A historical analysis of Debian package conflicts

Sébastien Drobisz*, Tom Mens* and Roberto Di Cosmo†
* Software Engineering Lab, COMPLEXYS Research Institute, University of Mons, Belgium
firstname.lastname@umons.ac.be
† Univ Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126 CNRS and INRIA, F-75205 Paris, France
roberto@dicosmo.org

***Abstract*—Users and developers of software distributions are often confronted with installation problems due to conflicting packages. A prototypical example of this are the GNU Linux distributions such as Debian. While the problem of co-installability of Debian packages has been studied in the past, little is known about how this problem evolves over time. In this article, we present preliminary results of an empirical analysis of co-installability problems of individual packages throughout seven years of history of the Debian distribution.**

I. Introduction

Package-based software distributions (such as the Debian Linux distribution) and other large component-based software repositories have been shown to suffer from maintainability and scalability problems due to the so-called "co-installability problem" [1]. Desired software packages or components may not be installable together due to conflicting dependencies, and detecting such conflicts is algorithmically hard.

While efficient algorithms and tools have been proposed for detecting co-installation conflicts and supporting their resolution [2], [3], [4], [5], little is known about how this problem evolves throughout the history of the considered software repository.

Therefore, the goal of our research is to empirically analyse the evolution history of package-based software repositories, in order to assess the extent of the "co-installability problem" and how this problem evolves over a longer time period, involving many different versions of the software package repository. This is different in scope from the research presented in [6], where the evolution of the co-installability problem is studied between pairs of successive versions of a package repository, in order to identify the so-called "broken sets".

The case study that we have used for our empirical analysis is the Debian Linux testing distribution for the i386 architecture, over a 7-year time period. In this paper, we present some research results obtained while studying the evolution of this distribution.

II. Context

A. About Debian

The Debian GNU/Linux distribution is a free distribution of the Linux operating system, initially released in 1993. While t is available for a multitude of architectures, in this article we will focus on the i386 architecture that can be considered historically as the first one for which Debian was available. To facilitate maintenance and facilitate collaborative work, Debian is composed of tens of thousands of different packages, developed by thousands of developers.

TABLE I
Major production releases of Debian

Version	Code name	Release date	Number of packages
4.0	etch	2007-04-08	about 18K
5.0	lenny	2009-02-15	about 23K
6.0	squeeze	2013-02-06	about 29K
7.0	wheezy	2013-03-04	about 37K

There are essentially three types of Debian distributions. The *stable* distribution is the latest official release, and only contains stable, well-tested packages. The *testing* distribution contains package versions that should be considered for inclusion of the next stable Debian release. The *unstable* distribution contains packages that are not thoroughly tested and that may still suffer from stability and security problems.

Because we are interested in studying the evolution of Debian *development* activity, our empirical study will be restricted to the *testing* distribution, as it is what we consider to correspond most closely to a development version: package versions in the testing distribution are likely to become part of the next public release.

For this distribution, we have extracted daily snapshots, during the 7-year period from 1 January 2007 (>18K packages) to 31 December 2013 (>38K packages). Table I lists the major production release versions of Debian created during this timeframe.

B. Mining package co-installability data

A major problem when analysing co-installability of packages is the sheer size of the package dependency graph: there are typically thousands of different packages with implicit or explicit dependencies to many other packages. Vouillon et al. [5] addressed this problem by proposing an algorithm and theoretical framework to compress such a dependency graph to a much smaller one with a simpler structure, but with equivalent co-installability properties. The idea is that sets of packages are bundled together into an equivalence class if these packages are co-installable together, while they are not co-installable with the same other packages. As an example, the full graph for the Debian testing distribution on 22 August 2010 contained 28,919 packages, 124,246 dependencies and 1,146 co-installability conflicts. After simplification, the resulting graph contained only 1,038 packages, 619 dependencies

and 985 conflicts.

The COINST tool (http://coinst.irill.org)) was developed specifically for extracting and visualizing coinstallability kernels for GNU/Linux distributions. We used the output of this tool as the basis of our analysis. For each daily snapshot, we used a shell script to browse and extract all names of packages contained in the distribution[1]. To retrieve the information about the co-installation conflicts of these packages we used the JSON output files generated by the COINST tool with the command `coinst -conflicts conflicts -stats -o graph.dot Packages.bz2 >& log`.

