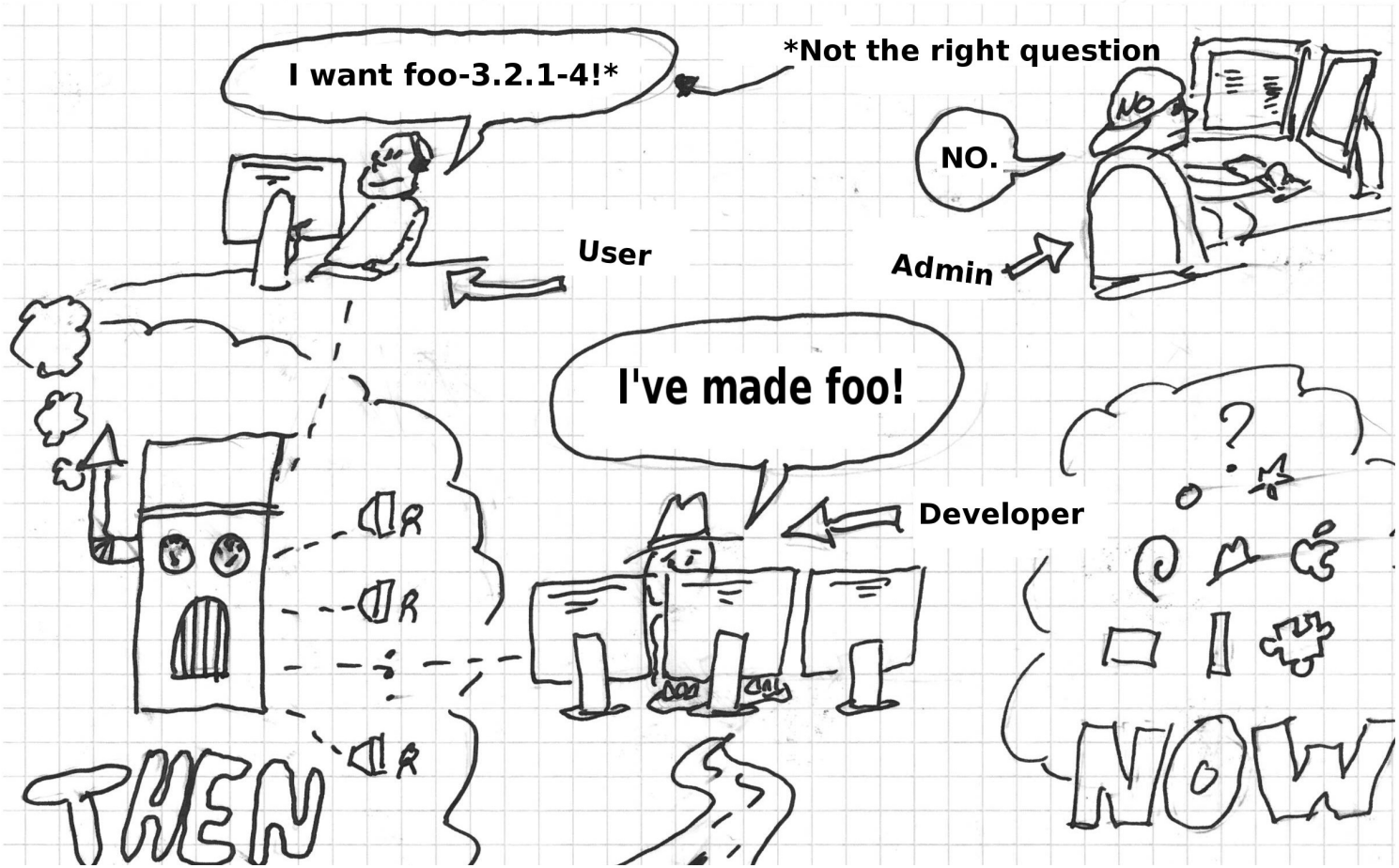
Experiences with moving to open source standards for building and packaging

