From Toy Story to Toy History: a deep analysis of Debian GNU/Linux

Gregorio Robles Jesús M. González-Barahona

September 2003

Abstract

The Debian operating system is one of the more popular GNU/Linux distributions today. From its birth, one decade ago, it has undergone many technical, structural and organizational changes. This article tries to study the evolution of Debian in the last five years, comparing the last the four stable versions of this distribution in size, programming languages and packages. Also the evolution of the number of voluntary developers (maintainers) in the Debian project is analyzed and an estimation of the effort is made in human and economic terms that would be necessary to produce a software of this size. The main evidences that we have found are that Debian approximately doubles to the size in lines of code and number of packages every two years, whereas the mean value of the packages' sizes remain constant. A great majority of the packages of the first version considered in this study "have survived" in time and are present in more modern versions of Debian, many of them even with the same version number. The most widely used programming language is C, although its importance is decreasing with time. At last, the results of the evolution of Debian during these last five years are used to make a prediction of how the next stable version of Debian could be, when it cuold be shipped and what challenges (limiting factors) it will have to surpass.

1 Introduction

At the beginning of the nineties, the first distributions arose from the union of the GNU tools with the Linux kernel. Its purpose was to facilitate the installation of free tools as far as possible, an arduous task that required a great patience, being sometimes even more an artisan work. The second big innovation of distributions -already in the mid-nineties- is due to the package management systems

that not only allowed to install in a simple way a distribution into the users' hard disk, but in addition allowed the management of packages once installed.

Distributions occupied, consequently, a space that in the world of propietary software rare time reaches important proportions: integrators. Its work consists of taking the sources - generally from their original author(s)-, to group them 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 end user.

Organizations and companies that create distributions are also responsible for the quality of the end 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 responsible in front of their users for the stability and security of the resulting distribution. As a result of all these situations, it is not difficult to imagine why the distributions soon occupied an important place as far as the popularization of libre software in general and GNU/Linux systems in particular is concerned.

A multitude of different distributions, each one with their own peculiarities, exist. Between the most remarkable differences we can name their commercial character (some have companies behind), their size as far as the number of packages that incorporate, their publication policy for new versions, etc. Of between all of them, this study is going to be centered in a particular distribution, although enough extended and very popular: Debian.

This paper shows the most interesting results in general way and, in many occasions, without entering detail. We suggest the interested reader to visits the web page where he will find more statistical information, graphs and more data [DebianCounting]. Also, in [Libresoft] he will find more articles and information on Libre Software Engineering, the branch of software engineering in which we classify this type of studies.

2 About Debian

Debian is a libre operating system that at the present time uses the Linux kernel to carry out its distribution (although some efforts are put in the fact that future Debian distributions may be based on other kernels, such as The HURD). At the moment Debian is available for several architectures, including Intel x86, ARM, Motorola, 680x0, PowerPC, Alpha and SPARC.

Debian is not only the biggest GNU/Linux distribution at present time, it also is one of most stable and enjoys several user preference prizes. Although its base of users is difficult to consider, since the Debian project does not sell CDs and the software that it contains can be redistributed by anyone who desires to do it, we can suppose without any doubt that is an important distribution within the GNU/Linux market.

There exists a categorization within Debian according to the license and the distribution requirements of the software packages. The main part of the Debian distribution (the section called "main" contains a great variety of packages) is compound only of libre software in agreement with [DFSG] (the Debian Free Software Guidelines). It is available for download from the Internet and many redistribuidores sell it in CDs or by other means.

The Debian distributin is created by around thousand of volunteers (generally computer professional). The work of these volunteers consists in 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. This may sound at first simple, but it becomes more complex as soon as factors as the dependencies between the different

packages (the package A needs, to be able to work, of package B) are introduced and the existence of different versions for all these packages.

The work of the members of the Debian project is the same that the one that is made in any other distribution: software integration for its correct joint operation. In addition to work of adaptation and packing, the Debian developers are in charge of the maintainance of an infrastructure of services based on Internet (Web site, on-line archives, bug management systems, mailing lists, support and development, etc.), and for several translation and internationalization projects, the development of several Debian-specific tools and, in general, any element that makes the Debian distribution possible.

Besides its voluntary nature, the Debian project has a characteristic that makes it specially singular: the Debian Social Contract [DebianSocialContract]. This document contains not only the primary goals of the Debian project, but also the means that will be used to carry them out.

Debian also is well-known to have a very strict package and versioning policy with the purpose of obtaining to a greater product quality [DebianPol]. Thus, at any moment three different "flavors" of Debian exist: a stable, an unstable and testing version. As their name point out, the stable version is the version targeted to systems and people looking for high stability. Its software has to pass a freezing period in which only critical errors corrected. The norm is that when a stable version is released it should not contain any known critical error. On the other hand, for the reason of the freeze the stable version usually does not include the most recent version of software applications.

For the ones that wish to have a version with the current software other two contemporary versions to the stable one exist. The testing version includes packages that are on the way of becoming stabilized, whereas the unstable version, as its own name points out, is more inclined to fail and contains the latest of the latest on libre software applications and tools.

At the moment of this study, the stable version of Debian is Debian 3,0 (also well-known as "Woody"), the unstable one receives the codename of "Sid" and the one that is in testing is "Sarge". But in the past, Woody also passed through an unstable stage and, before that, it was in testing. This is important, because what we are going to consider in this article are the different stable versions of Debian since version 2.0 when it was published in 1998. Thus, we have to Debian 2,0 (alias "Hamm"), Debian 2,1 ("Slink"), Debian 2,2 ("Potato") and, finally, Debian 3,0 ("Woody").

The codenames of the versions in Debian correspond to the protagonists of the "Toy Story" animated cartoon film, a tradition that started with version 2,0 when Bruce Perens, then leader of the Debian project and later founder of the Open Source Initiative and the term Open Source, worked for the company that was in charge for make this film. More details on the history of Debian and the Debian distribution in general can be found in [DebianHistory].

3 Metodology of the study

The methodology which we have used for the analysis of the stable versions of Debian is very simple. First all the packages that composed them were unloaded. For each package the number of source lines of code is counted and the programming language in which the code is written is recognized.

The couting is made by means of a tool called tool SLOCCount [SLOCCount]. SLOCCount takes as input a directory where the sources are stored, it identifies by means of a series of heuristic the files which contain source code, by means of other heuristics identifies the programming language in which they are written and finally it counts the number of source lines of code they contain. As we will see more ahead in the formal definition of source lines of code, these include neither commentaries nor blank, reason why the identification of the programming language in which a file is written is essential

taking into account that the syntax of the commentaries differ between languages.

