Chapter I

Volunteers in Large Libre Software Projects:
A Quantitative Analysis Over Time

Martin Michlmayr, University of Cambridge, UK

Gregorio Robles, Universidad Rey Juan Carlos, Spain

Jesus M. Gonzalez-Barahona, Universidad Rey Juan Carlos, Spain

Abstract

Most libre (free, open source) software projects rely on the work of volunteers. Therefore, attracting people who contribute their time and technical skills is of paramount importance, both in technical and economic terms. This reliance on volunteers leads to some fundamental management challenges: Volunteer contributions are inherently difficult to predict, plan, and manage, especially in the case of large projects. In this chapter we present an analysis of the evolution over time of the human resources in large libre software projects, using the Debian project, one of the largest and most complex libre software projects based mainly in voluntary work, as a case study. We have performed a quantitative investigation of data corresponding to roughly seven years, studying how volunteer involvement has affected the software released by the project, and the developer community itself.

Introduction

Volunteer contributions are the basis of most libre[1] software projects. However, the characteristics, and the way of working of volunteers, can be quite different from those of employees who are the main force behind traditional software development. Volunteers can contribute with the amount of effort they want, can commit for the time period they consider convenient, and can devote their time to the tasks they may prefer, if the context of the project makes that possible (Michlmayr & Hill, 2003). But even in this apparently difficult environment, many libre software projects have produced systems with enough quality and functionality to gain significant popularity. Therefore, the fairly unstructured collaboration of volunteers has been demonstrated as a viable software development strategy, even if it is associated with certain challenges related to project management and quality (Michlmayr, 2004). In this chapter we explore how these voluntary contributions evolve over time in one of the largest libre software projects, Debian.

For our purposes, we will define volunteers as those who collaborate in libre software projects in their spare time, not profiting economically in a direct way from their effort. Volunteers can be professionals related to information technologies, but in that case their activity in the libre software project is not done as a part of their professional activity. Although the vast majority of participants in libre software projects comply with our definition, it is important to note that there are also non-volunteers, that is, paid people (normally hired or contracted), who produce libre software. German has studied paid employees from various companies in the GNOME project (German, 2004). He notes that they are usually responsible for less attractive tasks, such as project design and coordination, testing, documentation, and bug fixing. Also, "[m]ost of the paid developers in GNOME were, at some point, volunteers. Essentially for the volunteers, their hobby became their job."

The involvement of volunteers, of course, raises new economic and business model issues that have to be taken into account in commercial strategies around libre software. Collaboration from volunteers is difficult to predict, but if it is given, it may add value to a software system in very economic terms for a software company.

The structure of this chapter will be as follows. In the second section we discuss the nature and, in particular, the tasks performed by volunteers, paying special attention to those who contribute to Debian, the case study investigated in this chapter. Following this section, a set of research questions regarding volunteer participation will be raised. The primary goal of this chapter is to answer these questions based on quantitative data. The methodology for retrieving the quantitative data used in this study is first given. In this section, we also propose a number of measures that will allow us to answer the questions. The results we have obtained as part of this study will be presented and commented on in depth for the Debian project. Finally, conclusions, applicability of the methodology, and further research will be discussed.

The Debian Project and its Volunteers

Debian is an operating system completely based on libre software (Monga, 2004; O'Mahony, 2003). It includes a large number of applications, such as the GNU tools and Mozilla, and the system is known for its solid integration of different software components. Debian's most popular distribution, Debian GNU/Linux, is based on the Linux kernel. Ports to other kernels, such as Hurd and FreeBSD, are in development.

One of the main characteristics of the Debian distribution is that during the whole life of the project it has been maintained by a group of volunteers, which has grown to a substantial number. These individuals devote their own time and technical skills to the creation and integration of software packages, trying to supply users with a robust system which provides a lot of functionality and technical features.

Following our definition of volunteers, all maintainers in Debian are volunteers. Some employers of people who act as Debian maintainers in their spare time permit their staff to devote some of their time to Debian during work hours. Nevertheless, the majority of work by most Debian maintainers is performed in their spare time. In contrast to some projects, such as the Linux kernel and GNOME, there are no Debian maintainers who are paid to work on the system fulltime, even though a number of organizations have a commercial interest in Debian and contribute varying degrees of manpower (and other resources) to the project. For example, a number of regions in Spain have their own operating systems based on Debian; HP made Debian more suitable for large telecom customers, and Credativ provides commercial services for Debian and similar systems.

There are several tasks that volunteers can do in Debian: maintaining software packages, supporting the server infrastructure, developing Debian-specific software, for instance, the installation routine and package management tool, translating documentation and Web pages, and so forth. From all these tasks, we will focus in this chapter on package maintainers, whose task it is to take existing libre software packages and to create a ready-to-install Debian package. Debian maintainers are also called Debian developers, although their task is really not to develop software but to take already developed software for the creation of a Debian package. This, of course, does not mean that a Debian maintainer may not develop and maintain software, but this is not usually the case: the original author (or developer), known as the 'upstream' developer, and the Debian maintainer, are usually, but not necessarily, not the same person.

Besides its voluntary nature, the Debian project is unique among libre software projects because of its social contract (Debian Social Contract, 2006). This document contains not only the primary goals of the Debian project, but also makes several promises to its users. Additionally, there are a number of documents Debian maintainers have to follow in order to assure quality, stability, and security of the

resulting distribution. In particular, Debian's Policy document ensures that the large number of volunteers working independently will produce a well-integrated system rather than merely an aggregation of software packages which do not play together very well (Garzarelli & Galoppini, 2003).

There has recently been some interest in studying how the voluntary status of Debian members affects the quality of the resulting product. Managing volunteer contributors is associated with certain problems that "traditional" software development usually does not confront (Michlmayr & Hill, 2003). It is known that there are some intrinsic problems when the development process is carried out in a distributed fashion (Herbsleb et al., 2001). The situation in Debian and similar projects is even more complex because the development process is not only distributed but also largely based on volunteers. This can lead to certain challenges, such as the unpredictability of the level of their involvement (Michlmayr, 2004).

To some degree, the volatility of voluntary contributors can be limited by the introduction of more redundancy, such as the creation of maintainer teams. The creation of teams and committees for specific purposes, such as management, or for complex tasks has been already reported in other libre software projects (as for instance, German's work on the GNOME project [2004]).