Dennis van Dok, Mischa Sallé and Oscar Koeroo

Nikhef Amsterdam

CHEP 2013 Amsterdam

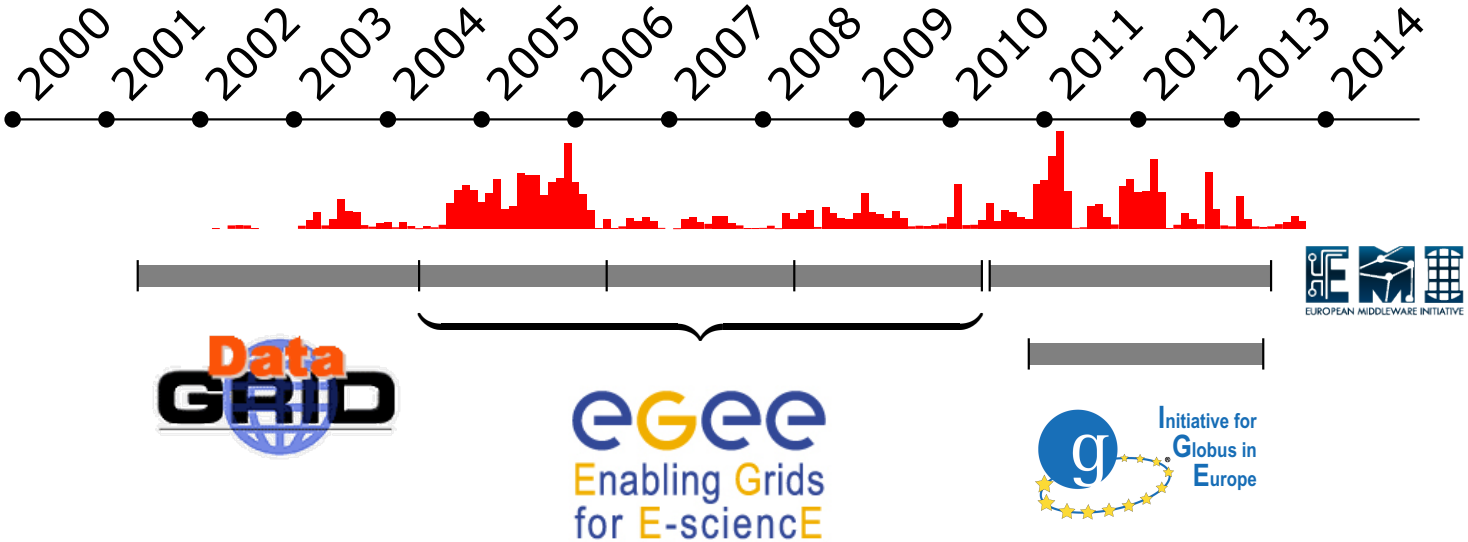
Software midwifery



Software midwifery II

Where we come from

- Middleware contributions in a series of EU funded grid projects: DataGrid, EGEE (I, II and III), EMI and IGE.
- current sustained 'maintenance mode' through SURF e-infrastructure.



The ETICS era

The EGEE era saw increased scale in every way:

- the EGEE Grid
- the middleware stack
- the code complexity and interdependencies

EGEE II had to deliver more reliable and stable software. To manage this, a build system was introduced called **ETICS**.
This system had a few shortcomings.

ETICS' shortcomings

- It was not easy to build outside of ETICS
- Building in ETICS was too slow for debugging/test cycles
- The system required specific `m4` macros to build
- There was little or no focus on portability
- The output consisted of binary products and no rebuildable source material.
- The ETICS project stopped in 2013 and the portal went off-line.

We had to do things ourselves.