C. Research Methodology

Di Cosmo et al. have previously used co-installation conflict graphs to determine appropriate solutions to co-installability problems. These solutions, however, did not take into account the "package conflict history", i.e., the evolution over time of package co-installation conflicts. It is our goal to determine to which extent this historical information provides additional information to understand and predict how co-installation conflicts evolve over time, and to improve support for addressing co-installability problems.

To work towards this goal, section III starts by performing a high-level analysis of the Debian package conflict history. Subsection III-A provides some visualisations and descriptive statistics of how the co-installability problem evolves over time for the Debian testing distribution as a whole. Subsection III-B focuses on specific observed trend breaks and identifies problematic packages in Debian based on this. Section IV carries out a statistical analysis based on the individual conflict evolution histories of all packages contained in the Debian testing distribution.

III. Qualitative Analysis

A. Descriptive statistics

In the remainder of this article we will use the term *conflicting package* to denote a package that has at least one co-installability conflict with at least one other package at a specific point in time.

Let us start by presenting some plots showing the evolution of (conflicting) packages belonging to the Debian testing distribution. Fig. 1 (top) compares the daily evolution of the total number of packages against the number of conflicting packages. They both show a linearly increasing trend. Fig. 1 (bottom) displays the percentage of conflicting packages over time. We see that (with some exceptions in 2009), the ratio of conflicting packages remains more or less constant (between 15% and 20%) over time, despite the fact that the number of Debian packages continues to increase.

Fig. 2 displays, per snapshot, the absolute (top figure) and relative (bottom figure) number of co-installation conflicts per package. Most of the time there are between 2,000 and

[1]The information for a given snapshot date <*DATE*> (using the format *YYYYMMDD*) is available on http://snapshot.debian.org/archive/debian/<*DATE*>T060000Z/dists/testing/main/binary-i386/Packages.bz2.

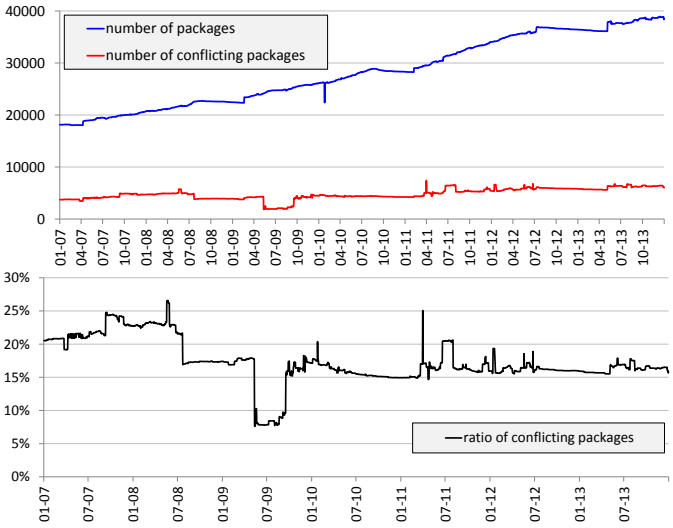

Fig. 1. Daily evolution of the number of packages and conflicting packages in the Debian testing distribution.

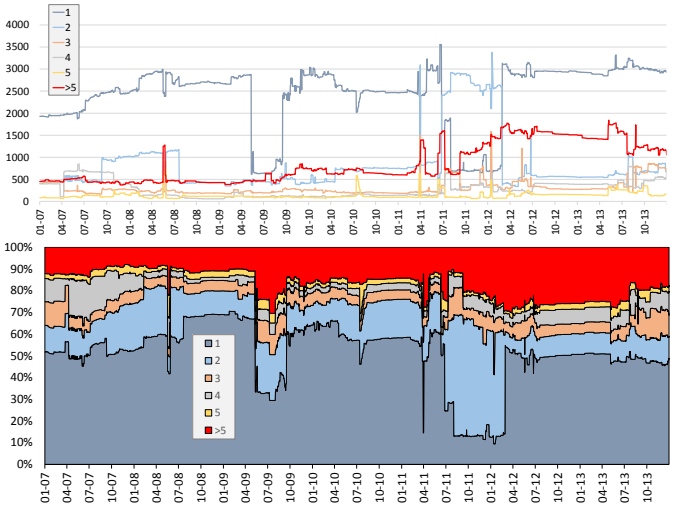

Fig. 2. Daily evolution of the number of packages in the Debian testing distribution having a co-installability conflict with 1, 2, 3, 4, 5 or more packages.