Research Objectives and Goals

Related research has been very active in studying the static picture of a libre software project community over the last years, as can be seen from studies performed on Apache and Mozilla (Mockus et al., 2002), FreeBSD (Dinh-Trong & Bieman, 2005) or GNOME (Koch & Schneider, 2002), leading to models that discuss onion-like structures of libre software projects (Crowston & Howison, 2005). The main goal of this chapter is to introduce the time axis in these kinds of studies, focusing most notably on the contribution of volunteers.

We have therefore set up a list of research questions which we would like to answer for several large libre software projects. In the case of Debian, we have additional information that permits us to link the work done by a volunteer with a piece of software, so we can, for example, study what has happened to packages from volunteers who have left the project. In the following, the set of questions that we are raising will be answered in detail for the Debian project.

The specific questions we aim to answer with this chapter are the following:

a. **How many volunteers does the project have, and how does this number change over time?** This will provide us with some basic data, useful when

working with subsequent questions. When we started the study, we expected a steady increase of volunteers over time, as it is already known that the number of packages included in the system has been growing in that way (Gonzalez-Barahona et al., 2004). In addition, we will try to find out if the work ratio (measured as activity or output per developer) has increased over time or not.

b. **How many volunteers from previous releases remain active?** We want to measure the volatility of the volunteers in large libre software projects. That is, do volunteers join the project and work on it for short periods of time, or, on the contrary, do they stay for many years? Specifically, we want to calculate the half-life of contributors of the project. The half-life is defined as the time required for a certain population of maintainers to fall to half of its initial size. This figure could be easily compared with other libre software projects and, of course, with statistics from companies from software and other industries.

c. **What is the activity of volunteers who remain in subsequent releases?** Answering this question will allow us to know if "older" volunteers strengthen their contributions as time passes, contributing more to the project, or whether they become less active. There are two possible hypotheses one could propose. On the one hand, those volunteers who have been involved for a long time may be very experienced and therefore more efficient in their work than less experienced developers. On the other hand, young developers may have more time or energy to devote to the project and therefore contribute more. Both theories are possible and mutually compatible.

d. **What happens to packages maintained by volunteers who leave the project?** Our intention is to see if we can find a regeneration process in libre software projects that allows them to survive the loss of some of their human resources.

   Since Debian maintainers are volunteers, they may quit the project at almost any time, leaving their packages unmaintained. There are two possible outcomes regarding the future of those packages: First, they can be taken over (adopted) by another maintainer. Alternatively, if nobody is interested in adopting them, they will eventually be removed from the archive and excluded from future stable releases. Such removals of unmaintained packages are part of Debian's Quality Assurance effort. Our intention was to know how this inherent characteristic of the voluntary contributors affects Debian, and how this is damped down by other (possibly new) maintainers.

e. **Are more "important" and commonly used packages maintained by more experienced maintainers?** It can be interesting to know whether packages which are considered crucial for the functioning of the system are maintained primarily by volunteers who have more experience. For this, we have considered the most used packages as the targets of the study. We have defined as crucial

packages those which are usually installed on every system, as, for instance, the base system which in the case of the Debian GNU/Linux operating system is composed, among others, of the Linux kernel and the GNU tools. This does not necessarily mean, of course, that crucial packages are more difficult to maintain than other packages, but as they are used by all users of the system and the rest of the software heavily depends on their proper functioning, these packages have to be maintained with special care. Data about the importance of each package will be obtained from the Debian Popularity Contest[2] that tracks how many people have installed a given package.

Methodology and Sources of Data

Debian consists of four parallel versions (stable, testing, unstable, experimental) which can be downloaded from the Internet. The focus of this study is on the stable versions from Debian 2.0 onwards, up to version 3.1, which provide good snapshots of the history of the distribution. These releases comprise a period of time from July 1998 to June 2005. There have been releases of Debian before 1998 (Lameter, 2002), but they have not been taken into consideration for this study since the sources of data we have used in this study were not available for them. For each release, we have retrieved the corresponding Sources.gz file (see next section) from the Debian archive. We have then extracted information about packages and their maintainers from this file and stored the results in a database. After that, we performed some semi-automatic cleaning and massaging of the data that will be explained in more detail below. Final results were obtained through queries to the database, and correlations that have been implemented by another set of scripts.[3]

In addition to the analysis of official releases, we have enriched the findings by additionally taking a more fine-grained data source into account. While releases are only done occasionally, in the case of Debian with years between releases, uploads to the Debian archive are done on a continuous basis. We have analyzed the activity related to these uploads to clarify some of the findings of the paper on which this chapter is based (Robles et al., 2005). The estimations of the size of the releases have been done using a software that counts source lines of code and avoids double-counting the code included in various packages (this methodology is described in detail (Gonzalez-Barahona et al., 2001)), using previously published data (Gonzalez-Barahona et al., 2004), except for release 3.1, which was calculated specifically for this study. The data related to the importance of packages has been retrieved from the Debian Popularity Contest (see section on this topic).

Debian Sources File

Since version 2.0, the Debian repository contains a Sources.gz file for each release, listing information about every source package in it. Every source package contains the name and version, list of binary packages built from it, name and e-mail address of the maintainer, and some other information which is not relevant for this study. As an example, see an excerpt of the entry for the mozilla source package in Debian 2.2 below. It can be seen how it corresponds to version M18-3, provides four binary packages, and is maintained by Frank Belew.

[...]
Package: mozilla
Binary: mozilla, mozilla-dev, libnspr4, libnspr4-dev
Version: M18-3
Priority: optional
Section: web
Maintainer: Frank Belew (Myth) <frb@debian.org>
Architecture: any
Standards-Version: 3.2.0
Format: 1.0
Directory: dists/potato/main/source/web
Files: 57ee230b97ccc69444ccccd0bc66908a 719 mozilla_M18-3.dsc
532934635ad426255036ee070bad03c8 28642415 mozilla_M18.orig.tar.gz
3adf83de7e74bf940ee02c0deca20372 18277 mozilla_M18-3.diff.gz
[...]