Another tasks that SLOCCount does, although in a quite primitive way, is the identification of identical files and automatically generated code. For the first it builds a database of hashes with the files' MD5 sum, that compares two to two to see if they are identical, whereas for the second it establishes another series of heuristic by means of which it tries to find files generated in an automatic way. Without a doubt, these mechanisms have remarkable deficiencies: to find identical files with slight modifications (for instance, the automatic identifier included for the CVS) by means of the use of such hashes is definitely not very effective, whereas for the second heuristics only take care of well-known and common cases, but not for that reason it detects all of them or others thay may appear in future.

The results of SLOCCount analysis SLOCCount is transformed later on to a XML format that allows easy visualization, manipulation and transformations to other formats. Among the most interesting transformations we encounter the one responsible for having the data in SQL format and inserting it into a database. Then, by means of a simple web interface anyone can have access to the data in crude and even to other more elaborated visualization forms as graphs that facilitate a first analysis. The research group that has carried out this study offers, consequently, a web site where all these data (and some more), statistics and graphs shown in this study can be found ([DebianCounting]).

A more description of the methodology used, as well as its main error sources can be found in [GBarahona2001] and [GBarahona2003].

4 Evolution of the number of Debian developers

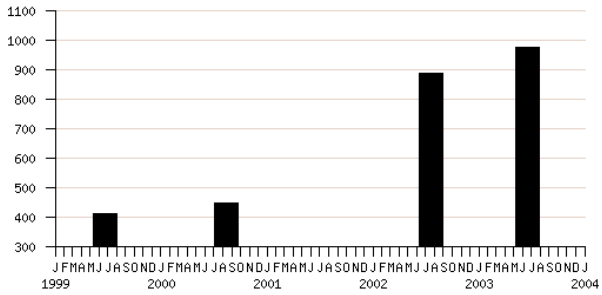
From June 1999 onwards, Debian holds to a database [DBDebian] with data related to the members of the project, in order to ensure communication with and between them. The data that is contained is the name, their nick or username, their e-mail address and the PGP/GPG key. In addition, it includes data on the country of residence - interesting for knowing Debian developers that live near and the date of entrance to the project (if this one is later to the date of creation of the database, else the database creation date, June 20th 1999, is shown). For this paper, we have taken some of these data and we have processed them properly to preserve on the one hand the anonymity of the developers and on the other one to obtain information the evolution of the number of developers and countries in which they reside. In [Robles2001] a similar study has already been made.

In Figure 1 we can see the number of Debian developers at the moments of releasing a new stable version. It is possible to observe that between versions 2.1 and 2.2 (for the 2.0 we do not have data) a slight growth exists, that is accentuated remarkably in the space of time between 2.2 Debian and Debian 3.0. In those two years, the number of developer of Debian doubles. The last column corresponds to the number of developers entered in the Debian database at the moment. We can see how the Debian project continues growing good rate, although not as firmly as in the time between 2.2 Debian and Debian 3.0.

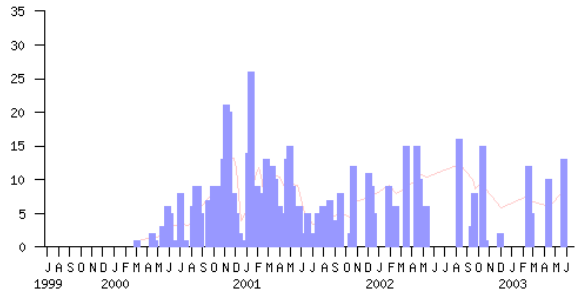
We also have included a figure in which we can see the entrance rate of new members per week into the Debian project. For it, as commented before, we only counted on data from June 21st 1999 onwards. The first and most surprising thing is to observe a period of freezing as far as the number of developer that extends from June 1999 (or perhaps before, since we do not have previous data) to March 2000. This stop can be explained by a change of the new developer policy. It seems that there wer some members who had entered without knowing, understanding or agreeing with the philosophical lines of Debian written down in [DebianSocialContract], so that discussions became

unbearable. The members of the project decided then that they had to put in practice a mechanism to avoid these cases in the future and meanwhile no more developers were admitted.

Once the admission process reopened, the number of Debian developers grew without stopping and at good rate during the rest of the year 2000 and 2001 until in the middle of 2002 the incorporations seem to slow down in an obvious way. In January 2001 we found the peak of incorporations with 26 incorporations in one week. On the other hand, the fact that it seems that new developers enter in groups from mid-2002 onwards and not continuously is possibly caused by the fact that the database is updated periodically.



(a) Number of Debian developers when releasing stable versions



(b) New developers that enter the Debian project

The figure at the left hand side shows the number of developers in the dates where a new stable version was released, while the figure at the right shows the number of new developers that enter the project in time.

Figure 1: Number of Debian developers when releasing stable versions

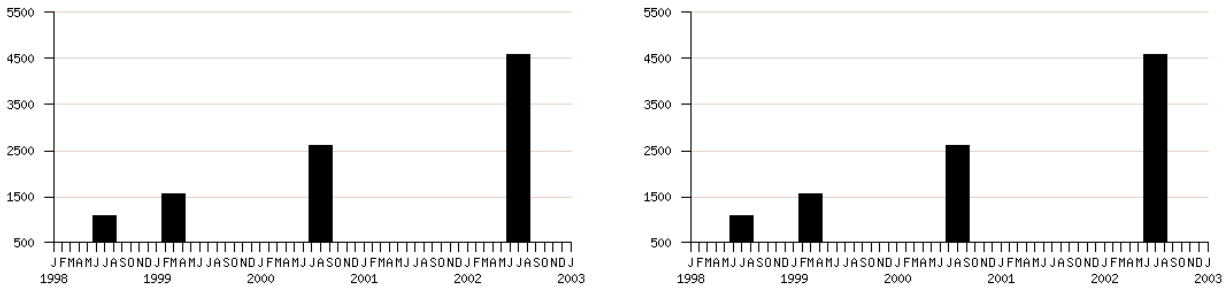
In table Table 1 the distribution of the Debian developers can be seen according to the countries of residence and throug the time for the 11 countries that have more developer. The remaining 36 countries that also have at least one Debian developer fall off the table. A tendency to decentralization can be observed:, something that is stated by the fact that the growth of the number of developers in the United States - the country that contributes most- is inferior to the average. Generally, countries have been able to double their number of volunteers in the last four years, being France the clearest example in this sense, since its presence has grown by a factor of five. Considering that the first steps of Debian took place in America (particularly in the United States and Canada), we can see that in the last four years the project has undergone an “europeanization”. We suppose that the following step will be the longed for globalisation with the incorporation of South American, African and Asian countries (with the exception of Korea and Japan, already well represented), although the data that we manage (2 developers in Egypt, China and India, 1 in Mexico, Turkey and Colombia in June of 2003) is not very flattering in this sense.

