Community Driven Sci-Soft Projects

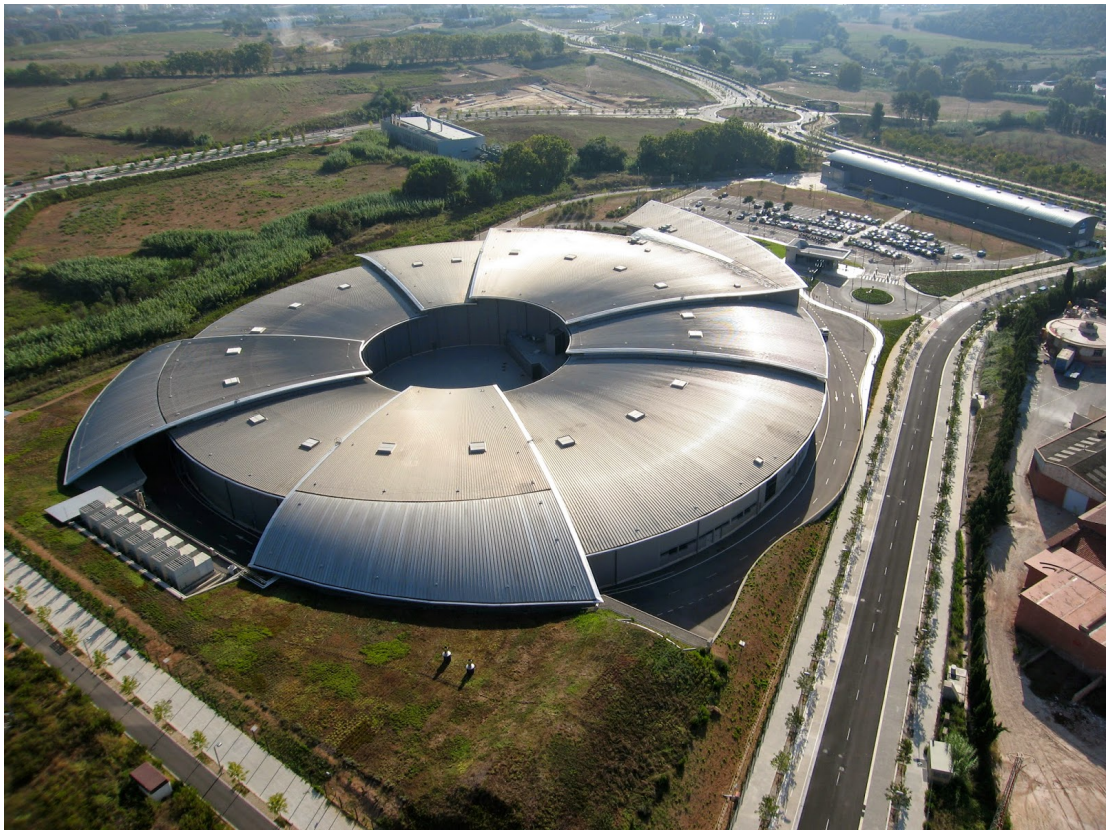
lessons learned on tools and practices

by

Carlos Pascual-Izarra

(On behalf of ALBA's GenSoft group and Taurus/Sardana communities)

ALBA is...

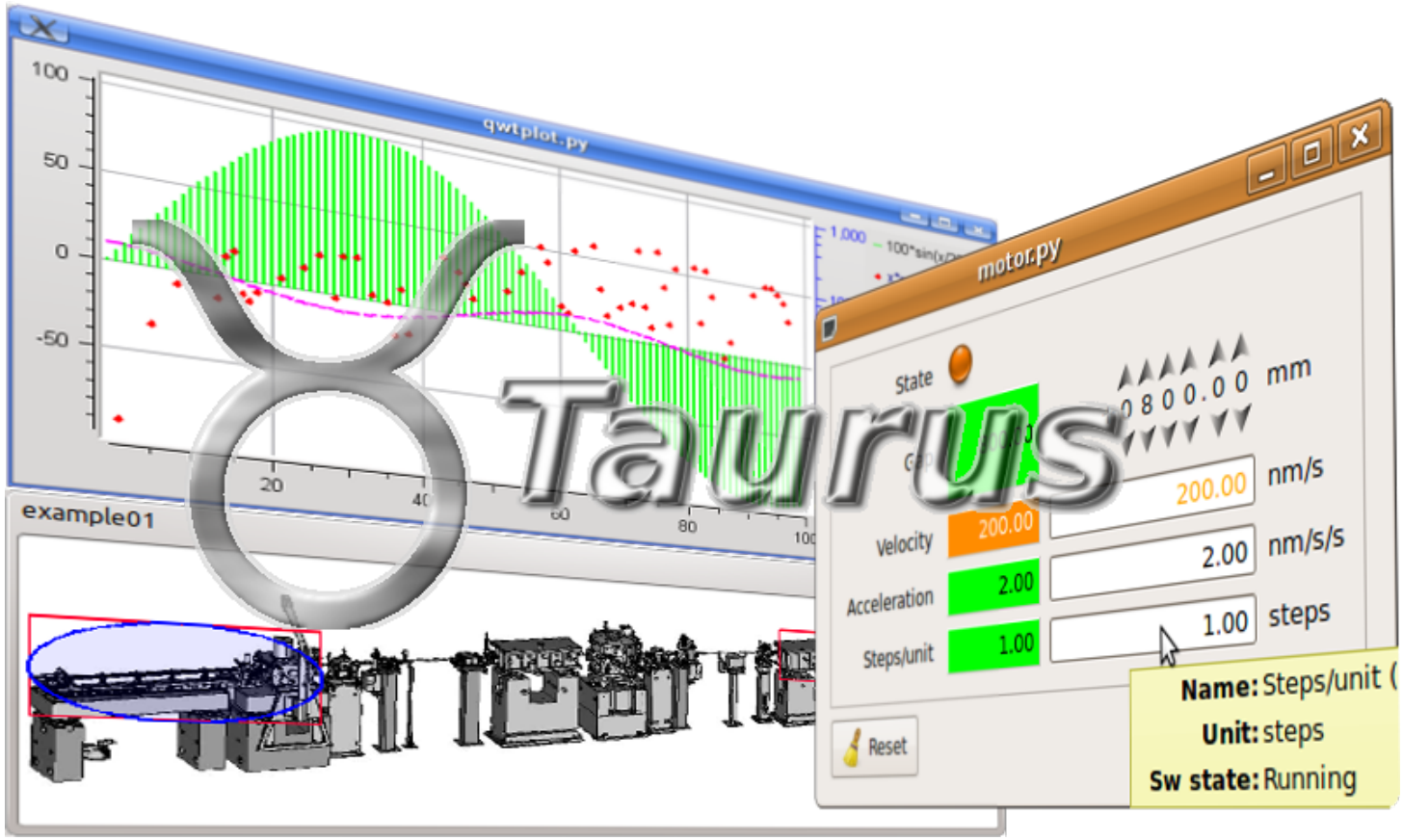


ALBA *is a 3rd generation Synchrotron Radiation Facility built in* ***Barcelona*** *and in operation since* ***2010****.*

It currently operates ***8 beamlines*** *and 2 more are under construction.*

The control and data acquisition software for its accelerators and Beamlines is based on ***Tango*** *(with* ***Taurus*** *and* ***Sardana****)*

Taurus is...