Debian Popularity Contest

The Debian Popularity Contest is an attempt to map the usage of Debian packages. Its main goal is to know what software packages are actually installed and used. Information from the Popularity Contest is used by Debian, for example, to decide which software to put on the first CD.

The system functions as follows: Debian users may install the popcon package, which sends a message every week with the list of packages installed on the machine as well as the access time of some files which may give hints regarding the last usage of these packages. Of course, privacy issues are considered in a number of ways: Upon installation, users are explicitly asked if they want to send this information to Debian, and the server which collects the data anonymizes it as much as possible.

Table 1. Debian Popularity Contest statistics

rank	name	inst	vote	old	recent	no-files	(maintainer)
1	adduser	6881	6471	94	316	0	(Adduser Development)
2	debianutils	6881	6517	50	314	0	(Clint Adams)
3	diff	6881	6425	261	195	0	(Santiago Vila)
4	e2fsprogs	6881	5448	825	608	0	(Theodore Y. Ts'o)
5	findutils	6881	6449	233	199	0	(Andreas Metzler)
6	grep	6881	6436	126	319	0	(Ryan M. Golbeck)
7	gzip	6881	6558	245	78	0	(Bdale Garbee)
8	hostname	6881	6112	715	54	0	(Graham Wilson)
9	login	6881	6407	56	418	0	(Karl Ramm)
10	ncurses-base	6881	56	143	6	6676	(Daniel Jacobowitz)

The resulting statistical information of all users participating in this scheme is publicly available on the Web site of the project. For every package, it includes the number of machines on which it is installed (inst), the number of machines which make regular use of that package (vote), the number of machines with old versions of the package (old), the number of recent updates (recent), the number of machines where not enough information is available (no-file), and the maintainer of the package. Below is an excerpt of the available data, in this case the top ten packages ordered by installations as of December 4, 2004. The first 66 packages are installed on all machines, with 6881 installations.

Debian Developer Database

From June 1999 onwards, Debian has held a database (http://db.debian.org) with data related to members of the project. Some information, such as the full name and user name, can be retrieved publicly through the Internet. This database also contains information about the digital keys used by a developer. Debian makes use of digital signatures through the use of the tools PGP (Pretty Good Privacy) and GPG (GNU Privacy Guard) to approve uploads to their software archive. The use of digital signatures provides two guarantees. First, the signature will show that the package comes from a trusted source, that is, from an official Debian developer whose PGP or GPG key is stored in the Debian keyring and this developer database. Second, by verifying the signature on the package, it can be ensured that the package has not been tampered with during the process of uploading it to the Debian archive.

Package Uploads

While the main data source of this chapter is the Sources.gz files from the last official Debian releases, we have additionally taken uploads to the archive into account to answer some of the research questions with more detail. As mentioned above, Debian's archive is separated into various branches. The official releases are known as the stable branch, whereas Debian's development tree is known as "unstable." Even though it is said that the development tree is usually fairly stable, it is where major development occurs and as such, major bugs are introduced from time to time.

When an upload is made to unstable, a summary of the changes is automatically sent to a mailing list known as "debian-devel-changes." By extracting data from the archives of this list, the uploads made to the Debian archive over the last few years can be studied. The archive of this mailing list starts towards the end of August 1997. Because August is not complete, we use the data starting with September, since having data for full months allows for better comparisons. We then divided the whole time period into periods of three months each, leading to 34 periods which cover 8.5 years (102 months), from September 1997 leading up to February 2006. In total, slightly more than 181,500 uploads have been observed in this period, leading to a mean number of uploads of around 1,800 each month.

The method used to extract this information is as follows: first, the archives of the "debian-devel-changes" mailing list are downloaded from Debian. These are then parsed, leading to one message for each upload. These messages are signed by the developer's PGP or GPG key as described above. The information from these digital signatures is then used to map each upload to a unique user id corresponding to the developer who made the upload. The mapping from PGP or GPG key to user name is obtained from Debian's developer database and missing entries for old developers are manually supplemented. After this information is obtained, information about the developer (user name) and the date of each upload is stored in a database together with the message id from the posting stored in the archive.

The extraction of this data leads to fairly precise results but there are some limiting factors. First of all, it is important to take into account that uploads are not a measure of effort. We use the data as an indication of activity of a developer but the information is not rich enough to give specific information about how much effort was involved with a specific upload. Second, for the last few years Debian had the concept of "sponsorship," whereby an official Debian developer would upload a package created by a prospective developer (who is not part of Debian yet and whose GPG key is therefore not recognized). In such cases, the main effort was done by the prospective developer but the signature shows the name of the official Debian developer. Since we are concerned with activity of developers and not with effort, this is not an obstacle but it has to be considered during the interpretation of the data.

Finally, while the archive software used by Debian now sends automatic notifications of new uploads to the "debian-devel-changes" list, this was done manually in the past. Therefore, data from the past can be slightly unreliable. For example, we observed a number of messages which were not signed by PGP or GPG. Out of about 185,000, we have only detected about three thousand uploads for which no information about the developer could be extracted automatically. We believe that these are not significant and that data from uploads greatly enriches the findings from studies using the Source.gz file from official Debian releases.

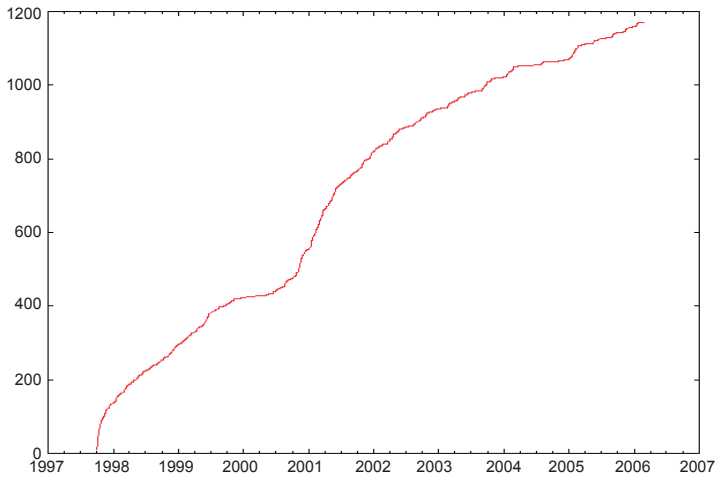
Evolution of the Number of Debian Maintainers