Table 1: Countries with higher number of Debian developers

Countries	1999.07.01	2000.07.01	2001.07.01	2002.07.101	2003.06.20
United States	162	169	256	278	297
Germany	54	58	101	121	136
United Kingdom	34	34	55	63	75
Australia	23	26	41	49	52
France	11	11	24	44	51
Canada	20	22	41	47	49
Spain	10	11	25	31	34
Japón	15	15	27	33	33
Italy	9	9	22	26	31
The Netherlands	14	14	27	29	29
Sweden	13	13	20	24	27

5 Physical source lines of code (SLOC)

The physical number of source lines of source code is one of the measures used commonly to compare software. From SLOCs established methods for effort estimation and optimal timing can be used (as it is the case for COCOMO). The definition of a physical source line of code in this context is 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”. The acronym of the unit of physical source line of code SLOC, although the use in KSLOC is commonest. Due to the size of software which we considered in this paper, sometimes the required unit for measurement is in million source lines of code, MSLOC. In figure Figure 2 the number of MSLOC and source packages for the considered stable versions of Debian can be seen.



(a) MSLOC for each version

(b) Number of packages for each version

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

Figure 2: Size, in MSLOC, and number of packages for the versions in study.

Debian 2.0 included 1,096 source packages that had more than 25 MSLOC. The following stable version of Debian, the 2.1 (published around nine months later) had more than 37 MSLOC distributed in 1,551 source packages. Debian 2.2 (that arrived 15 months after Debian 2.1) was summed up around 59 MSLOC in 2,611 packages, whereas the last stable version for the moment, Debian 3.0 (published two years after Debian 2.2), grouped 4,579 packages of source code with almost 105 MSLOC.

Table 2: 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

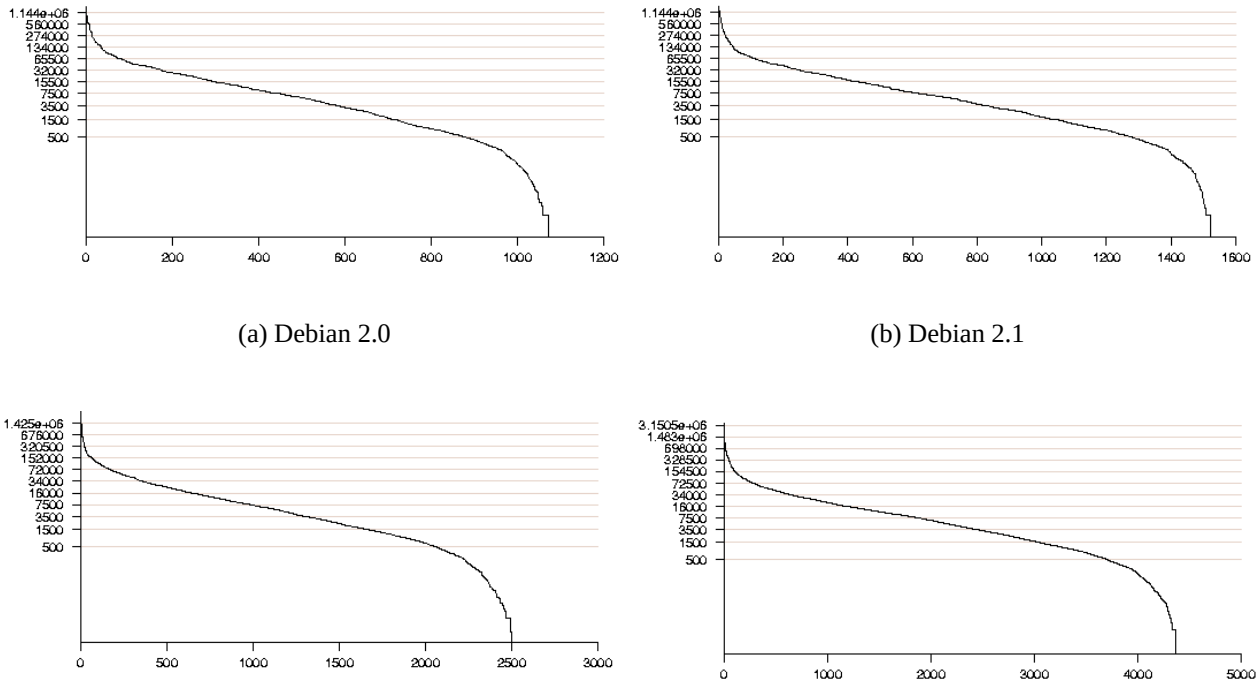
6 Packages

Distributions are organized internally in packages. Packages correspond almost univocally usually to applications or libraries, although commonly in Debian it is tried to modularize packages to the maximum, reason why usually sources are separated from documentation and data, for example. This does not affect much to our results, since the accounts that we have made consider the source lines of code solely and the packages with documentation in general contain little or nothing of code. On the other hand, we have to differentiate two types of packages: binary packages and source packages. The former ones contain sources of the 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.

In the following figure we can see graphs for the distribution of package sizes included in the different versions of Debian. It is possible to observe that there is a small number of great packages (over 100,000 source lines of code) and that the size of these packages tends, as one of Lehman's law of software evolution states to increase in time [**?**]. Nevertheless, it seems surprising that in spite of the growth that has undergone Debian in time, the graph does not show great variations. But certainly what is still more interesting is the fact that the mean size for the packages included in Debian are surprisingly regular (around 23,000 SLOC for Debian 2.0, 2.1, 2.2 and 3.0). With the data that we have at the present time is difficult to give a forceful explanation to 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.

The histogram with the size of the packages shows the data from another perspective. It can be clearly observed how the great packages increase in size with time, while at the same time more and more packages near the origin exit. This fact can be stated specially for the case of very small packages (less than thousand lines of code), small (less than ten thousands) and medium (between ten thousand and fifty thousand lines of code).