Taurus *is a framework for building control and data acquisition* ***CLIs*** *and* ***GUIs***

It is based on ***Python*** *and extends* ***PyQt***

*It supports plugins for various control systems (****Tango****,* ***EPICS****,...) or data sources (****HDF5****,* ***Python eval****,...)*

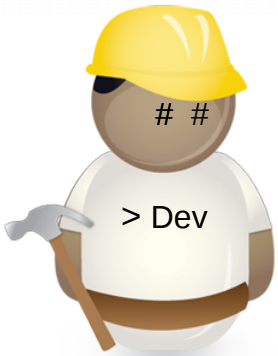
Sardana is...

Sardana *is a SCADA for scientific installations originally developed at ALBA.*

It is built on top of ***Taurus*** *and* ***PyTango****.*

It provides ***automation*** *of procedures and* ***synchronization*** *in a distributed control system.*

Taurus & Sardana Communities



ALBA

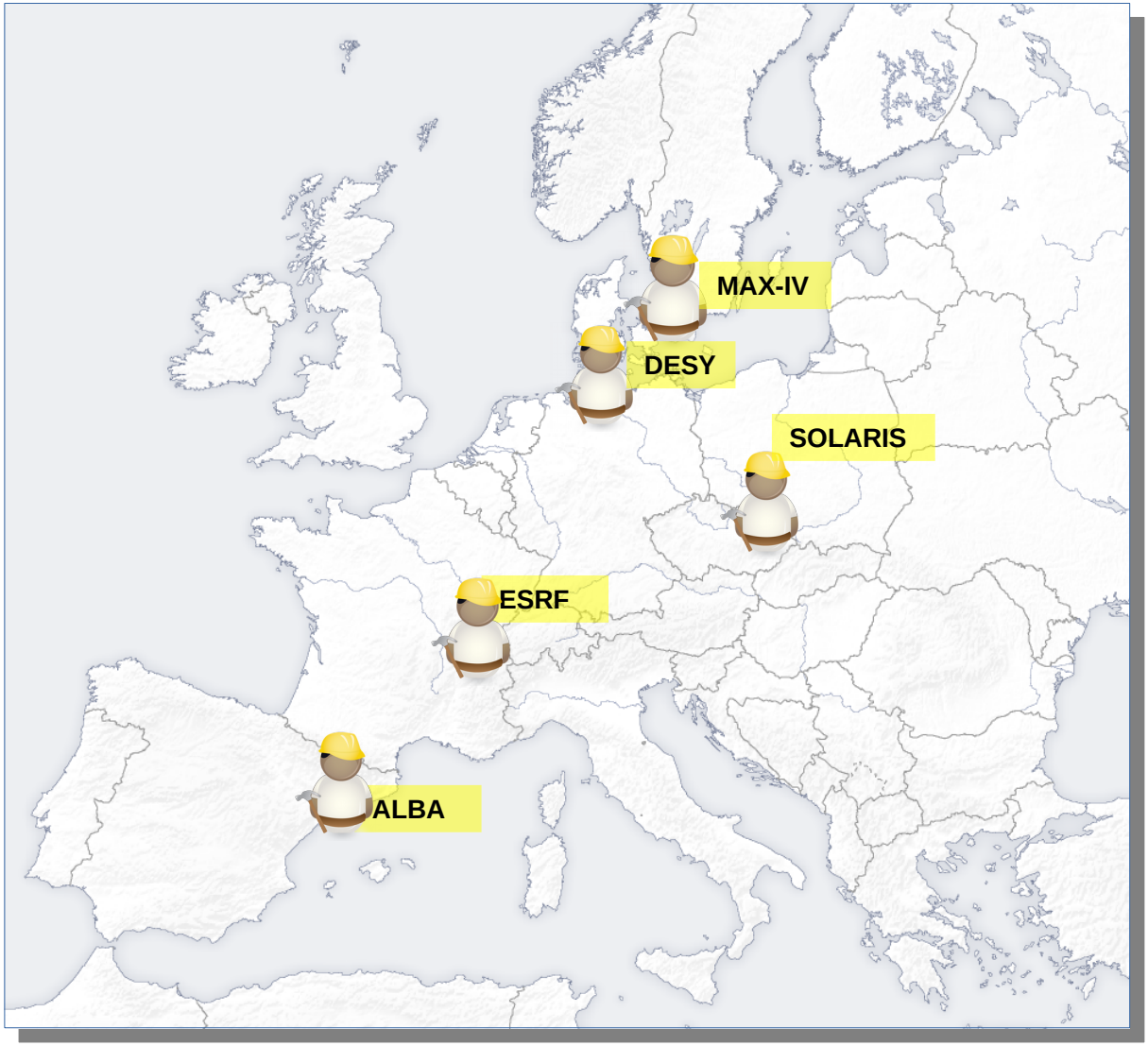
