UPGRADE
The European Online Magazine for the IT Professional
http://www.upgrade-cepis.org
Vol. II, No. 6, December 2001

UPGRADE is the European Online Magazine for the Information Technology Professional, published bimonthly at http://www.upgrade-cepis.org/

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, http://www.cepis.org/) by Novática (http://www.ati.es/novatica/) and Informatik/Informatique (http://www.svifsi.ch/revue/)

Chief Editors
François Louis Nicolet, Zurich <nicolet@acm.org>
Rafael Fernández Calvo, Madrid <rfcalvo@ati.es>

Editorial Board
Peter Morrogh, CEPIS President
Prof. Wolffried Stucky, CEPIS President-Elect
Fernando Sanjuán de la Rocha and
Rafael Fernández Calvo, ATI
Prof. Carl August Zehnder and François Louis Nicolet, SVI/FSI

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson

Cover page designed by Antonio Crespo Foix, © ATI 2001

Layout: Pascale Schürmann

E-mail addresses for editorial correspondence: <nicolet@acm.org> and <rfcalvo@ati.es>

E-mail address for advertising correspondence: <novatica@ati.es>

Copyright
© Novática and Informatik/Informatique. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

Coming issue:
"**Knowledge Management**"

Open Source / Free Software: Towards Maturity

Guest Editors: Joe Ammann, Jesús M. González-Barahona, Pedro de las Heras Quirós

Joint issue with NOVÁTICA and INFORMATIK/INFORMATIQUE

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

Debian is the largest Free Software distribution, with well over 2,800 source packages in the latest stable release (Debian 2.2) and more than 4,000 source packages in the release currently in preparation. But, how large is "the largest"? In this paper, we use David Wheeler's sloccount system to determine the number of physical source lines of code (SLOC) of Debian 2.2 (aka Potato). We show that Debian 2.2 includes over 56,000,000 physical SLOC (almost twice than Red Hat 7.1, released about 8 months later), showing that the Debian development model (based on the work of a large group of voluntary developers spread around the world) is at least as capable as other development methods (like the more centralized one, based on the work of employees, used by Red Hat or Microsoft) to manage distributions of this size.

Keywords: Debian, Free Software, Libre Software, SLOC, Lines of Code, Linux

1 Introduction

On August 14th of 2000 the Debian Project announced Debian GNU/Linux 2.2, the "Joel 'Espy' Klecker" release [Debian22Ann] [Debian22Rel]. Code named "Potato", it is the latest (to date) release of the Debian GNU/Linux Operating System. In this paper, we have counted this distribution, showing its size, and comparing it to other distributions.

Debian is not only the largest GNU/Linux distribution, it is also one of the more reliable, and enjoys several awards based on users preferences. Although its user base is difficult to estimate, since the Debian Project does not sell CDs or other media with the software, it is for sure important within the Linux market. It also takes special care to benefit from one of the freedoms that Free Software provides to users: the availability of source code. Therefore, source packages are carefully crafted for easy compilation and reconstruction of original (upstream) sources. This makes it also convenient to measure and in general, to get statistics about it.

The idea of this paper came after reading David Wheeler's fine paper [Wheeler 01]. We encourage the reader to at least browse it, and compare the data it offers for Red Hat Linux with the ones found here.

The structure of this paper is as follows. Next section provides some background about the Debian Project and the Debian 2.2 GNU/Linux distribution. After that, we discuss the method we have used for collecting the data shown in this paper. Later on, we offer the results of counting Debian 2.2 (including total counts, counts by language, counts for the largest packages, etc.). The following section offers some comments on the numbers and how they should be understood, and some comparisons with Red Hat Linux and other operating systems. To finish, we include some conclusions and references.

2 Some Background about Debian

The Debian 2.2 GNU/Linux distribution is put together and maintained by the Debian project. In this section, we offer some background data about Debian as a project, and about the Debian 2.2 release.