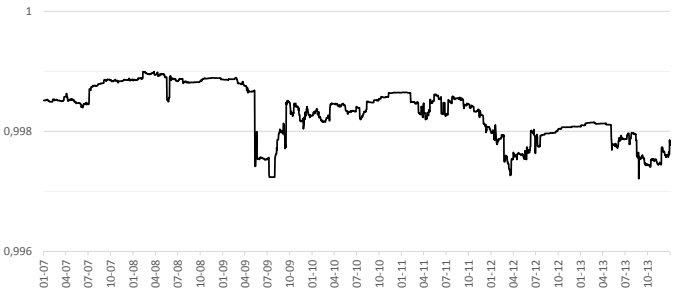

Fig. 3. Evolution of the Gini index (aggregating the number of co-installability conflicts per package) over time.

3,000 packages with exactly one co-installation conflict. This corresponds to a ratio of about 50% of all conflicting packages.

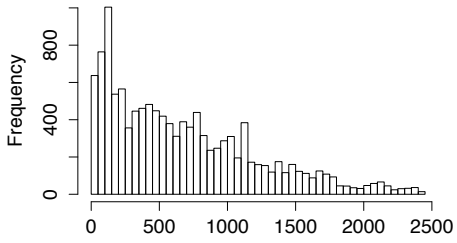

Fig. 4. Lifetime (in days) of packages that were both created and removed during the considered period.

Fig. 2 (bottom) reveals that there are gradually less packages that have more co-installation conflicts, indicating a skewed distribution. We used the Gini index to measure inequality of the distribution [7]. The evolution of this index is visualised in Fig. 3. For all snapshots we found a very high inequality, reflected by a Gini value that is very close to 1 (>0.997 in all cases). Besides this, the evolution plot of the Gini index reveals the same patterns as those observed in Fig. 1 (bottom) and Fig. 2 (bottom).

Fig. 4 displays the lifetime (in days) of each considered Debian package. We have excluded from the figure all packages that already existed at the first day of the considered period and/or that still existed at the last day of the considered period. This leaves us with 12,182 packages (out of a total of 58,478). The reason for doing so is that we cannot be sure about the lifetime of these packages: they may have existed already before or continue to exist after the considered period. We observe that the lifetime of packages is not uniformly spread, and there appears to be a decreasing trend. The median value is 546, implying that 50% of the packages survive less than 1,5 years.

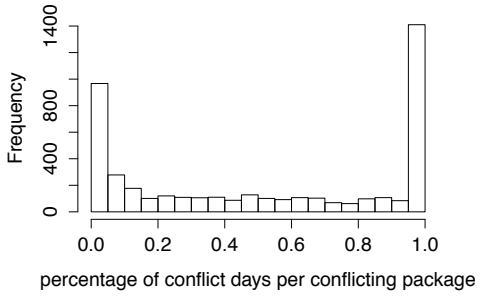

Fig. 5. Percentage of days of their lifetime that conflicting packages were in conflict.

Of the list of packages considered in the previous figure, let us focus on only those 4,416 packages that had a conflict at least once in their lifetime. The number of conflicting days for these packages is visualised horizontally as a percentage of their total lifetime in Fig. 5. We observe that 965 (i.e., 21,9%) packages were almost never conflicting (<5% of the time). Another peak is observed at the other side of the spectrum, were we find 1,410 (i.e., 31,9%) packages that were conflicting most of their lifetime (>95% of the time). More specifically, 1,181 of these packages (i.e., 26,7%) were conflicting during their entire lifetime. Many of these packages can be considered as having "stable" co-installability problems that do not cause problems for their maintenance. We will study this in more detail in the remainder of this article.

