Managing the complexity of component-based systems

Pietro Abate
pietro.abate@{pps.jussieu.fr}
http://mancoosi.org/~abate

Laboratoire PPS, University Paris Diderot

04/02/2012
Bruxelles

Contents

Outline

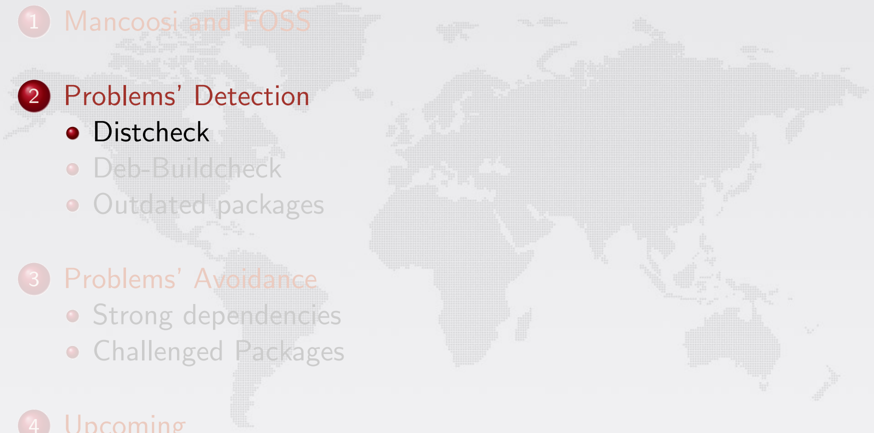
The Mancoosi Project

The mancoosi project (now concluded) focused among other things on the development of QA tool for FOSS distributions. More specifically we developed a suite of tools to:

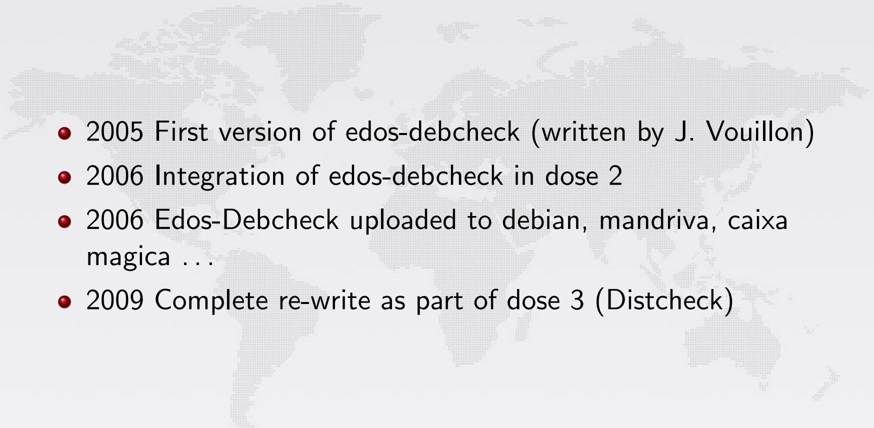
- detect packages that are not installable or that cannot be compiled.
- detect packages that are outdated in the archive.
- identify classes of “important” packages.
- identify packages the if migrated to a specific future will break a large number of other packages.

Outline

Outline



Distcheck - History



- 2005 First version of edos-debcheck (written by J. Vouillon)
- 2006 Integration of edos-debcheck in dose 2
- 2006 Edos-Debcheck uploaded to debian, mandriva, caixa magica . . .
- 2009 Complete re-write as part of dose 3 (Distcheck)

Features

- Use special purpose sat solver
- Accept in input different formats (deb, rpm, eclipse)
- Handle compressed archives
- Use CUDF as internal generic format
- YAML output for simple parsing / integration
- twice as fast compared to edos-debcheck (with more room for improvement)

Output Anatomy 1/3

```
report:
 -
  package ...
 -
  package ...
 ....

background-packages: 0
foreground-packages: 29589
total-packages: 29589
broken-packages: 143
```

