From Pigs to Stripes: A Travel through Debian[*]

Juan José Amor, Gregorio Robles, Jesús M. González-Barahona, and Israel Herraiz

June 2005

Keywords

libre software, Debian, GNU/Linux, libre software engineering, software evolution, lines of code, COCOMO

Abstract

The Debian GNU/Linux is one of today's most popular Linux-based distributions. It is intended not only for final users, but also as a basis for other projects which can build on top of it, as is the case of some well-known "live" distributions and meta-distributions. Since its beginnings, more than one decade ago, it has undergone many technical and organizational changes, while experimenting an spectacular growth. This paper is a study of the evolution of Debian GNU/Linux in time for the last five stable releases (from 2.0 to the upcoming 3.1 version). We show results for the size in packages of the different releases, the size in number of source lines of code, the importance of the various programming languages in which the software is written, the evolution of the number of developers. We also apply the COCOMO "classical" cost estimation model to the whole distribution, which can be considered as an first approximation of the effort (and cost) for creating the software in Debian from scratch, using traditional development models.

1 Introduction

In the early nineties, the first GNU/Linux distributions arose from the union of the GNU tools, the Linux kernel, and some other pieces of libre software[1]. Their main purpose was to facilitate the installation and use of the software as much as possible, something that at the time was an arduous task,

[*]This work has been funded in part by the European Commission, under the CALIBRE CA, IST program, contract number 004337, by the Universidad Rey Juan Carlos under project PPR-2004-42 and by the Spanish CICyT under project TIN2004-07296.

[1]Through this article, we use "libre software" as a way of referring both to free software and open source software. Though open source software and free software communities are very different, the software is not, since almost all licenses considered to be "free" are also considered "open source", and the other way around.

requiring large amounts of patience and knowledge. Later, during the mid nineties, the package management systems emerged as another important innovation in libre software distributions. Those tools allowed not only to install a distribution easily, but also offered the possibility to manage (remove, add or update) packages once they had been installed.

Distributions filled, consequently, a space that in the world of proprietary software seldom reaches important proportions: integrators. Their work consists on taking the sources, usually from their original (upstream) author(s), to group them together with other tools and applications that could be interesting, and to pack everything together in such a way that the task of installing or of updating enormous amounts of packages is easy enough for the common end-user.

Organizations and companies that create distributions are also responsible for the quality of the final product, a very important task if we consider that most of the libre software projects are managed by volunteers [Michlmayr2003]. In this sense, they are the responsible face to their users for the stability and security of the resulting distribution. Therefore, it is not difficult to imagine why distributions soon occupied an important position, as far as the popularization of libre software in general and GNU/Linux systems in particular is concerned.

A large number of distributions do exist, each with its own peculiarities: commercial aim (or not), size (in number of packages, in lines of code, etc.), release policy, etc. This study is focused on one of them, the widely extended and popular Debian GNU/Linux. We analyze its latest five stable releases (from Debian 2.0 to the upcoming Debian 3.1), which cover a period of about seven years. The approach followed for the study is based mainly on the inspection of the source code for the packages, with the aim of getting some insight on how this code is evolving from a macro point of view. As a complement, a (less detailed) analysis of the developer population is also performed.

Some important findings of the study are: the current size of Debian, about 230 MSLOC (millions of source lines of code); the evolution of this size in time (doubling about every two years); the packages remaining from latest releases (for instance, 158 of the 1,096 Debian 2.0 source packages are still in Debian 3.1, with the same version number); the share of the dominant languages (C is the first one, but descending from more than 75% in Debian 2.0 to about 57% in Debian 3.1); and the COCOMO-estimated substitution cost for all software in Debian, about USD 8 billion.

Most of the data in this paper are presented as a brief summary, without digging into details. We suggest the interested reader to visit the Debian Counting web site [DebianCounting], where more detailed statistical information, graphs and data can be found. In addition, the Libresoft web site [Libresoft] contains more papers and information on libre software engineering [GBarahona2003b].

The structure of this paper is as follows. Next, the Debian project is briefly introduced, followed by an study on the evolution of the number of voluntary Debian developers. In the fourth section we introduce the methodology we have used for collecting the data shown later: size of the systems as a whole, size of packages, and usage of languages in the distribution. Some of these data are used, in the next section, for estimating substition cost, using COCOMO. The ninth section compares the size of Debian GNU/Linux with other GNU/Linux distributions, followed by a comparison with other operating systems and applications. The paper ends with the usual section on conclusions.

2 About Debian

Debian is a complete operating system, with thousands of applications, composed exclusively of libre software (according to the Debian Free Software Guidelines [DFSG]), that uses Linux as the underlying kernel (although there are some efforts to support in the future some other kernels, such as the HURD or FreeBSD). It is available for several hardware architectures, including Intel x86, ARM, 680x0, PowerPC, Alpha and SPARC. Debian is not only the largest GNU/Linux distribution, it is also one of the most stable ones, enjoying several prizes based on user preferences. Although its number of users is difficult to estimate, since the Debian project does not directly market CDs, and the software that it contains can be redistributed by anyone who desires to do so (and is in fact redistributed, as such or modified, by many parties), we can assume with no doubt that it is an important distribution within the GNU/Linux market. One of the key characteristics of Debian as a project is the Debian Social Contract [DebianSocialContract] (agreed by all developers) which details not only the primary goals of the Debian project, but also the means that will be used to carry them out.

The Debian distribution has been created by around one thousand developers (volunteers, mostly IT-related professionals and students). The work of these volunteers consists on taking the source programs (in most of the cases from their original, upstream authors), to configure, compile, test and finally pack them. The final product of this work is one or more packages, which a user can select for installation it her system (and later for update or removal). In principle, the work of the Debian developers may seem easy, but the needs of customization (to conform the the Debian standards) and coordination (to define relationships and dependencies with other packages) make is, in many cases, quite complex.

The work performed by the Debian project is similar that by any other distribution producer: software integration. However, in addition to the adaptation and packaging work, Debian developers are responsible for the maintenance of the Internet infrastructure of services for the project (web site, on-line archives, bug management systems, mailing lists, support and development, etc.), for several translation and internationalization projects, for the development of several Debian-specific tools and, in general, for any other element that makes the Debian distribution possible.

Debian is also well-known for following a strict packaging and versioning policy, with the aim of improving quality [DebianPol]. As a consequence of this policy, three different "flavors" of Debian exist: stable, unstable and testing (yet another one exists: Debian experimental, which will be described below). As the name suggests, the stable version is targeted to systems and people looking for high stability. Its software has to pass a freezing period in which only critical errors are corrected. The rule is that when a stable version is released it should not contain known critical bugs. On the other hand, because of the freezing period, stable versions usually do not include the most recent software versions. Those who want to use a version with more up-to-date software, can download testing, which includes packages that are on the way of becoming stabilized, or even unstable version, more inclined to fail but containing the latest of the latest on libre software applications and tools. The fourth "flavor", experimental, includes packages still in early stages of development.

Distributions are composed of packages, usually corresponding to applications or libraries (although they can also contain documentation or even data). In Debian, there are two quite different kinds of packages: source and binary. The former contains the sources of applications and libraries that, once compiled and linked, may produce several binary packages. Binary packages are those which users generally install in their computers. For example, Debian 3.0 consists of about 4,500 source packages, but has around 10,000 binary packages.

At the moment of this study, the stable version of Debian is Debian 3.0 (also known as "Woody"). The next new release, still in test phase, has been announced for June 2005. Its release number will be 3.1, and is codenamed "Sarge". When this release becomes stable, a new testing release will start. Its codename will be "Etch". As a rule, unstable releases are always called "Sid". In this paper we are going to consider the stable versions of Debian since version 2.0, which was published in 1998, to the present (in fact, to the foreseeable future): Debian 2.0 (alias “Hamm”), Debian 2.1 (“Slink”), Debian 2.2 (“Potato”), Debian 3.0 (“Woody”) and, finally, Debian 3.1 (“Sarge”).