What we did

code improvements

- better `m4` macros for dependency discovery
- stick to standards (ANSI C, POSIX, Autotools)
- Build without compiler warnings
- Implement secure coding standards
- active portability testing to many platforms (solving bugs along the way)
- documented software process

what we also did

version control: imported all CVS history for our components from CERN CVS, with full history

packaging and building: wrote Fedora and Debian compliant packaging sources, built native packages suitable for inclusion in mainstream distributions

Implementation of guidelines: adopt Fedora packaging guidelines and Debian Policy

automation: set up Koji for automated building

software delivery: deliver software through our own (signed) repositories

improved documentation in the form of manpages and Wiki pages.

What we (eventually) managed

Most (autotools based) open source software can do this out of the box:

```
./configure
make
make install
```

We actually stuck to the mantra:

```
make distcheck
```

which builds outside the source tree and tests if an install with `DESTDIR` works.

Fedora packaging

- Tagging in SVN (of the `SPEC` file) triggers a Koji build
- Koji does mock builds of the source and binary RPMs for all targeted platforms, i.e. the latest Fedora releases and EPEL5 and EPEL6.
- The builds are signed with a tool called *sigul.*
- The *mash* utility generates the repositories that can be installed through yum

koji, sigul and mash are also used by the Fedora project.

Debian packaging

For Debian, the procedure is slightly different. There is no equivalent of Koji for Debian ☹.

- a Debian source package is created for currently supported Ubuntu versions, Debian unstable, stable and oldstable,
- each source package is build with cowpoke/cowbuilder/pbuilder (equivalent to mock).
- the resulting packages are signed with the packager's GPG key
- The package is uploaded to a software repository from where it can be installed with apt-get.

Catching common errors

For Fedora, use `rpmlint`; for Debian, `lintian` to see if packages do not contain silly mistakes (lintian helped find several common spelling errors.)

The automated build logs revealed more warnings due to using different compiler settings.

This is not a substitute for real testing, of course.

Communi{ty,cation}

- Mailing lists

  We've set up a few mailing lists:

  - grid-mw-security-support@nikhef.nl for support questions and
  - grid-mw-security-announce@nikhef.nl for announcements of new versions. This list has an open subscription policy.

  No general discuss mailing list (yet)... no sizable community either (besides wLCG/EGI there is OSG).

Sources, binaries and bugs

- Version control in SVN
  https://ndpfsvn.nikhef.nl/viewvc/mwsec/
  https://ndpfsvn.nikhef.nl/ro/mwsec/
- Download sources
  http://software.nikhef.nl/security
- RPM/Deb distribution
  http://software.nikhef.nl/dist
- Bug tracking

  NEW: https://bugzilla.nikhef.nl/, moving away from CERN Savannah.

Why we did it

Changes were implemented gradually over time, with each step bringing new benefits.
Being good netizens and playing along with common open source practices is rewarding even without drawing a crowd.
Supporting OSG was much easier once we took control of the process.
We believe we have greatly improved sustainability of our software.

What we got in return

Some of the benefits our work rendered:

1. playing fair with package management avoids conflicts
2. installation of software becomes trivial
3. reproducing bugs becomes easier
4. pin-pointing bugs to source lines is easier
5. cycle time to deliver updates becomes shorter
6. uncovered some lurking bugs
7. improved portability
8. easier integration with third parties
9. using common technology makes it easier to pass the support to future staff members.

Last slide

This slide is intentionally left blank.

How to tell if a FLOSS project is doomed to FAIL

Our overall fail score: 55 points.

References

- Guide to setting up the Koji build system
  http://fedoraproject.org/wiki/Koji/ServerHowTo
- Nikhef Security Access Control sofware procedures
  https://wiki.nikhef.nl/grid/SAC_software_procedures
- Fedora packaging guidelines
  https://fedoraproject.org/wiki/Packaging:Guidelines
- Debian Policy
  http://www.debian.org/doc/debian-policy/
- Debian upstream guide
  https://wiki.debian.org/UpstreamGuide