News from the EDOS project: improving the maintenance of free software distributions [*]

Jaap Boender [1] , Roberto Di Cosmo [1] , Berke Durak [2]
Xavier Leroy [2] , Fabio Mancinelli [1] , Mario Morgado [3]
David Pinheiro [3] , Ralf Treinen [4], Paulo Trezentos [3] , Jérôme Vouillon [1]

[1] PPS, University of Paris 7, `Firstname.Lastname@pps.jussieu.fr`

[2]INRIA Rocquencourt, `Firstname.Lastname@inria.fr`

[3]Caixa Magica, `Firstname.Lastname@caixamagica.pt`

[4]LSV, ENS de Cachan, `Firstname.Lastname@lsv.ens-cachan.fr`

Abstract. *The EDOS research project aims at contributing to the quality assurance of free software distributions. This is a major technical and engineering challenge, due to the size and complexity of these distributions (tens of thousands of software packages). We present here some of the challenges that we have tackled so far, and some of the advanced tools that are already available to the community as an outcome of the first year of work.*
Keywords: *free software, open source software, dependency management, constraint satisfaction, rollback, EDOS project.*

1 Introduction

So-called *distribution editors* like Caixa Magica, Conectiva, Debian, Mandriva, RedHat, Suse, Ubuntu and many others try to offer some kind of reference viewpoint over the breathtaking variety of free and open source software (FOSS) available today: they take care of packaging, integrating and distributing tens of thousands of software packages, very few of them being developed in-house and almost all coming from independent developers. As a consequence, most FOSS distributions today simply rely on the general notion of a software *package*[1]: a bundle of files containing data, programs, and configuration information, with some metadata attached. Most of the metadata information is about *dependencies*, i.e., the relationships with other packages that may be needed in order to run or install a given package, or that conflict with its presence on the system.

How can one ensure the quality of a distribution? This problem, which is the focus of the European Sixth Framework Programme project EDOS (Environment for the development and Distribution of Open Source software), can essentially be divided into three main tasks:

Upstream tracking makes sure that a package in the distribution closely follows the evolution of the software development, almost always done by some team beyond direct control by the distributor.

[*]This work was supported by the EDOS Specific Targeted Research Project of the 6th European Union Framework Programme.
[1]Not to be mistaken for the software organizational units such as librairies, modules or classes.

Testing and integration makes sure that a program performs as expected in combination with other packages in the distribution. If this is not the case then bug reports should be sent to the upstream developer.

Dependency management makes sure that, in a distribution, packages can be installed and user installations can be upgraded when new versions of packages are produced, while respecting the constraints imposed by the dependency metadata.

Inside the EDOS project, work package 2 (WP2) is about dependency management. This task is surprisingly complex [Tuura 2003, Van der Storm 2004] due to the large number of packages present in a typical distribution and to the complexity and richness of their interdependencies. More specifically, our focus is on the issues related to dependency management for large sets of software packages, with a particular emphasis on maintaining consistency of a software distribution *on the repository side*, as opposed to maintaining a set of packages installed *on a client machine*. This choice is justified by the following observation: maintaining consistency of a distribution of software packages is *fundamental* to the quality and scalability of current and future distributions; yet, it is also an *invisible* task, since the smooth working it ensures on the end user side tends to be considered as normal and obvious as the smooth working of packet routing on the Internet. In other words, we are tackling an essential *infrastructure* problem that has long been ignored: while there are a wealth of client-side tools to maintain a user installation (`apt`, `urpmi`, `smart` and many others [Silva 2004, Mandriva 2005, Niemeyer 2005]), there are surprisingly little literature and publically available tools that address server-side requirements. We found very little significant prior work in this area, despite it being critical to the success of FOSS in the long term.