The codenames of the versions in Debian correspond to the main characters of the animated cartoon film “Toy Story”, a tradition that started with version 1.1 when Bruce Perens, then leader of the Debian project, worked for the company that was in charge of producing these movies. More details on the history of Debian and the Debian distribution in general can be found in [DebianHistory].

3 Evolution of the number of Debian developers

From June 1999 onwards, the Debian project maintains a database [DBDebian] with data related to its members. Some data can be retrieved publicly from it: name, nick or username, e-mail address and PGP/GPG key. In addition, it includes information about the country of residence and the date when joining the project (if later than the database creation, else June 20th 1999), except for the period between June 2003 and June 2005, for which we have not found data about joining date. For this paper, we have processed this information to obtain information about the evolution of the number of developers and countries in which they reside. A similar study to the one presented here (dating from 2001) can be found in [Robles2001], while for a more detailed study on how the participation of Debian package maintainers has evolved in the last has been reported in [Robles2005].

In Figure 1 we can see the number of Debian developers at the moments of releasing a new stable version. We have included in this picture data from [Lameter2002] for releases that go back to Debian 0.96R3 version in 1995. These numbers seem to be approximative.

Table 1 Number of Debian developers

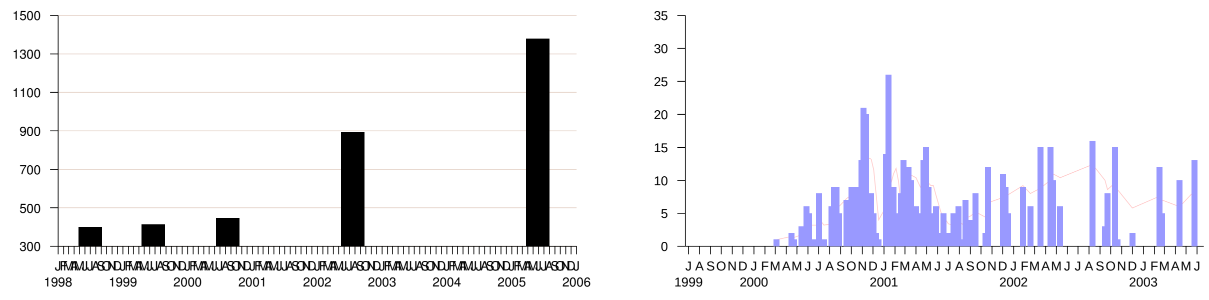
Date	Number of Debian developers	Release	Source
1995/08/01	60	Debian 0.96R3	[Lameter2002]
1996/06/01	90	Debian 1.1	[Lameter2002]
1996/12/01	120	Debian 1.2	[Lameter2002]
1997/07/01	200	Debian 1.3	[Lameter2002]
1998/07/01	400	Debian 2.0	[Lameter2002]
1999/06/21	413	Debian 2.1	[DBDebian]
2000/08/15	448	Debian 2.2	[DBDebian]
2002/07/19	892	Debian 3.0	[DBDebian]
2005/05/28	1380	Debian 3.1 prerelease	[DBDebian]

Between versions 2.1 and 2.2 a small growth can be observed. It is clearly increased in the period between Debian 2.2 and Debian 3.0: in two years the number of developers doubles. The last row corresponds to the number of developers in the Debian database at the moment of this study (May 2005). We can see how the Debian project continues its growth at a good rate, although not as firmly as in the period between Debian 2.2 and Debian 3.1.

We have included a picture in Figure 1 where we can see the incoming figures per week. As stated before, we only count on data from June 21st 1999 to June 2003. The first and most surprising fact is to observe a period from June 1999 (or perhaps before, since we do not have previous data) to March 2000 when no new developers were admitted. This situation can be explained by a change in the policy for accepting new developers. It seems that there were some members who entered Debian without knowing, understanding or agreeing with the philosophical lines of Debian [DebianSocialContract], and some discussions became unbearable. Then, the members of the project decided that a mechanism to avoid such cases in the future had to be set up. Until that mechanism was established, no more developers were admitted.

Once the admission process reopened, the number of Debian developers grew without stop and at a good rate during the rest of the year 2000 and 2001, as we can see from the figure. We can find the peak of incorporations with 26 incorporations in one week in January 2001. From mid-2002 the incorporations seem to slow down. On the other hand, it seems that new developers entered in groups from mid-2002 onwards, and not continuously. This is possibly caused by the fact that the database is updated periodically and not every time a new developer enters the project.

Figure 1 Number of Debian developers when releasing stable versions. The figure on the left hand side shows the number of developers in the dates where a new stable version was released. Data from [Lameter2002] has been included for releases before June 1999. The figure on the right shows the number of new developers that enter the project in time since June 1999.



(a) Number of Debian developers when releasing stable versions

(b) New developers that enter the Debian project

Table 2 shows the top 11 countries according to the number of residential Debian developers for the last 5 years. 36 other countries have at least one developer. We can observe a trend toward the decentralization of the project. Considering that the first steps of Debian took place in America (particularly in the United States and Canada), we can see that during the last four years the project has grown significantly in the European region. The United States -the country that contributes most in number of developers- has an average growth rate that is inferior to the mean, while most countries have at least doubled the number of developers in the last 5 years, being France the most significant case with a multiplicative factor of five.

However, the decentralization seems to be limited to some regions, since the incorporation of developers residing in South America, Africa and Asian countries (with the exception of Japan and Korea, which are well represented) is testimonial. In June 2003, we can see in the Debian developer database two developers living in Egypt, China and India and only one in Mexico, Turkey and Colombia.

Table 3 shows a more precise distribution of the Debian developers: the cities with larger number of Debian developers. This data has been obtained by means of the geographical position that developers voluntarily entered for a survey, in December 2003.

Table 2 Top countries in number of Debian developers.

Country	1999.07.01	2000.07.01	2001.07.01	2002.07.01	2003.06.20	2005.05.28
United States	162	169	256	278	297	319
Germany	54	58	101	121	136	158
United Kingdom	34	34	55	63	75	84
Australia	23	26	41	49	52	55
France	11	11	24	44	51	64
Canada	20	22	41	47	49	55
Spain	10	11	25	31	34	38
Japan	15	15	27	33	33	36
Italy	9	9	22	26	31	38
The Netherlands	14	14	27	29	29	34
Sweden	13	13	20	24	27	29

Table 3 Top cities in number of Debian developers.

Rank	City	Country	Developers
1.	Paris	France	15
1.	Cambridge	United Kingdom	15
3.	Madrid	Spain	13
4.	Stuttgart	Germany	10
5.	Berlin	Germany	9
5.	Munich	Germany	9
5.	Tokyo	Japan	9
5.	Seattle	United States	9
9.	Vienna	Austria	8
9.	Sidney	Australia	8
9.	Montreal	Canada	8
9.	Budapest	Hungary	8

4 Source code study: introduction and methodology

The methodology that we have used for the analysis of the stable versions of Debian is simple. To begin with, all the packages of the studied versions were downloaded. For each package the number of source lines of code was counted, and the programming language(s) in which the code is written were recognized.

