Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time*

Gregorio Robles, Santiago Dueñas, Jesus M. Gonzalez-Barahona

GSyC/LibreSoft, Universidad Rey Juan Carlos (Madrid, Spain)
{grex,sduenas,jgb}@gsyc.escet.urjc.es

Abstract. Although much of the research on the libre (free, open source) phenomenon has been focused on the involvement of volunteers, the role of companies is also important in many projects. In fact, during the last years, the involvement of companies in the libre software world seems to be raising. In this paper we present an study that shows, quantitatively, how important this involvement is in the production of the largest collection of code available for Linux: the Debian GNU/Linux distribution. By studying copyright attributions in source code, we have identified those companies with more attributed code, and the trend of corporate presence in Debian from 1998 to 2004.
Keywords: open source, libre software, involvement of companies, empirical study, software business

1 Introduction

For companies producing computer programs, libre software[2] is not yet another competitor playing with the same rules. The production of libre software differs from *traditional* software development in many fundamental aspects, ranging from ethical and psychological motivation to new economic and marketing premises, to new practices and procedures in the development process itself.

One of the key differences is the different role of users. While in the *classical* software development environment the development team can be clearly distinguished from the users, most of the libre software projects develop around themselves a community [7]. This community is usually formed by people with many different involvements, from pure users to core developers, including many mixed roles, such as that of users contributing with patches (small modifications) to the code. Therefore, in most libre software projects we may observe a *continuum* of commitment to the project which includes a wide range of occasional contributors.

* This work has been funded in part by the European Commission, under the FLOSS-METRICS (FP6-IST-5-033547) and FLOSSWORLD (FP6-IST-015722) projects.

[2] Through this paper the term "libre software" will be used to refer to code that conforms either to the definition of "free software" (according to the Free Software Foundation) or of "open source software" (according to the Open Source Initiative).

Some software companies have realized this fact, sponsoring and promoting projects with the aim of benefitting from the development of a strong community around them. Some of the most known cases in this realm are Sun Microsystems (OpenOffice.org, OpenSolaris, GNOME, among others), IBM (Apache, Eclipse, etc.) or Apple (Darwin, the kernel of Mac OS X). Be it for this reason or for any other, the involvement of companies in libre software development is strong, and increasing with time.

While there has been some empirical research on self-organized software development in libre software (among others, see [6, 19]) and especially on activities performed by volunteers [12, 17], including their integration process [20], the involvement of software companies in the phenomenon has been rarely attended and if it has been mainly from the point of view of business models, business case studies, and the motivations behind companies [4, 5, 2, 3]. Hence, the focus of these papers can be understood from the perspective of companies wanting to understand, invest or to guide them to successfully get integrated into the libre software phenomenon.

Many of the companies that work with libre software just take already written code and adapt it without providing feedback to the community, but some others actively participate in libre software projects. In general, what these companies are looking for in libre software is to obtain a surrounding user community which serves both as a basic and fast feedback mechanism, but also as a marketing strategy, with the aim of getting software of better quality by letting external brainware access the project's source code, to lower the cost by letting volunteers enhance or fix the software, among others.

The target of this paper is to measure the involvement of libre software companies in libre software, specifically of those that deliver the code to the community. For this, we will analyze the source code available in the Debian GNU/Linux stable distributions, which contain in its latest version more than 10,000 source code packages (usually applications, but also libraries and other components). As the sources of several Debian stable releases are available, we will apply our methodology to five of them, spawning from 1998 to 2005, therefore tracking the evolution of the participation of companies during a period of 7 years.

The rest of this paper is structured as follows: next, the methodology for our empirical study will be presented. In this section, we will present the data sources and the procedures we have used to extract and analyze them. We will discuss why the use of copyright statements is significant for our approach and will include some refinements in our methodology to avoid double counting files. In the following section, we will present the results of applying the methodology to Debian, the largest libre software GNU/Linux distribution. We will provide results over time to compare the evolution of the involvement. Finally, conclusions, limitations and future research opportunities are presented.