B. Identifying problematic packages

Figures 1 to 3 reveal, in the first half of 2009, a sudden very important *decrease* in number and ratio of conflicting packages. Pinpointing the source of the problem, we found that on 12 May 2009 a lot of packages were no longer conflicting compared to the previous day. This disappearance of conflicts coincides with the removal of package `ppmtofb` that appears to be the source of the problem. It had a very high number of conflicts just before its removal (2,664 conflicts, a value much higher than for all other identified packages). Removing this package from the distribution removes conflicts in many other packages (that were only co-uninstallable with this problematic package).

The figures also reveal the opposite behaviour (i.e., an important sudden *increase*) a couple of months later due to the introduction of package `liboss-salsa-asound2`. On 17 September 2009 it provoked by itself 1,758 new conflicts. On 29 February 2012 this package was removed, resulting in a decrease of the number of conflicts for many packages, as can be observed in Fig. 2 (bottom).

Manual inspection of these packages and their related entries in the Debian bug tracking system revealed that both of these packages were highly problematic: `ppmtofb` was a package no longer maintained, that depended on an obsolete library, and `liboss-salsa-asound2` was not intended to be used on the i386 architecture. The obvious solution was to remove them, which eventually happened, but the absence of clear metrics to pinpoint these problems, together with the marginal popularity of these packages on the i386 architecture made this process extremely long.

In an attempt to automate the detection of this class of problematic packages, we computed the following metrics for each package: its *minimum* and *maximum* number of conflicts, its *spread* (i.e., $max - min$), and the number of days with more conflicts than $\frac{max-min}{2}$. For packages with a high value for most of these metrics we checked whether they have been reported as problematic by the Debian community. This allowed us to find back both aforementioned problematic packages, and many more such as:

- `fdpowermon-icons` is incompatible with all of KDE (it changes the icons used by KDE), and should therefore be avoided by a normal user.
- `libqt4-phonon` is a package for temporary usage only during the cross-port of a distribution. Its number of conflicts is therefore not really problematic in the sense that a "normal" user does not need it, and those that need it only need it once.
- `odbc-postgresql` and some of its variants are known to have had incompatibilities with the `libiodbc2` package for a certain time. This problem is no longer present today.

Using the same metrics, we also found quite a few packages with a high maximum number of conflicts that only lasted for relatively short periods. A typical example is `libgdk-pixbuf2.0-0`, whose introduction with 4,100 conflicts causes a peak on 31 March 2011 that is clearly visible at the bottom of Fig. 1 and disappears completely after 2 days.

Manual inspection revealed that these conflicts are caused by the arrival in the testing distribution of new versions of a package, that rely on a new version of another package that has not yet been introduced: these transient episodes are generally solved in a matter of days or weeks with the arrival of the expected new version of the other packages, and do not indicate any long lasting problem. The problem can be avoided altogether by using more advanced tools for maintaining Debian repositories, such as the `comigrate` tool described in [8].

IV. Statistical Analysis

The reasons for particular trend breaks (sudden decreases or jumps) in the evolution of conflicting packages can be analysed according to specific classes of packages (as we have illustrated in section III). We are, however, also interested in understanding the package conflict history from the point of view of individual packages. We also wish to study to which extent the introduction and removal of packages affects package conflicts in the distribution. To address these goals, we will focus our statistical analysis on specific research questions. Ultimately, answers to these questions will allow us, at the longer term, to come up with prediction models of how likely it is that package conflicts get introduced or resolved over time.

RQ_1 How long does it take before a conflict is introduced in a package?

For this research question, we are interested in the first appearance of a conflict in a package. We hypothesise that newly introduced packages have a high likelihood of introducing conflicts. To analyse this, we first have to exclude all packages that are present at the first day of the considered 7-year period, since we cannot know when a conflict first appeared in them.

This leaves us with 40,332 packages that are introduced somewhere during the considered timeframe. Of these, 26,397 (i.e., 65,4%) never encountered a conflict, and 7,541 of them (i.e., 18,7%) were introduced with at least one conflict at the moment of their introduction. For the remaining 6,394 packages (i.e., 15,9 %), the distribution of the number of days before the first conflict is introduced has a median value of slightly over one year (375 days to be precise) and follows a decreasing trend (see Fig. 6). This means that the longer a package exists without conflicts, the less likely it is that a conflict will appear.