The counting has been performed using SLOCCount [SLOCCount]. SLOCCount takes as input a directory where the sources are stored, identifies (by means of a series of heuristics) the files which contain source code, recognizes for each of them (also by means of heuristics) the programming language, and finally counts the number of source lines of code they contain. SLOCCount counts SLOCs (physical lines of source code), defined as “a line that finishes in a mark of new line or a mark of end of file, and that contains at least a character that is not a blank space nor commentary”. Therefore, SLOCS are parsed differently for different languages, which forces to the stage on language identification. SLOCs is one of the common metrics to compare software systems. There are some well known effort estimation and optimal timing methods (such as COCOMO) that use SLOCs as input (as will be shown later, when those methods are presented in this paper). The acronym SLOC and its derivatives, KSLOC (1,000 SLOC) and MSLOC (1,000,000 SLOC), are common, and we use them in this paper.

SLOCCount also identifies identical files (by using MD5 hashes), and includes heuristics to detect (and avoid counting) automatically generated code. These mechanisms are helpful when analyzing the code, but have some deficiencies. For instance, finding almost identical files with slight modifications (for instance, the automatic identifier included for the CVS) using such hashes is definitely not effective. In the second case, heuristics only take care of well-known and/or common cases, but do not detect all of them, or others that may appear in future.

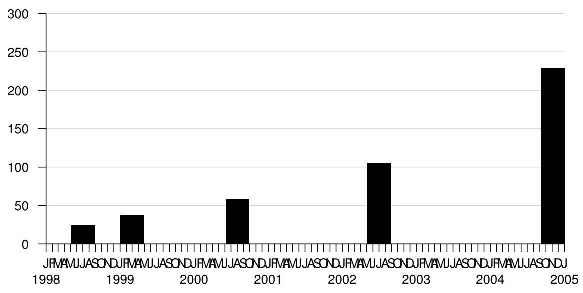
The results of the SLOCCount analysis are transformed by a collection of scripts into an XML file which is later used for visualization, manipulation and transformations into other formats. Among the most interesting transformations, we can insert the data into a relational database, by using SQL commands. Then, with a simple web interface anyone can have access to raw data and other more elaborated visualization forms that facilitate a first analysis (graphs, maps, etc.) ([DebianCounting]).

A more detailed description of the methodology used, as well as a discussion on the main constraints, can be found in [GBarahona2001] and [Amor2004].

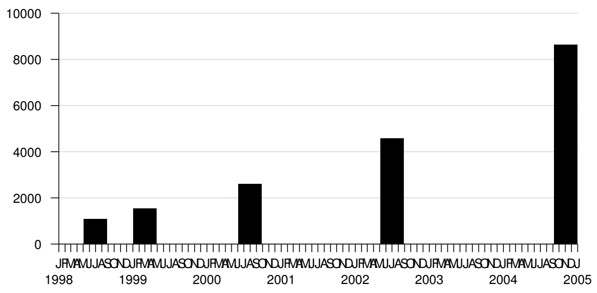
5 The size of Debian

The number of MSLOC (millions of physical lines of source code) and of source packages for the analyzed versions of Debian is shown in figure Figure 2. Debian 2.0 included 1,096 source packages, with a total of more than 25 MSLOC. The next stable version of Debian, 2.1 (published around nine months later) had more than 37 MSLOC and 1,551 source packages. Debian 2.2 (released 15 months after Debian 2.1) summed up around 59 MSLOC in 2,611 packages. The last stable version at the moment of writing this paper, Debian 3.0 (published two years after Debian 2.2), grouped 4,579 packages of source code with almost 105 MSLOC. Almost three years later Debian 3.1 is about to be released, with 8,633 packages and about 230 MSLOC.

Figure 2 Size, in MSLOC, and number of packages for the versions in study. In both graphics of this figure, the studied versions are spaced in time along the X axis according to their release date. On the left we can see the number of MSLOC that includes each version, while the right graph shows the evolution of the number of packages.



(a) MSLOC for each version



(b) Number of packages for each version

Table 4 Size of the Debian distributions under study

Version	Release date	Source packages	Size (MSLOC)	Mean package size (SLOC)
Debian 2.0	July 1998	1,096	25	23,050
Debian 2.1	March 1999	1,551	37	23,910
Debian 2.2	August 2000	2,611	59	22,650
Debian 3.0	July 2002	4,579	105	22,860
Debian 3.1	June 2005 (estimated)	8,633	229	26,584

These figures show how Debian total size (both in MSLOC and in number of packages) is doubling every two-three years. This is an interesting finding, which implies that, for instance between releases 3.0 and 3.1 the Debian project has included as much code as in all its previous history. In addition, having into account that Debian includes an important share of the mature and usable libre software ported to GNU/Linux, we can extrapolate that this kind of software is growing at the same rate.

6 Packages

Graphs representing the distribution of package sizes in the different versions of Debian are shown in Figure 3 and Figure 4. It is possible to observe that there is a small number of large packages (with more than 100,000 lines of code) and that the size of these packages tends to increase with time, as one of Lehman's law of software evolution states [Lehman1997]. Nevertheless, it seems surprising that despite the growth that has undergone Debian in time, the graph does not show large variations. But certainly, what is still more interesting is the fact that the mean size for the packages included in Debian is surprisingly regular (around 23,000 SLOC for Debian 2.0, 2.1, 2.2, 3.0 and 3.1). With the data currently available it is difficult to give a forceful explanation of this fact, but we can suggest some thoughts: perhaps the "ecosystem" in Debian is so rich that while many packages grow in size, smaller ones are included causing that the average stays approximately constant over time.

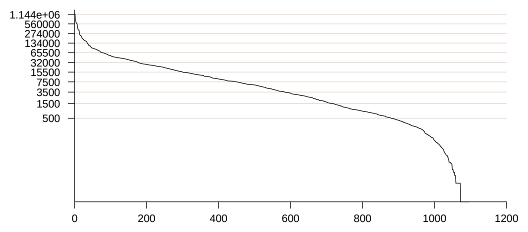
It is also interesting to follow the evolution of the largest packages included in each of the stable versions of Debian. Many of these packages correspond to significant applications, well-known and popular, which have been documented in detail in several scientific papers. From the study of these packages we can infer some information about the nature of the Debian distributions.

There is a lot of movement in the list of the top ten packages in size, from release to release. Almost no package remains in the top-ten of Debian 3.1 from the top-ten of Debian 2.0, about sever years earlier. Some of the "new ones" have been included in later versions (as it is the case for the Mozilla navigator), whereas in the case of others they are compositions from some smaller packages (this is the case for mingw32, a C/C++ cross compiler for Win32).

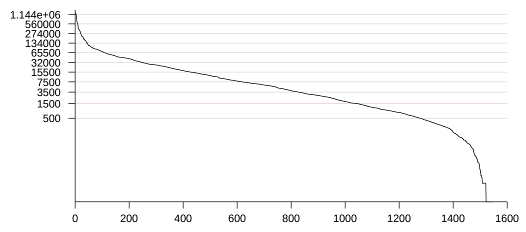
On the other hand, there is a clear trend for the inferior limit of the top ten packages to increase in size with time: whereas in Debian 2.0 we can see how GCC with 460,000 SLOC was located in the tenth position, the tenth largest package for Debian 3.1, kfreebsd5-source (the kernel source code of FreeBSD for building a Debian GNU/kFreeBSD distribution) consisted on more than 1,600,000 lines of code.

But top packages in size do not only tend to have more source code, they also show a trend to have larger source code files. While the average in SLOC per file is in a rank between 352 and 359 for packages among the top ten, the average for all the packages in the same versions lies between 228 and 243 of source code lines per file. It exists, nevertheless, a big variance among the top packages.

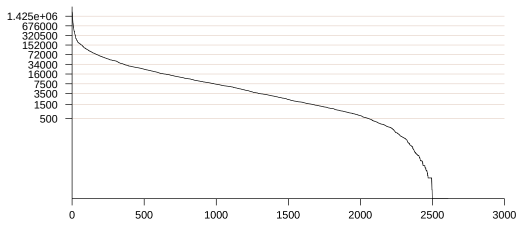
Figure 3 Package sizes for Debian distributions. Packages are ordered by their size along the X axis, while the counts in SLOCs are represented along the Y axis (in logarithmic scale).



