Current Trends in Free Software Research

Ramon Navarro Bosch
ramon@epsem.upc.edu

Sebastià Vila-Martà
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 analyzed 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 ................................................. 7

2 Free Software ............................................. 8
  2.1 Legal Framework ........................................ 8
  2.2 Historical considerations .............................. 9
  2.3 Other Aspects of Free Software ....................... 10

3 Data Set Acquisition .................................... 11

4 Data Set Characteristics ................................ 12

5 Data Analysis .............................................. 15
  5.1 Preliminary analysis ................................... 15
  5.2 Principal Components Analysis ...................... 15
    5.2.1 Subject + Publication ............................ 16
    5.2.2 Subject + Research Group ....................... 18
    5.2.3 Subject + Isjournal ................................ 18
    5.2.4 Subject + Publication + Year + Isjournal ....... 19
    5.2.5 Subject + Impact Factor ......................... 20
    5.2.6 MCA analysis D.2 .................................. 20
    5.2.7 MCA analysis of K.6 .............................. 20
  5.3 Conclusions ............................................ 22

6 Free Software Main Research Subjects ............... 22
  6.1 K.4 Computers and Society ......................... 23
  6.2 H.3 Information Storage and Retrieval .............. 23
  6.3 K.6 Management of Computing and Information Systems and D.2 Software Engineering ................................................. 23
    6.3.1 D.2.0 General .................................... 24
    6.3.2 D.2.2 Design Tools and Techniques ................ 24
    6.3.3 D.2.4 Software/Program Verification
          K.6.5 Security and Protection .................... 24
    6.3.4 D.2.7 Distribution, Maintenance, and Enhancement 25
    6.3.5 D.2.8 Metrics .................................... 25
    6.3.6 D.2.9 Management
          K.6.4 System Management
          K.6.3 Software Management
          K.6.1 Project and People Management ........... 25
    6.3.7 D.2.12 Interoperability .......................... 27
    6.3.8 D.2.13 Reusable Software ....................... 27

7 Conclusions and future work ........................... 28
1 Introduction

This paper analyzes the scientific 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 Fitzgerald, [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 Fitzgerald, 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 knowledge area. That research activity is mature, is published in journals and have well established 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 meaningful 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. Although 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 proprietary 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 proprietary 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 surrenders 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 unusual 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 modifiable 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 philosophy 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 commercial 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 attention 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 became important players in free software arena. In those companies, free software plays an important strategical 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 communities 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 opportunities for the development teams. The collaboration 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 approach 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 community success factors or free software quality assessment.

Many hardware manufacturers use free software in their products. Network equipment, storage units or printing devices that embed free software are very common. More interesting are manufacturers like Nokia, [7], OpenMoko, [9] or Arduino, [1]. These are involved in developing and selling electronic 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 successful suite of permissive 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 conducted 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 knowledge area.

We proceed by choosing only the publications whose central topic is related to computer science.
Table 1: Number of documents in the data set published each year

<table>
<thead>
<tr>
<th>Year</th>
<th>Number of documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>3</td>
</tr>
<tr>
<td>2000</td>
<td>1</td>
</tr>
<tr>
<td>2001</td>
<td>1</td>
</tr>
<tr>
<td>2002</td>
<td>4</td>
</tr>
<tr>
<td>2003</td>
<td>7</td>
</tr>
<tr>
<td>2004</td>
<td>7</td>
</tr>
<tr>
<td>2005</td>
<td>16</td>
</tr>
<tr>
<td>2006</td>
<td>11</td>
</tr>
<tr>
<td>2007</td>
<td>32</td>
</tr>
<tr>
<td>2008</td>
<td>9</td>
</tr>
</tbody>
</table>

4 Data Set Characteristics

In this section we describe the main characteristics of the publications data set that will be analyzed. This data set is built 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 classification 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 according 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.
<table>
<thead>
<tr>
<th>Area</th>
<th>Name</th>
<th>Number of documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Introductory and survey</td>
<td>1</td>
</tr>
<tr>
<td>C.2</td>
<td>Computer-communication networks</td>
<td>4</td>
</tr>
<tr>
<td>C.5</td>
<td>Computer system implementation</td>
<td>1</td>
</tr>
<tr>
<td>D.1</td>
<td>Programming techniques</td>
<td>1</td>
</tr>
<tr>
<td>D.2</td>
<td>Software engineering</td>
<td>69</td>
</tr>
<tr>
<td>D.3</td>
<td>Programming languages</td>
<td>3</td>
</tr>
<tr>
<td>D.4</td>
<td>Operating systems</td>
<td>5</td>
</tr>
<tr>
<td>F.3</td>
<td>Logics and meanings of programs</td>
<td>1</td>
</tr>
<tr>
<td>H.1</td>
<td>Models and principles</td>
<td>2</td>
</tr>
<tr>
<td>H.2</td>
<td>Database management</td>
<td>3</td>
</tr>
<tr>
<td>H.3</td>
<td>Information storage and retrieval</td>
<td>3</td>
</tr>
<tr>
<td>H.4</td>
<td>Information systems applications</td>
<td>4</td>
</tr>
<tr>
<td>H.5</td>
<td>Information interfaces and presentation</td>
<td>10</td>
</tr>
<tr>
<td>I.2</td>
<td>Artificial intelligence</td>
<td>2</td>
</tr>
<tr>
<td>I.6</td>
<td>Simulation and modeling</td>
<td>2</td>
</tr>
<tr>
<td>J.1</td>
<td>Administrative data processing</td>
<td>3</td>
</tr>
<tr>
<td>J.m</td>
<td>Computer applications miscellaneous</td>
<td>1</td>
</tr>
<tr>
<td>K.1</td>
<td>The computer industry</td>
<td>4</td>
</tr>
<tr>
<td>K.2</td>
<td>History of computing</td>
<td>1</td>
</tr>
<tr>
<td>K.3</td>
<td>Computers and education</td>
<td>4</td>
</tr>
<tr>
<td>K.4</td>
<td>Computers and society</td>
<td>5</td>
</tr>
<tr>
<td>K.6</td>
<td>Management of computing and information systems</td>
<td>53</td>
</tr>
<tr>
<td>K.8</td>
<td>Personal computing</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2: ACM subjects found in the data set