2 Methodology

Contrary to popular belief, libre software authors rely on copyright law to enforce their licenses, and therefore include copyright marks in their programs. This is especially true for companies interested in maintaining ownership on the code their (hired) developers have produced. Even when the software is licensed under libre software licenses, the copyright owner has some privileges (such as relicensing under other licenses) which are usually appealing to companies. Companies also tend to have strict policies about copyright notices in source code files.

But it is not only in the interest of companies to retain copyright. For instance, non-profit organizations such as the FSF or the GNOME Foundation actively ask developers to assign the copyright to them. The reason for this is that these bodies may have stronger legal bodies to defend themselves from license infringement.

Therefore, it is very likely that if a source code file is owned by a company or one of these organizations, its copyright notice will appear in it. Usually, individual authors also include their copyright attribution, but they may be not that strict about that. In any case, the methodology of the study described in this paper is based on the assumption of the existence of those notices.

As the rest of the software industry does, copyright notices are included (among other places) at the beginning of each file with source code [18]. That means that information about the copyright holder can be extracted from source code files. For instance, the notice in the apps/units.c file of the GIMP project (see figure 1) clearly states that the copyright holders are Spencer Kimball and Peter Mattis, and that the license in use is the GNU General Public License. In any case, it should be noted that the way of stating the copyright is not unique and may change from project to project, even from file to file.

```
/* The GIMP -- an image manipulation program
 * Copyright (C) 1995 Spencer Kimball and Peter Mattis
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
[...]
```

Fig. 1. First lines of file apps/units.c, of the GIMP project

To identify those copyright notices, and extract from them information about the copyright owner, we designed the methodology described in the following subsections, and implemented it by producing pyTernity[3]. The structure

[3] The most current version of pyTernity can be found at `https://forge.morfeo-project.org/projects/libresoft-tools/`

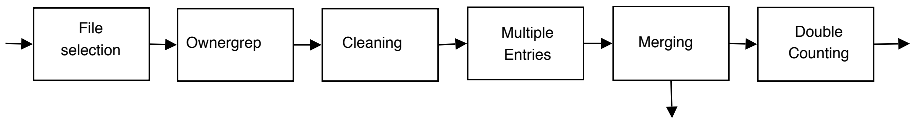

Fig. 2. Block diagram with the various components of pyTernity

of the methodology, which corresponds to the architecture of pyTernity, is shown in figure 2. The result is a list of files (avoiding similar files), with their size (in SLOC, lines, and characters) and the copyright holders identified in them.

2.1 File selection and counting

First step in the methodology is the identification of source code files. This is usually performed by using some ad-hoc heuristics, which may vary in their accuracy as well as in the granularity of their results. We use two sets of heuristics in our discrimination process: the extension in the file name, and the content. Detailed information about the process can be obtained from [13]. With these heuristics, almost all source code files are identified [15].

Every file identified as source code is then counted using SLOCCount[4], a tool authored by David Wheeler that calculates the number of SLOC (source lines of code). SLOCCount has been used in many studies about the size of software collections [21, 10, 1, 11, 14]. It calculates the number of physical source lines of code (SLOC) of a software program. The Unix wc command is also used to estimate the number of characters and lines of the file. All this information is stored in a database, linked to the file name.

2.2 Ownergrep

The second step is to search for copyright notices in source code files. For that, the *ownergrep* expression is compared with every line in the file. Since there is no standard or widely-used way of stating copyright in files, the pattern requires flexibility, which is achieved by the use of regular expressions that allow matching multiple, slightly different ways of expressing the copyright notice.

The *ownergrep* expression is a modification of the original one by Prakash and Ghosh (cite) and looks like this:

```
[1] .*copyright (?:\(c\))?[\d\,\-\s\:]+(?:by\s+)?([^\d]*)
```

Figure 3 presents some copyright entries that can be identified by the *ownergrep* expression. Identities found are stored in a database, linked to the files in which they were present (and therefore, also to their size).

[4] http://www.dwheeler.com/sloccount/

```
Copyright (c) 1998, 1999 by Sun Microsystems, Inc. All Rights Reserved.
Copyright (c) 2001-2, Vipul Ved Prakash.  All rights reserved.
Copyright (c) 2006 IBM Corporation and others.
```

Fig. 3. Some copyright entries that can be matched by the *ownergrep* expression.