It is also interesting to see the evolution of the greatest packages included in each one of the stable versions of Debian. Many of these packages correspond to significant applications, very well-known



(a) Debian 2.0

(b) Debian 2.1

(c) Debian 2.2

(d) Debian 3.0

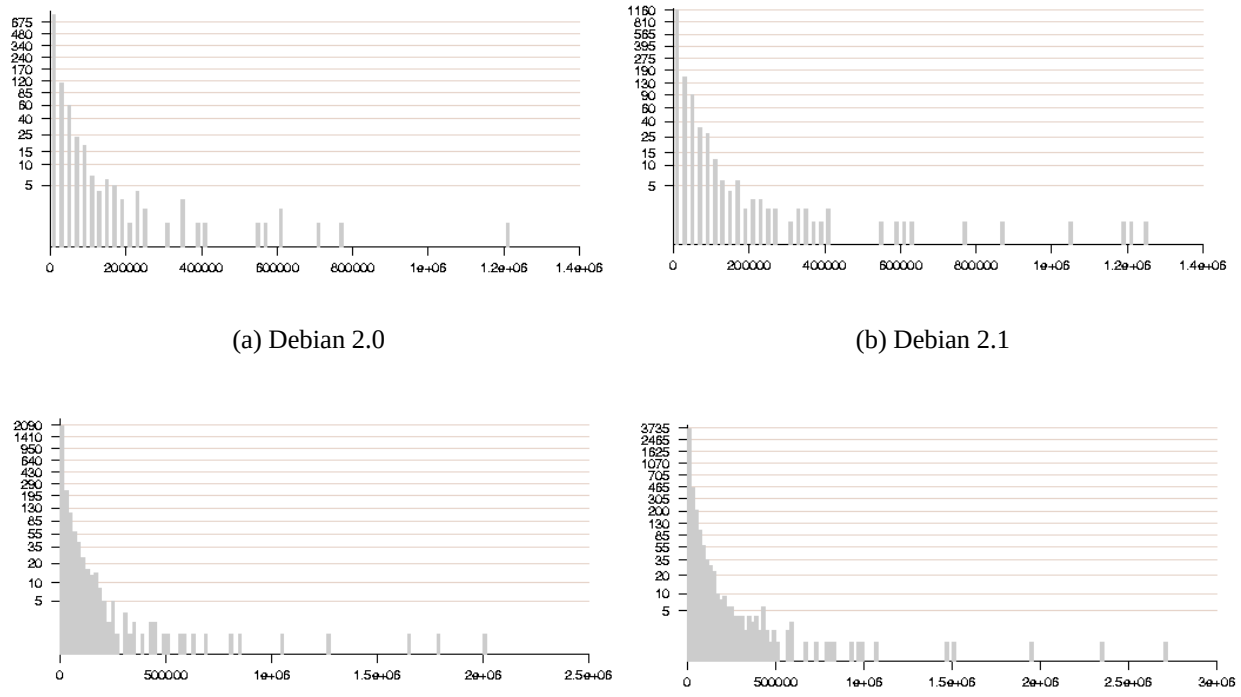
Figure 3: Package sizes in Debian distributions. Packages are ordered by their size along the X axis, while the counts in SLOCs are represented along the Y axis (in logaritmic scale)

and popular and that have been documented in detail in several scientific articles. The study of how these packages evolve in size, as well as having a look at the evolution of the composition of the 10 biggest packages in time can offer an interesting perspective of the Debian distributions.

There is much movement between the select group of top packages in size. The fact that only three of them prevail in Debian 3.0 from the first considered version in this study, Debian 2.0, after almost four years is indicative in this sense. Some of the “new ones” in the club of top packages in size have been included in later versions (as it is the case for the Mozilla navigator), whereas in the case of others we can see that they are compositions made from other packages (so is the case for mingw32, a cross compiler feasible C/C++ for Win32).

On the other hand, it is possible to observ that a clear tendency in time exists for the inferior limit of the top ten packages in size: Whereas in Debian 2.0 we can see how GCC with a 460,000 SLOC was located in the tenth position, the tenth biggest package for Debian 3.0, ncbi-tools (a series of libraries for applications of the scope of Biology) consisted of more than 700,000 lines of code. Another fact supporting this is that only the greatest package of Debian 2.0 would enter between top ten in Debian 3.0.

But top packages in size do not only tend to have more source code, they also show a tendency to have bigger files of source code. While the average of SLOC by file is in the rank between 352 and 359 for packages among the top ten, the average one for all the packages in those versions has between 228 and 243 lines of source code per file. It exists, nevertheless, a great variance in this



(a) Debian 2.0

(b) Debian 2.1

(c) Debian 2.2

(d) Debian 3.0

Figure 4: Histogram with the SLOC distribution for Debian packages

sense, that goes 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.

Table 3: Top 10 packages in size for Debian 2.0

Rank	Package name	Versión	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

From the point of view of the application domain, significant differences in the top packages in size are not seen. We can find at the top of this classification system tools (compilers, debuggers...),

Table 4: Top 10 packages in size for Debian 2.1

Rank	Package name	Versión	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

Table 5: Top 10 packages in size for Debian 2.2

Rank	Package name	Versión	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

specific-purpose libraries and a web navigator (Mozilla). The kernel of the operating system, Linux, packed like kernel-source is a consolidated application in this section.

Until now we have been able to verify how through the last stable versions, Debian has been growing as far as number of packages and number of SLOC in concerned. In the following paragraphs, nevertheless, we would like to center to us in the opposite: what has not changed. We have seen previously in the list of the top packages in size that there are packages that have been added in more recent stable versions of Debian. Other packages, nevertheless, may ”have fallen” away.

Although it can seem surprising, of the 1096 packages included in Debian 2.0, only 754 appear in the last version of Debian considered in this study. This means that less more than 25% of the packages have disappeared from Debian in the last four years. But, this that could be explained because quite a long time has passed as long as the world of software is refered, can be also stated if we watched that the number of packages in Debian 2.2 included also in Debian 3.0 is 1,920 from a total of 2,610, so we get a similar percentage of packages that ”disappear” between these two versions.

Tables Table 7, Table 8, Table 9 and Table 10 show the packages in common between different stable versions. We suppose that two versions have a package in common, if that package is including

Table 6: Top 10 packages in size for Debian 3.0

Rank	Package name	Versión	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

in both, independently from the version number of the package. Each table displays in his 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 own version of Debian that is compared in included. As it is logical, Debian 2.0 will have in common with itself the 1,096 source packages of which it consists.

On the other hand, we also to have consider that distributions contain applications and libraries that evolve in time. This can be observed from the fact that the own version number of included packages also evolves. For example, the Linux sources come generally packaged in a package called kernel-source, as we could see in the tables with the top packages in size. In each version of Debian, the version number of kernel-source changes, reason why we see that Linux has been evolving in time and that this changes and improvements have been introduced in Debian. This thus does not have to be for all the packages. If previously we were interested in packages in common without mattering if their version numbers changed, now we are going to consider those whose version number does not vary among distributions. We consider therefore as common packages those with the same version that are including in two different versions of Debian with the same package version number. We include in the comparison the own Debian version being compared, so that version 2.0 has all of its packages (1,096) with common package version number with itself.