In this short paper we want to give an overview of some of the tools developed inside the EDOS Project that have the potential to improve the management of distributions from the point of view of dependencies. The paper is organized as follows. Section 2 contains a formal description of the main characteristics of a software package found in mainstream FOSS distributions as far as dependencies are concerned. The algorithmic aspects of solving the dependency constraints are underlined in Section 3. Section 4 describes the usage of some of these tools for a Linux distribution (Caixa Magica) and their extension of APT with a rollback feature. Some metadata exploration tools developed by the WP2 and taking into account the historical evolution of the repositories are described in Section 5.

2 Basic definitions

Every package management system [Debian 2005, Bailey 1997] takes to various extends into account the interrelationships among packages. We will call these relationships *requirements*. Several kinds of requirements can be considered. The most common one is a *dependency* requirement: in order to install package P_1, it is necessary that package P_2 be installed as well. Less often, we find *conflict* requirements: package P_1 cannot coexist with package P_2.

Some package management systems specialize these basic types of requirements by allowing to specify the *time frame* during which the requirement must be satisfied. For example, it is customary to be able to express *pre-dependencies*, a kind of dependency stating that some package P_1 needs some package P_2 to be present on the system *before* P_1 can be installed [Debian 2005].

These notions can be made precise [Di Cosmo et al. 2006], and we refer the interested reader to that paper for a more detailed discussion. For the current presentation we do not need such a level of formal precision, and we will rely on the intuitive meaning of commonly used terms like package, dependency, conflict, repository and installation.

The first, most basic quality requirement for a distribution is that for each package P being part of a distribution there should exist at least one installation of the distribution that satisfies all dependency constraints and that contains P. Otherwise, P is useless: nobody will ever be able to install it without breaking the dependency constraints, which in turn breaks the package management system.

Definition 1 (Installability). *A package π of a repository R is* installable *if there exists an installation I that contains π and that is* healthy*, i.e. with no broken dependencies.*

We say that a repository R is *trimmed* when every package of R is installable w.r.t. R. The intuition behind this terminology is that a non-trimmed repository contains packages that cannot be installed in any configuration. We call those packages *broken*, and they should be excluded from the repository.

Our first set of tools is able to formally check whether a repository is trimmed, and if not it explains which packages are not installable for which reason.

3 Algorithmic considerations

It is not obvious that checking a repository for broken packages is actually tractable in practice: due to the rich language allowed to describe package dependencies in the mainstream FOSS distributions, this task may involve verifications over a large number of packages. During our first investigations of these problems we have indeed already proven the following complexity result.

Theorem 1 (Package installability is an NP-complete problem). *Checking whether a single package P can be installed, given a repository R, is NP-complete.*

Nevertheless, the actual instances of these problems, as found in real repositories, turn out to be quite simple in the average.

We implemented various checking tools, using custom solvers as well as a SAT solver [Eèn and Sörensson 2004] and CP solvers, and ran them over both the Debian pool (about 15,000 source packages giving 20,000 units, totaling 30,000 packages in the different distributions, including ”contrib” and ”non-free” packages) and the Mandriva Cooker distribution (around 5,000 packages). The execution time is completely satisfactory, and the tools found a number of non-installable packages in both distributions. These tools are available from the subversion server of the EDOS project at `http://www.edos-project.org/`.

Notice that, unlike scripts that are actually used in some distributions[2], these EDOS tools are *correct and complete*, that is, they find *all* broken packages, and *only*

[2]See EDOS Deliverable 2.2 on the EDOS website for an analysis of the limitations of pre-existing tools.

the broken packages, and they are *highly efficient*, as they can analyze the whole Debian repository in just a couple of minutes.

You can of course simply go to the EDOS subversion repository and download the tools to run them on the command line, but we have also two real-world deployment examples which show how a distribution manager could use them in a production environment.