Jesús M. González Barahona is a lecturer at the *Universidad Rey Juan Carlos*, and collaborator of *BarraPunto.Com*. He began working on the promotion of Free Software in 1991, in the PD-SOFT group (later the *Sobre* group). Since then he has been involved in many activities in this field, such as the organisation of seminars, giving courses and taking part in Free Software working groups. He is currently collaborating on several Free Software projects (Debian and The Espiral among others), he takes part in the Working Group on Free software promoted by the DG-INFO of the European Commission, he collaborates with associations like Hispalinux and EuroLinux, he writes for several publications about Free Software, and he advises companies in their strategies regarding this subject. Co-editor of the Free Software section of Novática. <jgb@gsyc.escet.urjc.es>

Miguel A. Ortuño Pérez is engineer in informatics, professor at the *Universidad Rey Juan Carlos* in Madrid where his domains of interests are mobile computation and Free Software. Previously he worked at the University of Oviedo in various projects related to remote teaching.

José Centeno González is professor at the *Universidad Rey Juan Carlos* in Madrid. He joined the Informatics department of the *Universidad Carlos III* in Madrid in 1993 where he worked until 1999. His research interests include distributed systems programming, communications protocols and mobility. He is also interested in the impact of Free Software in the domain of software engineering and industry. He holds a degree in telecommunication engineering from the *Universidad Politécnica de Madrid* where he expects to obtain the doctor's degree this year.

Vicente Matellán Oliveras is professor at the *Universidad Rey Juan Carlos* in Madrid. He has been active in the fields of Free Software, among others in the creation of OpenResources.com and BarraPunto.com.

© Novática and Informatik/Informatique

The Debian Project

Debian is a Free Operating System, that currently uses the Linux kernel to put together the Debian GNU/Linux software distribution (although other distributions based in other kernels, like the Hurd, are expected in the near future). This distribution is available for several architectures, including Intel x86, ARM, Motorola 680x0, PowerPC, Alpha, and SPARC.

The core of the Debian distribution (called section "main", which amounts for the vast majority of the packages) is composed only of Free Software, according to the DFSG (Debian Free Software Guidelines) [DFSG]. It is available on the Web for download, and many redistributors sell it in CDs or other media. The Debian distribution is put together by the Debian project, a group of over 900 volunteer developers spread around the world, collaborating via the Internet. They take care not only of adapting and packaging all the software included in the distribution, but also of the web infrastructure, the Debian bug tracking system, the internationalization efforts, the Debian support and development lists, and in a wide sense, of all the infrastructure that makes the Debian distribution possible.

Debian developers package software which they obtain from the original (upstream) authors, ensuring that it works smoothly with the rest of the programs in the Debian system. For this matter, there is a set of rules that the package should comply with, the so called Debian Policy Manual [DebianPol]. Most of the work for packaging a given program is usually make it compliant with those rules. Developers also take bug reports, try to fix them (reporting fixes and problems upstream), follow new upstream developments, and build all the software glue needed for making Debian system work. Bugs and security holes are discussed openly, and updates fixing important problems are made available for stable releases on a daily basis so that users can maintain their systems secure and as much bug free as possible.

Debian is unique for many reasons. Its dedication to Free Software, its non-profit nature, and its open development model (where most of the discussions are addressed openly in public lists) are remarkable. The project is committed to Free Software, as is reflected in the Debian Social Contract. The definition of what Debian understand as Free Software can be found in the Debian Free Software Guidelines (DFSG), and in essence is the same software which can be considered "Open Source" software (which is not strange, since the Open Source Definition was actually derived from the DFSG).

Debian Potato

Debian 2.2 (Potato) is the latest official release, the one currently considered as "stable". It was released in August 2000, and includes all the major Free Software packages available at that time. Only in its main distribution, composed only of Free Software (according to the Debian Free Software Guidelines), there are more than 2,600 source packages. The whole release includes almost 4,000 binary packages, which the user can install easily from various media or from Internet servers.

Debian 2.2 is split in two archives: the "regular" one, and the *non*-**US** archive. In *non-US* are archived those packages which have some legal impediment to be exported from the United States of America (usually the US legislation on strong cryptography).

Each archive is composed of several so called "distributions": *main*, *contrib* and *non-free*.

For the work referenced in this paper we have considered only the *main* distribution of the "regular" archive. It is (by far) the largest fraction of the archive, is composed of Free Software only, and has no export restrictions. In many respects, it is the largest coordinated collection of Free Software available on the Internet.