The fact that Debian 3.0 includes 221 packages that have not evolved since their Debian 2.0 (four years before) is very surprising, as it comes to say that 20% of the source packages included in Debian 2.0 have stayed almost inalterable since they were released in Debian 2.0. As it is logical, on the other hand, the number of packages with versions in common increases when the distributions are nearer in the time.

7 Programming languages

As we have already commented in the section dedicated to the methodology of this study, before counting the number of SLOC the programming language in which a file is written is identified. Thanks to this, we can know their significance and the use the different programming languages in Debian. The language most used in all versions is C with percentages that are situated between 60%

Table 7: 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	2,5267,766
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

Table 8: 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

and 85% and with a big advantage on his immediate pursuer, C++. It can be observed, nevertheless, that the importance of C is diminishing gradually, whereas other programming languages grow at a good rate.

For example, in the table Table Table 11 the evolution of the most significant languages - those that surpass 1% of code in Debian 3.0 - is shown. Below the 1% border we can find, in this order, PHP, Ada, Modula3, Objective C, Java, Yacc and ML (all with percentages between 0,30% and 0,60% for Debian 3.0).

There exist some programming languages that we could consider as minor languages and that reach a quite high position in the abovve classification. This is because still being present in a reduced number of packages, these are quite big in size. So is the case of Ada, that in three packages (gnat, an Ada compiler, libgtkada, a binding to the GTK library, and Asis, a system to manage sources in ADA) sums up 430,000 SLOC of a total of 576,000 SLOC that have been included in Debian 3.0

Table 9: 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

Table 10: 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

Table 11: Top programming languages in Debian

Language	KSLOC Debian 2.0	Percentage Debian 2.0	KSLOC Debian 2.1	Percentage Debian 2.1	KSLOC Debian 2.2	Percentage Debian 2.2	KSLOC Debian 3.0	Percentage Debian 3.0
C	19,371	76.67%	27,773	74.89%	40,878	69.12%	66,550	63.08%
C++	1,557	6.16%	2,809	7.57%	5,978	10.11%	13,067	12.39%
Shell	645	2.55%	1,151	3.10%	2,712	4.59%	8,636	8.19%
Lisp	1,425	5.64%	1,892	5.10%	3,197	5.41%	4,087	3.87%
Perl	425	1.68%	774	2.09%	1,395	2.36%	3,199	3.03%
Fortran	494	1.96%	735	1.98%	1,182	1.99%	1,939	1.84%
Python	122	0.48%	211	0.57%	349	0.59%	1,459	1.38%
Tcl	311	1.23%	458	1.24%	557	0.94%	1,081	1.02%

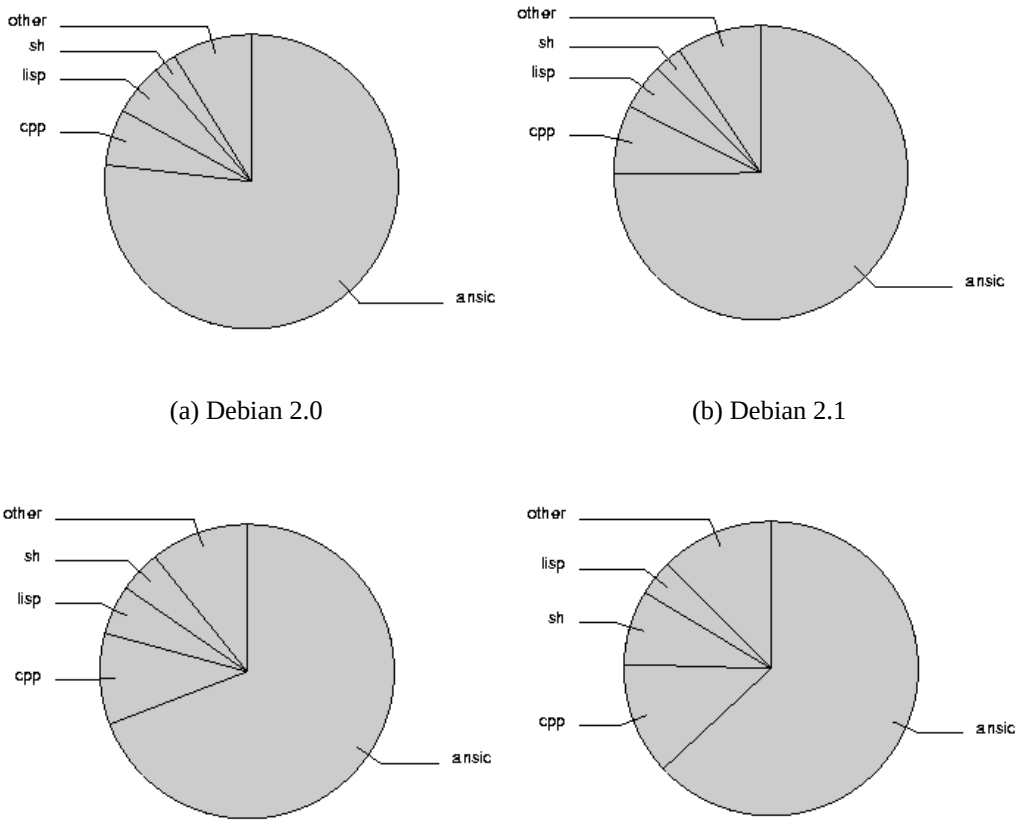
for Ada. Another similar case is the one for Lisp, that counts only in GNU Emacs and XEmacs with more than 1,200,000 SLOC of around 4 MSLOC in all distribution.

The programming language distribution pies show a clear tendency in decline as far as the contribution of C to the global system is concerned. Something similar seems to happen to Lisp, that is to be the third most used language in Debian 2.0 to become the fouth in Debian 3.0, 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 increases.

The graph with the relative evolution of programming languages gives a new perspective of the growth in the last the four stable Debian versions. We took as reference the Debian 2.0 version and suppose that the presence of each language in it is 100% so that growth is shown relative to it..

Previous pies showed that C is backing down as far as its relative presence is concerned. In the next one we can see that even so, C has grown more than 300% throughout the four versions, a fact that should be not despreciable. But we can see that scripting languages (shell, Python and Perl) that have undergone an extraordinary growth, all of them multiplying their presence by factors superior to seven, accompanied by C++. Languages that grow in smaller quantity are the compiled languages (Fortran and Ada). This can give an idea of the importance that the interepreted languages have begun to have in the libre software world.

