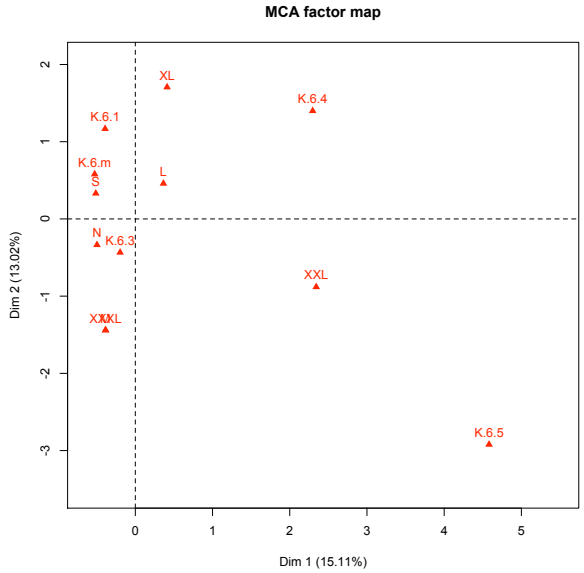

Figure 7: MCA analysys results for variables `subject` and `impact factor`

the impact factor of their publications. It only consideres subareas with more than 3 publications in order to remove outliers.

The analysis result is the graphic shown in Figure 6. In the graphic it can be identified that K.6.5[security and protection] and K.6.4[system management] are the subtopics with a higher impact factor.

5.3 Conclusions

From the analysis of the data set we can sketch the following conclusions:

- The year of publication has no influence over any other variable. However, the number of publications increases over the time.
- The research in free software is highly biased towards the following subjects:
  - K.6 [Management of computing and information systems]
    - K.6.4 [System Management]
    - K.6.5 [Security and protection]
  - D.2 [Software engineering]
    - D.2.8 [Metrics]
    - D.2.9 [Management]
    - D.2.13 [Reusable software]
- The subjects K.4 [Computers and society] and H.3 [Information storage and retrieval] concentrate the highest impact factor despite of not being the most frequent subjects.
- The research groups identified work mainly on software engineering related subjects. However, they sporadically publish works in other areas.
- Journal papers are mainly from K.4 [Computers and society], and D.2 [Software engineering] subjects. It is reasonable to think in these subjects as the more mature ones.

6 Free Software Main Research Subjects

In this Section we review the publications of the main free software research subjects. According to the results of the previous Section, we review the subjects K.4 [Computers and society] and H.3 [Information storage and retrieval], which concentrate the highest impact factor publications, and the subjects K.6 [Management of computing and information systems] and D.2 [Software Engineering] which contain the highest number of publications.

6.1 K.4 Computers and Society

This is a new subject on free software research as the oldest article dates from 2007 and the data set contains no more than five publications on this subject.

Scacchi, in [146, 145], discusses how cooperation, coordination and control are realized in free software projects from a social cause standpoint. He also studies why individuals participate in these projects. Scacchi uses surveys coming from free software projects empirical studies as the principal source of information. Barcellini, [26], does an analysis of the design process in free software communities. He works by modeling the process dynamics using the project mailing lists information. Both authors agree that free software offers new kind of practices, processes and organizational forms to discover, observe, analyze, model and simulate.

Okoli, [130], investigates the motivation of participants who contribute their time freely to free software projects. Okoli bases his contribution in the data obtained from the english version of the wikipedia. He concludes that economic incentives might dilute the original spirit and may have negative consequences for the project.

6.2 H.3 Information Storage and Retrieval

There are few documents in the data set about this subject. However, they achieve an important impact factor. How to store the information of critical large scale systems, like those needed during human disaster emergency, is the main goal of Currion, [54], and Chae, [43]. They conclude that free software can be an interesting model for these kind of applications because the distributed and collaborative development model. They also point out that the free software model offers a higher autonomy from third party companies.

Dinh-Trong in [59], explains how shared data is stored in the online information services that are used on free software development.

6.3 K.6 Management of Computing and Information Systems and D.2 Software Engineering

This Section is about K.6 [Management of computing and information systems] and D.2 [Software engineering]. We study them together because both subjects share several second level subjects, the mayority of the articles on D.2 are also on K.6.

Given the big amount of documents in this subject, we review them below following the corresponding three levels deep clasification.

6.3.1 D.2.0 General

From a general point of view O'Reilly, [132], explains how free software helps to improve innovation. Moreover, he argues about the importance of extensibility in free software projects and the free software based commercial product development.

Crowston, [45], and English, [68], investigate the success factors in free source projects. Scacchi, [146, 145], and Crowston in [51], inquire into empirical evidence of how free source software development teams self-organize their work. The publication of Heckman in [91] look into decision-making practices in technology-supported self-organizing distributed teams. Free software projects are the target.

Free source software as a teaching tool is the goal of Ellis paper, [66]. This publication explains how the engineering process can be analyzed and taught using the Sahana project, [14]. It concludes that a real-world free source project can successfully support a range of software engineering learning practices.