3 Collecting the Data

The approach used for collecting the data presented in this paper is, in summary, as follows:

- Which source code makes a Debian release?
  Fortunately enough, source code for current and past Debian releases is archived, and available for everyone on the Internet. The only problem is to determine the list of source packages for any given release, and where to access them.
- Downloading and collecting data
  Once we know what files to download, we have to download all of them before being able to gather data. Since the size of the unpackaged sources for a Debian release is very large, we chose to work on a per-package basis, gathering all the relevant data from it (mainly, the number of lines of code) before deleting it and following on with the download of the next one.
- Final analysis
  Analyse the collected data and get some statistics regarding the total number of physical SLOC of the release, the SLOC for each package, the SLOC for each of several programming languages considered, etc.

In the following sections these three steps are described in more detail.

Which Source Code Makes a Debian Release?

The Debian packaging system considers two kind of packages: source and binary. One of more binary packages can be built automatically from each source package. For this paper, only source packages are relevant, and therefore we will no longer refer to binary packages.

When building a source package, a Debian developer starts with the "original" source directory for the piece of software. In Debian parlance, that source is called "upstream". The Debian developer patches upstream sources if needed, and creates a directory *debian* with all the Debian configuration files (including data needed to build the binary package). Then, the source package is built, usually (but not always) consisting of three files: the upstream sources (a *tar.gz* file), the patches to get the Debian source directory (a *diff.gz* file, including both patches to upstream sources and the *debian* directory), and a description file (with extension *dsc*). Only in latest releases *dsc* files are present. Patches files are not present for "native" source packages (those developed for Debian, with no upstream sources).

© Novática and Informatik/Informatique

Source packages of current Debian releases are part of the Debian archive. For every release, they reside in the *source* directory. There are sites in the Internet including the source packages for every official Debian release to date (usually, mirrors of archive.debian.org). Since Debian 2.0, for every release a *Sources.gz* file is present in the *source* directory, with information about the source packages for the release, including the files that compose each package. This is the information we use to determine which source packages, and which files, have to be considered for Debian 2.2.

However, not all packages in *Sources.gz* should be analysed when counting lines of code, because there are, in some cases, several versions of the same piece of software. For instance, in Debian 2.2 we can find source packages *emacs19* (for *emacs-19.34*), and *emacs20* (for *emacs-20.7*). Counting both packages will imply counting Emacs twice, which is not intended. Therefore, the list of packages must be inspected manually for every release, detecting those which are versions of the same software, and choosing one "representative" for each family of versions.

These cases may cause an underestimation of the number of lines of the release, since different versions of the same package may share a lot of code, but not all (consider for instance PHP4 and PHP3, with the former being an almost complete rewrite of the latter). However, we think this effect is negligible, and compensated with some overestimations (see below).

In other cases, we have decided to analyse packages which may have significant quantities of code in common. This is the case, for instance, of *emacs* and *xemacs*. Being the latter a code fork of the former, both share a good quantity of lines which, even when not being exactly equal, are evolutions of the same "ancestors". Other similar case is *gcc* and *gnat*. The latter, an Ada compiler, is built upon the former (a C compiler), adding many patches and lots of new code. In those cases, we have considered that the code is different enough to consider them as separate packages. This probably leads to some overestimation of the number of lines of code of the release.

The final result of this step is the list of packages (and the files composing them) that we consider for analysing the size of a Debian release. This list is done by hand (with the help of some really simple scripts) for each release.

Downloading and Collecting Data

Once the packages and files composing Debian 2.2 are determined, they are downloaded from some server of the net of Debian mirrors. Some simple Perl scripts where used to automate this process, which (for each package) consists of the following phases:

- Downloading of the files composing the package
- Extraction of the source directory corresponding to the upstream package (by untaring the *tar.gz* file. After extraction, data about this upstream source is gathered.
- Patching of the upstream directory with the *diff.gz* file, to get the Debian source directory. After extraction, data about it is gathered.
- Deletion of the *debian* directory, to avoid counting maintainer scripts (stored in this directory), and gathering data about this sans-debian Debian source package.

Not all packages have upstream version. Therefore, during this process, some care has to be taken to differentiate this situations.

The fetching of data is done using *sloccount* scripts, three times for each package (one in each phase, see above), which stores the count of lines of code for each package in a separate directory, ready for later inspection and reporting.