The graph includes the most representative languages in Debian, but excluding Java and PHP, since the growth of these two is enormous, mainlye because their presence in Debian 2.0 was testi-



(a) Debian 2.0

(b) Debian 2.1

(c) Debian 2.2

(d) Debian 3.0

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

monial.

As far as the average file size is concerned, for the most important programming languages it is interesting to verify how in spite of the spectacular increase of some of them, their means file size remains usually constant. Thus, for C the average length is around 260 to 280 source lines of code per file, whereas in C++ it is located in a bracket that goes 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 peculiar to see how the structured languages usually have average file lengths that are greater than object-oriented languages. Thus the files in C (or Yacc) usually are much greater, in average, than those in C++. This makes us think that the modularity of programming languages is also reflected in the mean file size.

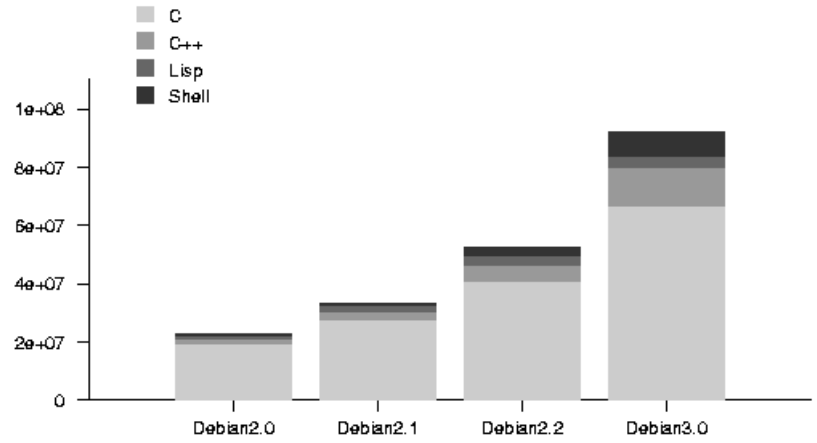


Figure 6: Evolution of the four most used languages in Debian

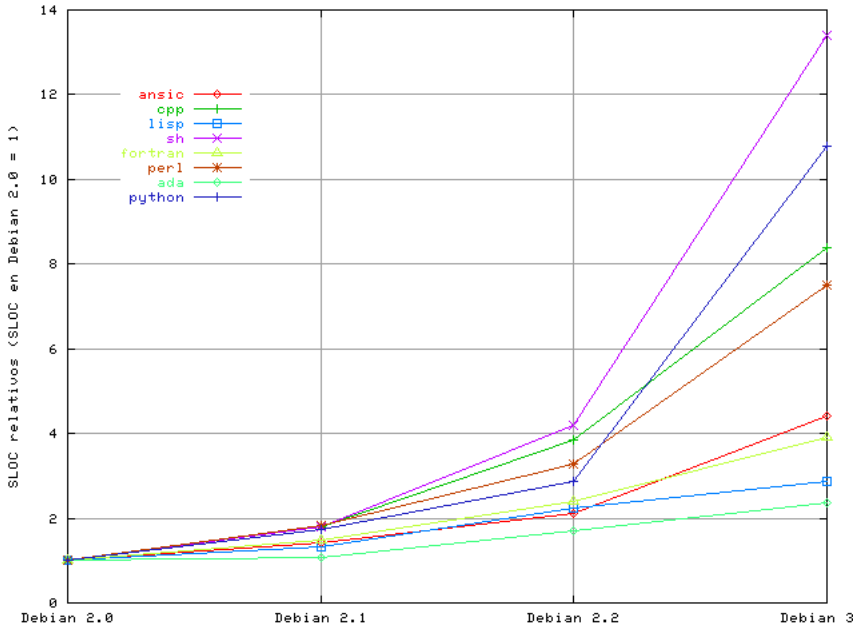


Figure 7: Relative growth of some programming languages in Debian

8 COCOMO

The COCOMO model [Boehm1981] gives an estimation of the human and monetary effort that is necessary to generate software for a given size. It takes as entrance the measurement of the number of source lines of code. COCOMO is a model thought for the “classic” processes of software generation (development in cascade or V) and for big projects, reason why the numbers that it offers to us in our case have to be taken with a lot of care. In any case, the results can give us an idea of the order of magnitude in which we are moving, giving us the necessary optimal efforts if a proprietary development model had been used.

Table 12: Mean file size for some programming languages

Language	Debian 2.0	Debian 2.1	Debian 2.2	Debian 3.0
C	262,88	268,42	268,64	283,33
C++	142,5	158,62	169,22	184,22
Lisp	394,82	393,99	394,19	383,60
shell	98,65	116,06	163,66	288,75
Yacc	789,43	743,79	762,24	619,30
Media	228,49	229,92	229,46	243,35

In general, the most astonishing result COCOMO offers is its cost estimation. In this estimation two 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 [**?**]. "Overhead" is the overhead cost that any company has to assume independently from the programmers' salaries so that the product hits finally 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. 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 which they programmers would perceive to make software. Once this is understotd, cost calculations seem less bulky.

In table Table 13 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 that have been later being summed up. It should be noted that as COCOMO is a non-linear model, the sum of the separated cost of the different packages is not equal to the cost of the sum of all packages. The first result would give us the inferior effort limit, since the integration tasks are not considered, whereas in the second case we have a superior limit, since savings from having independient projects are not considered. In [DebianCounting] the two numbers can be obtained for their comparison. For the goals in this paper it is enough with an estimation of the order of magnitude and therefore only one of them appears.

Table 13: Effort, time and development cost estimation for each Debian version

Version	MSLOC	Effort (man-years)	Time (years)	Cost (USD)
Debian 2.0	25	6,360	4.93	860,000,000
Debian 2.1	37	9,425	4.99	1,275,000,000
Debian 2.2	59	14,950	6.04	2,020,000,000
Debian 3.0	105	26,835	6.81	3,625,000,000

9 Comparison with other distributions

A similar study to this one exists, although in this case the distribution that has served as study object is Red Hat. Red Hat can be considered as the canonical distribution among the commercial ones. It

has a very different strategy and philosophy to the one that has been presented in this paper for Debian. But for a comparison, Red Hat perfectly serves for the purpose of our intentions. We should not forget that the Red Hat Package Manager (RPM) is the one that is used by a great majority of distributions, followed - at a big 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. The results of Red Hat have been extracted in part from [Wheeler2000] and [Wheeler2001] while others have been added by the authors of this paper in order to have a more complete picture.