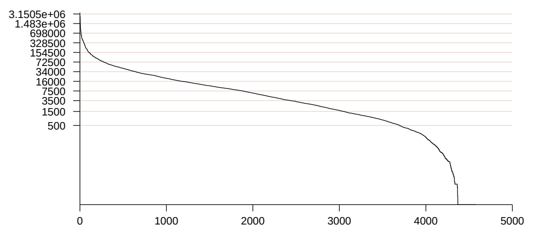
(a) Debian 2.0



(b) Debian 2.1

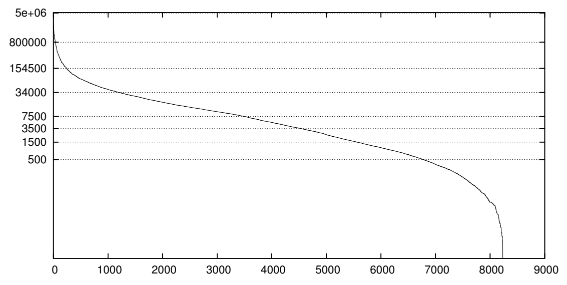


(c) Debian 2.2



(d) Debian 3.0

Figure 4 Package sizes for Debian 3.1. Packages are ordered by their size along the X axis, while the counts in SLOCs are represented along the Y axis (in logarithmic scale).



We have a range going from the 138 SLOC per file in version 1.1.2 of egcs (a derivative of the GNU GCC compiler) to the 806 SLOC per file in bigloo (a system for Scheme compilation) in its version 2.4b.

Regarding the application domain, there are no significant differences in the top packages. We can find at the top of this classification system tools (compilers, debuggers...), specific-purpose libraries and end-user application as a web navigator (Mozilla) or office suite (OpenOffice.org). The kernel of the operating system, Linux, packed as kernel-source, is present in the top-ten of all releases. From the evolution of the top packages in size for the latest versions, we can infer a certain shift from low-level programmer-oriented software (kernels, compilers, debuggers...) to end-user targeted applications (web browsers and office suites), which may mirror the expansion of GNU/Linux systems from administrators and computer scientists to “regular” end users.

Table 5 Top 10 packages in size for Debian 2.0.

Rank	Package name	Version	SLOC	files	SLOC/file
1.	xfree86	3.3.2.3	1,189,621	4,100	290.15
2.	xemacs20	20.4	777,350	1,794	433.31
3.	egcs	1.0.3a	705,802	4,437	159.07
4.	gnat	3.10p	599,311	1,939	309.08
5.	kernel-source	2.0.34	572,855	1,827	313.55
6.	gdb	4.17	569,865	1,845	308.87
7.	emacs20	20.2	557,285	1,061	525.25
8.	lapack	2.0.1	395,011	2,387	165.48
9.	binutils	2.9.1	392,538	1,105	355.24
10.	gcc	2.7.2.3	351,580	753	466.91

Table 6 Top 10 packages in size for Debian 2.1.

Rank	Package name	Version	SLOC	files	SLOC/file
1.	mozilla	M18	1,269,186	4,981	254.81
2.	xfree86	3.3.2.3a	1,196,989	4,153	288.22
3.	kernel-source	2.2.1	1,137,796	3,927	289.74
4.	prc-tools	0.5.0r	103,5230	3,025	342.22
5.	egcs	1.1.2	846,610	6,106	138.65
6.	xemacs20	20.4	777,976	1,796	433.17
7.	emacs20	20.5a	630,052	1,116	564.56
8.	gnat	3.10p	599,311	1,939	309.08
9.	gdb	4.17	582,834	1,862	313.02
10.	ncbi-tools6	6.0	554,949	951	583.54

6.1 Package versions

It has already been shown how, throughout the latest stable releases, Debian has changed a lot, growing both in number of packages and in number of SLOC. Now we will focus on the opposite direction: matters that have not changed. We have already shown how there are packages in the list of the top packages in size that have been added in more recent stable versions of Debian. Other packages may have "fallen" and do not appear anymore, while still some others remain invariant in time. In this section we will discuss these later two groups.

Although it can seem surprising, out of the 1096 packages included in Debian 2.0 only about 800 appear in the latest version of Debian considered in this study. This means that about 25% of the packages have disappeared from Debian in six years. The number of packages of the 3.0 version that are still included in 3.1 is 3,848 out of 4,578 which gives us a similar percentage of "disappeared" packages.

Table 10, Table 11, Table 12, Table 13 and Table 14 show the packages that the different stable versions have in common. We assume that two versions have a package in common if it is included in both, independently of the version number of the package. Each table displays in its second column the number of packages in common that a version of Debian has with the other versions. To facilitate the comparison in relative and absolute terms, the same version of Debian that is compared is itself

Table 7 Top 10 packages in size for Debian 2.2.

Rank	Package name	Version	SLOC	files	SLOC/file
1.	mozilla	M18	1,940,167	9,315	208.28
2.	kernel-source	2.2.19.1	1,731,335	5,082	340.68
3.	pm3	1.1.13	1,649,480	10,260	160.77
4.	xfree86	3.3.6	1,256,423	4,351	288.77
5.	prc-tools	0.5.0r	1,035,125	3,023	342.42
6.	oskit	0.97.20000202	851,659	5,043	168.88
7.	gdb	4.18.19990928	797,735	2,428	328.56
8.	gnat	3.12p	678,700	2,036	333.35
9.	emacs20	20.7	63,0424	1,115	565.4
10.	ncbi-tools6	6.0.2	591,987	988	599.18

Table 8 Top 10 packages in size for Debian 3.0.

Rank	Package name	Version	SLOC	files	SLOC/file
1.	kernel-source	2.4.18	2,574,266	8,527	301.9
2.	mozilla	1.0.0	2,362,285	11,095	212.91
3.	xfree86	4.1.0	1,927,810	6,493	296.91
4.	pm3	1.1.15	1,501,446	7,382	203.39
5.	mingw32	2.95.3.7	1,291,194	6,840	188.77
6.	bigloo	2.4b	1,064,509	1,320	806.45
7.	gdb	5.2.cvs20020401	986,101	2,767	356.38
8.	crash	3.3	969,036	2,740	353.66
9.	oskit	0.97.20020317	921,194	5,584	164.97
10.	ncbi-tools6	6.1.20011220a	830,659	1,178	705.14

included. As it is logical, Debian 2.0 will have in common with itself 1,096 (all) source packages.

On the other hand, we have to consider that distributions contain applications and libraries that evolve over time. This can be inferred from the fact that the own version number of packages evolve. For example, the Linux sources comes generally in a package called kernel-source, as we have seen before. In each version of Debian, the version number of kernel-source changes, so we can state that Linux has evolved and that these changes and improvements have been introduced in Debian. Thus this does not have to be the case for all packages. In the same way that previously we were interested in packages in common without mattering if the package version numbers changed, now we are going to consider those whose package version number does not vary among distributions. So, now we assume as a common package that with the same package version, being included in two different Debian versions. Again, we add the own Debian version being compared, and because of that Debian 2.0 will have all of its packages (1,096) in common with itself.

The fact that Debian 3.1 includes 158 packages that have not evolved since Debian 2.0 is very surprising, as it comes to say that 15% of the source packages included in Debian 2.0 have stayed almost inalterable since they were released six years ago. As it is logical, on the other hand, the number of packages with versions in common increases when the distributions are nearer in time.