DESY

ESRF

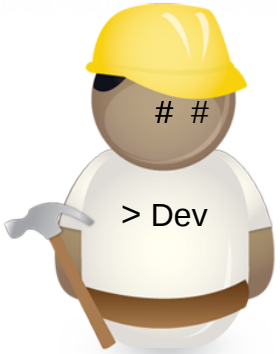
MAX IV

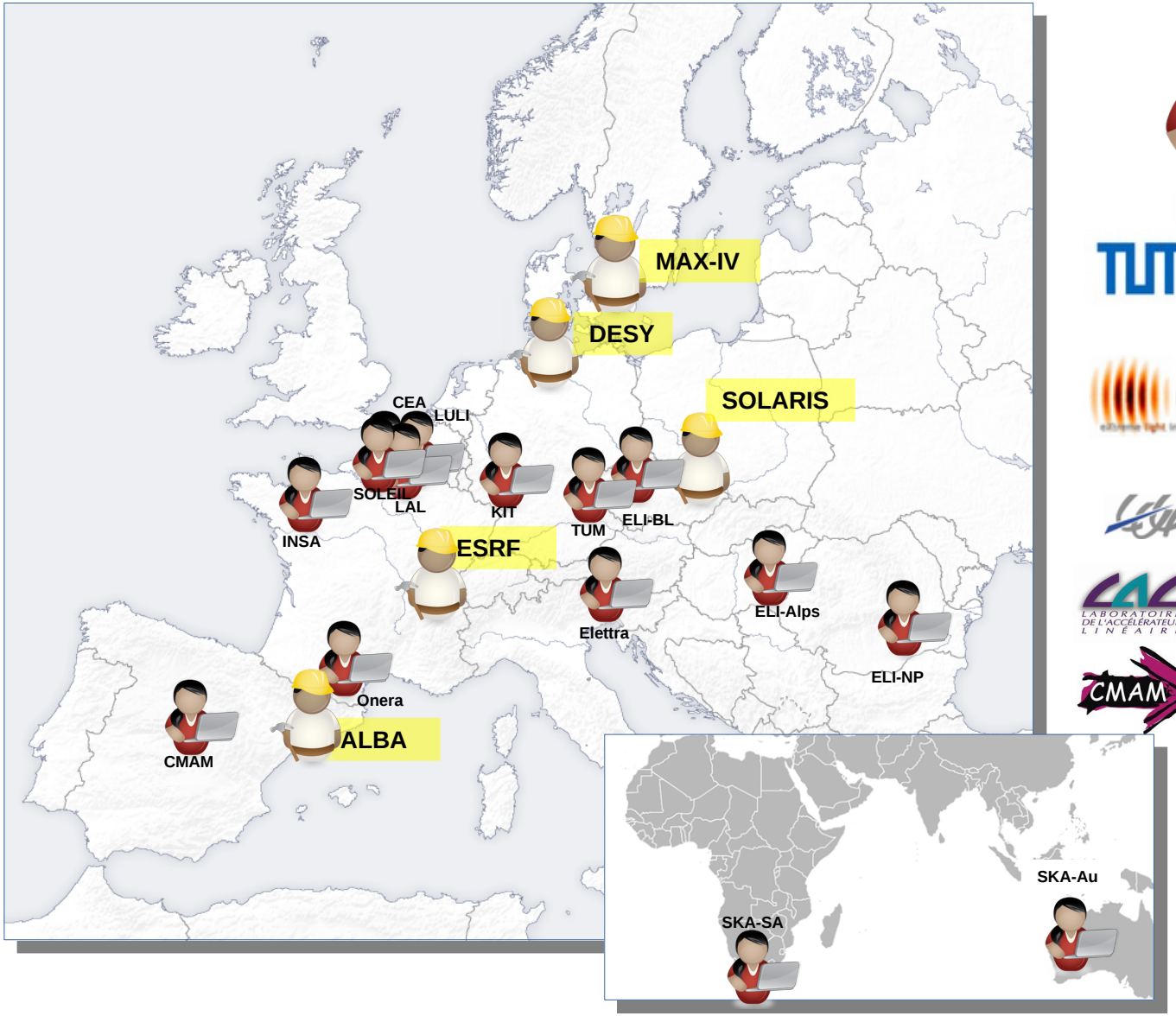
SOLARIS
NATIONAL SYNCHROTRON RADIATION CENTRE

debian

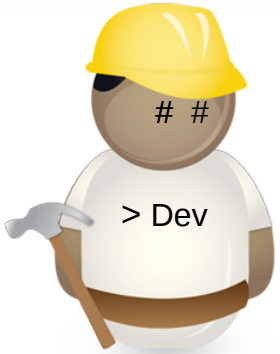


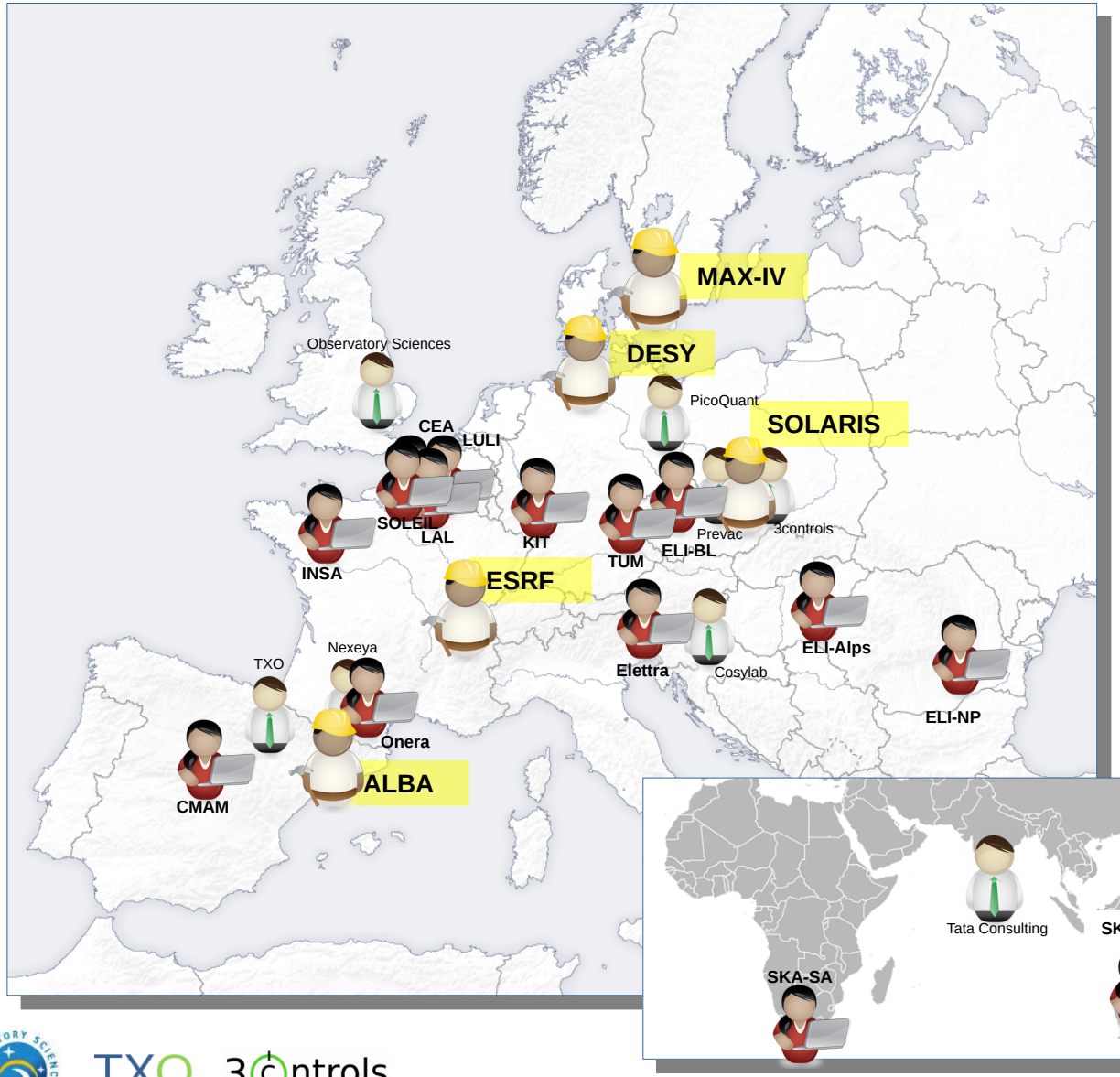
Taurus & Sardana Communities



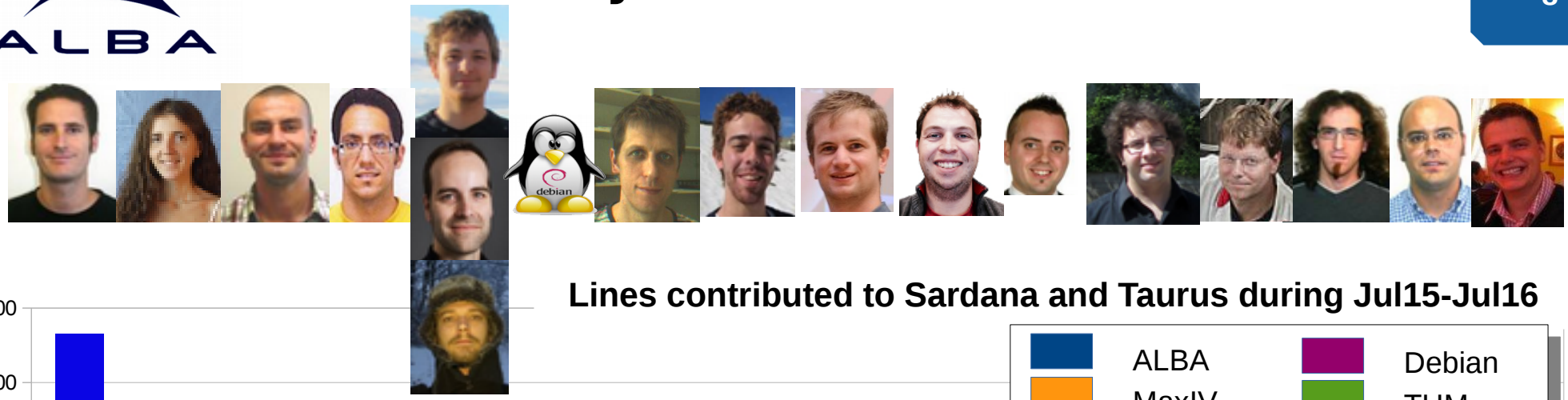