4 Deployment and usage of the tools at Caixa Magica

Caixa Magica is a Portuguese Linux distribution used nationwide in Portugal. It is used not only in schools and public administrations but also by companies and private citizens. It is based on the RPM format although it uses `apt` (namely `apt-rpm`) since 2004. As in other Linux distributions, it has FTP servers with the official software packages (RPMs) and servers with unofficial RPMs submitted by the community. The company encourages the submission of these unofficial packages since they are much more up-to-date than the stable and official ones. For that purpose Caixa Magica has created a website, named "ContribWare" (`http://contribware.caixamagica.pt/`), which maintains the submission of unofficial packages. ContribWare is now 3 months old and hundreds of packages have been submitted through it. One workflow-related problem of such systems is detecting the broken dependencies. The contributors who submit packages usually have a lot of software installed and thus may fail to identify dependencies. These dependencies can also be on not yet packaged software.

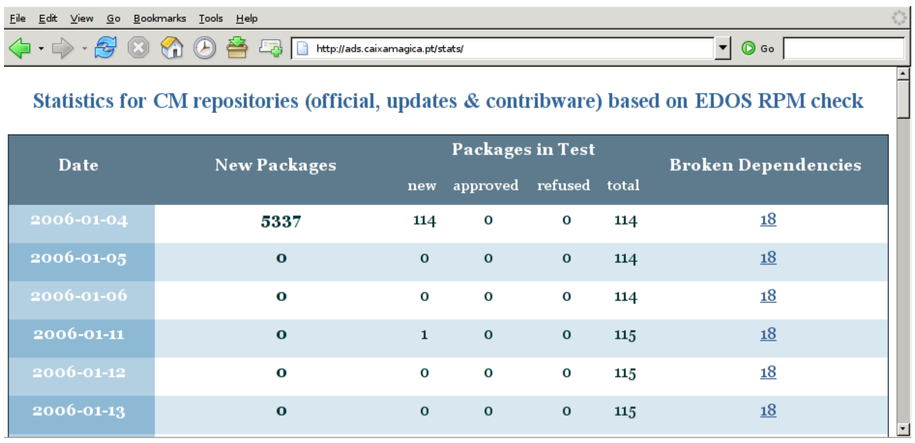
4.1 Description of graphical statistics interface

Using the WP2 `rpmcheck` tool, we were able to identify broken dependencies in ContribWare. A Python script has been developed for processing the information and displaying an HTML page containing a table summarizing the detected problems, as in Figure 1. The table has the following columns:

- New Packages: new packages added to the repository. It began with 5,337 packages.
- Packages in Test: packages that have been submitted to ContribWare by users and that have been moved to the "on test" state by Caixa Magica editors. This column has 4 sub-columns: *new*, *approved*, *refused* and *total*. Packages that are approved go to "New packages" and leave the "on test" state.
- Broken Dependencies: this column gives the number of packages with broken dependencies, with a link to a more detailed description. The latter gives the broken packages with their unsatisfied dependencies.

4.2 Apt rollback - extending package maintenance

The EDOS team is also developing some enhancements to the package management process. One of them is the APT rollback mechanism. As some upgrades are not always successful, customer requirements on quality assurance opened the need for implementing a rollback mechanism into `apt-rpm`. This mechanism relies on registering the requests for installation, upgrade, downgrade and removal of packages from the system as well as saving, in some situations, the packages' configuration files (depending on the operation


File Edit View Go Bookmarks Tools Help
http://ads.caixamagica.pt/stats/
Go

Statistics for CM repositories (official, updates & contribware) based on EDOS RPM check

Date	New Packages	Packages in Test				Broken Dependencies
		new	approved	refused	total	
2006-01-04	5337	114	0	0	114	18
2006-01-05	0	0	0	0	114	18
2006-01-06	0	0	0	0	114	18
2006-01-11	0	1	0	0	115	18
2006-01-12	0	0	0	0	115	18
2006-01-13	0	0	0	0	115	18

Figure 1. Daily log of Caixa Magica archives.

to be performed on the packages: upgrade, downgrade or removal). This mechanism permits to restore the system back to its state before any `apt` operation. For example, if an error is detected after an upgrade, the system can quickly restore its previous state. With every `apt-rpm` operation we save the following information:

- the package's name and version before the operation,
- the package's version after the operation (only for upgrades or downgrades),
- the operation's type (install, upgrade, downgrade or remove),
- a transaction ID,
- a timestamp and
- the package's configuration files.