Table 9 Top 10 packages in size for Debian 3.1.

Rank	Package name	Version	SLOC	files	SLOC/file
1.	openoffice.org	1.1.3	5,181,285	19,011	272.54
2.	kernel-source-2.6.8	2.6.8	4,043,242	13,974	289.34
3.	nvu	0.80	2,476,391	10,780	229.72
4.	mozilla	2:1.7.3	2,437,724	10,713	227.55
5.	gcc-3.4	3.4.3	2,422,056	22,992	105,34
6.	xfs-xtt	1:1.4.1	2,326,623	7,081	328.57
7.	xfree86	4.3.0	2,316,842	7,216	321,07
8.	vnc4	4.0	2,055,178	6,790	302.68
9.	insight	6.1+cvs.2004.08.11	1,690,058	4,063	415.96
10.	kfreebsd5-source	5.3	1,630,452	4,968	328.19

Table 10 Packages and versions in common for Debian 2.0

Debian Version	Common packages	Common versions	SLOC of common versions	Files of common versions	SLOC of common packages
Debian 2.0	1,096	1,096	25,267,766	110,587	25,267,766
Debian 2.1	1,066	666	11,518,285	11,5126	26,515,690
Debian 2.2	973	367	3,538,329	86,810	19,388,048
Debian 3.0	754	221	1,863,799	70,326	15,888,347
Debian 3.1	813	158	1,271,377	15,296	15,594,976

7 Programming languages

As we have already commented in the section devoted to methodology, before counting the number of SLOC, the programming language in which a file is written is identified. Thanks to this, we are able to know the significance and the use of the different programming languages in Debian. The most used language in all Debian versions is C, with shares ranging from 57% to 85%, and with a large advantage over its immediate follower, C++. It can be observed, nevertheless, that the importance of C is diminishing gradually, whereas some other programming languages are growing at a good rate.

For example, in Table 15 the evolution of the most significant languages (those that surpass 1%

Table 11 Packages and versions in common for Debian 2.1

Debian Version	Common packages	Common versions	SLOC of common versions	Files of common versions	SLOC of common packages
Debian 2.0	1,066	666	11,518,285	115,126	26,515,690
Debian 2.1	1,551	1,551	37,086,828	161,303	37,086,828
Debian 2.2	1,384	602	8,460,239	133,140	30,052,890
Debian 3.0	1,076	322	3,152,790	108,071	24,743,063
Debian 3.1	1,124	231	2,306,969	27,543	23,630,211

Table 12 Packages and versions in common for Debian 2.2

Debian Version	Common packages	Common versions	SLOC of common versions	Files of common versions	SLOC of common packages
Debian 2.0	973	367	3,538,329	86,810	19,388,048
Debian 2.1	1,384	602	8,460,239	133,140	30,052,890
Debian 2.2	2,610	2,610	59,138,348	257,724	59,138,348
Debian 3.0	1,921	771	8,356,302	186,508	42,938,562
Debian 3.1	1,946	508	4,992,308	60,525	36,584,110

Table 13 Packages and versions in common for Debian 3.0

Debian Version	Common packages	Common versions	SLOC of common versions	Files of common versions	SLOC of common packages
Debian 2.0	754	221	1,863,799	70,326	15,888,347
Debian 2.1	1,076	322	3,152,790	108,071	24,743,063
Debian 2.2	1,921	771	8,356,302	186,508	42,938,562
Debian 3.0	4,578	4,578	104,305,557	403,285	104,702,397
Debian 3.1	3,848	1,567	16,042,810	211,299	78,451,818

of code in Debian 3.1) is shown. Below the 1% border we can find a long list starting with PHP, Tcl, Pascal, Ada, ObjC, ML, Yacc and some other languages.

There exist some programming languages that we could consider as minor languages but which are found ate a strange high position in the classification given above. This is because although being present in a reduced number of packages, these are quite large in size. That is the case of Ada, that sums near to 500,000 SLOC in three packages: gnat (an Ada compiler), libgtkada2 (a binding to the GTK library) and Asis (a system to manage sources in Ada) of a total of 1,401,000 SLOC that have been identified as code written in that language in Debian 3.1. Another similar case is Lisp, that counts, only for GNU Emacs and XEmacs, with more than 1,200,000 SLOC of around 7 MSLOC in the whole distribution.

The pie-charts showing programming language shares expose a clear trend to the decline of C. Something similar seems to happen to Lisp, that was the third most used language in Debian 2.0, but

Table 14 Packages and versions in common for Debian 3.1

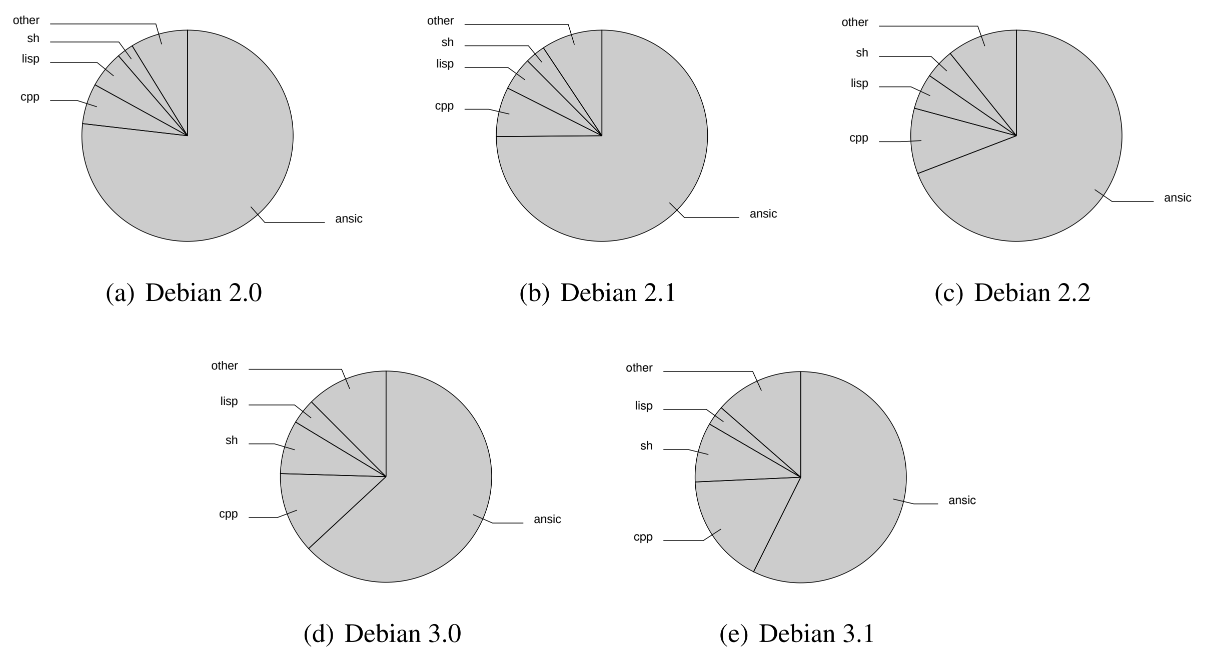
Debian Version	Common packages	Common versions	SLOC of common versions	Files of common versions	SLOC of common packages
Debian 2.0	813	158	1,271,377	15,296	15,594,976
Debian 2.1	1,124	231	2,306,969	27,543	23,630,211
Debian 2.2	1,946	508	4,992,308	60,525	36,584,110
Debian 3.0	3,848	1,567	16,042,810	211,299	78,451,818
Debian 3.1	8,633	8,633	229,495,824	972,121	229,495,824

Table 15 Top programming languages in Debian.