The reason for fetching data three times for every package is to analyse the impact of the Debian developer on the source package. This impact can be in the form of patches to the source (usually to make it more stable and secure, to conform to Debian installation policy, or to add some functionality to it) or in installation scripts (which can be singled out when counting sans-debian source packages).

The final result of this step is the collection of all the data fetched from the downloaded packages, organized by package, and ready to be analysed. These data consist mainly of lists of files and line counts for them, split by language.

Final Analysis

The last step is the generation of reports, using *sloccount* and some scripts, to study the gathered data. Since in this step all the fetched data is available locally, and in a simple to parse form, the analysis can be done pretty quickly, and can be repeated easily, looking for different kinds of information.

The final result of this step is a set or reports and statistical analysis, using the data fetched in the previous step, and considering them from different points of view. These results are presented in the following section.

4 Results of Counting Debian

The main results of our current analysis of the Debian 2.2 GNU/Linux release can be organized in the following categories:

- Size of Debian Potato.
- Importance of the most used programming languages.
- Analysis of the evolution in the size of the most relevant packages.
- Effort estimations.

Size of Debian Potato

We have counted the number of source lines of code of Debian GNU/Linux 2.2 in three different ways, with the following results (all numbers are approximate, see appendix for details):

- Count of upstream packages "as such": 52,810,000 SLOC
- Count of Debian source packages: 56,180,000 SLOC
- Count of Debian source packages without *debian* directory: 55,920,000 SLOC

For details on the meaning of each category, the reader may revisit the subsection "Downloading and collecting data". In short, the count of upstream packages could be considered as the size of the original software used in Debian. The count of Debian source packages represents the amount of code actually

© Novática and Informatik/Informatique

present in the Debian 2.2 release, including both the work of the original authors and the work of Debian developers. This latter work includes Debian-related scripts and patches. Patches can be the work of Debian developers (for instance to adapt a package to the Debian policy) or be the downloaded from elsewhere. The count of Debian packages without the *debian* directory excludes Debian-related scripts, and therefore is a good measure of the size of the packages as they are found in Debian, excluding the specific Debian-related scripts.

It is also important to notice that packages developed specifically for Debian have usually no upstream source package. This is, for instance the case of *apt*, which is present only as a Debian source package.

Programming languages

The number of physical SLOC, classified by programming language, are (roughly rounded) as follows (numbers for Debian source packages):

- C: 39,960,000 SLOC (71.12%)
- C++: 5,500,000 SLOC (9.79%)
- LISP: 2,800,000 SLOC (4.98%)
- Shell: 2,640,000 SLOC (4.70%)
- Perl: 1,330,000 SLOC (2.36%)
- FORTRAN: 1,150,000 SLOC (2.04%)
- Tcl: 550,000 SLOC (0.99%)
- Objective C: 425,000 SLOC (0.76%)
- Assembler: 425,000 SLOC (0.75%)
- Ada: 405,000 SLOC (0.73%)
- Python: 360,000 SLOC (0.65%)

Below 0.5% we find some other languages: Yacc (0.46%), Java (0.20%), Expect (0.20%), Lex (0.13%), and others below 0.1%.

When we count the lines in the Debian source packages without the *debian* directory (which contains package configuration files and maintainer scripts), the numbers are similar. This means that the maintainer scripts are not a significant part of the distribution. The main difference is in Shell lines (about 150,000 less) and in Perl lines (about 80,000 less), which uncovers the preferred languages for those scripts.

However, when we count original (upstream) source packages there are some remarkable differences: about 2,000,000 lines of C code, 300,000 lines of LISP, 200,000 lines of FORTRAN, and minor variations in other languages. This differences can usually be amounted to patches to upstream packages made by the Debian developer. Therefore, looking at this numbers, we can know in which languages are written the most patched packages.