2.3 Cleaning

Identities stored in the database have to be cleaned. This means removing all non-relevant information to convert identified identities to their canonical format. This ranges from removing additional white-spaces to the deletion of dots. Some ad-hoc heuristics are used, along with the complement of a database of common transformations. Cleaning also includes splitting up an entry when it corresponds to two or more authors. So, the entry "Spencer Kimball and Peter Mattis" will result in two, one for Spencer Kimball and another one for Peter Mattis. If this is the case, both names appear as authors of the file and get attributed half of its length.

2.4 Multiple entries

After cleaning found identities, those corresponding to the same real entity are identified. Developers and companies may appear in several forms while corresponding to one single entity. The first idea in this line resulted in the construction of a large database where the various entries identified for a given developer were noted (manually). This method proved to enhance results considerably. However, the consideration of other methods, and the rising in complexity and size of the database have finally lead to the construction of a different tool, Seal, that returns a unique identifier for any given identity [16]. It is responsibility of this external tool to track all developers and to manage them properly.

2.5 Merging

Once cleaning has been performed and multiple entries have been identified, pyTernity merges the identities in the database so that authors appear only once in a file. This procedure also includes adding the size of all the files corresponding to each real identity.

2.6 Avoiding double-counting

In a large collection of software, some files may appear in several packages. That means that the copyright owner of one such file will be attributed the same code several times, which leads to inconsistencies. Therefore, similar files have to be identifies and removed from the count. For this, we use Nilsimsa[5], a hashing

[5] The Nilsimsa code can be retrieved from `http://ixazon.dynip.com/\%7ecmeclax/nilsimsa.html`

algorithm that produces similar hashes (according to a certain metric, based on Hamming distance) for similar texts. For our methodology, 32 bits of Hamming distance are used as the threshold for considering two files too similar to count them twice.

Unfortunately, comparing proximity of Nilsimsa hashes is meaningfully slower than comparing for equality. Therefore, comparing Nilsimsa hashes for every pair of files is not practical for large quantities of software. To avoid this situation, we propose a simplified use of Nilsimsa by (1) identifying files with the same Nilsimsa and similar amount of code in order to avoid false positives and (2) comparing by pairs all those files with the same filename. Table 1 shows the number of files with the same name and a similar nilsimsa hash that have been discovered for all versions of Debian.

Version	Total files	Same filename, similar Nilsimsa	Percentage
Debian 2.0	243,057	45,850	18.86%
Debian 2.1	367,463	80,551	21.92%
Debian 2.2	838,834	238,601	28.44%
Debian 3.0	1,340,081	292,367	21.82%
Debian 3.1	2,497,636	420,885	16.85%

Table 1. Total number of files and files that have the same file name and a similar Nilsimsa hash for every version under study.

2.7 Previous work

CODD, a tool designed by Rishab A. Ghosh and Vipul Prakash [8], was the first tool to extract authorship information from source code by tracking copyright notices. Its main aim is to assign the length (in bytes) of each file to the corresponding authors. It was successfully used in the Orbiten Survey [8], the source code survey in the FLOSS study [9], and some other research projects.

CODD is a very powerful tool which implements a methodology similar (in part) to pyTernity, but shows also some weaknesses. The most important one is that it lacks a way of merging the various ways in which an author may appear. So, authors may appear several times with different names or e-mail addresses. For instance, we have found that some developers have up to 15 different identities which may appear in copyright notices. In the case of companies and organizations, the same may happen: IBM or the MIT appear in several ways (up to twenty!) with slightly different wordings.

CODD also includes some heuristics for cleaning the extracted data. Although they have proven to be very powerful, these heuristics can not deal with enough accuracy with the fact that developers use different conventions to assign copyright.

Both limitations are important, and were the main reasons to create pyTernity. The *ownergrep* expression used in pyTernity is a modification of the original one in CODD.

3 Results on Debian

The methodology presented in the previous section has been applied to several releases of the Debian GNU/Linux distribution. In the next subsections an introduction to the Debian project and the results of our study are shown.