Language	KSLOC Debian 2.0	Percent. Debian 2.0	KSLOC Debian 2.1	Percent. Debian 2.1	KSLOC Debian 2.2	Percent. Debian 2.2	KSLOC Debian 3.0	Percent. Debian 3.0	KSLOC Debian 3.1	Percent. Debian 3.1
C	19,371	76.67%	27,773	74.89%	40,878	69.12%	66,550	63.08%	130,847	57,01%
C++	1,557	6.16%	2,809	7.57%	5,978	10.11%	13,067	12.39%	38,601	16,82%
Shell	645	2.55%	1,151	3.10%	2,712	4.59%	8,636	8.19%	20,792	9.06%
Lisp	1,425	5.64%	1,892	5.10%	3,197	5.41%	4,087	3.87%	6,919	3.02%
Perl	425	1.68%	774	2.09%	1,395	2.36%	3,199	3.03%	6,416	2.80%
Python	122	0.48%	211	0.57%	349	0.59%	1,459	1.38%	4,129	1.80%
Java	22	0.09%	58	0.16%	183	0.31%	531	0.51%	3,679	1.60%
Fortran	494	1.96%	735	1.98%	1,182	1.99%	1,939	1.84%	2,724	1.19%

has become the fifth in Debian 3.1, and that foreseeably will continue backing down in the future. On the other hand, the part of the pie corresponding to C++, shell and other programming languages grows.

Figure 5 Pie with the distribution of source lines of code for the predominant languages in Debian



(a) Debian 2.0 (b) Debian 2.1 (c) Debian 2.2

(d) Debian 3.0 (e) Debian 3.1

Figure 7 provides the relative evolution of programming languages, giving a new perspective of the growth in the last four stable Debian releases. We take as reference the Debian 2.0 release and assume that the presence of each language in it is 100%, so that growth for a programming language is shown relative to itself.

Previous pies showed that C is backing down as far as its relative importance is concerned. Here we can observe that C has grown more than 300% throughout the four versions. But we can see that scripting languages (shell, Python and Perl) have undergone an extraordinary growth, all of them multiplying their presence by factors superior to seven, accompanied by C++. Languages that grow a smaller quantity are the traditional, compiled ones (Fortran and Ada). This can give an idea of the importance that interpreted languages have begun to have in the libre software world.

Figure 7 includes the most representative languages in Debian, but excludes Java and PHP, since the growth of these two is enormous, partly because their presence in Debian 2.0 was testimonial, partly because their popularity in the latest time is beyond doubt.

Figure 6 Evolution of the four most used languages in Debian

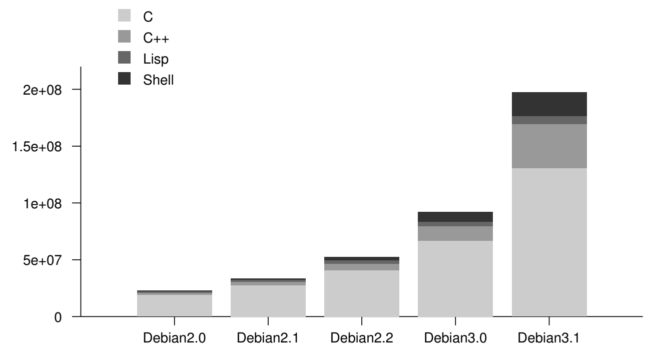
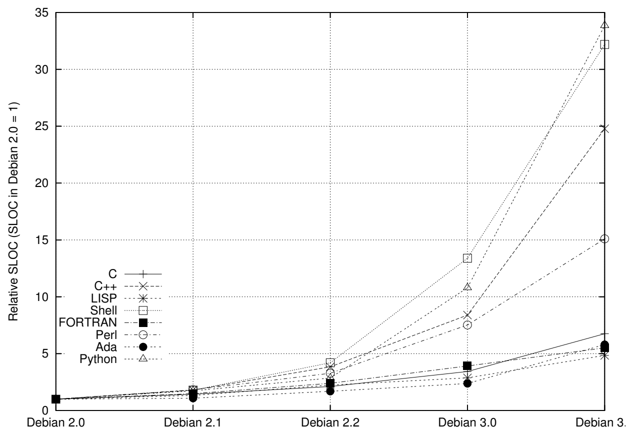


Figure 7 Relative growth of some programming languages in Debian



7.1 File sizes

Some of the most important programming languages have spectacular increases in their use, but, interestingly, the mean file sizes, for files in each of those languages, remain basically constant (see Table 16). Thus, for C the average length lies around 260 to 280 SLOC per file, whereas in C++ this value is located in an interval going from 140 to 185. We can find the exception to this rule in the shell language, that triples its mean size. This is because the shell language is very singular: almost all the packages include something in shell for their installation, configuration or as "glue". It is probable that this type of scripts get more complex in time.

It is also quote peculiar to learn that structured languages usually have larger average file lengths than object-oriented languages. Thus the files in C (or Yacc) usually have higher sizes, in average, than those in C++. This makes us think that the modularity of programming languages is also reflected in the mean file size.

Table 16 Mean file size for some programming languages. Average computed for all programming languages (not only those shown)

Language	Debian 2.0	Debian 2.1	Debian 2.2	Debian 3.0	Debian 3.1
C	262.88	268.42	268.64	283.33	283.43
C++	142.5	158.62	169.22	184.22	186.09
Lisp	394.82	393.99	394.19	383.60	350.76
shell	98.65	116.06	163.66	288.75	340.46
Yacc	789.43	743.79	762.24	619.30	607.17
Average	228.49	229.92	229.46	243.35	236.08

8 Effort and time estimation

The COCOMO model [Boehm1981] provides an estimation of the human and monetary effort needed to generate software of a given size. It takes as input metric the number of source lines of code. COCOMO is a model thought for the "classical" software generation processes (V or waterfall development model) and for large projects, so that the results that it offers when applied to Debian packages should be considered with caution. In any case, the results may give us an idea of the order of magnitude of the costs that creating Debian would represent, showing the necessary optimal efforts, if a proprietary development model had been used. In other words, COCOMO can be a good estimator for the substition cost of Debian.

All in all, the most astonishing result that COCOMO offers when applied to Debian is the cost estimation. Some words should be said in order to clarify the concept. In this estimation two special factors are considered: the average developer salary and the factor of "overhead". In the calculation of the cost estimation, the average wage for a full-time system programmer has been taken from the year 2000 salary survey [ComWorld2000]. "Overhead" is the overhead cost that any company has to assume independently from the programmers' salaries so that the product hits the streets. Secretaries, marketing team and so on have to be added to the costs of photocopies, electricity, equipment, hardware, etc. and that all is computed in the "overhead" factor (which for our calculations has been supposed to be 2,4). In summary, the final cost calculated by COCOMO is the total cost that a company would have to confront to create a software of the specified size and not simply the money that programmers would perceive to make the software. Once this is understood, cost calculations seem less bulky.

In Table 17 we can observe the results of applying the basic COCOMO model to the different Debian stable versions. The results have been obtained by means of the separate calculation of the cost that each package would suppose and has been summed up to give the global cost. It should be noted that as COCOMO output is a non-linear, the sum of the separate cost for packages is not equal to the cost of considering everything as one single project. The first result would give us the inferior effort limit, since the integration tasks are not considered. On the contrary, the second case offers an upper limit, since savings from having independent projects are not considered. In [DebianCounting] both figures can be obtained for their comparison. As stated before, for our current goals it is enough with an estimation of the order of magnitude and therefore only the former cost appears.

Table 17 Effort, time and development cost estimation for each Debian version.