We can find the main differences 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 homogenous way in order to facilitate its installation and its configuration and update. In other words, while in Debian the packages included in the distribution depend on if there are voluntary collaborators who manage to package 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 their conception results 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 well-known inferior to contemporary Debian versions. Thus, Debian 2.2 doubles Red Hat 7.1 in size when in fact its publication was some months later.

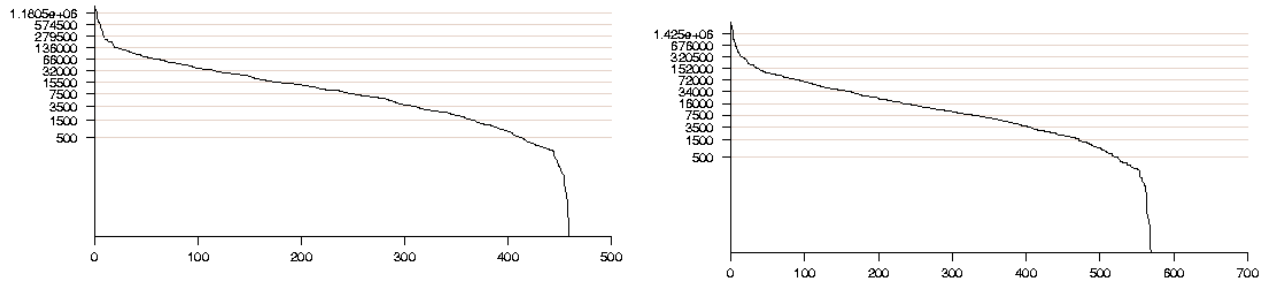
Table 14: Comparison with other GNU/Linux distributions

Nambe	Release date	MSLOC	Effort (man-years)	Time (years)	Cost (USD)
Red Hat 5.2	April 1998	12	3,216	4.93	434,500,000
Red Hat 6.0	April 1999	15	3,951	5.08	534,000,000
Red Hat 6.2	March 2000	17	4,550	5,45	615,000,000
Debian 2.0	July 1998	25	6,360	4.93	860,000,000
Red Hat 7.1	April 2001	30	7,950	6.53	1,075,000,000
Debian 2.1	March 1999	37	9,425	4.99	1,275,000,000
Red Hat 8.0	September 2002	50	13,315	7.35	1,800,000,000
Debian 2.2	August 2000	59	14,950	6.04	2,020,000,000
Debian 3.0	July 2002	105	26,835	6.81	3,625,000,000

On the other hand, Red Hat distributions usually include the most recent versions of software, whereas in Debian freezing intervals before the release have as effect that stable versions do not include the latest of the latest. This can be easily demonstrated studying the package versions included in Red Hat and Debian. For example, we can see that many packages in Red Hat 6.2 and Debian 2.2 agree even though Debian 2.2 was published five months later. In some cases, Debian 2.2 even includes older versions than those obtained in Red Hat 6.2.

Another interesting aspect is that Red Hat seems to show a smaller 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 we remember that with Debian it kept approximately a constant value.

For a more detailed comparison of the Debian and Red Hat versions we refer the reader to [GBarahona2003b].



(a) Red Hat 6.2

(b) Red Hat 7.1

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

10 Comparison with other operating systems

If comparisons are always difficult, those of libre software with propietary one are more. All this study on Debian has been possible by its condition of libre software. The access to the code (and mucho other information that has been exposed in this article) is essential to study thoroughly the different versions as far as number of lines, packages, programming languages that have been used... But the advantages of libre software (and, therefore, of libre software engineering, see [GBarahona2003c]) go further on, because in addition they facilitate the possibility of revision by third parties, being them research groups or just interested people.

In propietary systems, in general, making a study as the one performed in this paper is thus an impossible task. In fact, the accounts that will be offered next have their sources mostly in the own companies that are behind the software development, reason why we cannot guarantee their veracity. In many cases we do not even know if the results 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. It is necessary to add that we do not know either for sure what they consider in their software, reason why for some versions of Microsoft Windows we do not know if they include the Microsoft Office suite or not.

In any case, and considering everything what has commented on the matter in previous paragraphs, we thought that including this comparative is always interesting, since it helps to locate the different Debian versions within a ampler panorama. What it seems to be beyond all doubt is that Debian as well as Red Hat, but specially first one, is the biggest software collection ever build up until the moment.

The cited numbers in the following table proceed from [Lucovsky2000] for Windows 2000, [SunPressRelease] for StarOffice 5.2, [McGraw] for Windows XP and [Schneier2000] for the rest of systems. In table Table 15 the comparative is shown ordered by increasing size.

Table 15: Comparison with proprietary systems

System	Release Date	”Lines of Code”
Microsoft Windows 3.1	April 1992	3,000,000
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
Debian 2.0	July 1998	25,000,000
Microsoft Windows 2000	February 2000	29,000,000
Debian 2.1	March 1999	37,000,000
Windows NT 4.0	July 1996	40,000,000
Debian 2.2	August 2000	55,000,000
Debian 3.0	July 2002	105,000,000

11 Looking into the crystal ball: How will Debian’s next version be?

From the collected data for the four last stable versions of Debian, we have been able to see how this distribution has been evolving. Entering a little into the predictive area, we can use these results to try to guess how the following stable version could be. It has to be noted that if in some cases with the historical Debian versions we have given estimations (for instance the cost and effort calculation), now we are entering the world of speculations and as such should following paragraphs be understood. On the other hand, that our predictions adjust more or less to reality does not only depend on the technical parameters that have been presented in this paper, but also on other organizational and structural issues, which still make the evolution of Debian more unforseeable.

One of the key points for our prediction is to know the release date for the following version. This is, without a doubt, a question to which the members of the Debian project will have to give an answer and that without a doubt will not depend only on technical parameters, but also organizational and human ones. Until now the space of time between stable versions been increasing gradually (between Debian 2.0 and version 2.1 we have only 8 months, whereas between the next versions the time gap was of 17 and 23 months respectively). If we part from these numbers and we suppose that Debian will keep on increasing in size (and therefore its integration is going to be more difficult), we venture to locate the following stable version in December 2004, 28 months after the last one.

As far as the size estimation, in number of source lines of code, we have two different suppositions, due partly to the fact that we are considering a too small number (four) of elements to be able to predict the following one with exactitude. On one hand, if we followed the idea that the code doubles every two years approximately, the following version of Debian would have to contain around 220 MSLOC - assuming it will hit the streets after about 28 months. On the other hand, the factor of growth between stable versions never has been so big. We can see how Debian 2.1 supposed a growth of 50% over Debian 2.0, whereas Debian 2.2 grew 60% and Debian 3.0 80%. This can be because integration becomes more complex as the number of packages increase, something that is very logical on the other hand, and that the freezing period before releasing the new stable version has to increase. The estimation therefore in this second case throws a new stable version that will have around 185 MSLOC.