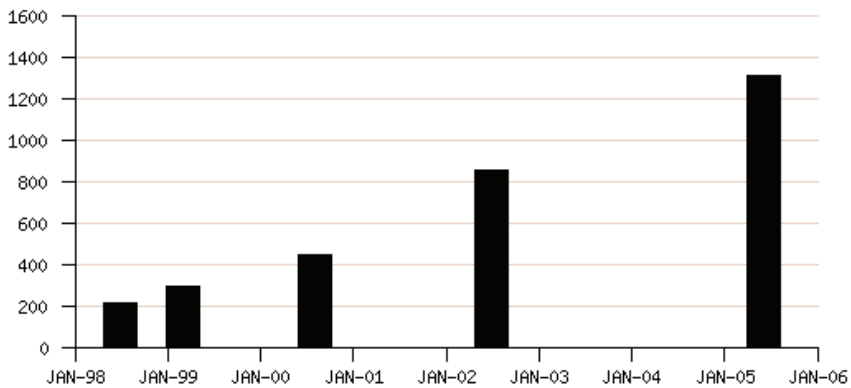
The information of the evolution of the number of Debian maintainers will provide us with some basic data useful when working with subsequent research questions. When we started the study, we expected a steady increase of maintainers over time, as it is already known that the number of packages included in the system has been growing linearly (Gonzalez-Barahona et al., 2004). In fact, we expected the packages-to-maintainer ratio to be nearly constant, since it seems reasonable to consider that volunteers devote similar amounts of effort over time, which would lead to a constant number of packages per maintainer.

Figure 1 shows on the left side the evolution of the number of Debian maintainers for the latest five stable releases. As we have expected, the number of Debian individual maintainers has been growing over time. Debian 2.0 (July 1998) was put together by 216 individual maintainers, while the number of maintainers for later releases are 859 for 3.0 (July 2002) and 1,314 for 3.1 (June 2005). This shows a growth of about 35 percent every year. The right side of the figure shows the cumulative number of developers who have made uploads to Debian (starting as of September 1997 and finishing with February 2006). This chart gives more precise information as to the growth over time but it does not include some information which the other chart captures. As mentioned above, only official Debian developers are considered and therefore the numbers are lower than in the chart on the left side which considers all maintainers of packages in Debian, regardless of their official status. The more detailed information the chart on the right conveys is very interesting, though. A remarkable stagnation of the growth can be observed. This is because the New Maintainer process, Debian's admission process, was stopped for several months at the end of 1999. The growth continues in the middle of 2000 and, interestingly enough, the pause in the admission of the developers did not have any significant effect on the overall growth of the project.

Based on the data from official releases, we have conducted a small statistical analysis; the results are shown in Table 2. The ratio of packages per maintainer

Figure 1. Number of maintainers over time. The left chart is based on Debian releases while the right one is based on continuous uploads.





(see column “Pkg/Maint”) grows over time, contrary to our initial hypothesis. The growth of packages is actually bigger than that of volunteers who contribute to the project. There are some possible explanations for this finding. First, it is possible that improvements of the development tools or in the practices employed have led

Table 2. Statistical analysis of the growth in number of Debian maintainers

Date	Release	Maint	Packages	Pkg/Maint	Median	Mode	Std. Dev	Gini	Max
Jul 98	2.0	216	1,101	5.1	3	1 (52)	5.8	0.492	50
Mar 99	2.1	296	1,559	5.3	3	1 (76)	6.5	0.521	55
Aug 00	2.2	453	2,601	5.7	3	1 (122)	7.4	0.535	69
Jul 02	3.0	859	5,119	6.0	4	1 (208)	8.2	0.539	79
Jun 05	3.1	1,314	7,989	6.1	3	1 (386)	9.1	0.577	127

to an increase in the efficiency of developers. Second, due to increased interest in libre software, the development speed in general has accelerated and volunteers are more committed.

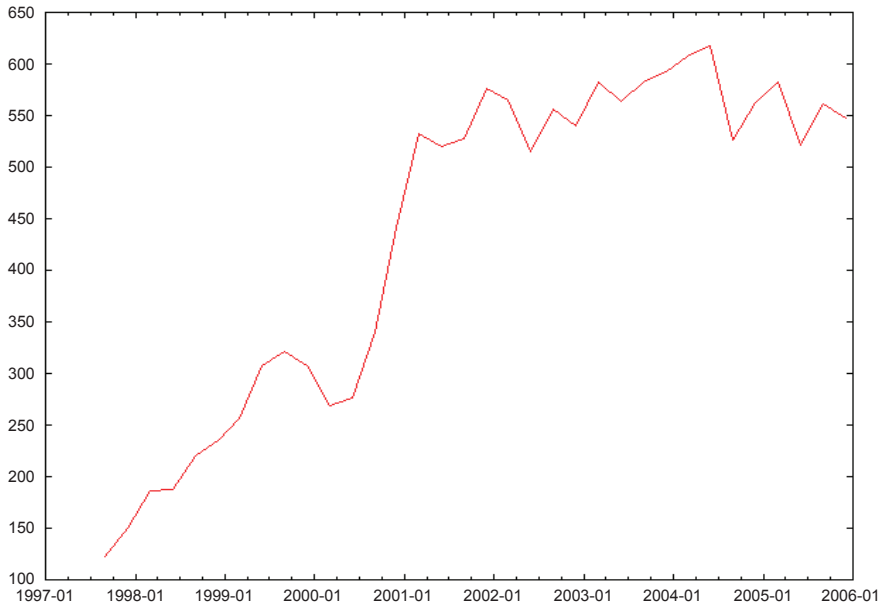
Interesting enough, the median does not vary (with the exception of Debian 3.0) over time in these last years. Half of the maintainer population does not have more than three packages to maintain. Furthermore, the mode shows that the most frequent situation is a maintainer who is in charge of one package. In brackets we can find the number of developers who actually maintain only one package, which is around one forth of the total population of Debian maintainers.