The Largest Packages

The largest packages in the Debian potato distribution are:

- Mozilla (M18): 2,010,000 SLOC (2,010,000). C++ amounts for 1,260,000 SLOC, C for 702,000. Mozilla is the well known Open Source WWW browser.
- Linux kernel (2.2.19): 1,780,000 SLOC (1,780,000). C amounts for 1,700,000 SLOC, Assembler for 65,000. The Linux 2.x kernels were the stable series at the time of the Debian 2.2 release.
- XFree86 (3.3.6): 1,270,000 SLOC (1,265,000). Mainly 1,222,000 SLOC of C. This is an X Window implementation, including graphics server and basic programs.
- PM3 (1.1.13): 1,115,000 SLOC (1,114,000). 983,000 SLOC of C, 57,000 of C++. PM3 is the Modula-3 distribution of the Ecole Polytechnique de Montréal, including a compiler and libraries.
- OSKit (0.97): 859,000 SLOC (859,000). Amounts for 842,000 SLOC of C. OSKit is the Flux Operating System Toolkit, a framework for operating system design.
- GDB (4.18): 801,000 SLOC (800,000). Includes 727,000 lines of C and 38,000 of Expect. GDB is the GNU source-level debugger.
- GNAT (3.12p): 688,000 SLOC (687,000). About 410,000 SLOC of C and 248,000 SLOC of Ada. GNAT is the GNU Ada 95 compiler, including libraries.
- Emacs (20.7): 630,000 SLOC (629,000). 454,000 SLOC of LISP, 171,000 SLOC of C. Emacs is the well known extensible text editor (and many, many things more).
- NCBI Libraries (6.0.2): 591,000 SLOC (591,000). Almost only C is found, 590,000 SLOC. This package includes libraries for biology applications.
- EGCS (1.1.2): 578,000 SLOC (562,000). Includes 470,000 SLOC of C and 55,000 SLOC of C++. This package includes the GNU C++ extension library.
- XEmacs, base support (21): 513,000 SLOC (513,000). An almost pure LISP package, 510,000 SLOC. Includes the base extra Emacs LISP files needed to have a working XEmacs.

Numbers in parenthesis are approximate number of SLOC of upstream packages, the rest of the numbers are approximate number of SLOC of the Debian source packages. Only data for the more relevant languages found in each package are reported. The reader may notice that in most cases, the numbers in both cases are roughly equal, showing evidence that, in those cases, the additions done by Debian developers are minimal (although modifications could be more important).

The release numbers of the packages are obviously not current, but those were the ones available at the time of the freeze for Debian 2.2 (Spring 2000). The classification could be different had Debian developers packaged things in other ways. For instance, if all Emacs extensions were in the Emacs package, it would have been much larger. However, a Debian source package usually matches well with what upstream authors consider as a package, and with the general idea about what is a package.

The next packages by SLOC size (between 350,000 and 500,000 SLOC) are Binutils (GNU assembler, linker, and binary utilities), TenDRA (C and C++ compiler and checker), LAPACK (a set of linear algebra routines), and Gimp (the GNU Image Manipulation Package). Except for LAPACK (which is composed mainly of FORTRAN files), these packages are mainly written in C.

Effort and Cost Estimations

Using the basic COCOMO model [Boehm 81], the effort to build a system with the same size as Debian 2.2 can be estimat-

© Novática and Informatik/Informatique

ed. This estimation assumes a "classical", proprietary development model, and therefore is not valid to estimate the effort which has been applied to build this software. But it can give us at least an order of magnitude of the effort which would be needed in case a proprietary development model had been used.

Using the SLOC count for the Debian source packages, the data provided by the basic COCOMO model are as follows:

- Total physical SLOC count: 56,184,171
- Estimated effort: 171,141 person-months (14,261 person-years)
- Formula: 2.4 * (KSLOC**1.05)
- Estimated schedule: 72.53 months (6.04 years)
  Formula: 2.5 * (Effort**0.38)
- Estimated cost to develop: 1,848,225,000 USD

For calculating the cost estimation, we have used the mean salary for a full-time systems programmer during 2000, according to Computer World [ComWorld 00], which is of 54,000 USD per year, and an overhead factor of 2.4.

5 Comments and Comparisons

The numbers offered in the previous section are estimations. They give us least orders of magnitude, and allow for comparisons. But they should not be taken as exact data, there are too much sources of error and field for interpretation. In this section, we will discuss some of the more important assumptions made, and the possible sources of error. We will also compare the SLOC counts with the SLOC counts for other system, with the aim of giving the reader some context to interpret the numbers.