For the stable versions of Debian that have been studied in this papaer, we have been able to see, for our surprise, that mean size of packages stays constant in time. If we suppose that this is going to continue being , by means of a simple conversion of the previous calculations made in source lines of code, we will obtain the number of packages included in the next version. For the first approach (about 220 MSLOC), we would count on the extraordinary number of 9,600 source packages, whereas for the second approach (about 185 MSLOC) the number of source packages included is slightly superior to 8,000.

Taking as input about 200 MSLOC, the estimations that throws the COCOMO model are astronomical. In order to generate a software of such dimensions, a million man-months (something more than 80,000 man-years) would be required and the considered optimal time corresponds to 450 months (more than 37 years!) in which two thousand developers would have to work on the project. The considered total cost would ascend to approximately 10,000 million euros/USD. It is important to notice that we have assumed in this calculus that Debian is a single project (and not the sum of many smaller projects), since in our forecasts we have not made an estimation of package sizes. As always, these numbers are orientative and so they have to be understood.

In this paragraph we are going to watch in the crystal ball to see the biggest packages included in the next stable version. Until now, we have seen that in spite of having mobility in this section, usually system tools, specific-purpose libraries, compilers and a navigator, Mozilla, prevail. We can assume that with all the controversy that the technology NET is cousing and the success of initiatives like MONO or dotGNU, we will be able to see a compiler, or at least a suite of C# classes among the top packages in size. The turn of libre software towards the end user and the desktop will also be reflected in this category with the more than probable inclusion of the OpenOffice.org office suite. In any case, the toll that has to be payed to enter this select club of top pacakges is going to be very big: almost with complete certainty it will be necessary to surpass the barrier of the million source lines of code.

As far as the distribution of programming languages, we can assure that C will continue being the language with greatest presence within Debian. Nevertheless, its supremacy will continue diminishing until the point that we can affirm that almost half of the code will not be written in C. C++, on the other hand, will continue growing in relative importance, reaching foreseeably 15% of the total code and summing up 30 MSLOC (helped by more than probable inclusion of OpenOffice.org which is made up of about 4 MSLOC written mainly in this language). Nevertheless, it will be the scripting programming languages of script, as PHP, Python and Perl (we dare to indicate that it is even going to be in this order) that will undergo a remarkable increase in code and importance. The compiled languages will follow with their relative tendency to fall, as it will be the case for Fortran, Ada or Pascal.

Regarding Java, we think it will get rid of this tendency and we augure a remarkable ascent, due to two causes: first, the inclusion of several packages from the Apache project (Jakarta, etc.) based on Java that are already nowadays quite big and, second, Debian will be affected by the big number of projects in Java which in the last three years have been launched - probably due to the new generation of developers that have learned Java in their university courses. The C# programming language will enter Debian for its first time, but its presence will be small. This is beacause, although we bet that the following version of Debian has a compiler and a hierarchy of classes for this language, applications that are created with them will still not be the sufficiently mature to be included. In any case, we are sure that C# will be an important language for the version next to the following one.

As far as the number of voluntary developers that participate in the Debian project, we suppose

that it will be continue maintaining the growth rate for the last 18 months, so that the project will have around 1,100 developers. This fact offers serious doubts about the future growth of Debian since if the forecasts stay, the ratio of packages per developer in the following version will be of nine, whereas in the last versions this ratio has moved been between the 4 for version 2.1 and the 6 for Debian 2.2 (in Debian 3.0 the number of source packages per developer was approximately 5). It is possible that this will become the limiting factor in the growth of Debian, since as it has been commented this distribution depends basically on voluntaries wanting to pack a program.

12 Conclusions

In this article the results of studying in depth the stable versions from Debian 2.0 ahead have been shown. We have been able to see the evolution of physical source lines of code, the number and size of the packages and the used programming languages. These data have been supported by the number of voluntary developers with which Debian counts to create its distributions, as well as the effort, time and cost estimations using the well-known COCOMO method. 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 big propietary systems. Finally a prediction of how the following stable version of Debian will be has been made.

Among the most important evidences that have appeared we encounter that the stable versions seem approximately to double in number of source lines of code and packages every two years, a evolution that we think can only be possible to 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 approximately remained constant, so that the number of packages grows linearly with the number of code lines. 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 last version of Debian (3.0) makes us think that we are in front of one of the biggest software collections in the history of humanity, if not the biggest. In order to create their 105 MSLOC, according to the COCOMO model, 27,000 person-years would be necessary and the cost would go up to around the 3,600 million dollars. None of the other systems with which we have compared Debian (Red Hat, Solaris, Windows, etc.) can compete at the present time in size with Debian.

The top applications in size contained in Debian consist predominantly of low level applications (kernel, development software, specific-purpose libraries...), although lately with the inclusion of Mozilla there has been an upset towards end user applicactions. We suppose that in future versions, the OpenOffice.org office suite will make this tendency more patent.

As far as the programming languages are concerned, C is the most used language more, 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, we would like to insist that we have been mainly offering a small amount of measures and some estimations, although we considere that they are sufficient to draw some conclusions, to compare with other systems and to make some predictions about Debian's future.

References

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

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

[DBDebian] *Debian Developers Database*, Debian Project, http://db.debian.org . 4

[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.dat.escet.urjc.es/debian-counting/ . 1, 3, 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, 4

[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/ . 3

[GBarahona2003] *Measuring Woody: The size of Debian 3.0*, 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, Pending publication. Will be available at http://people.debian.org/~jgb/debian-counting/ . 3

[GBarahona2003b] *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, Pending publication in the book "Free/Open Source Software Development" edited by Stefan Koch and published by Idea Group, Inc. . 9

[GBarahona2003c] *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 . 10

[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 .

[Libresoft] *Libre Software Engineering*, Jesús M. González-Barahona and Gregorio Robles, http://libresoft.dat.escet.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

[McGraw] *Building Secure Software: How to avoid security problems the right way*, Gary McGraw, Cited by David A. Wheeler in http://www.dwheeler.com/sloc/ . 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 . 4

[Robles2002] *Ingeniería del Software Libre - Una visión alternativa a la ingeniería del software tradicional*, Gregorio Robles, http://es.tldp.org/Presentaciones/200211hispalinux/robles/robles-ponencia-hispalinux-2002.pdf .

[SLOCCount] *SLOCCount*, David Wheeler, http://www.dwheeler.com/sloccount/ . 3

[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