Version	MSLOC	Effort (man-years)	Time (years)	Cost (Mill USD)
Debian 2.0	25	6,360	4.93	860
Debian 2.1	37	9,425	4.99	1,275
Debian 2.2	59	14,950	6.04	2,020
Debian 3.0	105	26,835	6.81	3,625
Debian 3.1	229	59,537	8,82	8,042

9 Comparison with other GNU/Linux distributions

There exist a similar study (in which the preliminary research that has lead to this paper, several years later, was inspired) with Red Hat as the studied distribution [Wheeler2000], [GBarahona2004]. Red Hat can be considered as the canonical distribution among the commercial ones. It has a very different strategy and philosophy than the Debian project has. In any case, Red Hat perfectly serves for our purposes of comparison. We should not forget that the Red Hat Package Manager (RPM) is used by a large majority of distributions, followed (at a long distance) by the one used in Debian (known as deb) [DistroWatch]. We are, therefore, probably comparing the two most significant GNU/Linux distributions of the libre software world. Some of the presented Red Hat information has been extracted from [Wheeler2000] and [Wheeler2001]. The rest has been added by the authors in order to obtain a more complete picture.

In 2003, Red Hat decided to transfer their freely available distribution to a community supported project, called Fedora. So, we have considered Fedora as a continuation of Red Hat Linux distribution; however we think it needs to be studied separately because we can imagine that growth of community supported distributions must be faster than traditional enterprise distributions as Red Hat did. However, in this paper Fedora and Red Hat distributions will be considered as same yet. In the future, we will study the evolution of Fedora distributions and will be compared with RedHat and other.

We can find the main organizational difference with Debian in the fact that there is a company behind Red Hat. This means that this company will have a determined number of dedicated employees that integrate all the software in an homogeneous way in order to facilitate its installation, configuration and update. In other words, while in Debian the packages included in the distribution depend on the presence of voluntary collaborators who manage to pack them, in Red Hat certain economic calculations have to be made to see the effort that supposes a new distribution and if that is affordable for the company's staff.

These divergences in conception result in a series of differences between Red Hat and Debian that we can analyze and compare. One of the main differences is the fact that the number of packages in Red Hat is much more small than contemporary Debian versions. Thus, Debian 2.2 doubles Red Hat 7.1 in size, although its release happened some months later.

On the other hand, Red Hat distributions usually include most recent versions of software, whereas in Debian, freezing intervals before the release have as an effect that stable versions do not include

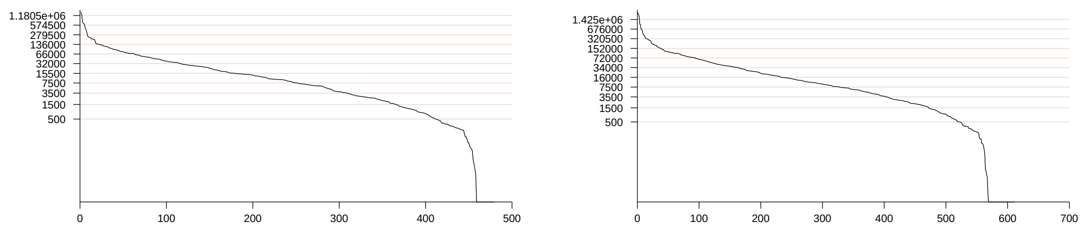
Table 18 Comparison with other GNU/Linux distributions.

Name	Release date	MSLOC	Effort (man-years)	Time (years)	Cost (Mill USD)
Red Hat 5.2	April 1998	12	3,216	4.93	434
Red Hat 6.0	April 1999	15	3,951	5.08	534
Red Hat 6.2	March 2000	17	4,550	5,45	615
Debian 2.0	July 1998	25	6,360	4.93	860
Red Hat 7.1	April 2001	30	7,950	6.53	1,075
Debian 2.1	March 1999	37	9,425	4.99	1,275
Red Hat 8.0	September 2002	50	13,315	7.35	1,800
Debian 2.2	August 2000	59	14,950	6.04	2,020
Debian 3.0	July 2002	105	26,835	6.81	3,625
Redhat 9.0	March 2003	53	14,092	7.43	1,904
Fedora Core 2	May 2004	67	17,961	7.97	2,426
Fedora Core 4 (pre.)	May 2005	76	20,782	10.18	2,807
Debian 3.1	June 2005 (est.)	229	59,537	8.82	8,043

the latest of the latest. This can be easily demonstrated by studying the package versions included in Red Hat and Debian. For example, as we can see in [GBarahona2004], many packages in Red Hat 6.2 and Debian 2.2 have identical versions even though Debian 2.2 was published five months later. In some cases, Debian 2.2 even includes older versions than those that could be obtained in Red Hat 6.2.

Another remarkable detail is that Red Hat seems to have lower interest in small packages, as it can be seen from figure Figure 8. As a result of this, the number of SLOC per package grows in time, whereas (as we already showed) in Debian this value kept approximately constant.

Figure 8 Package sizes for Red Hat distributions. Packages are ordered by size along the X axis. the number of SLOCs for each package is represented in logarithmic scale in the vertical axis.



(a) Red Hat 6.2 (b) Red Hat 7.1

For a more detailed comparison of the Debian and Red Hat versions we refer the reader to [GBarahona2004].

10 Comparison with other operating systems and applications

If comparisons are always difficult, those of libre software with proprietary one are more. All this study on Debian has been possible because of its condition as libre software. The access to the code (and many other information that has been exposed in this article) is essential to study thoroughly the different versions in number of lines, packages, programming languages... But the advantages of libre software (and, therefore, of libre software engineering, see [GBarahona2003b]) go further, because in addition they offer the possibility of being reviewed by third parties that could be research groups or just people interested in the topic.

In proprietary systems, in general, making a study as the one performed in this paper is a difficult, if not impossible task. In fact, the numbers that will be offered below have their sources mostly in the own companies that are behind the software development, so we cannot guarantee their veracity. In many cases we do not even know if the figures being offered correspond to physical source lines of code (SLOC) as we have been doing throughout this article or if they also include blank lines and comments. We do not know either exactly what software they consider in their studies. So, for some versions of Microsoft Windows we are not able to know if the Microsoft Office suite is included or not.

In any case, we believe that including this comparative is relevant, since it helps to locate the different Debian versions within a broader panorama. What seems to be beyond any doubt is that Debian and Red Hat (specially Debian) are among the largest coordinated software systems ever built up.

The numbers cited in Table 19 proceed from [Lucovsky2000] for Windows 2000, [McGraw2003] for the rest of sizes of Windows systems, [SunPressRelease] for StarOffice 5.2 and [Schneier2000] for the rest of systems.

Table 19 Comparison with proprietary systems

System	Release Date	"Lines of Code"
Microsoft Windows 3.1	April 1992	3,000,000
Microsoft NT	1995	4,000,000
SUN Solaris 7	October 1998	7,500,000
SUN StarOffice 5.2	June 2000	7,600,000
Microsoft Windows 95	August 1995	15,000,000
Windows NT 4.0	July 1996	16,000,000
Microsoft Windows 98	1999	18,000,000
Microsoft NT 5.0	1995	4,000,000
Debian 2.0	July 1998	25,000,000
Microsoft Windows 2000	February 2000	29,000,000
Debian 2.1	March 1999	37,000,000
Microsoft XP	2002	40,000,000
Debian 2.2	August 2000	55,000,000
Debian 3.0	July 2002	105,000,000
Debian 3.1	June 2005 (est.)	229,000,000

11 Conclusions

In this paper we have shown the results of studying in depth the stable versions of Debian, from 2.0 on. We have shown the evolution of physical source lines of code, the number and size of the packages, and the share of programming languages. All this source code has been put together, and is supported, by the number of voluntary developers with which Debian counts to create its distributions. The different stable versions have been compared with the others as well as with other GNU/Linux distributions, in our case Red Hat, and with large proprietary systems.