3.1 Introduction to Debian and global results

Debian is a libre operating system that, at present time, uses the Linux kernel to carry out its distribution (although there are some efforts to make that future Debian distributions could be based on other kernels). The distribution has a categorization of software packages according to their license and their distribution requirements. The main part of the Debian distribution (the section called *main*, which contains a large variety of packages) is compound only of libre software in agreement with the Debian Free Software Guidelines. It is available for download from the Internet and many resellers supply it on CDs or by other means.

The Debian distribution is created by over a thousand volunteers (generally computer professionals). The work of these volunteers consists on taking the source programs -in most of the cases from their original author(s)-, to configure them, to compile them and to pack them, so that a typical user of a Debian distribution only has to select the package to be installed/updated/removed. Hence, being a software included in Debian depends only on a volunteer performing the aforementioned tasks.

	Codename	Release	Packages	Total SLOC	Companies SLOC	%	# Companies
2.0	Hamm	Jul 1998	1,096	28,750,853	4,259,164	6.75%	249
2.1	Slink	Mar 1999	1,551	44,352,088	6,477,981	6.85%	312
2.2	Potato	Aug 2000	2,611	95,738,163	14,934,951	6.41%	482
3.0	Woody	Jul 2002	4,579	151,023,303	23,271,027	6.49%	782
3.1	Sarge	Jun 2005	8,560	239,580,490	40,421,751	5.93%	1455

Fig. 4. Some information about the Debian distributions under study: version number, *Toy Story* codename, release date, number of source code packages, total number of SLOC, SLOC attributed to companies, share of code by companies and number of different companies identified.

Table 4 gives further details about the various releases that have been studied in this paper and about the involvement of firms in them. As it is already known from previous studies [14] the size of Debian seems to double almost every two years. It is noteworthy that the amount of code that can be attributed to companies stays almost constant over time with values that lie around 6%-7%, throwing that the number of lines of code contributed by companies grows at the same pace than the distribution. The number of companies that could be identified has also increased significantly from over 200 in 1998 to almost 1500 in the most recent stable version. In any case, from our empirical analysis we can conclude that the involvement of industry in the libre software phenomenon has grown substantially in the last 8 years, although its relative importance has remained constant. It should be noted that this may not mean that the participation of the software industry has not been raising in the last years as other activities different from development such as support, consultancy and deployment without providing feedback to the community are not considered with our methodology.

3.2 Top companies by non-double-counted SLOC

Tables 5, 6 and 7 give the contribution of companies found in the various Debian releases under study. Contributions are measured in SLOC, avoiding double counting as explained in the methodology. These stats may give an idea of the effort spent by the company in the development of libre software.

Company name	SLOC	Files
sun microsystems inc.	801,632	2644
digital equipment corp.	434,152	1119
silicon graphics corp.	277,992	1274
xerox corp.	207,623	736
aladdin enterprises	92,172	475
age logic inc.	79,458	217
nec corp.	78,538	135
e.i. du pont de nemours	76,458	45
hewlett packard corp.	71,201	283
evans & sutherland	66,840	95

Company name	SLOC	Files
netscape comm. corp.	1,129,302	3934
sun microsystems inc.	810,437	2716
digital equipment corp.	428,176	1100
silicon graphics corp.	277,409	1207
aladdin enterprises	141,652	656
xerox corp.	98,071	342
lucent technologies inc.	85,586	139
at&t	80,140	223
age logic inc.	79,458	217
nec corp.	78,538	135

Fig. 5. Top-contributing companies for Debian 2.0 and Debian 2.1 (non-double-counted SLOC).

SUN Microsystems has historically been among the most contributing firms in terms of lines of code. For the first four Debian versions considered, its contribution was slightly less than one million lines of code, but with the inclusion of OpenOffice.org in Debian 3.1 its share has increased notably with over 5 MSLOC. IBM is another software *giant* present in this list, although its appearance is more recent (it enters the top 10 only in Debian 2.2). Interestingly enough, we find that the third place in Debian 3.1 is occupied by a company which is the main driver of a competing distribution to Debian, Red Hat Corp.