Robles, [141], wrote his PhD thesis about empirical software engineering research on free software. He studies how to obtain a better understanding of the free software phenomenon methodically and empirically analyzing the public available traces from the software development process. A similar objective is pursued by Krafft, [110], who developed a framework that captures the factors which have an effect on the developers' decision to adopt or reject a development method.

6.3.2 D.2.2 Design Tools and Techniques

Halverson, [89], studies the management of change requests in free software projects. The work describes a tool to help during design phase of free software projects. Souza, [56], search for tools to manage the evolution of source code in free software projects. They conclude that free software is an interesting domain to visualize the evolution of a project, that individual and software components may act as "passage points" and that the use of computational tools help to see the structures. Barcellini in [25] dig into the mailing lists of free software projects trying to discover how they are used as a tool to design the software. The Python PEP, [12], is one of the processes studied.

6.3.3 D.2.4 Software/Program Verification
K.6.5 Security and Protection

According to Hoepman, [95], the free software development process helps on to the security assessment of the developed systems. Kuru, [109], concludes that the efect of the free software project size is significant on the quality and verification of the software. He quantifies the influence of project size on defect proneness.

6.3.4 D.2.7 Distribution, Maintenance, and Enhancement

The study of the changes on software that are done by patches, [31], and the study of bug tracking systems combined to version control system information, [72], helps on the analysis of the evolution and maturity of the source code and the identification of error prone classes with affectes components or products. Yu, in [163], studies the self-organization organizational methodology that is used on free software projects to fulfill accomplish the functional and quality requirements of the software. To know how and when affects to a project the learning process is the goal of Shaikh, [148]. He concludes that the technology, license and the learning of development tools affect the evolution of the software. Raja, [136], studies the quality of software on large scale free software projects and Koru, [108], uses free software to define a model that helps identifying the change-prone modules.

Kopenen in [105] compares the ISO maintenance process to free software maintenance process. He finds four similar activities: process implementation, problem and modification analysis, implementation and modification review and acceptance. The ISO migration process corresponds release management in free software. There isn't a ISO retirement process on free software.

6.3.5 D.2.8 Metrics

Measuring free software projects is easy as all the data is available to study. Dinh-Trong in [59], Wang in [158], Mockus in [123] and Raja in [136] obtain and analyze data from version control systems, bug tracking and mailing lists, looking for the number of developers, the defect density of code, the time to solve problems and measuring the evolution of free software projects together their communities. Koru, [108, 107, 109], and Paulson, [134], also studied the changing rate, growing rate, change top metrics and defect handling authors. Contributions before commit to version control system and reviews after a commit is the topic of Rigby in [139]. Bird in [32] analyzes the mailing lists to evaluate the coordination activities of the participants. Capiluppi, [39], adapts the staged model for software evolution to free software in order to take free software projects evolution closer to that of commercial software.

6.3.6 D.2.9 Management
K.6.4 System Management
K.6.3 Software Management
K.6.1 Project and People Management

The process of patch submission and acceptance is the target of Bird, [31], who defines a methodology to analyze it. Feller in [69] defines a framework to study the free source development paradigm. The self-organization process is studied by Yu, [163]. He concludes that the initial adaptation to self-organization moves the system towards an unstable state, after that transitory the system moves toward a stable state. Koch, [104], concludes that a significant percentage of projects are able to sustain super-linear

growth and that there is an evidence for a group of projects of moderate size which shows decreasing growth rates, while small projects in general exhibit linear growth. Paulson in [134] and Koru in [108, 107, 106] compare the evolution of privative with free software concluding that free software development doesn't implies a faster evolution but more creativity and more quickly defects are fixed. Gofrey, [86], concludes that free software evolution doesn't follow "Lehman's laws". Katsamakas, [103], explains that complex interaction between participation and development processes affects crucially success of failure. Crowston in [45] exposes that a successful project needs recognition, involvement of the users and to be ported to different systems. Finally, Herraiz in [94] defines predictor models for the evolution of participation and activity on free software projects

The analysis of quality of free software development model is the topic of Zhao, [165], and Ajila, [21]. Both conclude that this methodology has introduced a new dimension in large-scale distributed software development. That this methodology in not exploitable under all scenarios. They also conclude projects produce a high quality reusable components. Sohn in [149] analyses the relationship between the quality factors based on ISO/IEC 9126 and free software utilization concluding on how to improve programmer satisfaction during free software utilization. Norris in [127] shows how the free software development methods obtains great results in mission-critical subjects. Norris think that the reason in the facility to collaborate with third parties that the framework offer.

Release management is the focus of Fischer in [72] who studies how to populate a release history. Michelmayr wrote his PhD thesis, [121], about the impact of release management on free software projects. He defenses that time based releases is the best approach.

The analysis of the distribution of developers on free software projects is the goal of Robles, [142], obtaining a map of libre software developers.