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

Table 3: The data set contains papers from these journals.
<table>
<thead>
<tr>
<th>Name</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM workshop on Interdisciplinary software engineering research</td>
<td>WISER</td>
</tr>
<tr>
<td>Computer Supported Cooperative Work</td>
<td>CSCW</td>
</tr>
<tr>
<td>Conference on Software Engineering Education and Training</td>
<td>CSEET</td>
</tr>
<tr>
<td>Conference on Supporting Group Work</td>
<td>GROUP</td>
</tr>
<tr>
<td>ESEC FSE: International Workshop on Principles of Software Evolution</td>
<td>IWPSE</td>
</tr>
<tr>
<td>European Software Engineering Conference</td>
<td>ESEC-FSE</td>
</tr>
<tr>
<td>Free Libre Open Source Software Conference</td>
<td>FLOSS</td>
</tr>
<tr>
<td>Hawaii International Conference on System Sciences</td>
<td>HICSS</td>
</tr>
<tr>
<td>Human-Computer Interaction</td>
<td>HCI</td>
</tr>
<tr>
<td>ICSE: International Workshop on Mining Software Repositories</td>
<td>MSR</td>
</tr>
<tr>
<td>ICSE: Workshop on Open Source Software Engineering</td>
<td>WOSSE</td>
</tr>
<tr>
<td>IEEE International Conference on Software Maintenance</td>
<td>ICSM</td>
</tr>
<tr>
<td>International Conference on Information Systems</td>
<td>ICIS</td>
</tr>
<tr>
<td>International Conference on Software Engineering</td>
<td>ICSE</td>
</tr>
<tr>
<td>Technical Symposium on Computer Science Education</td>
<td>SIGCSE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Label</th>
<th>Impact factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXXXL</td>
<td>[4,3)</td>
</tr>
<tr>
<td>XXXL</td>
<td>[3,2)</td>
</tr>
<tr>
<td>XXL</td>
<td>[2,1.5)</td>
</tr>
<tr>
<td>XL</td>
<td>[1.5,1)</td>
</tr>
<tr>
<td>L</td>
<td>[1,0.7)</td>
</tr>
<tr>
<td>M</td>
<td>[0.7,0.5)</td>
</tr>
<tr>
<td>S</td>
<td>[0.5,0.2)</td>
</tr>
<tr>
<td>XS</td>
<td>[0.2, 0)</td>
</tr>
</tbody>
</table>

Table 5: Conversion table used to discretize impact factor.
<table>
<thead>
<tr>
<th>Group</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependability at School of Computing Science</td>
<td>Newcastle University</td>
</tr>
<tr>
<td>Institute for Software Research - Open Software Development</td>
<td>California University</td>
</tr>
<tr>
<td>Libre Software Group GSyC</td>
<td>Universidad Rey Juan Carlos</td>
</tr>
<tr>
<td>Software Engineering Group</td>
<td>Aristotle University of Thessaloniki</td>
</tr>
<tr>
<td>Software Engineering Group</td>
<td>University of Victoria</td>
</tr>
<tr>
<td>Syracuse FLOSS Research Group</td>
<td>Syracuse University</td>
</tr>
<tr>
<td>The interaction lab</td>
<td>University of Saskatchewan</td>
</tr>
</tbody>
</table>

Table 6: Research Groups

**isjournal** This variable that discriminates 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 Preliminary analysis

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 $\mathbb{R}^n$ 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 $\mathbb{R}^n$ 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 \textit{R}, [13], together with the \texttt{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 precisely. 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 conduct 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 assigning 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 individually 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 intelligence”, 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 suggests 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.

\subsection*{5.2.1 Subject + Publication}

This analysis studies the relationship between the variable \texttt{subject} and \texttt{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 analysis results for variables subject and publication

Figure 2: MCA analysis results for variables subject and group
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:
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 majority of publications are on group topic conferences (GROUP, WOSSE, CSCW, CHI).
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 analysis results for variables subject and impact factor

Figure 7: MCA analysis 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 majority 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 classification.
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 effect 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 imply 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], Spinelli 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 different roles and the different paths from peripheral 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 interoperability, Cerri concludes that free software promotes interoperability.

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 studying the benefits of reutilization of free software components on commercial 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 reusing 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 analyzed. 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 exponential 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


[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).