The next three values (the standard deviation, the Gini coefficient,[4] and the maximum number of packages maintained by a single maintainer) strengthen the idea that the distribution of work tends to be distributed in a more unequal way, with a small number of maintainers maintaining more and more packages while the number of packages the vast majority is in charge of does not change much. Compared to other libre software applications and, in general, to other studies which have looked at the distribution of work in libre software projects (Ghosh &d Prakash, 2000; Koch & Schneider, 2002; Hunt & Johnson, 2002; Mockus et al., 2002; Ghosh et al., 2002), we can see that, unlike other projects, Debian is far away from a Pareto distribution. In terms of the Gini coefficient, Debian shows values from roughly 0.5 to 0.6 while studies of the activity in CVS repositories of other projects have found Gini to be in the range between 0.7 and 0.9 (Robles, 2006).

Finally, in Figure 2 we see the number of individual developers making uploads to the Debian archive for each three month period. A significant growth in the number of contributors can be seen in the first few years of the observed period. Since roughly the beginning of 2001, a fairly constant number of individuals makes contributions to Debian. There are about 550 unique contributors, even if they change over time.

The first column gives the date of the release specified in the second one. “Maint” is the number of maintainers that maintain at least a package, “Packages” the number of total packages for that release, “Pkg/Maint,” the mean number of packages per maintainer, “Median” the median number of packages, “Mode” gives the most

Figure 2. Mean number of developers making uploads per each three month period



frequent contribution in number of packages and in brackets the number of maintainers who contribute to it, "Std. Dev," the standard deviation of our sample," "Gini" the Gini coefficient, and "Max" the maximum number of packages that a unique maintainer is responsible for.

Tracking Remaining Debian Maintainers

At the time of the release of Debian 2.0 in July 1998 there were 216 voluntary developers contributing to Debian. We have studied how the involvement of these 216 contributors to Debian 2.0 has changed over time. Table 3 gives an overview of the number of contributors from the original group left at each release, as well as the number of packages maintained by them. As the figure shows, the number decreases steadily, with only 117 of the original 216 contributors (54.2%) still possessing ownership of a package in June 2005. Based on these figures, we concluded in a previous paper that the half-life value had not been reached after six and a half years and estimated that the half-life value would be around 7.5 years (or 90 months) (Robles et al., 2005).

Taking the more fine-grained information from uploads into account, we can now revise these findings. Taking package ownership as an indication for activity is error prone, since it has been shown that a number of "maintainers" are actually inactive and do not maintain their packages (Michlmayr, 2004). It can take several months or longer until the situation is resolved, in particular if maintainers are busy but do not want to admit to themselves that they do not actually have enough time anymore. Uploads are therefore a much better measure in this case since they show activity. While this measure does not show the effort done by a maintainer it shows that they are still active, which is the question being asked here.

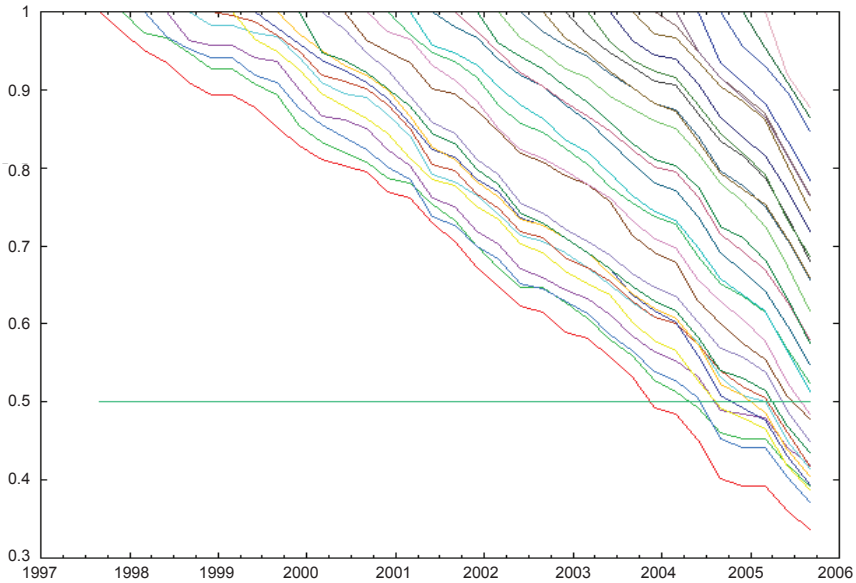
Based on data from uploads, we can see that as of the three-month period starting in June 1998, only 187 out of the original 216 contributors (which still possess packages in the release done in July 1998) are still active. Taking these 187 developers as the new population, we find that the group reaches their half-life in the periods between June and September 2004. This leads to a half-life value of less than 78 months (6.5 years). This is in line with the originally estimated value of 7.5 years, a slight over-prediction due to the fact that it takes some time until inactivity is reflected in the maintainer field of a package. The value obtained from this population is also in line with those obtained from other populations observed as part of the investigation of uploads to Debian, as can be seen in Figure 3. They all show a half-life of between 75 and 90 months.

It would be interesting to perform further analysis about which factors influence how long volunteers remain active. There is already evidence that some volunteers face feelings of burn-out (Hertel et al., 2003), but further studies into human-resource management and motivation in libre software projects could have positive effects on extending the half-life of volunteer contributions.

The number of packages for which these developers are responsible is also interesting. The initial number of packages maintained by the 216 contributors of Debian 2.0 was 1,101. The corresponding number of packages in Debian 2.1 (around nine months later) for the developers remaining rose to 1,351 and then to 1,457 for Debian 2.2, where the maximum number of packages was achieved. Then it decreased to 1,305 for Debian 3.0, and in the last Debian version it had similar figures as in their first release, although now with half of the maintainers.

This data shows that there has been a continuous increase in the mean number of packages that maintainers are responsible for. While the number of packages per maintainer was slightly above five for the 2.0 release, this number has grown to nine packages per maintainer in release 3.1. It also seems that the mean value keeps on growing, although at a lower pace, and that it has a tendency towards a value around nine as we can see from the last two releases. Given the large amount of time between the last two releases, we can assume that this observed pattern is stable. We have already discussed possible explanations for this behavior with the data about the evolution in number of Debian maintainers (see Table 2).