The effectiveness of work teams, coordination and collective mind on free software development is the topic of Crowston in [46, 52, 53, 51, 50, 47, 91]. He defines a framework to analyze the distributed development teams. He also defines a theoretical model to explain the performance of free software teams and finds that "self-assignment" is the most common mechanism to assign tasks. He exposes that the problems with voluntary assignment of tasks makes hard or undesirable to transfer this practices to classical software development. Finally he explains the importance of face-to-face meetings in technology supported self-organizing distributed teams as it allow to speed up certain kind of tasks. Delorey, [57], concludes that the productivity on software development is affected depending on which programming language is used. The management of the team knowledge is the goal of Sowe in [150] and Scozzi in [147], they conclude that there is a high activity on sharing knowledge between participants on free software projects. They conclude that there is a "Fractal Cubic Ditribution" to describe the knowledge distribution. Turnu, [155], explains that mixing free software with agile practices yields better results in terms of code.

How the free software development methods and results affect commercial software is the topic on Watson in [160], Spinellis in [151], Laplante in [111], Hecker in [90], Ferris in [70], Lahey in [115], Leibovitch in [116] and Karels in [102]. They conclude that is challenging the status quo in the software marketplace increasing the efficiency, demand, innovation risks. Gurbani, [87], explains how the creation of a corporate development model creates shared technologies that are highly competitive, of higher quality and reduces product generation costs.

Dinh-Trong, [59], studies FreeBSD and Mockus , [123], studies Apache and Mozilla. Both conclude that on free software projects defects are repaired by a larger group than the core group, defect density on free software releases will generally be lower than privative code and that in successful developments, the developers will also be users of the software.

Discover how people is involved on free software projects is the main topic of Ducheneaut, [63], the analyses the relationships that newcomers develop over time and the individual and political learning process. On the same subject, Michlmayr in [120] studies why people administration and tracking is difficult on large projects, detecting who is active or not on Debian project, he concludes that free software participants are relatively unreliable. The migration of roles of the people involved on a project and its career is the main topic of Jensen in [100] who defines the diferent roles and the different paths from peripherical roles to core roles in a community. A study of the evolution of the social network is done by Ngamkajornwiwat, [125], who concludes that it changes over time in certain ways and that the study of it can help managers to understand better their free software projects.

6.3.7 D.2.12 Interoperability

Cerri in [42] explains that free software development methodology and projects are close to open standards. As using open standards is fundamental to software interaporability, Cerri concludes that free software promotes interaporability.

6.3.8 D.2.13 Reusable Software

Reusing software components is one of the basic ideas behind free software projects as you can reuse them freely for them and you can contribute and improve on it. Mockus in [122] explains that more than 50% of the files on free software projects were used in more than one project, the most widely reused components were small templates requiring major and minor modifications and groups of files reused without any change. Also Capiluppi, [38], studies the potential as shareable and small-grained reusable software components in other free software projects.

The reuse of free software components on mission-critical development is the topic of Norris in [127] where they explain the experience of developing NASA software using free software pieces. They conclude that free software methods help on mission-critical

development as you can know how developed and participate in the development of the different components.

Karels in [102], Barton in [28], Ferris in [70] and Spinellis in [151] focus on studing the benefits of reutilitzation of free software components on comercial products and how it has enabled a fast growing market. An important point is that reusing the source code may introduce problems on maintaining the security and features updates on your copied code and reuing components may introduce problems on API backward compatibility breaks. There is also an affectation on the license of the final result because of the different reused code licences.

7 Conclusions and future work

In this work we have collected an important number of free software related publications that have been classified and analized. After that, the most salient free software research subjects were identified and the publications in these areas reviewed.

We found that research in free software is mainly concentrated in K.6 [Management of computing and information systems] and D.2 [Software engineering], which are consolidated research areas. The number of publications on these topics has been increasing very fast during last years.

The main research topics include: software metrics, reusability, quality management and team management and organization. These subjects exhibit a number of open problems that have been collected in this document. Free software articles are published on journals and conferences with no distinction. The number of papers grow every year in an exponencial way.

The work in this paper is mainly of empirical nature. An important part of the process is the collection of data. This has been done following a manual procedure. To automatize the data collection would be a significant advance as this would allow to reduce the biases introduced by the manual handling of data.

References

[1] Arduino. `http://www.arduino.cc`. visited on 6/2008.

[2] Free culture movement. `http://en.wikipedia.org/wiki/Free_culture_movement`. visited on 6/2008.

[3] Free software definitions at wikipedia. `http://en.wikipedia.org/wiki/Free_software`. visited on 6/2008.

[4] Google scholar. `http://scholar.google.es/`. visited on 6/2008.

[5] IDEAS. `http://ideas.repec.org/`. visited on 6/2008.

[6] ISI web of knowledge. http://isiknowledge.com/. visited on 6/2008.

[7] Nokia corporation. http://www.nokia.com. visited on 6/2008.

[8] Open cores at wikipedia. http://en.wikipedia.org/wiki/OpenCores. visited on 6/2008.

[9] Open moko. http://www.openmoko.com. visited on 6/2008.