If the operation is an upgrade, a downgrade or a removal, we query the package's metadata to check the existence of any configuration files, and if so we save them. Note that files that are saved are the ones available in the system, not the package's original configuration files, so that when a rollback is performed we ensure that we restore the configuration files as they were at the time of the operation, including user modifications.

A rollback is basically the inverse of a transaction. It includes the inverse package operation (downgrade for an upgrade, removal for an installation, etc.) as well as the restoration of the package's configuration files (if necessary) as shown in Table 1:

Operation	Rollback operation	Action taken with the Configuration Files
`install`	`remove`	None
`remove`	`install`	Restore configuration
`upgrade`	`downgrade`	Restore configuration files
`downgrade`	`upgrade`	Restore configuration files

Table 1. Rollback operation relationships.

We implemented this functionality into `libapt`, thus ensuring that tools like `synaptic` also register every operation performed and we added two more commands to `apt-get`:

- `apt-get rollback-hist`: For displaying the history of operations with their transaction IDs

- `apt-get rollback <transaction id>`: For rolling back the operations performed in the given transaction.

5 EDOS Tools for Exploring the Debian History

The `history` tool allows command-line exploration of the Debian metadata using an algebraic query language. Daily metadata is extracted from the archives on `snapshot.debian.net` and stored in a MySQL database. However, despite the data being stored in a structured manner and being appropriately indexed, performance of even simple SQL queries (such as finding the set of immediate potential dependencies of a set of packages) is poor. This is partly due to the complex nature of metadata, which is, due to disjunctive dependencies, not a graph but a a hypergraph, whose relational representation requires multiple levels of indirection. Also, it is well-known that standard SQL cannot handle transitive closures. Hence we have opted for using the SQL database only as an off-line storage engine; `history` loads the whole database into RAM, and can then answer complex queries efficiently. We now give a very short introduction to the query language.

The tool works as a classic read-evaluate-print loop. Expressions or directives are entered and their results printed. The basic data types handled by `history` are units, packages, sources, sets of the above, dates, integers and booleans. Care has been taken to provide concise notation for describing these. The basic features of a functional language with strict evaluation are provided. The usual Boolean operations on sets (intersection $\&$, union $|$ and difference $\backslash$) are allowed. Sets can be filtered by regular expressions or user-defined functions. With operators such as `exists` and `for_all`, this effectively provides first-order quantified queries. Besides directives that print the metadata in human-readable form, various operators are provided so that the metadata can be used in complex expressions. The metadata is historically represented as functions which map dates to sets of packages. For instance, if p is a package, $\texttt{provides}(p)$ is the set of units provided by p, and if s is a set of packages, $\texttt{closure}(s)$ is the dependency closure of s : all packages that packages of s might need to run are contained in s – but due to disjunctive dependencies, s will usually contain extraneous packages, and due to conflicts, it might not be possible to install all packages of s. Thus, the dependency solver of `debcheck` and `rpmcheck` has been integrated into `history` as an operator $\texttt{install}(p_1, p_2)$ which computes a set p of co-installable packages such that $p_1 \subseteq p \subseteq p_2$. Table 2 gives an overview of the available operators.

We are developing a web version of `history` with an interface similar to that of `ara`[3]. An early prototype integrating results from the dependency solver and the historical metadata database, called `anla`, is available at `http://brion.inria.fr/anla/`, see Figure 2 for a screenshot.