Output Anatomy 2/3

```
package: apt-rpm-repository
version: 0.5.15lorg3.2-6
architecture: amd64
source: apt-rpm (= 0.5.15lorg3.2-6)
status: broken
reasons:
 -
  missing:
   pkg:
    package: apt-rpm-repository
    version: 0.5.15lorg3.2-6
    architecture: amd64
    unsat-dependency: librpm4.4 (>= 4.4)
```

Output Anatomy 3/3

```
package: gforge-web-apache2
version: 4.8.3-1
architecture: all
source: gforge (= 4.8.3-1)
status: broken
reasons:
 -
  conflict:
   pkg1:
    package: python2.6-minimal
    version: 2.6.6-3
    architecture: amd64
    source: python2.6 (= 2.6.6-3)
    unsat-conflitc: gforge-web-apache2 (< 5.0.1+svn10155)
   pkg2:
    package: gforge-web-apache2
    version: 4.8.3-1
    architecture: all
    source: gforge (= 4.8.3-1)
    provides: gforge-web--virtual
```

```
depchain1:
 -
  depchain:
   -
    package: gforge-web-apache2
    version: 4.8.3-1
    architecture: all
    depends: python
   -
    package: python
    version: 2.6.6-1
    architecture: all
    depends: python-minimal (= 2.6.6-1)
   -
    package: python-minimal
    version: 2.6.6-1
    architecture: all
    depends: python2.6-minimal (>= 2.6.6-1~)
```

Machine parsable output

```
import yaml

doc = yaml.load(file("output-of-distcheck",'r'))
for p in doc['report'] :
    if p['status'] == "broken" :
        print "%s %s is broken" %
          (p['package'], p['version'])
```

For Example . . .

```
./debcheck -v --progress -f -e natty/Packages.bz2

report:
 -
  package: kubuntu-full
  version: 1.222
  architecture: i386
  source: kubuntu-meta (= 1.222)
  status: broken
  reasons:
   -
    missing:
     pkg:
      package: kubuntu-full
      version: 1.222
      architecture: i386
      unsat-dependency: fglrx


background-packages: 7569
foreground-packages: 7569
broken-packages: 1
```

Demo

A demo is better then a 1000 words

Where is it used

- EmDebian
- Mandriva QA team
- Caixa Magica QA team
- Debian Weather
- simple-cdd
- file-conflicts
- . . .

Outline

Deb-Buildcheck

- Deb-specific for the moment (rpm version upcoming)
- Faster then edos-buildcheck (native application now)
- Same output format of distcheck
- It works by encoding the build dependencies and check if they can be satisfied w.r.t. a set of binary packages

Output example

```
package: src:tulip
version: 3.1.2-2.3
architecture: any
source: tulip (= 3.1.2-2.3)
status: broken
reasons:
 -
  conflict:
   pkg1:
    package: libgl1-mesa-swx11-dev
    version: 7.11.2-1
    architecture: amd64
    source: mesa (= 7.11.2-1)
    unsat-conflitc: libgl-dev--virtual
   pkg2:
    package: libgl1-mesa-dev
    version: 7.11.2-1
    architecture: amd64
    source: mesa (= 7.11.2-1)
```

```
depchain1:
 -
  depchain:
   -
    package: src:tulip
    version: 3.1.2-2.3
    architecture: any
    depends: libgl1-mesa-swx11-dev
depchain2:
 -
  depchain:
   -
    package: src:tulip
    version: 3.1.2-2.3
    architecture: any
    depends: libqt4-opengl-dev
   -
    package: libqt4-opengl-dev
    version: 4:4.7.4-2
    architecture: amd64
    depends: libgl1-mesa-dev
```

Second Demo !

Outline

Why packages are broken (not installable) ?

- Transient problems
  - Depend on a package that is not available
  - Depend on a package that is not installable

Why packages are broken (not installable) ?

- Transient problems
  - Depend on a package that is not available
  - Depend on a package that is not installable
- Non Transient problems : Package needs update - there is a problem in the metadata of a package.

Trivial Example 1: Is (*foo*,1) installable?

```
Package: foo
Version: 1
Depends: bar ( = 2 )

Package: bar
Version: 1
```

Is it installable in a future?