[10] Open source hardware at wikipedia. http://en.wikipedia.org/wiki/Open_source_hardware. visited on 6/2008.

[11] Principal component analysis at wikipedia. http://en.wikipedia.org/wiki/Principal_components_analysis. visited on 6/2008.

[12] Python enhancement proposal. http://www.python.org/dev/peps/. visited on 6/2008.

[13] R project. http://www.r-project.org/. visited on 6/2007.

[14] Sahana project. http://www.sahana.lk/. visited on 6/2008.

[15] Science direct. http://www.sciencedirect.com/. visited on 6/2008.

[16] ACM. The ACM computing classification system (1998). http://www.acm.org/class/1998/ccs98.html. visited on 6/2008.

[17] ACM. ACM digital library. http://portal.acm.org. visited on 6/2008.

[18] ACM. How to classify works using ACM's computing classification system. http://www.acm.org/class/how_to_use.html. visited on 6/2008.

[19] Adams, P., Boldyreff, C., Nutter, D., and Rank, S. Adaptive reuse of libre software systems for supporting on-line collaboration. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering* (New York, NY, USA, 2005), ACM, pp. 1–4.

[20] Ajila, S. A., and Dumitrescu, R. T. Experimental use of code delta, code churn, and rate of change to understand software product line evolution. *Journal of Systems and Software 80* (2007), 74–91.

[21] Ajila, S. A., and Wu, D. Empirical study of the effects of open source adoption on software development economics. *Journal of Systems and Software 80* (2007), 1517–1529.

[22] Asundi, J. The need for effort estimation models for open source software projects. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering* (New York, NY, USA, 2005), ACM, pp. 1–3.

[23] Bach, P. M., Kirschner, B., and Carroll, J. M. Usability and free/libre/open source software SIG: HCI expertise and design rationale. In *CHI '07 extended abstracts on Human factors in computing systems* (2007).

[24] Baldi, S., Heier, H., and Mehler-Bicher, A. Open courseware and open source software. *Communications of the ACM 46*, 9 (2003), 105–107.

[25] Barcellini, F., Détienne, F., Burkhardt, J. M., and Sack, W. Thematic coherence and quotation practices in OSS design-oriented online discussions. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work* (2005).

[26] Barcellini, F., Détienne, F., Burkhardt, J. M., and Sack, W. A socio-cognitive analysis of online design discussions in an open source software community. *Interact. Comput. 20*, 1 (2008), 141–165.

[27] Barnes, M., Phillips, J., Coumans, E., McGuire, M., Bradski, G., and Roosendaal, T. Open source toolchains. In *Sandbox '07: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games* (2007).

[28] Barton, J. From server room to living room. *Queue 1*, 5 (2003), 20–32.

[29] Basili, V. R., Briand, L. C., and Melo, W. L. How reuse influences productivity in object-oriented systems. *Communications of the ACM 39* (1996), 104–116.

[30] Bauer, A., and Pizka, M. The contribution of free software to software evolution. In *Software Evolution* (2003), pp. 170–179.

[31] Bird, C., Gourley, A., and Devanbu, P. Detecting patch submission and acceptance in OSS projects. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories* (Washington, DC, USA, 2007), IEEE Computer Society, p. 26.

[32] Bird, C., Gourley, A., Devanbu, P., Gertz, M., and Swaminathan, A. Mining email social networks. *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories 1* (2006), 137–143.

[33] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A., and Hsu, G. Open borders? immigration in open source projects. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories* (2007).

[34] Biswas, S., and Morris, R. Opportunistic routing in multi-hop wireless networks. In *Proceedings of the ACM SIGCOMM '05 Conference* (2005).

[35] Boehm, B. A view of 20th and 21st century software engineering. In *ICSE '06: Proceeding of the 28th international conference on Software engineering* (2006).

[36] Bonaccorsi, A., Lorenzi, D., Merito, M., and Rossi, C. Business firms' engagement in community projects. empirical evidence and further developments of the research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[37] Bonaccorsi, A., Lorenzi, D., Merito, M., and Rossi, C. Firms' involvement in the projects of the os community some preliminary empirical evidence and a research agenda. In *FOSDEM* (2007).

[38] Capiluppi, A., and Boldyreff, C. Coupling patterns in the effective reuse of open source software. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[39] Capiluppi, A., González-Barahona, J. M., Herraiz, I., and Robles, G. Adapting the "staged model for software evolution" to free/libre/open source software. In *IWPSE '07: Ninth international workshop on Principles of software evolution* (New York, NY, USA, 2007), ACM, pp. 79–82.

[40] Carnot, M. Error-free software development for critical systems using the b-methodology. In *Software Reliability Engineering* (1992), pp. 274–281.

[41] Cass, S. Free as in freedom: Richard stallman's crusade for free software [book review]. *Spectrum, IEEE 39*, 6 (2002), 56–57.

[42] Cerri, D., and Fuggetta, A. Open standards, open formats, and open source. *Journal of Systems and Software 80* (2007), 1930–1937.