Company name	SLOC	Files	Company name	SLOC	Files
netscape comm. corp.	2,651,592	10423	ibm corp.	1,258,263	4832
sun microsystems inc.	1,086,765	4418	sun microsystems inc.	955,462	3276
digital equipment corp.	975,178	5355	digital equipment corp.	784,279	4810
silicon graphics corp.	310,640	1308	trolltech as	587,784	1836
aladdin enterprises	296,933	1403	silicon graphics corp.	575,810	2818
ibm corp.	226,386	479	red hat corp.	376,099	878
trolltech as	147,154	587	static free software	292,448	242
lucent technologies inc.	146,014	208	aladdin enterprises	284,422	1422
e.i. du pont de nemours	140,351	136	abisource inc.	232,795	1530
xerox corp.	128,922	444	hewlett packard corp.	208,903	707

Fig. 6. Top-contributing companies for Debian 2.2 and Debian 3.0 (non-double-counted SLOC).

Company name	SLOC	Files
sun microsystems inc.	6,025,680	22,720
ibm corp.	1,991,300	6,953
red hat corp.	1,366,298	4,807
silicon graphics corp.	1,111,431	4,422
sap ag	1,080,246	4548
mysql ab	852,394	2,425
netscape communications corp.	786,070	2,780
ximian inc.	750,761	2,924
realnetworks inc	673,167	2453
at&t	656,045	2,620

Fig. 7. Top-contributing companies for Debian 3.1 (non-double-counted SLOC).

This is due to its participation in projects such as GNOME or the GCC compiler collection[6].

4 Conclusions

In this paper we have described a methodology (and pyTernity, a tool implementing it) which can be used to scan source code files and find their copyright owners. This methodology is used to estimate (over time) the quantity of code owned by companies in the largest libre software collection: the Debian GNU/Linux.

The information resulting from this estimation is a first try with the aim of answering several questions about the presence of companies in libre software development. For instance, the share of code owned by companies has been calculated (being around 6%-7% for all the studied releases of Debian, with some tendency to lower), and the list of the companies contributing with more

[6] Cygnus Solutions Inc. was acquired by Red Hat in 1999.

code - which is leaded by giants like Sun Microsystems, IBM, SAP, Silicon Graphics or AT&T, but also includes more small, focused on libre software companies like Red Hat, Ximian (now owned by Novell) or MySQL.

The described methodology can provide this landscape of company involvement, but has to be considered with some care, since several sources of potential errors do exist. To begin with, it is completely based on the accurate identification of copyright notices, and correct extraction of identities from them. This is based in heuristics which, even having been validated in several ways, may not completely identify some copyright owners, or could wrongly assign code to others. In addition, the identification of multiple identities for a single identity, or the presence of several copies of some source code files could lead to miscalculations, which are dealt with by the methodology, but again using heuristics with a certain chance of error.

However, manual validation of a random set of results seem to lead to the conclusion that the results are good enough for using them to better understand how much code from companies can be found.

Several lines of research are still open in this area. First of all, further developments in heuristics to better identify companies and other institutions from copyright notices would improve the accuracy of results. In addition, improvements in the merging of different identifications corresponding to the same entity would also help.

Correlation of these data, and comparison with performance parameters of both the products and the companies with different levels of involvement in libre software development are also promising lines for interesting results. Methodologies to automatically assess companies and other entities about errors in copyright attributions, and about ownership of the code corresponding to software they use could also be of great interest to industry.

5 Acknowledgments

We would like to thank Diego Barceló and the rest of the GSyC/LibreSoft research group at the URJC for their invaluable help with the implementation and testing of the software used in this study. We are also very grateful to Rishab A. Ghosh, Rüdiger Glott and Kirsten Haaland from UNU-MERIT/University of Maastricht.

References

1. Juan José Amor, Jesús M. González-Barahona, Gregorio Robles, and Israel Herraiz. Measuring libre software using Debian 3.1 (sarge) as a case study: preliminary results. *Upgrade Magazine*, August 2005.
2. Andrea Bonaccorsi and Cristina Rossi. Comparing motivations of individual programmers and firms to take part in the open source movement. from community

to business. Technical report, University of Pisa; Sant'Anna School of Advanced Studies, Italy, 2003.