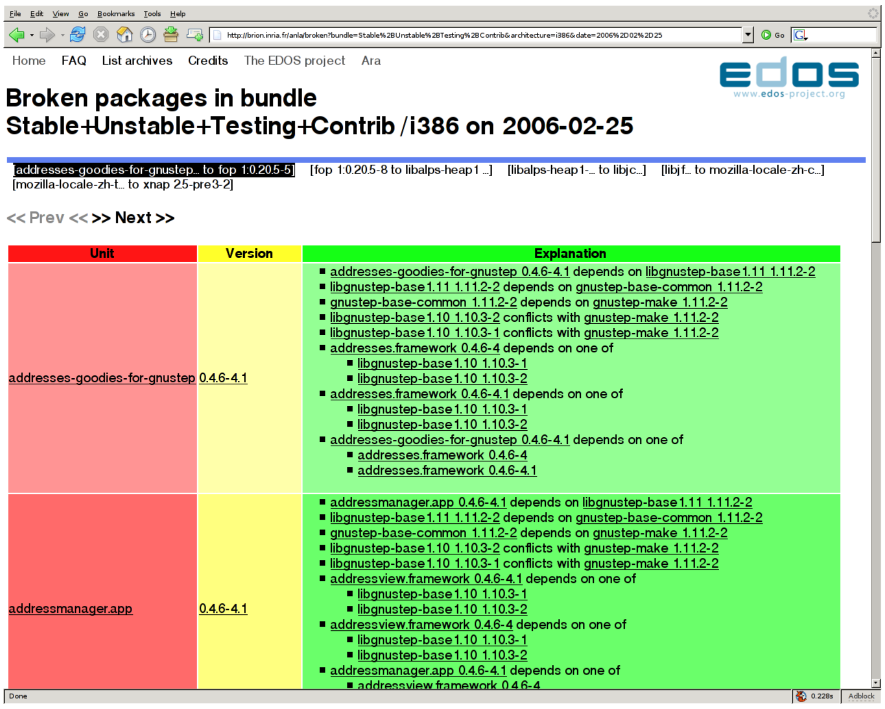
6 Conclusions

We hope that the efficient and formally based tools developed by the EDOS project will be soon adopted by distribution editors to improve their production cycle.

[3]`ara` is a text-based search engine for Debian packages which can compute Boolean combinations of field-restricted regular expressions, available at `http://ara.edos-project.org/`

Operator	Meaning
$s\&t, s\|t, s \setminus t$	Boolean set intersection, union and difference
$\texttt{provides}(p)$	Set of units provided by a package
$\texttt{conflicts}(p)$	Set of packages that conflict with a package
$\texttt{closure}(p)$	Dependency closure of a package
$\texttt{source}(p)$	Source of a package
$\texttt{unit}(p)$	Unit of a package
$\texttt{latest}(u)$	Latest version of a unit
$\texttt{versions}(u)$	All the versions of a unit
$\texttt{what\_provides}(u)$	The set of packages that provide a unit
$\texttt{replaces}(p)$	The set of packages replaced by a package
$\texttt{install}(p, q)$	Returns an installation of the set of packages p inside the set q
$\texttt{member}(x, s)$	True when the element `x` is a member of the set `s`
$\texttt{filter}(s, f)$	Return the elements of s for which $f(s)$ is true.
$\texttt{exists}(s, f)$	True when $f(s)$ is true for one element of s
$\texttt{for\_all}(s, f)$	True when $f(s)$ is true for all elements of s

Table 2. Operators. Most operators are overloaded to work on sets. Functions are first-class values with lexical scoping that can be defined anonymously.


File Edit View Go Bookmarks Tools Help
http://brion.inria.fr/anla/broken?bundle=Stable%2BUnstable%2BTesting%2BContrib&architecture=i386&date=2006%2D02%2D25
Go
Home FAQ List archives Credits The EDOS project Ara
edos www.edos-project.org

Broken packages in bundle
Stable+Unstable+Testing+Contrib /i386 on 2006-02-25

[addresses-goodies-for-gnustep... to fop 1:0.20.5-5] [fop 1:0.20.5-8 to libalps-heap1 ...] [libalps-heap1-... to libjc...] [libjf... to mozilla-locale-zh-c...] [mozilla-locale-zh-t... to xnap 2.5-pre3-2]