We can conclude that our hypothesis is not correct. Most new packages (>65%) never encounter a conflict. For those packages that do encounter a conflict, however, in the majority of the cases (54%) the conflict is already present at the moment of introduction of the package.

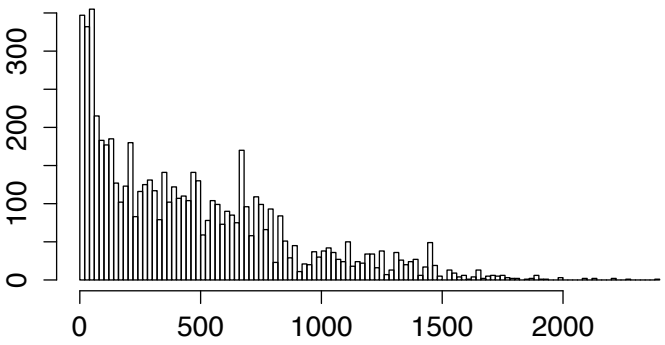

Fig. 6. Number of days before conflicts arise in newly introduced packages. Packages without conflicts or containing conflicts at the day of their creation are excluded.

RQ_2 What is the effect of conflicts on the longevity of packages?

For this research question we only consider those 12,182 packages that were both created and removed during the considered period. We address the question in three steps. In all these steps, we use a Wilcoxon rank sum test (a.k.a. Mann-Whitney test) and a Kolmogorov-Smirnoff test to find statistical evidence for a difference between two distributions. The null hypothesis assumes that there is no significant difference.

First, we hypothesise that the *absence* of conflicts upon *introduction* of a package has an effect on its longevity. To this extent, we compare the lifetime of packages that are introduced with conflicts and without conflicts. A two-sided test rejects the null hypothesis with statistical significance (p-value $<<0.001$). A similar one-sided test suggests that packages *with* conflicts upon their introduction actually live *longer* than packages that are introduced *without* conflicts. This is visually confirmed by the histograms in Fig. 7.

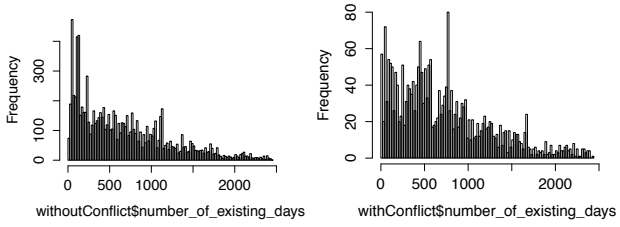

Fig. 7. Longevity of packages with or without conflicts at the time of their introduction.

Secondly, we hypothesise that the *absence* of conflicts during the lifetime of a package has an effect on its longevity. Again, a two-sided test observes a difference with statistical significance (p-value $<<0.001$). A one-sided test reveals that packages with conflicts during their lifetime tend to live longer than packages without conflicts.

Thirdly, we compare the longevity of packages that were conflicting during their *entire* lifetime (i.e., 100% of the time) with packages that only had conflicts occasionally (<100% of the time). Fig. 5 showed that 26,7% of all conflicting packages are in conflict 100% of the time. A two-sided test reveals a

difference with statistical significance. A one-sided test reveals that packages that were in conflict occasionally live longer than packages that were in conflict during their entire lifetime.

V. Threats to Validity

There are many potential threats to validity of our empirical study. Some threats relate to the generalisability of the results. We have restricted ourselves to Debian packages, so we cannot make any generalisations to other package-based systems. Even within Debian, we have only studied the Debian *testing* distribution for the i386 architecture. Results for the *stable* or *unstable* distribution are likely to be quite different. We have limited our analysis to a 7-year period. Extending this to a longer period before 2007 may change our obtained results.

For our analysis we based ourselves on the output produced by the COINST tool. The risk that possible bugs in this tool may affect the outcome of our results is limited because the conflicts identified by COINST can be independently checked using other existing tools, like `dose-deb-coinstall`. The obtained results may also be biased by some simplifying assumptions we have made during our analysis. First of all, we have considered *removed* packages as packages that did not exist any longer at the last day of the considered period. This is an overapproximation, in the sense that packages may sometimes disappear for a while from the Debian distribution, but get reintroduced again at a later time (beyond the considered time period). Secondly, we have ignored the issue of package renaming, assuming that a new package name effectively corresponds to a new package (which is the case for the majority of all considered packages). Sometimes, however, the name of a package can change from one snapshot to another. This is for example the case for some packages where the version number is explicitly encoded as part of the package name.