3. Andrea Bonaccorsi and Cristina Rossi. Altruistic individuals, selfish firms? the structure of motivation in open source software. *First Monday*, 1(9), January 2004.
4. Andrea Bonaccorsi and Cristina Rossi. Open source software, intrinsic motivations and profit-oriented firms. do not firms practise what they preach? In *Proceedings of the 1st International Conference on Open Source Systems*, Genoa, Italy, July 2005.
5. Andrea Bonaccorsi, Cristina Rossi, and Silvia Giannangeli. Adaptive entry strategies under dominant standards: Hybrid business models in the open source software industry. Technical report, University of Pisa; Sant'Anna School of Advanced Studies, Italy, 2003.
6. Kevin Crowston and James Howison. The social structure of open source software development teams. In *Proceedings of the International Conference on Information Systems*, Seattle, WA, USA, 2003.
7. Kevin Crowston and James Howison. The social structure of free and open source software development. *First Monday*, 10(2), February 2005.
8. Rishab A. Ghosh and Vipul Ved Prakash. The orbiten free software survey. *First Monday*, 5(7), May 2000.
9. Rishab Aiyer Ghosh, Gregorio Robles, and Ruediger Glott. Software source code survey (free/libre and open source software: Survey and study). Technical report, International Institute of Infonomics. University of Maastricht, The Netherlands, June 2002.
10. Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quiros, José Centeno González, and Vicente Matellán Olivera. Counting potatoes: the size of Debian 2.2. *Upgrade Magazine*, II(6):60–66, December 2001.
11. Jesús M. González-Barahona, Gregorio Robles, Miguel Ortuño Pérez, Luis Rodero-Merino, José Centeno González, Vicente Matellan-Olivera, Eva Castro-Barbero, and Pedro de-las Heras-Quirós. Analyzing the anatomy of GNU/Linux distributions: methodology and case studies (Red Hat and Debian). In Stefan Koch, editor, *Free/Open Source Software Development*, pages 27–58. Idea Group Publishing, Hershey, Pennsylvania, USA, 2004.
12. Martin Michlmayr. Managing volunteer activity in free software projects. In *Proceedings of the USENIX 2004 Annual Technical Conference, FREENIX Track*, pages 93–102, Boston, USA, 2004.
13. Gregorio Robles. *Empirical Software Engineering Research on Libre Software: Data Sources, Methodologies and Results*. PhD thesis, Escuela Superior de Ciencias Experimentales y Tecnología, Universidad Rey Juan Carlos, 2006.
14. Gregorio Robles, Jesus M. Gonzalez-Barahona, Martin Michlmayr, and Juan Jose Amor. Mining large software compilations over time: Another perspective of software evolution. In *Proceedings of the Third International Workshop on Mining Software Repositories*, pages 3–9, Shanghai, China, May 2006.
15. Gregorio Robles, Jesus M. González-Barahona, and Juan-Julián Merelo. Beyond executable source code: The importance of other source artifacts in software development (a case study). *Journal of Systems and Software*, 80(9):1233–1248, September 2006.
16. Gregorio Robles and Jesús M. González-Barahona. Developer identification methods for integrated data from various sources. In *Proceedings of the International*

Workshop on Mining Software Repositories, pages 106–110, St. Louis, Missouri, USA, May 2005.

17. Gregorio Robles, Jesús M. González-Barahona, and Martin Michlmayr. Evolution of volunteer participation in libre software projects: evidence from Debian. In *Proceedings of the 1st International Conference on Open Source Systems*, pages 100–107, Genoa, Italy, July 2005.
18. Diomidis Spinellis. *Code Reading: The Open Source Perspective.* Addison Wesley Professional, 2003.
19. Ilkka Tuomi. Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research. *First Monday*, 9(6), June 2004.
20. Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in Open Source Software innovation: A case study. *MIT Sloan Working Paper No. 4413-03*, 2003.
21. David A. Wheeler. More than a gigabuck: Estimating GNU/Linux's size, June 2001.