What Is a Source Line of Code?

Since we rely on David Wheeler's *sloccount* tool for counting physical SLOC, we also rely on his definition for "physical source lines of code". Therefore, we could say that we identify a SLOC when *sloccount* identifies a SLOC. However, sloccount has been carefully programmed to honour the usual definition for physical SLOC: *"A physical source line of code is a line ending in a newline or end-of-file marker, and which contains at least one non-whitespace non-comment character."*

There is similar measure, the "logical" SLOC, which sometimes is preferred. For instance, a line written in C with two semicolons would be counted as two logical SLOC, while it would be counted as one physical SLOC. However, for the purposes of this paper (and almost for any purpose), the differences between both definitions of SLOC are negligible, specially when compared to other sources of error and interpretation.

Sources of Inaccuracy in the SLOC Counts

The counts of lines of code presented in this paper are no more than estimations. We do not imply that they are exact, specially when they refer to aggregates of packages. Several factors cause inaccuracy of the numbers, some due to the tools used to count, some others due to the selection of packages:

- Some files may have not being counted accurately.
  Although *sloccount* includes carefully designed heuristics to detect source files, and to distinguish source lines from comments, those heuristics do not always work as expected. In addition, in many cases it is difficult to distinguish automatically generated files (which should not be counted), although sloccount makes also a good effort to recognize them.
- Not all programming languages are recognized.
  To fetch the data we used release 1.9 of *sloccount*, which recognizes about 20 different languages. However, some languages present in Debian (as is the case of Modula-3 or Erlang) are not currently supported. This obviously leads to some underestimation in the packages with files written in those languages.
- Different perceptions in the aggregation of package families and the selection of a representative.
  As we comment in the subsection where we discuss the selection of the list of packages to count, the reasons to take a given package in or out of the list are not unquestionable. Should we count different releases of the same package? Should we count only once code present in several packages? The usual criteria for measuring SLOC is "delivered source lines of code". From this point of view, all packages should be considered as they appear in the Debian release. However, this is difficult to apply when some packages are clearly evolutions of other packages. Instead of considering all of them as "delivered", it seems more productive to consider the older ones as "beta releases". However, in the Free Software world it is rather common to deliver stable releases every 6 or 12 months. Those stable releases have a lot of work behind them, only to ensure stability, even if they are also the foundation for later releases.

In most cases, we have adopted an intermediate decision: to count only once families of packages which are a line of evolution (as is the case of *emacs19* and *emacs20*, but to count separately families of packages which happen to share some code but are in themselves different developments (as is the case of gcc and gnat).

Estimation of Effort and Cost

Current estimation models, and specifically COCOMO, only consider classical, proprietary development models. But Free Software development models are different, and these models are not directly applicable. That way, we can only estimate the cost of the system, had it been developed using classical development models, but not the actual cost (in effort or in money) of the development of the software included in Debian 2.2.

Some of the differences that prevent from using these estimation models are:

- Continuous release process (frequent releases). The COCOMO model is based around the concept of "delivered SLOC", which implies one point in the history of the project where the product is released. From there on, the main development effort is devoted to maintenance. On the contrary, most Free Software projects deliver releases so frequently that it could be considered as a continuous release process. This process implies the almost continuous stabilizing of the code, at the same time that it evolves. Free Software projects are used to improve and modify their software at the same time that they prepare it for end users.

© Novática and Informatik/Informatique

- Bug reports and fixes. While every proprietary software system needs expensive debugging cycles, Free Software can count on the help of people external to the project, in the form of bug reports, and fixes for them.
- Reuse, evolution, and cross-fertilization of code. It is common in Free Software projects the reuse of code of other Free Software projects as an integral part of the system being developed. It is also common that several projects develop evolutions of the same base system, in many cases with all of them using code of the others all the time. Some of this cases can also happen in proprietary developments, but even in large companies, with many open projects, they are not common, while they are the norm in Free Software projects.
- Distributed development model. Although some proprietary systems are developed by geographically distributed teams, the degree of distributed development found in Free Software projects is orders of magnitude greater. There are exceptions, but usually Free Software projects are carried out by people from different countries, working for different companies, devoting different amount of effort to the project, interacting mainly through the Internet, and in most cases (specially in large projects), the developer team has never physically met.