<< Prev << >> Next >>

Unit	Version	Explanation
addresses-goodies-for-gnustep	0.4.6-4.1	■ addresses-goodies-for-gnustep 0.4.6-4.1 depends on libgnustep-base1.11 1.11.2-2 ■ libgnustep-base1.11 1.11.2-2 depends on gnustep-base-common 1.11.2-2 ■ gnustep-base-common 1.11.2-2 depends on gnustep-make 1.11.2-2 ■ libgnustep-base1.10 1.10.3-2 conflicts with gnustep-make 1.11.2-2 ■ libgnustep-base1.10 1.10.3-1 conflicts with gnustep-make 1.11.2-2 ■ addresses.framework 0.4.6-4 depends on one of ■ libgnustep-base1.10 1.10.3-1 ■ libgnustep-base1.10 1.10.3-2 ■ addresses.framework 0.4.6-4.1 depends on one of ■ libgnustep-base1.10 1.10.3-1 ■ libgnustep-base1.10 1.10.3-2 ■ addresses-goodies-for-gnustep 0.4.6-4.1 depends on one of ■ addresses.framework 0.4.6-4 ■ addresses.framework 0.4.6-4.1
addressmanager.app	0.4.6-4.1	■ addressmanager.app 0.4.6-4.1 depends on libgnustep-base1.11 1.11.2-2 ■ libgnustep-base1.11 1.11.2-2 depends on gnustep-base-common 1.11.2-2 ■ gnustep-base-common 1.11.2-2 depends on gnustep-make 1.11.2-2 ■ libgnustep-base1.10 1.10.3-2 conflicts with gnustep-make 1.11.2-2 ■ libgnustep-base1.10 1.10.3-1 conflicts with gnustep-make 1.11.2-2 ■ addressview.framework 0.4.6-4.1 depends on one of ■ libgnustep-base1.10 1.10.3-1 ■ libgnustep-base1.10 1.10.3-2 ■ addressview.framework 0.4.6-4 depends on one of ■ libgnustep-base1.10 1.10.3-1 ■ libgnustep-base1.10 1.10.3-2 ■ addressmanager.app 0.4.6-4.1 depends on one of ■ addressview.framework 0.4.6-4

Figure 2. Checking status of Debian archives. Most broken packages owe their status to dependency on packages that are not in their archives, for instance a package in unstable that depends on a package in stable is broken. To filter out these uninteresting cases, we have merged archives into a bundle before launching the checker. All the displayed metadata information is fully hyperlinked.

References

Edward C. Bailey. Maximum RPM, taking the Red Hat package manager to the limit. http://rikers.org/rpmbook/,http://www.rpm.org, 1997.

Manfred Broy and Ernst Denert. *Software Pioneers: Contributions to Software Engineering*. Springer-Verlag, 2002.

Debian Group. Debian policy manual. http://www.debian.org/doc/debian-policy/, 1996–2005.

David Eklund. The lib update/autoupdate suite. http://luau.sourceforge.net/, 2003–2005.

Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.

Roberto Di Cosmo, Berke Durak, Xavier Leroy, Fabio Mancinelli and Jérôme Vouillon. *Maintaining large software distributions: new challenges from the FOSS era*. FRCSS06, Vienna, 1st April 2006.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

Nathan LaBelle and Eugene Wallingford. Inter-package dependency networks in open-source software. *Submitted to Journal of Theoretical Computer Science*, 2005.

Mandriva. URPMI. http://www.urpmi.org/, 2005.

Gustavo Niemeyer. Smart package manager. http://labix.org/smart/, 2005.

Gustavo Noronha Silva. Apt-howto. http://www.debian.org/doc/manuals/apt-howto/, 2004.

Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison Wesley Professional, 1997.

L. Taylor and L. Tuura. Ignominy: a tool for software dependency and metric analysis with examples from large HEP packages. In *Proceedings of CHEP'01*, 2001.

L. A. Tuura. Ignominy: tool for analysing software dependencies and for reducing complexity in large software systems. In *Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, volume 502, pages 684–686, 2003.

Tijs van der Storm. Variability and component composition. In *Proceedings of the Eighth International Conference on Software Reuse (ICSR-8)*, 2004.