VI. Future Work

We plan to extend the research reported here in many ways. We plan to automate the identification of problematic packages into a metrics-based dashboard targeted to Debian package maintainers and users. This could be augmented with prediction models that analyse the package conflict history to predict survivability or future problems for certain (groups of) packages.

The current empirical analysis was only based on metrics related to packages and their co-installation conflicts. We plan to augment this analysis by taking the social aspects into account. Among others, we would like to assess the impact of package maintainer characteristics (such as number of maintained packages and seniority) on the likelihood of having package conflicts. We intend to take usage information into account as well by studying correlations between package conflicts and data collected by the Debian Popularity Contest.

Similar to the work in [9], we intend to study the evolution of the package conflict dependency graph. By studying the structure of this graph as well as specific nodes of this graph (e.g., packages that are in conflict with many other packages) we want to study how conflicts appear and disappear over time when packages get introduced, updated or removed.

VII. Conclusion

Building further upon the work by Di Cosmo et al. [2], [3], [5] we empirically analysed a 7-year history of the Debian package-based software distribution for the i386 architecture. We used data from the COINST tool to study how the problem of non co-installabe packages evolves over time.

While the number of Debian packages increases linearly, the ratio of conflicting packages stays more or less constant, with occasionally important decreases or increases in the number of conflicts, caused by the introduction, modification or removal of particularly problematic packages. We identified different type of problematic packages, whose presence coincides with high values for some simple conflict-based metrics.

We observed that the majority of newly introduced packages never encounters a conflict. If they do, in the majority of the cases there is already a conflict at the time of introduction of the package. About 1 out of 5 packages are in conflict less than 5% of the time. Close to one third of the conflicting packages are in conflict during their entire lifetime. As we expected, these packages live shorter than packages that are not in conflict all the time. Contrary to what we expected, however, packages that had at least one conflict during their lifetime tend to have a longer lifetime than packages without.

Acknowledgments

This research was carried out in the context of ARC research project AUWB-12/17-UMONS- 3.

References

[1] R. Di Cosmo and J. Vouillon, "On software component co-installability," in *SIGSOFT FSE*. ACM , 2011, pp. 256–266. [Online]. Available: http://dx.doi.org/10.1145/2025113.2025149

[2] C. Artho, K. Suzaki, R. Di Cosmo, R. Treinen, and S. Zacchiroli, "Why do software packages conflict?" in *Int'l Conf. Mining Software Repositories*, 2012, pp. 141–150.

[3] R. Di Cosmo, D. Di Ruscio, P. Pelliccione, A. Pierantonio, and S. Zacchiroli, "Supporting software evolution in component-based FOSS systems," *Sci. Comput. Program.*, vol. 76, no. 12, pp. 1144–1160, 2011.

[4] R. Di Cosmo and J. Boender, "Using strong conflicts to detect quality issues in component-based complex systems," in *Indian Software Engineering Conf.*, 2010, pp. 163–172.

[5] J. Vouillon and R. Di Cosmo, "On software component co-installability," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 4, p. 34, 2013.

[6] J. Vouillon and R. Di Cosmo, "Broken sets in software repository evolution," in *Int'l Conf. Software Engineering*, 2013, pp. 412–421.

[7] R. Vasa, M. Lumpe, P. Branch, and O. Nierstrasz, "Comparative analysis of evolving software systems using the Gini coefficient," in *Int'l Conf. Software Maintenance*, 2009, pp. 179–188.

[8] J. Vouillon, M. Dogguy, and R. D. Cosmo, "Easing software component repository evolution," in *Int'l Conf. Software Engineering*, P. Jalote, L. C. Briand, and A. van der Hoek, Eds. ACM, 2014, pp. 756–766.

[9] M. Claes, T. Mens, and P. Grosjean, "On the maintainability of CRAN packages," in *Int'l Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. IEEE, 2014, pp. 308–312.