Among the most important finding that have emerged, we encounter that the stable versions seem approximately to double in number of source lines of code and packages every two years, although, some slow down is observed for the latest period studied, from 3.0 to 3.1. We feel that this evolution can only be maintained if more voluntary developers enter the Debian project in the near future. That is, at least, what can be concluded from the fact that, until now, mean package size has remained approximately constant, so that the number of packages grows linearly with the number of lines of code. If the number of developers in Debian does not grow in those proportions, the number of packages that a developer will have to maintain will be too high.

The size of the upcoming version of Debian (3.1) makes us think that we are in front of one of the largest software collections in the history of humanity, if not the largest. In order to create their 229 MSLOC, according to the COCOMO model, 60,000 person-years would be necessary and the cost would go up to around the 8,000 million dollars. None of the other systems with which we have compared Debian (Red Hat, Solaris, Windows, etc.) can compete currently in size with Debian.

The top applications in size contained in Debian consist predominantly on low level applications (kernel, development software, specific-purpose libraries...), although in recent releases we have the inclusion of end-user applications, such as Mozilla and OpenOffice.org. So, there is a tendency to grow importance of desktop users versus more technical users.

As far as the programming languages are concerned, C is the most used language, although it gradually is losing weight. The scripting languages, C++ and Java are those that seem to have a higher growth in the following versions, whereas the traditional compiled languages have even inferior rates of growth than C.

To conclude, it is important to remark that we have been offering a relatively small amount of measurements and some estimations. Nevertheless, we consider that they are enough to draw some conclusions, to compare Debian with other systems and to make some predictions about Debian's future.

12 Acknowledgments

We thank Martin Michlmayr for his invaluable comments and suggestions on some of the data used for this report, and David A. Wheeler for his marvelous SLOCCount tool. Of course, we also thank the Debian project for offering us not only the opportunity of writing this paper, but mainly for devoting their time to producing such a fine distribution.

References

[Amor2004] *Measuring Woody: The size of Debian 3.0*, Juan José Amor, Gregorio Robles, and Jesús M. González-Barahona, Pending publication. Will be available at <http://people.debian.org/~jgb/debian-counting/>. 4

[Boehm1981] *Software Engineering Economics*, Barry W. Boehm, 1981, Prentice Hall. 8

[ComWorld2000] *Salary Survey 2000*, Computer World, <http://www.computerworld.com/cwi/careers/surveysandreports> . 8

[DBDebian] *Debian Developers Database*, Debian Project, <http://db.debian.org> . 3, 3

[DFSG] *Debian Free Software Guidelines (part of the Debian Social Contract)*, Debian Project, <http://www.debian.org/social_contract> . 2

[Debian22Ann] *Debian GNU/Linux 2.2, the "Joel 'Espy' Klecker" release, is officially released*, Debian Project, <http://www.debian.org/News/2000/20000815> .

[Debian22Rel] *Debian GNU/Linux 2.2 release information*, Debian Project, <http://www.debian.org/releases/2.2/> .

[DebianCounting] *Debian Counting*, Jesús M. González Barahona and Gregorio Robles, <http://libresoft.urjc.es/debian-counting/> . 1, 4, 8

[DebianHistory] *A Brief History of Debian*, Debian Documentation Team, <http://www.debian.org/doc/manuals/project-history/> . 2

[DebianPol] *Debian Policy Manual*, Debian Project, <http://www.debian.org/doc/debian-policy/> . 2

[DebianSocialContract] *Debian Social Contract*, Debian Project, <http://www.debian.org/social_contract> . 2, 3

[DistroWatch] *Linux Distributions - Facts and Figures*, Ladislav Bodnar, <http://www.distrowatch.com/stats.php?section=packagemanagement> . 9

[GBarahona2001] *Counting potatoes: The size of Debian 2.2*, Jesús M. González-Barahona, Miguel A. Ortuño-Pérez, Pedro de-las-Heras-Quirós, José Centeno-González, and Vicente Matellán-Olivera, <http://upgrade-cepis.org/issues/2001/6/up2-6Gonzalez.pdf> , Also available at <http://people.debian.org/~jgb/debian-counting/> . 4

[GBarahona2003b] *Free Software Engineering: A Field to Explore*, Jesús M. González-Barahona and Gregorio Robles, <http://www.upgrade-cepis.org/issues/2003/4/up4-4Gonzalez.pdf> . 1, 10

[GBarahona2004] *Anatomy of two GNU/Linux distributions*, Jesús M. González-Barahona, Gregorio Robles, Miguel Ortuño-Pérez, Luis Rodero-Merino, José Centeno-González, Vicente Matellán-Olivera, Eva Castro-Barbero, and Pedro de-las-Heras-Quirós, Chapter in book "Free/Open Source Software Development" edited by Stefan Koch and published by Idea Group, Inc. . 9, 9, 9

[GodfreyTu2000] *Evolution in Open Source Software: A Case Study*, Michael W. Godfrey and Qiang Tu, August 3-4, 2000, 2000 International Conference on Software Maintenance <http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf> .

[Lameter2002] *Debian GNU/Linux: The Past, The Present and The Future*, Christoph Lameter, Free Software Symposium 2002, Toky, Japan <http://u-os.org/tokyo/> . 3, 3, 1

[Lehman1997] *Metrics and laws of software evolution - the nineties view*, M.M. Lehman, J.F. Ramil, P.D. Wernick, and D.E. Perry, Proceedings of the Fourth International Software Metrics Symposium . 6

[Libresoft] *Libre Software Engineering*, Libre Software Engineering Lab at the Universidad Rey Juan Carlos (Madrid, Spain), <http://libresoft.urjc.es/> . 1

[Lucovsky2000] *From NT OS/2 to Windows 2000 and Beyond - A Software-Engineering Odyssey*, Mark Lucovsky, 4th USENIX Windows Systems Symposium, <http://www.usenix.org/events/usenix-win2000/invitedtalks/lucovsky_html/> . 10

[McGraw2003] *From the ground up: the DIMACS software security workshop*, Gary McGraw, IEEE Security and Privacy. March/April 2003. Volume 1, Number 2. pp. 59-66 . 10

[Michlmayr2003] *Quality and the Reliance on Individuals in Free Software Projects*, Martin Michlmayr and Benjamin Mako Hill, <http://opensource.ucc.ie/icse2003/3rd-WS-on-OSS-Engineering.pdf> . 1

[Robles2001] *WIDI - Who Is Doing It? A research on Libre Software developers*, Gregorio Robles, Henrik Scheider, Ingo Tretkowski, and Niels Weber, <http://widi.berlios.de/paper/study.pdf> . 3

[Robles2005] *Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian*, Gregorio Robles, Jesus M. Gonzalez-Barahona, and Martin Michlmayr, <http://www.cyrius.com/publications/robles_barahona_michlmayr-evolution_participation.html> . 3

[SLOCCount] *SLOCCount*, David A. Wheeler, <http://www.dwheeler.com/sloccount/> . 4

[Schneier2000] *Software Complexity and Security*, Bruce Schneier, March 15th 2000, Crypto-Gram Newsletter, <http://www.counterpane.com/crypto-gram-0003.html> . 10

[SunPressRelease] *Sun Microsystems Announces Availability of StarOffice(TM) Source Code on OpenOffice.org*, SUN Microsystems, <http://www.collab.net/news/press/2000/openoffice_live.html> . 10

[Wheeler2000] *Estimating Linux's Size*, David A. Wheeler, <http://www.dwheeler.com/sloc>. 9

[Wheeler2001] *More Than a Gigabuck: Estimating GNU/Linux's Size*, David A. Wheeler, <http://www.dwheeler.com/sloc>. 9