Debian Pure Blends
Making Debian the distribution of choice for specific work fields

Andreas Tille

DudesConf

A Coruña, April 11, 2010
Overview

1. Introduction
   - History
   - Goals

2. Used techniques
   - Blends features
   - Web tools

3. Future
   - Planned features
   - TODO
Term Custom Debian Distributions was always misunderstood

Main misunderstanding: CDD was regarded as “something else than Debian” even if people were told that it is a concept inside Debian explicitly

Dropped the misleading name in favour of a name where you just have to read the docs

Debian Pure Blend (in short Blend): a subset of Debian that is configured to support a particular target group out-of-the-box.
Examples of Blends

- Debian Jr
- Debian Med
- Debian Edu
- Debian Science
- Debian EzGo, BrDesktop
- Debian Accessibility, DebiChem
- Debian Lex, Debian GIS
- Debian Multimedia?
- ...
Basic goal of Blends

- Debian > 22,000 packages
- Users interested in *subset*
- Groups of specialised users
- Easy installation and configuration
- While Debian stays general support specialists as well
- **No derivative** from Debian

**Basic idea: Do not make a separate distribution but make Debian fit for special purpose instead**
Upstream - Debian Developer - User

- Tie a solid network of Debian developers, upstream developers (“developing experts”) and users
- Rationale: Experts in this field need help in build system / packaging
- Upstream anticipates enhancements of build system and security audit
- Finally support upstream developers to become Debian maintainers
- Penetrating specific work fields with Linux makes it even more acceptable in general
Looking from outside

- Doctor and friend of mine:
  "Debian developers == ‘secret society’” 😊
- We know we are everything but secret
- At least one feature of secrecy: concealment
  - Concealment inside advertising noise of proprietary products
  - Concealment by disunity

→ Breaking the secret by advertising complete solutions
Attracting people to use Blends

Developers

- Acceptance of new methods higher if the techniques provided are convincing enough
- Simple way to categorise packages (“tasks files”)
- Key documentation feature
- QA pages (Bugs of relevant packages)

Users

- I18n-ed web pages displaying relevant packages
- Promoting software that builds a complete working environment
- Rise user interest by providing ready to install software in the context of their work field
Special applications
Building a set of metapackages

- Define set of dependency relations
- Verify availability of Depends / Recommends
- Packages unavailable in main will be turned into Suggests
- Create proper debian/control file to build valid metapackages
- Create tasksel control file <BLENDF.Tasks.desc>
Similar to *debian/control*

**Task:** *taskname*

**Description:** *Shortdescription*

  *Longdescription*

**Depends:** *some dependant packages*

**Recommends:** *some recommended packages*

**Suggests:** *some suggested packages*
Verify availability of \texttt{Depends / Recommends}

Turn \texttt{Depends} into \texttt{Recommends}

Packages unavailable in \texttt{main} will be turned into \texttt{Suggests}

Create proper \texttt{debian/control} file to build valid metapackages

Create \texttt{tasksel} control file \texttt{<BLENDE>-tasks.desc}
Tasks and bugs pages

- Providing information about packages of interest
- Reading tasks files from Blends SVN containing
  - Dependency relations of packages inside Debian
  - Preliminary package information / WNPP
- Gathering all available information about the package dependencies defined in the tasks file
Intention of tasks pages

- Key entry point for users
- Quick overview about what’s inside Debian regarding their specific work field
- Turned out to be QA tool for developers as well
- Meta information like
  - Homepage
  - Maintainer and VCS of Debian packaging
  - Screenshot (http://screenshots.debian.net)
  - DEHS, versions and architectures
  - DebTags
  - Popcon
  - even scientific quotation if available

Demo http://blends.alioth.debian.org
Weighting bugs

- Try to find a measure for bugs of dependant packages
- Currently not normalised to the number of dependencies but rather regarding absolute number of bugs
- Weighting numbers for the different severities ranging from 10 for the RC bugs until 0 for wishlist bugs

Example calculation

1 serious bug in dependent pkg: \(1 \times 10 \times 3 = 30\)
2 important bugs in dependent pkg: \(2 \times 5 \times 3 = 30\)
1 important bug in suggested pkg: \(1 \times 5 \times 1 = 5\)
1 normal bug in dependent pkg: \(1 \times 3 \times 3 = 9\)
1 minor bug in dependent pkg: \(1 \times 1 \times 3 = 3\)

weighted sum = 77
Colouring according bugs weight

<table>
<thead>
<tr>
<th>assessment</th>
<th>limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>5</td>
</tr>
<tr>
<td>verygood</td>
<td>10</td>
</tr>
<tr>
<td>good</td>
<td>30</td>
</tr>
<tr>
<td>satisfactory</td>
<td>50</td>
</tr>
<tr>
<td>pass</td>
<td>70</td>
</tr>
<tr>
<td>bad</td>
<td>100</td>
</tr>
</tbody>
</table>

- Metapackage can not be in status "good" if there is at least serious (or higher) bug in a dependant package
- Not "very good" if there is a RC bug in a suggested package
- Two RC bugs in suggested packages might qualify for "good" - if there are only a very view other bugs
More QA overviews

- Lintian report overview
- Adding Ubuntu bugs
Make *blends-dev* use UDD

- Build metapackages based on UDD information
- Thus enabling `architecture=any` metapackages
- Include tasks file information into UDD
- I18n information of applications
Try to establish technique

- Further enhancements
- Rewrite *blends-dev* to use UDD
- Make even more projects like DebiChem and Debian-GIS actively using the framework
- Try to bring back external projects to Debian by providing attractive tools
This talk can be found at http://people.debian.org/~tille/talks/
Andreas Tille <tille@debian.org>