Yes, we just need to upgrade bar to version 2.

Trivial Example 2: Is (*foo*,1) installable?

```
Package: foo
Version: 1
Depends: bar ( < 2 )

Package: bar
Version: 2
```

Is it installable in a future?

No, because the dependency bar with version lesser then 2 and it will never be satisfied.

Example 1: Is (*foo*,1) installable?

```
Package: foo
Version: 1
Depends: baz (= 2.5) | bar (= 2.3),
  bar (> 2.6) | baz (< 2.3)

Package: bar
Version: 2

Package: baz
Version: 2
Conflicts: bar (< 3)
```

Is it installable in a future?

Yes, for example in a future with (*baz*,2.5) with no conflicts and *bar* with a version between 2.6 and 3.

Example 2: Will (*foo*,1) ever be installable?

```
Package: foo
Version: 1
Depends: baz (= 2.5) | bar (= 2.3),
  bar (> 2.6) | baz (< 2.3)

Package: bar
Version: 2.3

Package: baz
Version: 2.5
Conflicts: bar (> 2.6)
```

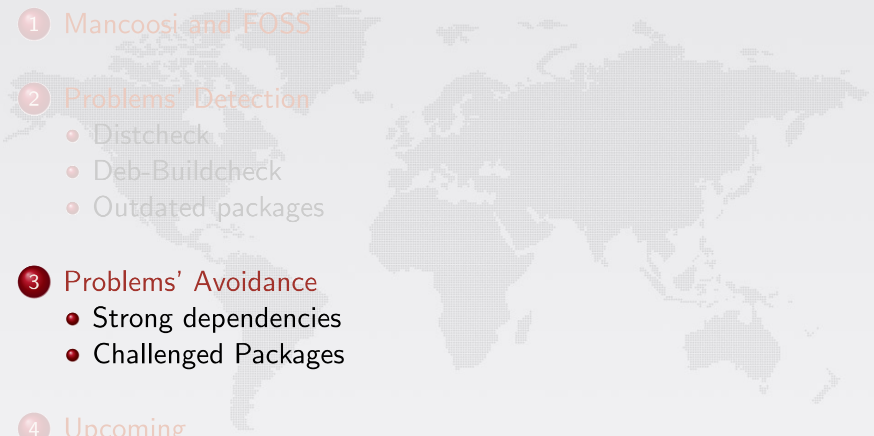
Is it installable in a future?

In this case this is not longer true, as if I choose baz $(= 2.5)$ this will conflict with any version of bar greater then 2.6 and I cannot choose bar $(= 2.3)$ because because baz is in the repository with version $= 2.5$

Demo again

A demo of an automatic tool is fun !

Outline



Outline

Problem with normal dependencies

The explicit, syntactic dependency relation $p \to q$ is too coarse grained to answer natural questions like:

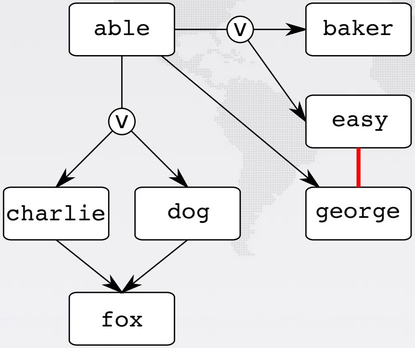
- *can I remove package p without affecting package q?*
- *how many / which packages will be affected by the upgrade of package p?*
    - Impact set: the set of packages potentially affected by changes in a given package.
    - Its size is the package sensitivity.
    - Direct dependencies: too little
    - Transitive closure of direct dependencies: too much

Answers do not depend on packages p and q only!
We need a stronger notion.