[43] Chae, B. K., and McHaney, R. Asian trio's adoption of linux-based open source development. *Communications of the ACM 49/9* (2006), 95–99.

[44] Creative Commons Team. Software: free and open source code. `http://creativecommons.org/software`. visited on 6/2008.

[45] Crowston, K., Annabi, H., and Howison, J. Defining open source software project success. In *Defining open source software project success. In Proc. of International Conference on Information Systems (ICIS)* (2003).

[46] Crowston, K., Annabi, H., Howison, J., and Masango, C. Effective work practices for software engineering: free/libre open source software development. In *WISER '04: Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research* (2004).

[47] Crowston, K., Annabi, H., Howison, J., and Masango, C. Effective work practices for FLOSS development: A model and propositions. In *Thirty-Eighth Hawai International Conference on System Science* (2005).

[48] Crowston, K., Howison, J.and Masango, C., and Eseryel, U. Y. Face-to-face interactions in self-organizing distributed teams. In *OCIS Division* (2005), Academy of Management Conference, Honolulu, HI.

[49] Crowston, K., and Howison, J. Assessing the health of open source communities. *Computer 39*, 5 (2006), 89–91.

[50] Crowston, K., Howison, J., Masango, C., and Eseryel, U. Y. The role of face-to-face meetings in technolog-supported self-organizing distributed teams. *IEEE Transactions on professional communication 50* (2007), 185–203.

[51] Crowston, K., Li, Q., Wei, K., Eseryel, U. Y., and Howison, J. Self-organization of teams for free/libre open source software development. *Inf. Softw. Technol. 49* (2007), 564–575.

[52] Crowston, K., and Scozzi, B. Open source software projects as virtual organisations: competency rallying for software development. *IEEE SOFTWARE Proceedings 149* (2002), 3–17.

[53] Crowston, K., Wei, K., Li, Q., Eseryel, U. Y., and Howison, J. Coordination of free/libre open source software development. In *Proceedings of the International Conference on Information Systems* (2005).

[54] Currion, P., de Silva, C., and de Walle, B. V. Open source software for disaster management. *Communications of the ACM 50/3* (2007), 61–65.

[55] Davis, M., O'Donovan, W., Fritz, J., and Childress, C. Linux and open source in the academic enterprise. In *SIGUCCS '00: Proceedings of the 28th annual ACM SIGUCCS conference on User services* (New York, NY, USA, 2000), pp. 65–69.

[56] de Souza, C., Froehlich, J., and Dourish, P. Seeking the source: software source code as a social and technical artifact. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work* (New York, NY, USA, 2005), ACM, pp. 197–206.

[57] Delorey, D. P., Knutson, C. D., and Chun, S. Do programming languages affect productivity? a case study using data from open source projects. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[58] Dempsey, B. J., Weiss, D., Jones, P., and Greenberg, J. Who is an open source software developer? *Communications of the ACM 45*, 2 (2002), 67–72.

[59] Dinh-Trong, T. T. The freeBSD project: A replication case study of open source development. *IEEE Trans. Softw. Eng. 31* (2005), 481–494.

[60] Dinkelacker, J., Garg, P., Miller, R., and Nelson, D. Progressive open source. In *ICSE 2002. Proceedings of the 24rd International Conference on* (2002).

[61] Dionisio, J. D. N., Dickson, C. L., August, S. E., Dorin, P. M., and Toal, R. An open source software culture in the undergraduate computer science curriculum. *SIGCSE Bull. 39* (2007), 70–74.

[62] Dittrich, Y., John, M., Singer, J., and Tessem, B. Editorial: For the special issue on qualitative software engineering research. *Inf. Softw. Technol. 49* (2007), 531–549.

[63] Ducheneaut, N. Socialization in an open source software community: A socio-technical analysis. *Comput. Supported Coop. Work 14*, 4 (2005), 323–368.

[64] Elliott, M., Ackerman, M. S., and Scacchi, W. Knowledge work artifacts: kernel cousins for free/open source software development. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work* (2007).

[65] Ellis, H. J. C., Morelli, R. A., de Lanerolle, T. R., Damon, J., and Raye, J. Can humanitarian open-source software development draw new students to CS? In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education* (New York, NY, USA, 2007), ACM, pp. 551–555.

[66] Ellis, H. J. C., Morelli, R. A., de Lanerolle, T. R., and Hislop, G. W. Holistic software engineering education based on a humanitarian open source project. In *CSEET '07: Proceedings of the 20th Conference on Software Engineering Education & Training* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 327–335.

[67] Engelstad, P., Tonnesen, A., Hafslund, A., and Egeland, G. Internet connectivity for multi-homed proactive ad hoc networks. In *Proceedings of IEEE International Conference on Communication* (2004).

[68] English, R., and Schweik, C. M. Identifying success and tragedy of FLOSS commons: A preliminary classification of sourceforge.net projects. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[69] Feller, J., and Fitzgerald, B. A framework analysis of the open source software development paradigm. In *ICIS '00: Proceedings of the twenty first international conference on Information systems* (2000).