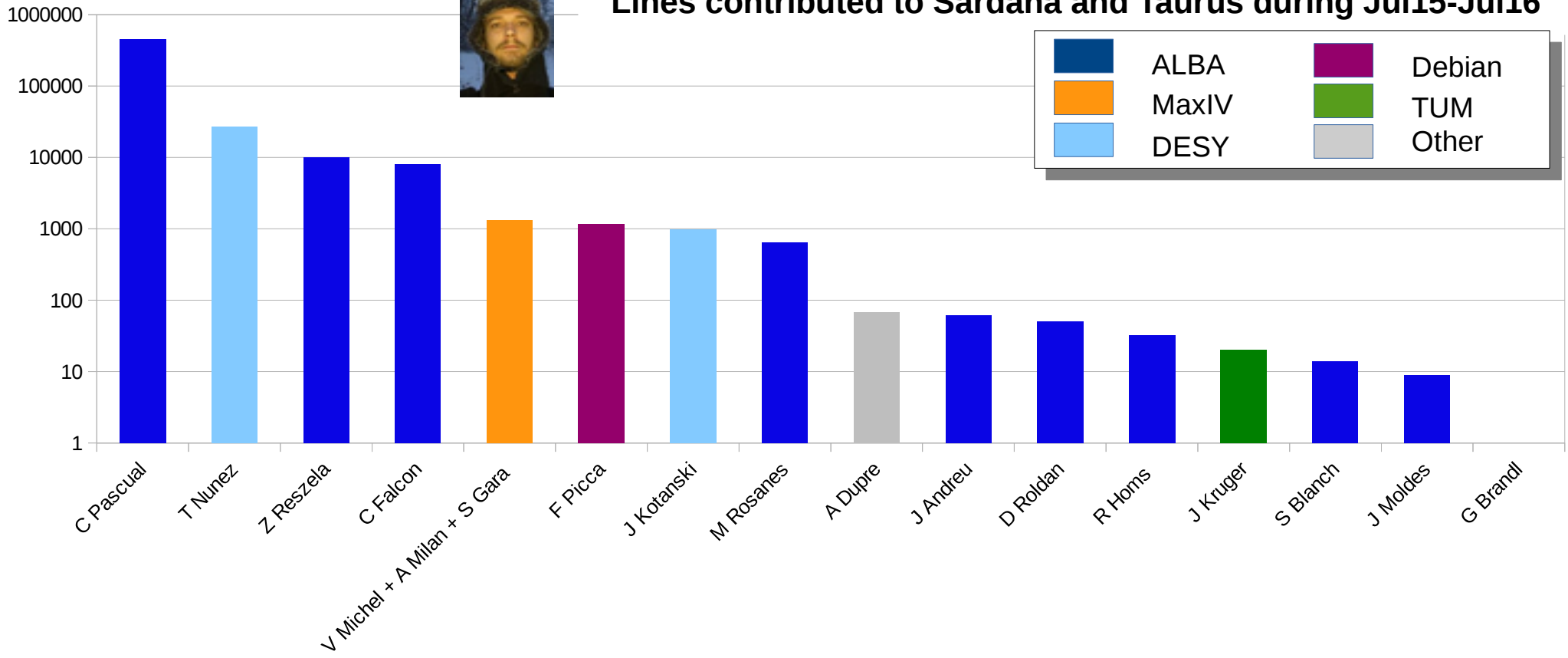
Taurus & Sardana Communities





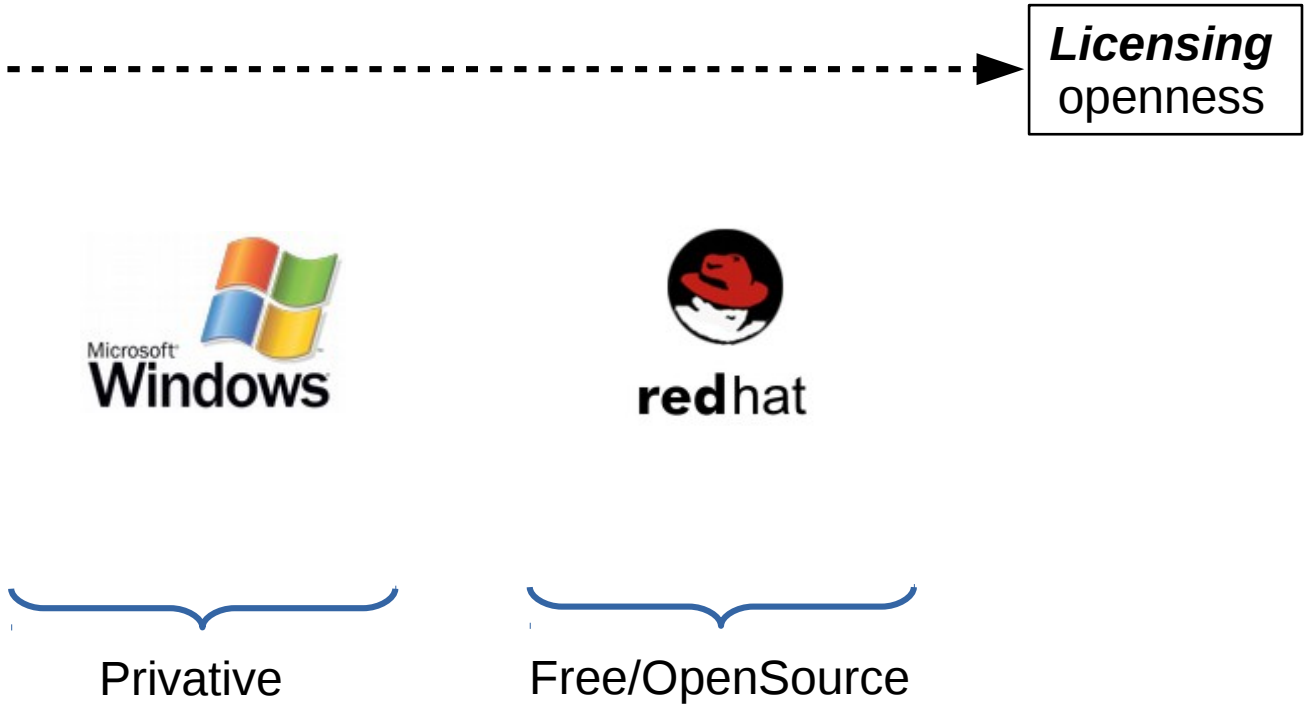
Last year's contributors



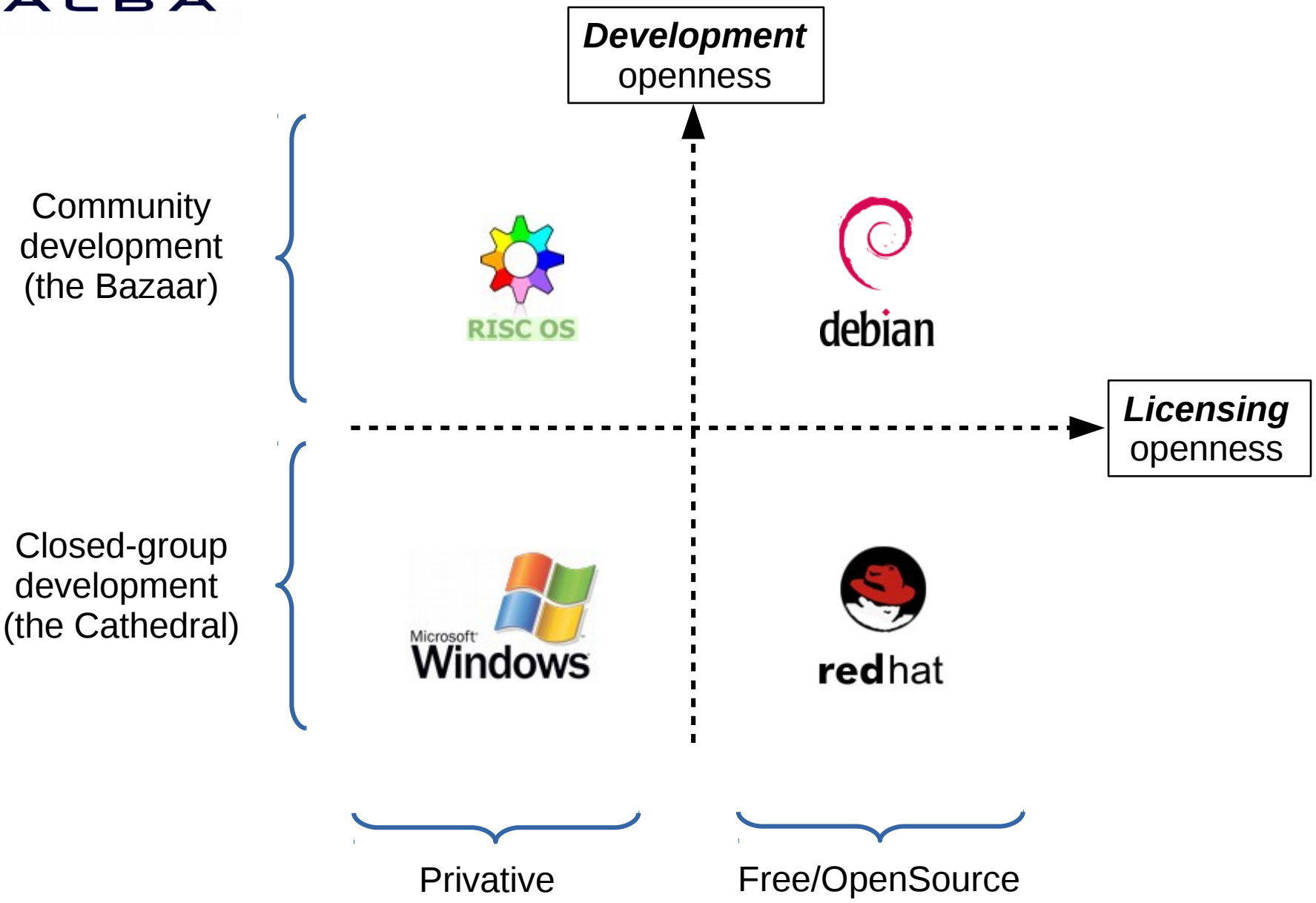


Credits are also due to former contributors, esp. T Coutinho (original author),
F Picca (Debian Maintainer) and the users (for reporting bugs and requesting features)