Figure 3. Half-life of Debian maintainers: How populations shrink over time. The horizontal line at 0.5 shows when a population reaches half-life



Regarding the involvement of maintainers, we can see from the median that there is a general shift towards maintaining more packages, as the median value starts with three packages and raises up to five for Debian 2.2 and Debian 3.0. The mode, on the other hand, shows that the number of maintainers who only maintain one package decreases over time more quickly than the number of total maintainers (the total number of maintainers drops from 216 to 117, a drop of 46%, while the number of maintainers who maintain only one package decreases from 52 to 20, a 62% drop; this means that the number of maintainers with more than one package shrinks from 164 to 97, which is only 34% less, almost half of the drop for maintainers in charge of a single package). The cause for this may be twofold: On one hand those maintainers could have left the project, and on the other they could have gotten more involved in it by maintaining more packages. In any case, maintaining one package could be seen as a "hot" zone in which nobody stays for a long time and where a decision has to be taken: to get more involved in the project or to leave.

The standard deviation and the Gini coefficient give an idea of the distribution of work. Both values show that there is tendency to have a less equally distributed load of work. Of particular interest is the Gini coefficient, which starts at almost 0.5 and grows up to 0.574. The maximum number of packages that a single maintainer is in charge of grows consequently, from 50 packages in Debian 2.0 to 83 in Debian 3.1. It should be noted that the maximum number of packages of the first three Debian versions under study correspond to a different person than the last two.

Table 3. Packages maintained by the Debian 2.0 maintainers

Date	Release	Maint	Packages	Pkg/Maint	Median	Mode	Std. Dev	Gini	Max
Jul 98	2.0	216	1,101	5.1	3	1 (52)	5.8	0.492	50
Mar 99	2.1	207	1,351	6.5	4	1 (38)	7.3	0.501	55
Aug 00	2.2	188	1,457	7.8	5	1 (33)	9.2	0.515	69
Jul 02	3.0	147	1,305	8.9	5	2 (20)	10.6	0.540	65
Jun 05	3.1	117	1,055	9.0	4	1 (20)	12.1	0.574	83

Investigating Maintainer Experience

In the previous paragraphs we tracked maintainers from Debian version 2.0 over time to see how their contributions evolved. In the following we are going to do the opposite; we will take the last Debian release (Debian 3.1) and will try to track when maintainers first started participating in the project. This will allow us to have a measure of experience in the project. Maintainers who entered in the same release will be grouped and analyzed together.

Figure 4 shows when currently active maintainers got involved in the project. For every maintainer of a package in the latest release, we have investigated in which release their first contribution can be found. In addition to the 117 developers who have made steady contributions since July 1998 (release 2.0), 55 participants got involved before Debian 2.1, and 106 arrived with Debian 2.2. In the last two stable releases, 384 and 652 new maintainers have been identified.

The evolution of the number of packages per maintainer given in Table 4 provides evidence about the impact of experience on the number of packages maintained.

Figure 4. First stable release to which Debian 3.1 maintainers have contributed

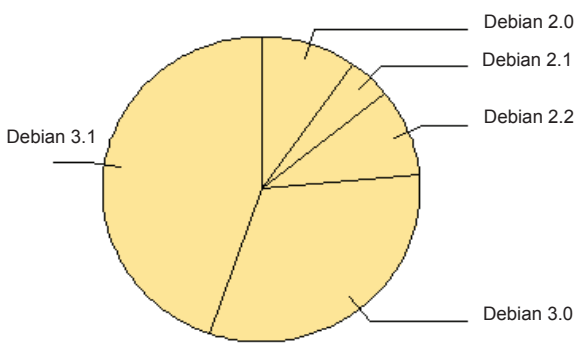


Table 4. First release as maintainer for maintainers in Debian 3.1

Date	Releale	Maint	Packages	Pkg/Maint	Median	Mode	Std. Dev	Gini	Max
Jul 98	2.0	117	1,055	9.0	4	1 (20)	12.1	0.574	83
Mar 99	2.1	55	631	11.5	6	5 (8)	15.1	0.544	81
Aug 00	2.2	106	1,008	8.8	6	1;2 (16)	9.7	0.515	55
Jul 02	3.0	384	2,835	7.2	5	1 (63)	9.5	0.511	121
Jun 05	3.1	652	2,221	4.0	2	1 (246)	7.4	0.570	106

We can see that for the first three versions considered, the values range around 9.0 up to 11.5 packages per maintainer, while in the last two the number of packages is lower. In general, the tendency is that older maintainers have more packages than those who joined later. The exceptions are maintainers who joined with Debian 2.1. For this version, we can see a statistical distortion in the mode as it has a value of 5 while the values for all other versions is 1 (or 1 and 2 in the case of Debian 2.2).

With regard to the median value, we can see that it is also higher for more experienced maintainers, although in this case it is not that clear as we have seen with the mean number of packages. The different behavior of the last two releases is interesting: While Debian 3.1 has a median of two with many (up to 246) maintainers only in charge of one package, those maintainers who entered in Debian 3.0 and who are still active, have a median value of five and a smaller proportion of them maintain a single package. Again, this supports our previous conclusion that maintaining a single package is only a temporary situation.

The standard deviation of the sample does not give us much information in this case. Maybe it stresses the distorted behavior of Debian 2.1 with such a high value; interestingly enough it shows that the data is more homogeneous as we come nearer to Debian 3.1. This is an expected effect, as "younger" maintainers should have a similar (smaller) involvement while "older" ones may vary more. Nonetheless, the Gini coefficient does not necessarily support this finding as the values show no clear tendency over time (the highest value is for Debian 2.0 followed closely by Debian 3.1). This is also the case for the maximum number of packages maintained by a single person which fluctuates from version to version without any predictable direction. In any case, we can see that very active maintainers enter any time in the project, some of them with a surprisingly high involvement. For instance, from July 2002 to June 2005 one new maintainer became in charge of 106 packages! Obviously, the effort needed for the maintenance of a package can vary widely. Further exploration is needed to estimate the effort associated with the maintenance of these packages.

