The DebianGis Project

Development and state of the art

“All your packages belong to us”

Francesco P. Lovergine <frankie@debian.org>

Affiliations:

CNR ISSIA (Italy)
Debian Project

Items of the talk

- Presenting the Debian Project, the DebianGis subproject and its goals.
- Where we are in development and support
- Infrastructure currently in use
- Tasks list for packaging (FHS,LSB, general and specific policies, good practices)
- Being a debian-compatible upstream
- What DebianGis experience could teach to other *something*-gis distributions
- Call for helpers
- Possible future activities and TODOs

The Debian project

Since 1993 Debian GNU/Linux is the very first community-based distribution based on an original package management system (dpkg -> .deb packages) and a public and totally open model of development. The development release has more than 10,000 source packages available and 11 architectures supported.

Currently the project counts ~1000 accounted developers (*maintainers*) all around the world and a good deal of *sponsored* (wonna-be-developers) contributors.

The project moved slowly from a one-man-show per package (single maintainer) to team maintainership. This is specifically useful for very complicated programs/suites and for improving general software quality.

Development roadmap

Three distribution branches:

stable (currently 3.1 aka Sarge): frozen, but for security updates.

testing (currently 4.0 aka Etch): the wanna-be-stable branch, semi automatically managed for inclusion/exclusion of packages due to the so called RC bugs. Currently should freeze around the end of this year.

unstable (aka Sid): where development really happens

"We release when we are ready!"

Debian sub-projects/derivations

In recent years some specialized Debian sub-projects or external projects were born to adapt the general distribution to more specific purposes or user targets.

Custom Debian Distributions (CDDs):

- DebianEdu
- DebianJunior
- ...

Derived Distributions:

- Ubuntu
- Linspire
- Knoppix
- ...

The DebianGis sub-project

http://lists.debian.org/debian-devel-announce/2004/10/msg00007.html

At the end of 2004, P. Cavallini and me issued a public manifest to startup team maintainership for all GIS related packages and creating a working group (both users and developers) to better focalize efforts on GIS applications.

This was just 8 months before 3.1 release.

Currently the project counts 130 subscribers to the general mailing list and 21 registered contributors. A few contributors are official debian developers. Most of the contributed packages are currently in the *main* pool (so will be in *Etch*).

All 'first class' applications or so are now packaged in main.

Why DebianGis?

- To provide a ready-to-go and stable distribution for end-users and developers interested in GIS applicatons.
- To well integrate and keep consistent all main free GIS tools and programs.
- To provide a first level of debugging help for users
- To return consistently feedbacks and hints to upstream teams about bugs, patches
- For world domination also in the GIS area :-)

The DebianGis infrastructure

But for main Debian facilities (BUILDD, BTS, archive, PTS, etc.) DebianGis uses the Debian Alioth Gforge site for all additional features:

- A general development mailing list:
http://lists.alioth.debian.org/mailman/listinfo/pkg-grass-general
- A commit and bugs only mailing list:
http://lists.alioth.debian.org/mailman/listinfo/pkg-grass-devel

- A subversion repository:
http://svn.debian.org/wsvn/pkg-grass
- An introductory wiki as main web site:
http://pkg-grass.alioth.debian.org/cgi-bin/wiki.pl

- IRC channel: #debian-gis on OFTC

Main packaging tasks

- Making maintainers scripts to perform pre and post installation or upgrade jobs
- Finding building and run-time dependencies, or conflicts
- Patching when needed for not in-sync building tools/libraries, uncommon architecture/platforms.
- Patching for Debian policies requirements
- Backporting relevant bug fixes from upstream development trees, whenever possible/suitable
- Providing first level documentation for Debian specific issues.

Relevant policies to follow

Main Debian Policy (currently 3.7.2)

apt-get install debian-policy

http://www.debian.org/doc/debian-policy/

Perl Policy

http://www.debian.org/doc/packaging-manuals/perl-policy/

Python Policy

http://www.debian.org/doc/packaging-manuals/python-policy/

Java Policy (now Java is distributable, packages go in *contrib* area)

http://www.debian.org/doc/packaging-manuals/java-policy/

Library packaging hints:
http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html

Tcl/Tk Policy (incoming, just started team maintainership)

Suggestions for upstreams

"How to make life of packagers easier"

Avoid custom/home-maiden building scripts

Autohell^Wtools are well supported in all distributions and per-architecture issues are already solved also on exotic platforms (GNU/Hurd or GNU/kBSD).
Cmake also has currently a better karma in the community: that should be considered probably for new projects.

Do not support only default paths for your depencies

LSB and FHS compliancy often require moving things around in the / tree. The same is done often to avoid conflicts with different versions of the same library/tool supported at the same time. Headers and library paths you require should be overridable at configuration time.

Suggestions for upstreams (2)

"How to make life of packagers easier"

Assign versions to your exported solibs and do that consistently
Better using libtool for portability among platforms. Anyway increase your soname versions, whenever your APIs (*any* APIs) changes in behavior/interfaces. Please document that :-)

Keep arch-dep and arch-indep stuffs apart at runtime (for paths) and at building time
All non meta distributions use autobuilders: that is required by policy in order to save CPU time and archive space.

Avoid pollution of headers and libs under /usr/include and /usr/lib
Use /usr/include/<your lib> and /usr/lib/<your lib> instead, versioned even for future portability.

Suggestions for upstreams (3)

"How to make life of packagers easier"

Do not embed a debian packaging directory in your sources
Upstream debian/ directories are obsolete by definition, we have to remove that and recreating the tarball from scratch. Better distributing that a part.

Providing a 'pkg-conf' like script is a Good Thing.
That simplifies things a lot for dependent packages when we do change paths, lib names, etc. for FHS and policy.

Do not embed third parties libraries in your sources. Static libs are evil, even.
We do package them a part and that practice is deprecated for security upgrades (no single point of upgrade and so on).

Known DebianGis issues

- Very few regular contributors and DDs.
- Debian release roadmap is typically much more slow than that of some prime time GIS applications.
- Development happens in sid, but many users work in testing: that creates misalignments and recurrent breakages for strictly interdepending packages (e.g. Grass vs Gdal vs Postgis, etc.) at user level.
- An always up-to-date archive of backports should be maintained because useful for stable users.
- More general QA tools focalized to the whole set of d-gis packages (*a la* packages.qa.debian.org) are required
- Few GIS contributors are pro developers
- We need more working people !

Possible enhancements

- Improving the DebianGis status report program to report RC bugs and other useful summary data for all relevant packages.
- Regular runs of *piuparts* to check installation/removing common problems
- Supporting backported packages with regular releasing plans
- Completing packaging of still missing and proposed applications (many Java-based still lack)
- Packaging special interest modules for common languages (e.g. GIS related modules at CPAN for Perl)
- Live CD ?
- Creating a true CDD for workstation and server use with a few core Debian applications and all useful GIS software installed (in progress now ...)

Please, join us !

questions

frankie@debian.org