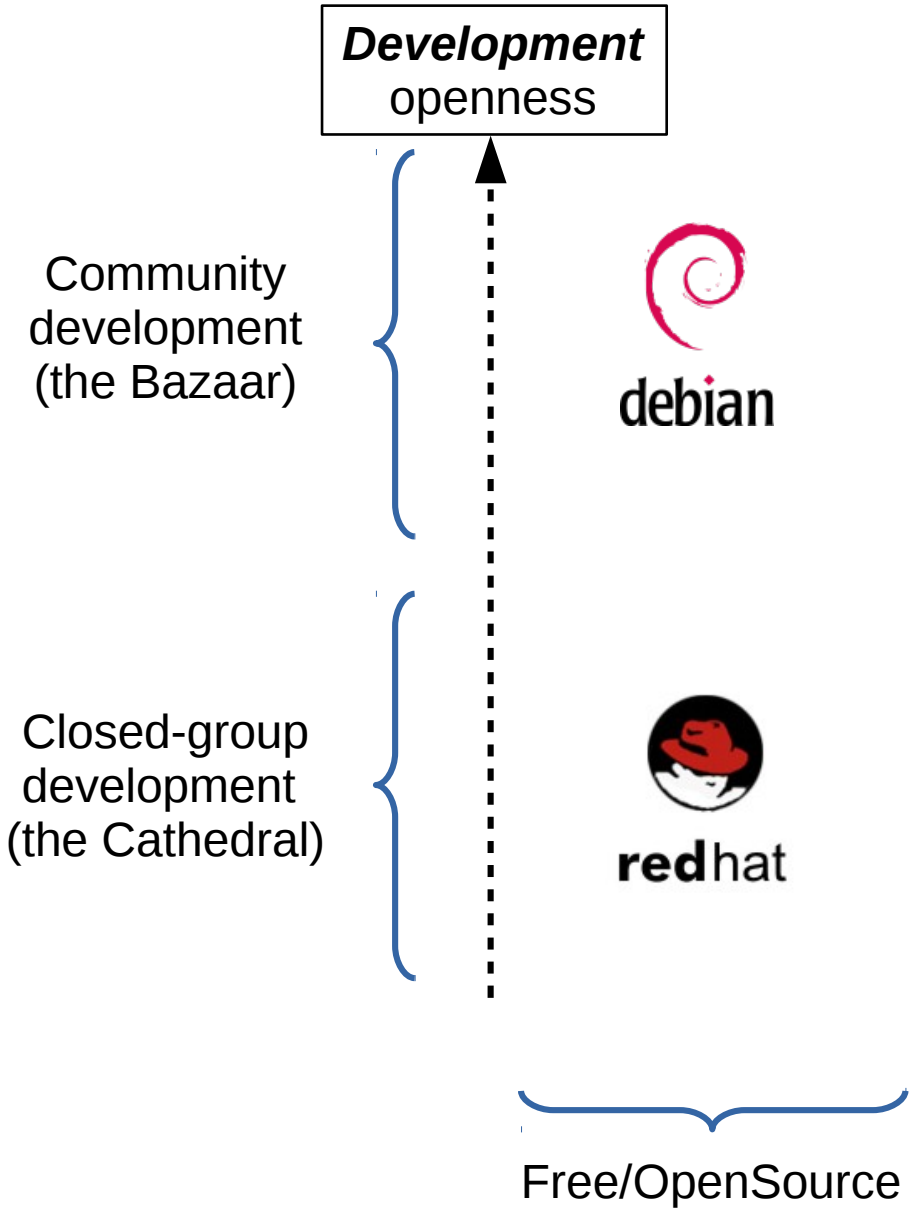
Open Development vs Open Source



Open Development vs Open Source

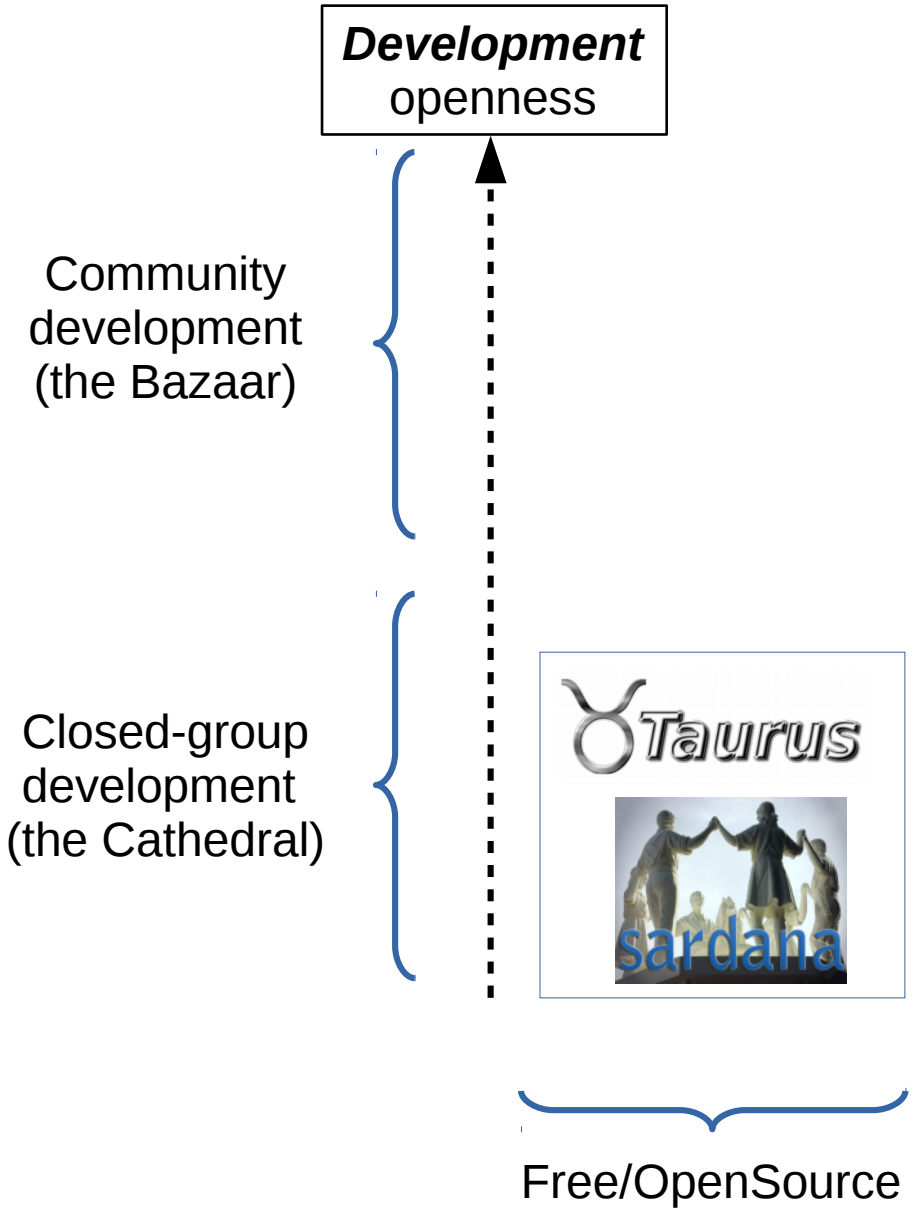


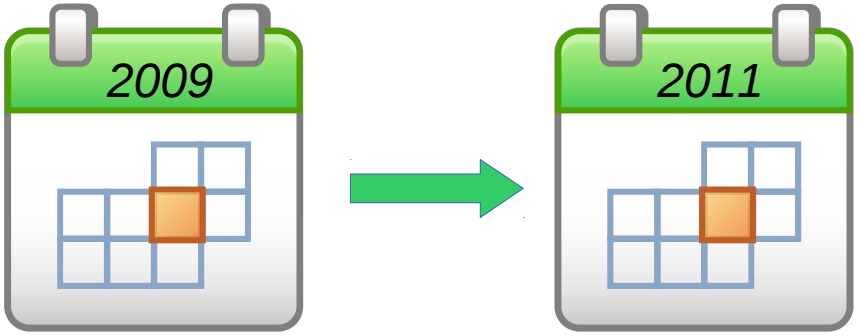
Why Open Development?



- ✔ project resilience
- ✔ code quality (more eyes to spot bugs)
- ✔ more contributors (agility and generality)
- ✔ more exposure
- ✔ encourages good practices
- ✖ time to setup (tidy up, upload, write policies,...)
- ✖ time to coordinate
- ✖ time to support new members
- ✖ time to review contributions
- ✖ less control over project direction