Some of these factors increase the effort needed to build the software, while some others reduce it. Without analysing in detail the impact of these (and other) factors, the estimation models in general, and COCOMO in particular, are not directly applicable to Free Software development.

Comparison with Size Estimations for Other Systems

To put the numbers shown above into context, we offer here estimations for the size of some operating systems, and a more detailed comparison with the estimations for the Red Hat Linux distribution.

As reported in [Lucovsky 00] (for Windows 2000), [Wheeler 01] (for Red Hat Linux), and [Schneier 00] (for the rest of the systems), this is the estimated size for several operating systems, in lines of code (all numbers are approximations):

- Microsoft Windows 3.1: 3,000,000
- Sun Solaris: 7,500,000
- Microsoft Windows 95: 15,000,000
- Red Hat Linux 6.2: 17,000,000
- Microsoft Windows 2000: 29,000,000
- Red Hat Linux 7.1: 30,000,000
- Debian 2.2: 56,000,000

Most estimations (in fact, all of them, except for Red Hat Linux) are not detailed, and is difficult to know what they consider as a line of code. However, the estimations should be close enough to SLOC counts to be suitable for comparison.

Note also that, while both Red Hat and Debian include many applications, in many cases even several applications in the same category, both Microsoft and Sun operating systems are much more limited in this way. If the more usual applications used in those environments were counted together, their size would be much larger. However, it is also true that all those applications are not developed neither put together by the same team of developers, as is the case in Linux-based distributions.

From these numbers, it can be seen that Linux-based distributions in general, and Debian 2.2 in particular, are some of the largest pieces of software ever put together by a group of developers.

Comparing with Red Hat Linux

The only operating system for which we have found detailed counts of source lines is Red Hat Linux (see "Estimating Linux's Size" and "More Than a Gigabuck: Estimating GNU/Linux's Size"). Since it is also a Linux-based distribution, and the software packages included in Debian and Red Hat distributions are quite similar, the comparison can be illustrative. In addition, since Red Hat Linux is very common, and probably the better known Linux-based distribution, comparing can provide a good context for the reader already familiar with it.

The first data that surprised us when we counted Debian 2.2 was its size compared to Red Hat 6.2 (released in March 2000) and Red Hat 7.1 (released in April 2001). Debian 2.2 was released in August 2000, and is roughly twice the size of Red Hat 7.1 (released about eight months later) and more than three times the size of Red Hat 6.2 (released five months earlier). Some of these differences might be due to different considerations of which packages to include when counting, but they provide a good idea of the relative sizes, even considering these considerations.

The main factor causing these differences is the number of packages included in each distribution: in the case of Debian we have considered 2,630 source packages (with a mean of about 21,300 SLOC per package), while Red Hat 7.1 includes only 612 packages (about 49,000 SLOC per package).

When comparing the largest packages in both distributions, we find in Debian all those included in Red Hat. The same is not true the other way around: several packages that amount a good quantity of SLOC to Debian are not present in Red Hat. For instance, among the 11 largest packages in Debian 2.2, the following are missing from Red Hat 7.1: PM3 (about 1,115,000 SLOC), OSKit (about 859,000 SLOC), GNAT (688,000), and NCBI (591,000). On the contrary, among the 11 largest packages in Red Hat 7.1, none is missing in Debian 2.2.

However, there is a large collection of software packages which is missing in Debian 2.2 and not in Red Hat 7.1: the KDE desktop environment and related utilities. Due to license problems, Debian decided not to include KDE software until after Debian 2.2, when the license for Qt changed to GPL. Therefore, we can say that Debian 2.2 is larger, even missing such a large piece of code as KDE. Just to give an idea, the largest KDE packages in Red Hat 7.1 are kdebase, kdelibs, koffice, and kdemultimedia, which amount for about 1,000,000 SLOC. All of them are missing from Debian. This suggest that should the measures had been made on the current Debian archive (still not officially delivered), the differences would have been greater.

The differences between the same package in each distribution are accountable to the different releases included in them.