[70] Ferris, P. The age of corporate open source enlightenment. *Queue 1*, 5 (2003), 34–44.

[71] Fielding, R. T. Software architecture in an open source world. In *ICSE '05: Proceedings of the 27th international conference on Software engineering* (New York, NY, USA, 2005), ACM, pp. 43–43.

[72] Fischer, M., Pinzger, M., and Gall, H. Populating a release history database from version control and bug tracking systems. In *ICSM '03: Proceedings of the International Conference on Software Maintenance* (2003).

[73] Frakes, W., and Terry, C. Software reuse: metrics and models. *ACM Computer Surveys 28* (1996), 415–435.

[74] Free Software Foundation. BSD license. `http://www.gnu.org/philosophy/bsd.html`. visited on 6/2008.

[75] Free Software Foundation. Copyleft. `http://www.gnu.org/copyleft/copyleft.html`. visited on 6/2008.

[76] Free Software Foundation. First bulletin on FSF. `http://www.gnu.org/bulletins/bull1.txt`. visited on 6/2008.

[77] Free Software Foundation. Free software definition at FSF. `http://www.gnu.org/philosophy/free-sw.html`. visited on 6/2008.

[78] Free Software Foundation. GNU general public license. `http://www.gnu.org/licenses/gpl.txt`. visited on 6/2008.

[79] Free Software Foundation. GNU lesser general public license. `http://www.gnu.org/licenses/lgpl.txt`. visited on 6/2008.

[80] Fuggetta, A. Open source software an evaluation. *Journal of Systems and Software 66* (2003), 77–90.

[81] Gabriel, R. P. The commons as new economy & what this means for research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[82] Gacek, C., and Arief, B. The many meanings of open source. *IEEE SOFTWARE 21* (2004), 34–40.

[83] Gao, Y., Antwerp, M. V., Christley, S., and Madey, G. A research collaboratory for open source software research. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[84] Georgas, J. C., Gorlick, M. M., and Taylor, R. N. Raging incrementalism: harnessing change with open-source software. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering* (2005).

[85] German, D. M. Using software distributions to understand the relationship among free and open source software projects. In *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories* (2007), IEEE Computer Society.

[86] Godfrey, M., and Tu, Q. Growth, evolution, and structural change in open source software. In *IWPSE '01: Proceedings of the 4th International Workshop on Principles of Software Evolution* (New York, NY, USA, 2001), ACM, pp. 103–106.

[87] Gurbani, V. K., Garvert, A., and Herbsleb, J. D. A case study of a corporate open source development model. In *ICSE '06: Proceeding of the 28th international conference on Software engineering* (2006).

[88] Gutwin, C., Penner, R., and Schneider, K. Group awareness in distributed software development. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work* (2004).

[89] Halverson, C. A., Ellis, J. B., Danis, C., and Kellogg, W. A. Designing task visualizations to support the coordination of work in software development. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work* (New York, NY, USA, 2006), ACM, pp. 39–48.

[90] Hecker, F. Setting up shop: The business of open-source software. *IEEE Softw. 16* (1999), 45–51.

[91] Heckman, R., Crowston, K., Li, Q., Allen, E., Eseryel, U. Y., Howison, J., and Wei, K. Emergent decision-making practices in technology-supported self-organizing distributed teams. In *International Conference on Information Systems (ICIS 2006)* (2006).

[92] Hedberg, H., Iivari, N., Rajanen, M., and Harjumaa, L. Assuring quality and usability in open soruce software development. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[93] Herbsleb, J., and Mockus, A. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on 29* (2003), 481–494.

[94] Herraiz, I., Robles, G., and Gonzalez-Barahona, J. M. Towards predictor models for large libre software projects. *SIGSOFT Softw. Eng. Notes 30*, 4 (2005), 1–6.

[95] Hoepman, J.-H., and Jacobs, B. Increased security through open source. *Communications of the ACM 50/1* (2007), 79–83.

[96] Howison, J. Taking research to FLOSS-curious engineers and managers. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[97] IEEE. IEEE explore. `http://ieeexplore.ieee.org/Xplore/guesthome.jsp`. visited on 6/2008.

[98] International Institute of Infonomics. Free/libre and open source software: Survey and study. `http://flossproject.org/report/index.htm`. visited on 6/2008.

[99] Jaccheri, L., and Osterlie, T. Open source software: A source of possibilities for software engineering education and empirical software engineering. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[100] Jensen, C., and Scacchi, W. Role migration and advancement processes in OSSD projects: A comparative case study. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 364–374.

[101] Jodnson-Eilola, J. Open source basics: Definitions, models, and questions. In *SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation* (2002).

[102] Karels, M. J. Commercializing open source software. *Queue 1*, 5 (2003), 46–55.

[103] Katsamakas, E., and Georgantzas, N. Why most open source development projects do not succeed? In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[104] Koch, S. Software evolution in open source projects—a large-scale investigation. *J. Softw. Maint. Evol. 19*, 6 (2007), 361–382.