Table 5. Orphaning and adoption of packages

Release 1	Release 2	Orphaned	Adopted	Adopt/Orph	Orph/Total1	Orph/Total2
2.0	2.1	15	14	93.3%	1.3%	1.0%
2.0	2.2	61	40	65.6%	5.5%	1.5%
2.0	3.0	231	171	74.0%	21.0%	4.5%
2.0	3.1	385	253	65.7%	35.0%	4.8%
2.1	2.2	47	31	66.0%	3.0%	1.8%
2.1	3.0	302	220	72.8%	19.4%	5.9%
2.1	3.1	516	332	64.3%	33.1%	6.5%
2.2	3.0	281	207	73.7%	10.8%	5.5%
2.2	3.1	685	433	63.2%	26.3%	8.6%
3.0	3.1	685	435	63.5%	13.4%	8.6%

Note: Each row shows packages present in the older release (first column) and not in the newer ("Orphaned" column), and which of those were adopted. The last columns show the percentages of package "saved" (adopted to orphaned, Adopt/Orph), and orphaned in the newer release to total in the older (Orph/Total1) and newer (Orph/Total2) releases.

Packages of Maintainers who Left the Project

When maintainers leave the project, their packages become unmaintained (the Debian project uses the expression "orphaned"). These packages may be taken up by others ("adopted" in the Debian jargon), or they will not be present in the next stable release. In Table 5 the ratios and numbers of orphaned and adopted packages between any pair of the studied releases are shown.

The table should be read as follows: The first column shows the number of packages, 15, for which their maintainers have left the project (column "Orphaned") and that have been adopted, 14, by other (possibly new) maintainers (column "Adopted") from Debian 2.0 to Debian 2.1. This means that the percentage of orphaned packages that have been adopted is 93.3% (column "Adopt/Orph"), so in this case few packages got lost. The last two columns help situating the amount of orphaned packages we are talking about, giving the share of orphaned packages in comparison to the total number of packages for each release.

Looking at the rest of the rows in the table, we can see that the percentage of adopted packages is very high: more than 60% for all releases considered. This happens even for releases with a very high portion of orphaned packages (for instance, between version 2.0 and 3.1). In other words, even though maintainers who left Debian between July 1997 and June 2005 were responsible for 35.0% of the packages in Debian 2.0, 65.7% of these packages can still be found in version 3.0. We can thus

affirm that Debian counts on a natural "regeneration" process for its voluntary contributors and that there is a high probability that the packages of a maintainer who leaves the project will be adopted by others.

Another interesting fact is that the ratio of adopted to orphaned packages is decreasing in later releases. This means that the number of orphaned packages grows more quickly than that of adopted, i.e., there are some packages missing in every new release. If a package is unmaintained and falls off the next release, it will probably not enter a future one. In this study we have only considered removed packages from maintainers who left the project, but it is likely that some software will also be abandoned by maintainers who still remain active and are therefore not covered by this study.

In any case, it should be noted that users are left unsupported when a package (maybe providing a unique functionality) from a previous release is not present in subsequent ones. It may therefore be beneficial to establish mechanisms to ensure that only packages which can be supported in the long term will not be introduced in the first place, or that at least that they be introduced only in a section of the Debian repository which is clearly marked as being less supported.

Experience and Importance

We have used data from the Debian Popularity Contest (presented in detail in the section Debian Popularity Contest) to find out whether more "important' packages are maintained by more experienced volunteers. Table 6 shows the data corresponding to installations and use of packages by developers which are still in the project, and which were already present in the studied releases. In it we can see, for instance, that Debian 2.0 and 3.1 have 117 common maintainers, who are responsible for 1,091 packages which have been installed 1,305,907 times and 576,991 that are regularly used.

Table 6. Installations and regular use of packages

Release	CMaint	CPkg	Installations	Votes	Inst/Maint	Votes/Maint
2.0	121	1,091	1,305,907	576,991	10792.6	4768.5
2.1	176	1,722	1,584,413	673,236	9002.3	3825.2
2.2	290	2,730	2,217,199	885,448	7645.5	3053.3
3.0	683	5,565	3,923,753	1,405,322	5744.9	2057.6
3.1	1315	7,989	5,248,869	1,711,496	3991.5	1301.5

The CMaint column shows how many maintainers Debian 3.1 had in common with the release in the first column, while the CPkg shows the number of packages maintained by them. Columns Installations and Votes give the sum of the packages installed and voted (used regularly) for those packages maintained by common maintainers. The last two columns show the ratios of both to common maintainers.

If we take the number of installations per maintainer and the number of regularly used packages per maintainer ("Votes/Maint') we can answer the question we proposed in the section Research Objectives and Goals. According to our hypothesis, these ratios decrease over time, which would mean that more experienced volunteers maintain packages which are installed and used more often. In fact, this can be observed through all Debian releases. An alternative (or complementary) explanation to our initial hypothesis is that many of the essential components of the Debian system were introduced in the first releases, and that new packages are mostly add-ons and software that are not installed and used that often.

Conclusion and Further Work

We have conducted a quantitative study of the evolution of the Debian maintainership over the last six-and-a-half years. We have retrieved and analyzed publicly available data in order to find out how Debian handles the volatility of the volunteers who made it happen.

Some of the most interesting findings are:

- Both the number of Debian maintainers and the number of packages per maintainer grow over time.
- The number of maintainers from previous releases who remain active is very high, with an estimated half-life of around 6.5 years (78 months). Slightly less than half of the maintainers from Debian 2.0 still contribute to the current release after more than 7 years.
- Developers tend to maintain more and more packages as they gain experience in the project.
- However, this does not mean that maintainers who have been in the project for more time maintain more packages than newer maintainers. In fact, in the latest release, the highest packages per maintainer ratio is shown by those entering the project around the year 2000.

From these facts, it can be said that Debian maintainers tend to commit to the project for long periods of time. However, there is a worrisome trend towards a higher and

higher ratio of packages per maintainer, which could imply scalability problems as the number of packages in the distribution increases, if the project doesn't admit a proportional number of developers.

Another issue on which we have focused is what happens to those packages that were maintained by developers who left the project. Most of them are taken over by other maintainers, so that we can state that a natural "regeneration" exists. Based on the data we have researched, those packages which are not adopted by other maintainers in the next release, and are therefore not present in it, are unlikely to be re-introduced in future releases.

Finally, we have also found that more experienced maintainers are responsible for packages which are installed more often and used more regularly.