© Novática and Informatik/Informatique

For instance, the Linux kernel amounts for 1,780,000 SLOC (release 2.2.19) in Debian 2.2, while the same package it amounts for 2,437,000 SLOC (release 2.4.2) in Red Hat 7.1, or XFree includes 1,270,000 SLOC (release 3.3.6) in Debian 2.2, while the release included in Red Hat 7.1 amounts for 1,838,000 (XFree 4.0.3). This differences in releases make it difficult to directly compare the figures for Red Hat and Debian.

The reader should also note that there is a methodological difference between the study on Red Hat and ours on Debian. The former extracts all the source code, and uses MD5 checksums to avoid duplicates across the whole distribution source code. In the case of Debian, we have extracted the packages one by one, only checking for duplicates within packages. However, the total count should not be very affected for this difference.

6 Conclusions and Related Work

It is important to notice that these counts may represent roughly the whole collection of stable Free Software packages available on GNU/Linux at the time of the Debian 2.2 release (August 2000). Of course, there is Free Software not included in Debian, but when we come to popular, stable and usable packages, most of them have been packaged by a Debian developer and included in the Debian distribution. Therefore, with some care, it could be said that this kind of software amounted for about 60,000,000 SLOC around the summer of 2001. Using the COCOMO model, this implies a cost (using traditional, proprietary software development models) close to 2,000 million USD and effort of more than 170,000 person-months.

We can also compare this count to that of other Linux-based distributions, notably Red Hat. Roughly speaking, Debian 2.2 is about twice the size of Red Hat 7.1, which was released about eight months later. It is also larger than the latest Microsoft operating systems (although, as is discussed in the corresponding section, this comparisons could be misleading).

When coming to the details, some interesting data can be shown. For instance, the most popular language in the distribution is C (more than 70%), followed by C++ (close to 10%), LISP and Shell (about 5%), and Perl and FORTRAN (about 2%). The largest packages in Debian 2.2 are Mozilla (about 2,000,000 SLOC), the Linux kernel (about 1,800,000 SLOC), XFree86 (1,250,000), and PM3 (more than 1,100,000).

There are not many detailed studies of the size of modern, complete operating systems. Of them, the work by David Wheeler, counting the size of Red Hat 6.2 and Red Hat 7.1 is the most comparable. Other interesting paper, with some intersection with this paper is [Godfrey/Tu 00], an study on the evolution over time of the Linux kernel. Some other papers, already referenced, provide total counts of some Sun and Microsoft operating systems, but they are not detailed enough, except for providing estimations for the whole of the system.

Finally, we find it important to repeat once more that we are offering only estimations, not actual figures. They depend too much on the selection of the software to measure, and on some other factors which were already discussed. But we believe they are accurate enough to draw some conclusions, and to compare with other systems.

Acknowledgements

This paper is obviously inspired by "More Than a Gigabuck: Estimating GNU/Linux's Size;", by David Wheeler [Wheeler 01]. We have also used his tool *sloccount*. Without his work, this paper would have been completely unthinkable.

We would also like to thank the comments and suggestions of many Debian developers, which have helped to improve this paper.

Bibliography

[Boehm 81]
Barry W. Boehm, 1981, Software Engineering Economics, Prentice Hall.

[ComWorld 00]
Computer World, Salary Survey 2000, http://www.computerworld.com/cwi/careers/surveysandreports.

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

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

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

[DFSG]
Debian Project, Debian Free Software Guidelines, http://www.debian.org/social_contract#guidelines.

[Godfrey/Tu 00]
Michael W. Godfrey, Qiang Tu, Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo, August 3–4, 2000, Evolution in Open Source Software: A Case Study, 2000 Intl Conference on Software Maintenance http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf.

[Lucovsky 00]
Mark Lucovsky, August 3–4, 2000, From NT OS/2 to Windows 2000 and Beyond – A Software-Engineering Odyssey, 4th USENIX Windows Systems Symposium, http://www.usenix.org/events/usenix-win2000/invitedtalks/lucovsky_html/.

[Schneier 00]
Bruc Schneier, March 15, 2000, Software Complexity and Security, Crypto-Gram Newsletter, http://www.counterpane.com/crypto-gram-0003.html.

[Wheeler 01]
David A. Wheeler, More Than a Gigabuck: Estimating GNU/Linux's Size, http://www.dwheeler.com/sloc.

© Novática and Informatik/Informatique