[105] Koponen, T., and Hotti, V. Open source software maintenance process framework. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering* (2005).

[106] Koru, A. G., and Liu, H. Identifying and characterizing change-prone classes in two large-scale open-source products. *Journal of Systems and Software 80*, 1 (2007), 63–73.

[107] Koru, A. G., and Tian, J. Defect handling in medium and large open source projects. *IEEE Softw. 21*, 4 (2004), 54–61.

[108] Koru, A. G., and Tian, J. Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products. *IEEE Trans. Softw. Eng. 31*, 8 (2005), 625–642.

[109] Koru, A. G., Zhang, D., and Liu, H. Modeling the effect of size on defect proneness for open-source software. In *PROMISE '07: Proceedings of the Third International Workshop on Predictor Models in Software Engineering* (2007), IEEE Computer Society.

[110] Krafft, M. *Method diffusion in large open-source projects.* PhD thesis, Lero - the Irish Software Engineering Research Center - CSIS, University of Limerick, 2007.

[111] Laplante, P., Gold, A., and Costello, T. Open source software: Is it worth converting? *IT Professional 9*, 4 (2007), 28–33.

[112] Lawrie, T., and Gacek, C. Issues of dependability in open source software development. *SIGSOFT Softw. Eng. Notes 27*, 3 (2002), 34–37.

[113] Lawton, G. Open source security: Opportunity or oxymoron? *Computer 35* (2002), 18–21.

[114] Lê, S., Josse, J., and Husson, F. FactoMineR: An R package for multivariate analysis. *Journal of Statistical Software 25*, 1 (1 2008), 1–18.

[115] Lehey, G. Closed source fights back. *Queue 1*, 5 (2003), 56–62.

[116] Leibovitch, E. The business case for linux. *IEEE Softw. 16*, 1 (1999), 40–44.

[117] Lin, Y.-W., and Zini, E. Free/libre open source software implementation in schools: Evidence from the field and implications for the future. *Comput. Educ. 50*, 3 (2008), 1092–1102.

[118] Matt Asay. Innovation with GNU licences. `http://www.linuxdevices.com/articles/AT4528760742.html`. visited on 6/2008.

[119] McDonald, C., Schadow, G., Barnes, M., Dexter, P., Overhage, J., Mamlin, B., and McCoy, J. Open source software in medical informatics–why, how and what. *Int J Med Inform 69* (2003), 175–84.

[120] Michlmayr, M. Managing volunteer activity in free software projects. In *USENIX 2004 Annual Technical Conference, FREENIX Track* (2004).

[121] Michlmayr, M. *Quality Improvement in Volunteer Free and Open Source Software Projects - Exploring the Impact of Release Management.* PhD thesis, Center for Technology Management - Institute for Manufacturing - University of Cambridge, 2007.

[122] Mockus, A. Large-scale code reuse in open source software. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[123] Mockus, A., Fielding, R. T., and Herbsleb, J. D. Two case studies of open source software development: Apache and mozilla. *ACM Transactions Software Engineering Methodology 11* (2002), 309–346.

[124] NetCraft. May 2008 web server survey. `http://news.netcraft.com/archives/2008/05/06/may_2008_web_server_survey.html`. visited on 6/2008.

[125] Ngamkajornwiwat, K., Zhang, D., Koru, A. G., Zhou, L., , and Nolker, R. An exploratory study on the evolution of OSS developer communities. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences* (Washington, DC, USA, 2008), IEEE Computer Society, p. 305.

[126] Nidy, D. R., and Kwok, F. Community source development: an emerging model with new opportunities. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2005), ACM, pp. 1697–1700.

[127] Norris, J. S., and Kamp, P.-H. Mission-critical development with open source software: Lessons learned. *IEEE Softw. 21* (2004), 42–49.

[128] O'Donnell, C. Making an open source case for offshoring commentary. *IEEE Transactions on professional communications 50*, 50 (June 2007), 85–87.

[129] Oezbek, C., and Prechelt, L. On understanding how to introduce an innovation to an open source project. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007), IEEE Computer Society.

[130] Okoli, C., and Oh, W. Investigating recognition-based performance in an open content community: A social capital perspective. *Inf. Manage. 44*, 3 (2007), 240–252.

[131] OneStat.com. Mozilla's firefox global usage share is still growing according to onestat.com. `http://www.onestat.com/html/aboutus_pressbox57-firefox-mozilla-ie-browser-market-share.html`. visited on 6/2008.

[132] O'Reilly, T. Lessons from open-source software development. *Communications of the ACM 42*, 4 (1999), 32–37.

[133] Oyri, K., and Murray, P. osni.info-using free/libre/open source software to build a virtual international community for open source nursing informatics. *Int J Med Inform. 74* (2005), 937–45.

[134] Paulson, J. W., Succi, G., and Eberlein, A. An empirical study of open-source and closed-source software products. *IEEE Trans. Softw. Eng. 30*, 4 (2004), 246–256.