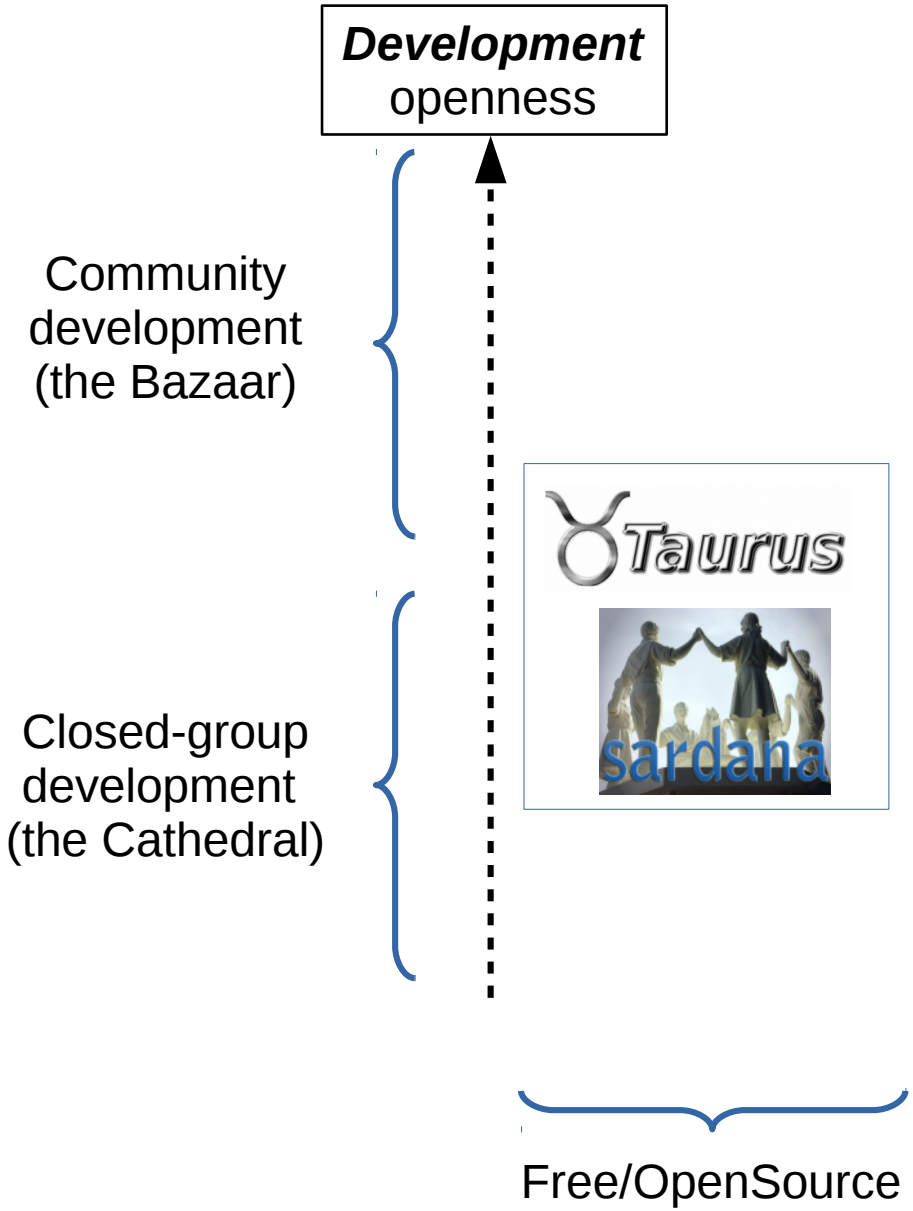
Our walk to the Bazaar

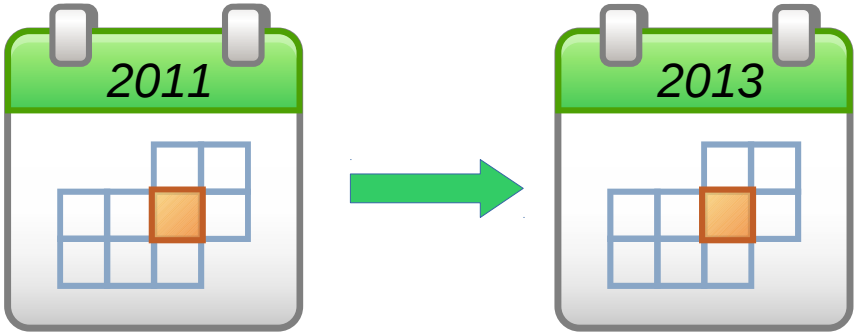




- In-house development
- Code was “Free”, but license unclear
- Code in **internal** ALBA SVN repository

Our walk to the Bazaar





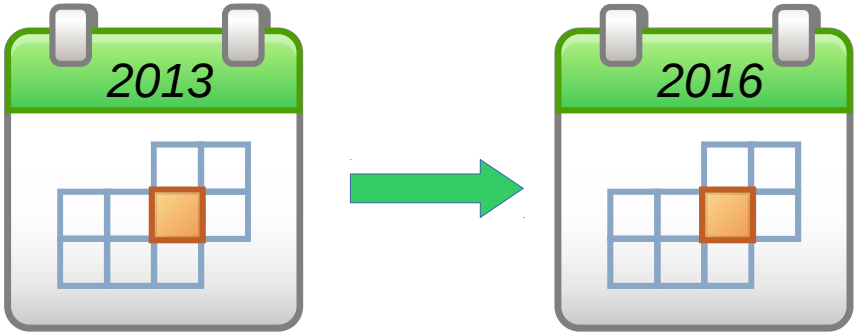
- ✖ In-house development
- ✔ License set as LGPL
- ✔ Code moved to SourceForge SVN

Our walk to the Bazaar

Community development (the Bazaar)

Closed-group development (the Cathedral)

Free/OpenSource



- Community-driven development:
  - Enhancement Proposals
  - Public code review
  - Mailing lists
  - ...

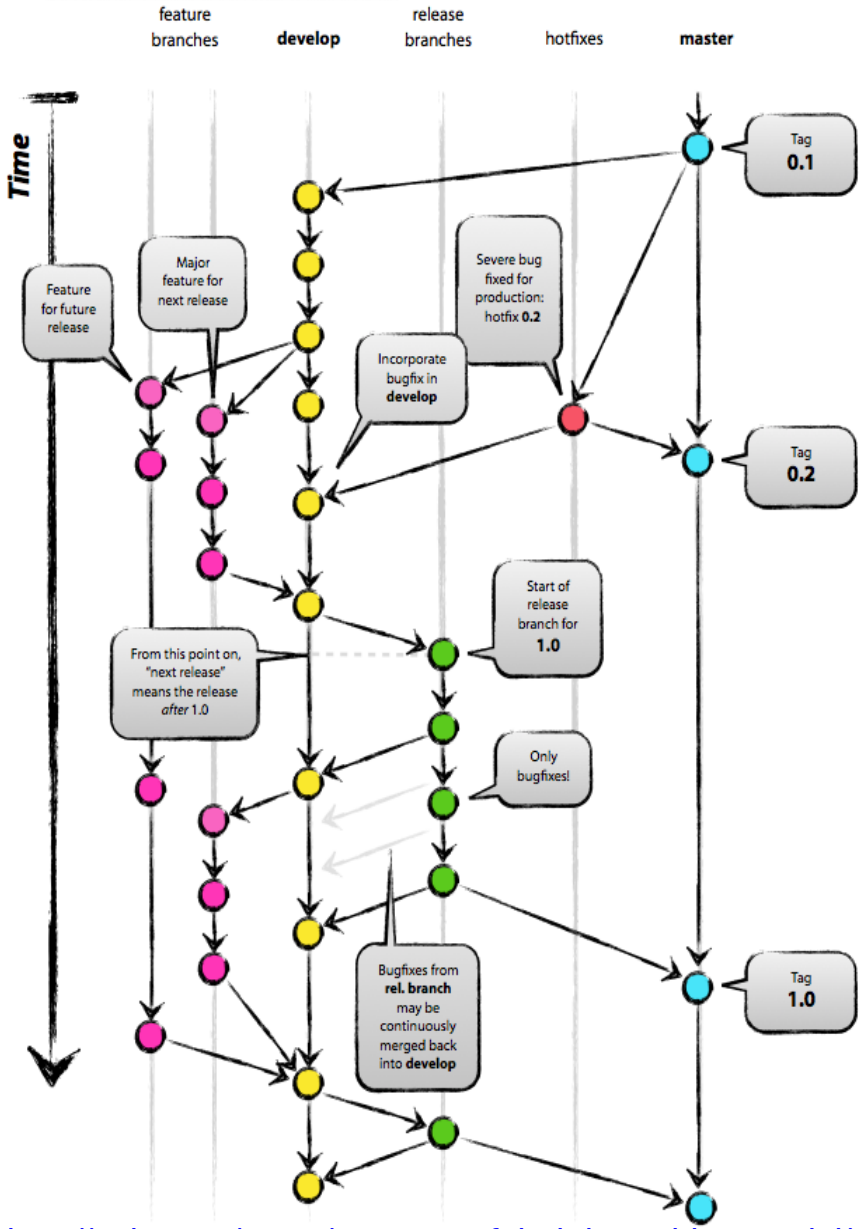
Our experience with...