In addition to the new insights gained in this investigation, we have proposed a number of further studies to elaborate on the findings of the present chapter. In particular, team maintenance and its impact on the quality of packages would be interesting to research. It is also not clear why there is an increase in the ratio of packages per maintainer. Possible explanations are that better tools and practices lead to more efficiency, or that with the success of libre software, new volunteers show more motivation and commitment, but more data is needed before these explanations can be conclusive.

From a more general point of view, this study explores the behavior of volunteers in libre software projects and provides some answers as to why these kinds of voluntary contributions are capable of producing such large, mature and stable systems over time, even when the project has no means for forcing any single developer to do any given task and when members may leave the project during important development phases. It is impossible to infer the behavior of volunteer developers just from the study of a single project, but given the size and relevance of the Debian project, at least some conclusions can be exposed as hypotheses for validating in later research efforts.

One of them is the stability of volunteer work over time. The mean life of contributors in the project is probably longer than in many software companies, which would have a clear impact on the maintenance of the software (it is likely that developers with experience in a module are available for its maintenance over long periods of time). Another one is that volunteers tend to take over more work with the passing of time if they remain in the project: In other words, they voluntarily increase their responsibilities in the project. Whether this is because it is easier for them because of their experience, or because they devote more effort to the project, is for now an open question. Yet a third one is the stability of the voluntary effort when some individuals leave the project: Most of their work is taken over by other developers. Therefore, despite being completely based on volunteers, the project organizes itself rather well with respect to drop-outs, which is an interesting lesson about how the project can survive in the long term.

As a final summary, we have found that given that there are no formal ways of forcing a developer to assume any given task, voluntary efforts seem to be more stable over time, and more reliable with respect to individuals leaving the project than we had initially expected.

Acknowledgment

The work of Martin Michlmayr has been funded in part by Google, Intel, and the EPSRC. The work of Gregorio Robles and Jesus M. Gonzalez-Barahona has been funded in part by the European Commission under the CALIBRE CA, IST program, contract number 004337. We would also like to thank the anonymous reviewers for their extensive comments.

References

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*, *10*(2).

Debian Social Contract. (2006). Debian social contract. Retrieved from http://www.debian.org/social_contract

Dinh-Trong, T. T., & Bieman, J. M. (2005). The FreeBSD project: A replication case study of Open Source development. *IEEE Transactions on Software Engineering*, *31*(6), 481-494.

Garzarelli, G., & Galoppini, R. (2003). *Capability coordination in modular organization: Voluntary FS/OSS production and the case of Debian GNU/Linux.*

German, D. (2004). The GNOME project: A case study of open source, global software development. J*ournal of Software Process: Improvement and Practice*, *8*(4), 201-215.

Ghosh, R. A., Glott, R., Krieger, B., & Robles, G. (2002). Survey of developers (Free/libre and open source software: Survey and study). Technical report, International Institute of Infonomics, University of Maastricht, The Netherlands. Retrieved from http://www.infonomics.nl/FLOSS/report

Ghosh, R. A., & Prakash, V. V. (2000). The orbiten free software survey. *First Monday*, *5*(7). Retrieved from http://www.firstmonday.dk/issues/issue5_7/ghosh/

Gonzalez-Barahona, J. M., Ortuno Perez, M. A., de las Heras Quiros, P., Centeno Gonzalez, J., & Matellan Olivera, V. (2001). Counting potatoes: The size of Debian 2.2. *Upgrade Magazine*, *II*(6), 60-66.

Gonzalez-Barahona, J. M., Robles, G., Ortuno Perez, M., Rodero-Merino, L., Centeno Gonzalez, J., et al., (2004). Analyzing the anatomy of GNU/Linux distributions: Methodology and case studies (Red Hat and Debian). In S. Koch (Ed.), *Free/open source software development* (pp. 27-58). Hershey, PA: Idea Group Publishing.

Herbsleb, J. D., Mockus, A., Finholt, T. A., & Grinter, R. E. (2001). An empirical study of global software development: Distance and speed. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering* (pp. 81-90).

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy*, *32*(7), 1159-1177.

Hunt, F., & Johnson, P. (2002). On the Pareto distribution of open source projects. In *Proceedings of Open Source Software Development Workshop*, Newcastle, UK.

Koch, S., & Schneider, G. (2002). Effort, cooperation and coordination in an open source software project: GNOME. *Information Systems Journal*, *12*(1), 27-42.

Lameter, C. (2002). *Debian GNU/Linux: The past, the present and the future.* Retrieved from http://telemetrybox.org/tokyo/

Michlmayr, M. (2004). Managing volunteer activity in free software projects. In *Proceedings of the USENIX 2004 Annual Technical Conference, FREENIX Track*, Boston (pp. 93-102).

Michlmayr, M., & Hill, B. M. (2003). Quality and the reliance on individuals in free software projects. In *Proceedings of the 3rd Workshop on Open Source Software Engineering*, Portland, OR (pp. 105-109).

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, *11*(3), 309-346.

Monga, M. (2004). From bazaar to kibbutz: How freedom deals with coherence in the Debian project. In *Proceedings of the 4th Workshop on Open Source Software Engineering*, Edinburg, Scotland, UK.

O'Mahony, S. (2003). Guarding the commons: How community managed software projects to protect their work. *Research Policy*, *32*, 1179-1198.

Robles, G. (2006). *Empirical software engineering research on libre software: Data sources, methodologies and results.* PhD thesis, Universidad Rey Juan Carlos.

Robles, G., Gonzalez-Barahona, J. M., & Michlmayr, M. (2005). Evolution of volunteer participation in libre software projects: Evidence from Debian. In

Proceedings of the 1st International Conference on Open Source Systems, Genoa, Italy (pp. 100-107).

Endnotes

1 In this chapter we will use the term "libre software" to refer to any software licensed under terms compliant with the Free Software Foundation definition of "free software," and the Open Source Initiative definition of "open source software," thus avoiding the controversy between those two terms.

2 http://popcon.debian.org

3 All the code used has been released as libre software, and can be obtained from http://libresoft.dat. escet.urjc.es/index.php?menu=Tools

4 The Gini coefficient is a normalized measure of inequality; values near 0 point out equal distributions while values close to 1 are indicative for high inequalities.