[135] Pedroni, M., Bay, T., Oriol, M., and Pedroni, A. Open source projects in programming courses. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education* (New York, NY, USA, 2007), ACM, pp. 454–458.

[136] Raja, U., and Barry, E. Investigating quality in large-scale open source software. *SIGSOFT Softw. Eng. Notes 30*, 4 (2005), 1–4.

[137] Raymond, E. S. *Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Inc., 2001. Foreword By-Bob Young.

[138] Richard Stallman. The GNU project. published in the book Open Sources. http://www.gnu.org/gnu/thegnuproject.html.

[139] Rigby, P. C., German, D. M., and Storey, M.-A. Open source software peer revier practices: A case study of the apache server. In *ICSE 08: International Conference on Software Engineering* (2008).

[140] Rigby, P. C., and Hassan, A. E. What can OSS mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list. In *MSR '07: Proceedings of the 2007 international workshop on Mining software repositories* (2007).

[141] Robles, G. *Empirical software engineering research on libre software: data sources, methodologies and results*. PhD thesis, Universidad Rey Juan Carlos - Escuela Superior de Ciencias Experimentales y tecnología, 2005.

[142] Robles, G., and Gonzalez-Barahona, J. M. Geographic location of developers at sourceforge. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories* (New York, NY, USA, 2006), ACM, pp. 144–150.

[143] Robles, G., Gonzalez-Barahona, J. M., Michlmayr, M., and Amor, J. J. Mining large software compilations over time: another perspective of software evolution. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories* (New York, NY, USA, 2006), ACM, pp. 3–9.

[144] Samuelson, P. IBM's pragmatic embrace of open source. *Communications of the ACM 49*, 10 (2006), 21–25.

[145] Scacchi, W. 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* (2007), ACM, pp. 459–468.

[146] Scacchi, W. Free/open source software development: Recent research results and methods. *Advances in Computers 69* (2007), 1–2.

[147] Scozzi, B., Crowston, K., Eseryel, U. Y., and Li, Q. Shared mental models among open source software developers. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences* (2008).

[148] Shaikh, M., and Cornford, T. Learning/organizing in linux: a study of the 'spaces in between'. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering* (New York, NY, USA, 2005), ACM, pp. 1–5.

[149] Sohn, S. Y., and Mok, M. S. A strategic analysis for successful open source software utilization based on a structural equation model. *Journal of Systems and Software 81*, 6 (2008), 1014–1024.

[150] Sowe, S. K., Stamelos, I., and Angelis, L. Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software 81*, 3 (2008), 431–446.

[151] Spinellis, D., and Szyperski, C. How is open source affecting software development? *Software, IEEE 21* (2004), 28–33.

[152] Spinuzzi, C. Documentation, participatory citizenship, and the web: the potential of open systems. In *SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation* (New York, NY, USA, 2002), ACM, pp. 194–199.

[153] Stewart, J., Mangalam, H., and Zhou, J. Open source software meets gene expression. *Brief Bioinform 2* (2001), 319–28.

[154] Theunissen, W. H. M., Boake, A., and Kourie, D. G. In search of the sweet spot: agile open collaborative corporate software development. In *SAICSIT '05: Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries* (2005).

[155] Turnu, I., Melis, M., Cau, A., Setzu, A., Concas, G., and Mannaro, K. Modeling and simulation of open source development using an agile practice. *J. Syst. Archit. 52*, 11 (2006), 610–618.

[156] von Krogh, G., and Spaeth, S. The open source software phenomenon: Characteristics that promote research. *J. Strateg. Inf. Syst. 16* (2007), 236–253.

[157] von Krogh, G., and von Hippel, E. The promise of research on open source software. *Manage. Sci. 52*, 7 (2006), 975–983.

[158] Wang, Y., Guo, D., and Shi, H. Measuring the evolution of open source software systems with their communities. *SIGSOFT Softw. Eng. Notes 32*, 6 (2007), 7.

[159] Wasserman, A. I., and Capra, E. Evaluating software engineering processes in commercial and community open source projects. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)* (2007).

[160] Watson, R. T., Boudreau, M.-C., York, P. T., Greiner, M. E., and Donald Wynn, J. The business of open source. *Communications of the ACM 51*, 4 (2008), 41–46.

[161] X Consortium. Other copyrights. `http://www.xfree86.org/3.3.6/COPYRIGHT2.html`. visited on 6/2008.

[162] Yackel, T. How the open-source development model can improve medical software. *Medinfo 10* (2001), 68–72.

[163] Yu, L. Self-organization process in open-source software: An empirical study. *Inf. Softw. Technol. 50*, 5 (2008), 361–374.

[164] Yu, L., and Chen, K. Evaluating the post-delivery fault reporting and correction process in closed-source and open-source software. In *WoSQ '07: Proceedings of the 5th International Workshop on Software Quality* (2007).

[165] Zhao, L., and Elbaum, S. Quality assurance under the open source development model. *Journal of Systems and Software 66*, 1 (2003), 65–75.