Strong dependencies

Definition

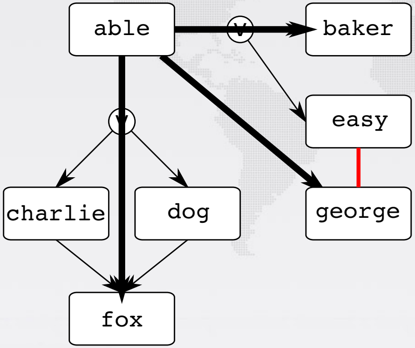
- p strongly depends on q with respect to R if it is not possible to install p without also installing q



Strong dependencies

Definition

- p strongly depends on q with respect to R if it is not possible to install p without also installing q



- `george`: conjunctive dependency
- `baker`: disjunctive, but `easy` not installable
- `fox`: dependency of both alternatives

Impact sets via *Strong* dependencies in Debian

Table: Debian Lenny, sorted by the size of the strong impact set.

Package	—IS(p)—
gcc-4.3-base	20128
libgcc1	20126
libc6	20126
libstdc++6	14964
libselinux1	14121
lzma	13534
libattr1	13489
libacl1	13467
coreutils	13454
dpkg	13450
perl-base	13310
debconf	11387
libncurses5	11017
zlib1g	10945
libdb4.6	9640
debianutils	8204
libgdbm3	8148
sed	8008
perl-modules	7898
perl	7898

Outline

Challenged Packages

- By looking at the impact set of a package P We can deduce which packages are affected by a problem in P.
- We want to know something more : if I upgrade a package P with version v to a future version w, how many packages are going to be affected by this upgrade ?
- We want to do this only by looking at the current repository and only using the available meta data information.
- We also consider clustering information to upgrade all packages in a cluster at once

Demo again ?

Nope, this one takes some time to run...

Results for Squeeze

Table: Top 20 cluster upgrades, by number of broken components

Source	Version	Target Version	#(BP)
perl	5.10.1-16	5.10.2 $<$. $<$ 5.12	2652
perl	5.10.1-16	5.10.1-16 $<$. $<$ 5.10.2	2652
perl	5.10.1-16	$>$ 006	2652
perl	5.10.1-16	5.12 $<$. $<$ 5.12.0	2651
perl	5.10.1-16	5.12.0 $<$. $<$ 006	2651
python-defaults	2.6.6-3+squeeze1	$>$ 3	1802
python-defaults	2.6.6-3+squeeze1	2.07 $<$. $<$ 2.008	1800
python-defaults	2.6.6-3+squeeze1	2.008 $<$. $<$ 3	1800
python-numpy	1:1.4.1-5	$>$ 1:1.5	542
pygobject	2.21.4+is.2.21.3-1	$>$ 2.21.4+is.2.21.3-1	522
pycairo	1.8.8-1	$>$ 1.8.8-1+b1	517
gtk+2.0	2.20.1-2	$>$ 2.20.1-2	482
udisks	1.0.1+git20100614-3	$>$ 1.1.0	417
eglibc	2.11.2-7	$>$ 2.12	395
eglibc	2.11.2-7	2.11.2-7 $<$. $<$ 2.12	382
ghc6	6.12.1-13	$>$ 6.12.1+	357
ghc6	6.12.1-13	6.12.1-13 $<$. $<$ 6.12.1+	357
libnotify	0.5.0-2	$>$ 0.5.0-2	331
ocaml	3.11.2-2	$>$ 3.11.2-2	252
apt	0.8.8	$>$ 0.8.8	219
haskell-mtl	1.1.0.2-10	$>$ 1.1.0.2+	173
haskell-mtl	1.1.0.2-10	1.1.0.2-10+b1 $<$. $<$ 1.1.0.2+	173
libdbi-perl	1.612-1	$>$ 1.612-1	172
pygtk	2.17.0-4	$>$ 2.17.0-4	129
libjpeg6b	6b1-1	$>$ 6b1-1	115
e2fsprogs	1.41.12-2	$>$ 1.41.12-2	115

Outline

Outline

Coints

- compute the installability kernel of a distribution
- it is used to identify the class of those packages that cannot be installed together
- author Jerome Vouillon
- http://coinst.irill.org/

That's all, folks.

Questions?

Pietro Abate
pietro.abate@pps.jussieu.fr
http://mancoosi.org/~abate

mancoosi
managing software complexity
http://www.mancoosi.org/