...stuff that worked and stuff that didn't

Our experience

Requirement	What we tried	OK?	Notes / lessons learned
Formal collaboration agreement	Institutional MoU	No	Never signed. Ignore it (enjoy the Bazaar!)
Community meetings	Yearly workshop + google hangouts	Yes	Google hangouts rarely used
Release cycle	Fixed *Jan* & *Jul* releases	Yes	Will Continuous Delivery change this?
Code modularity	Several *ad-hoc* plugin systems	mostly	Change to a unified and standardized system (setuptools entry-points / stevedore?)
Code repo organization	Git (at SourceForge) + gitflow	Yes	Feature branches are great!
Enhancement Proposals	Formal (inspired on Debian-EP)	Yes	Keep scope small. Use Pull Requests?
General discussion channel	mailing lists (-devel and -user) + SourceForge tickets	mostly	SF mailing list & tickets do not integrate well. devel list bloated with admin email
Automated tests	PyUnit + Docker	Yes	Simplifies code review. Enables TDD
Code Review	Public, mailing list-based (git format-patch + git send-email)	mostly	Pull requests are simpler for contributors and lighter for integrators
Continuous Documentation	ReadTheDocs	mostly	Mocks are a PITA (use Travis + Docker?)
Continuous Integration	Travis with Docker (+ Appveyor)	?	Pre-tests ok. Appveyor to be tested
Continuous Delivery	Travis + Appveyor + github releases?	?	Ongoing pre-tests

Code Repo organization



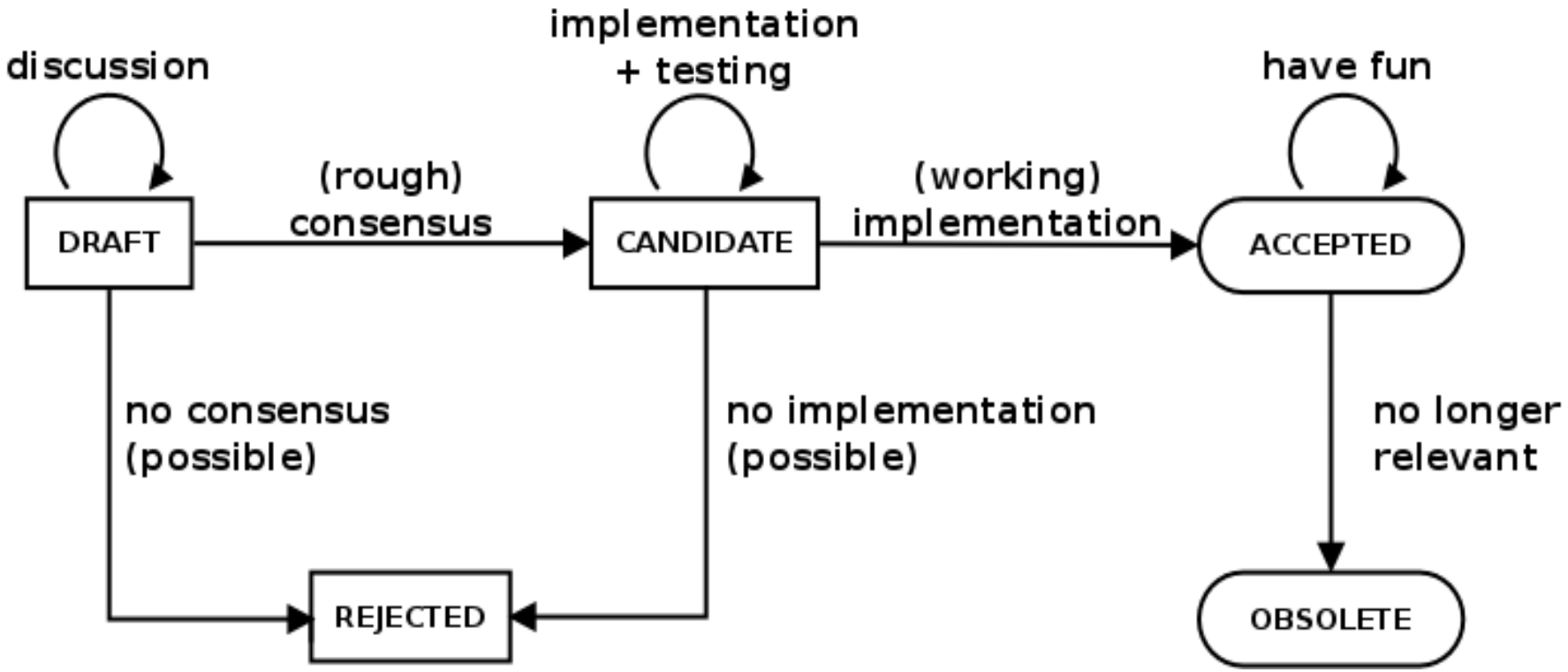
http://nvie.com/posts/a-successful-git-branching-model/

- The key is to use **feature branches**
- Many workflows are ok:
  - **githubflow** is the simplest. Good for CD
  - **gitflow** fits well with fixed releases (our choice)
  - **DMZflow** scales better

http://sf.net/p/sardana/wiki**/SEP7**

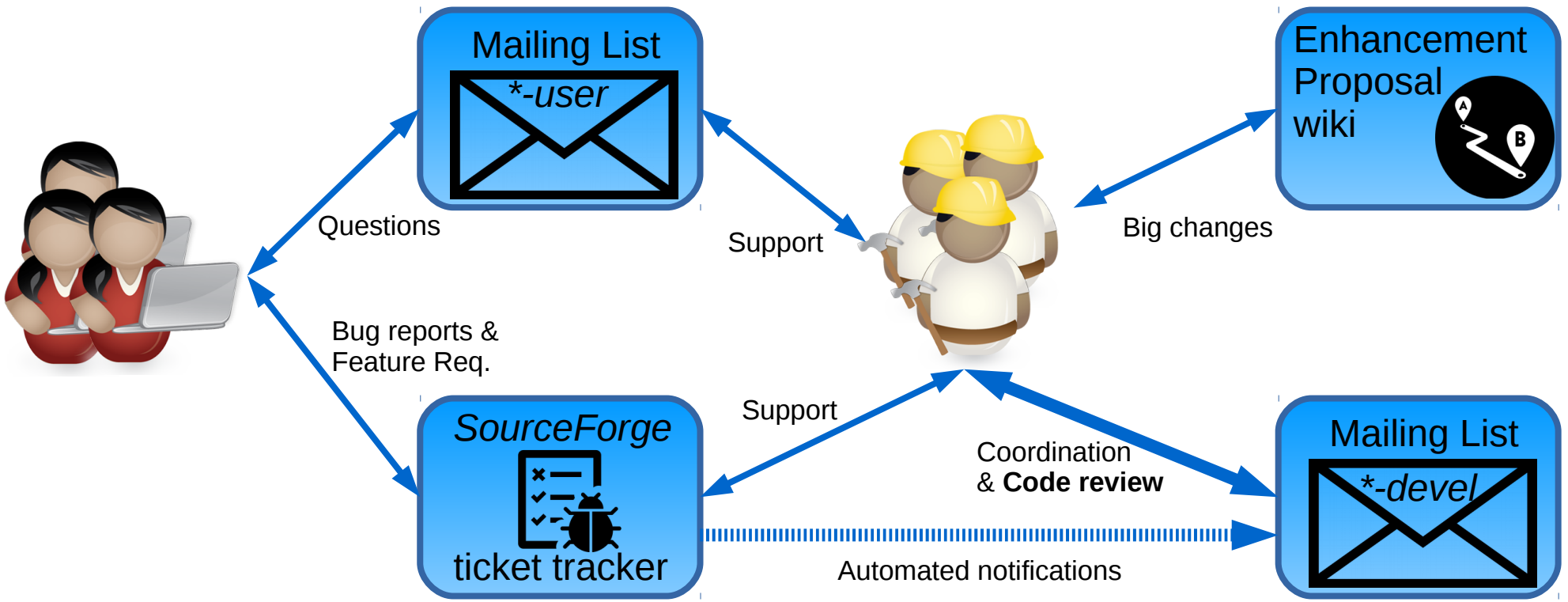
Enhancement Proposals

- The preferred way for introducing major **new features**
- Inspired by Debian DEPs and Python PEPs
- Promotes **shared decission-making** and **good documentation**



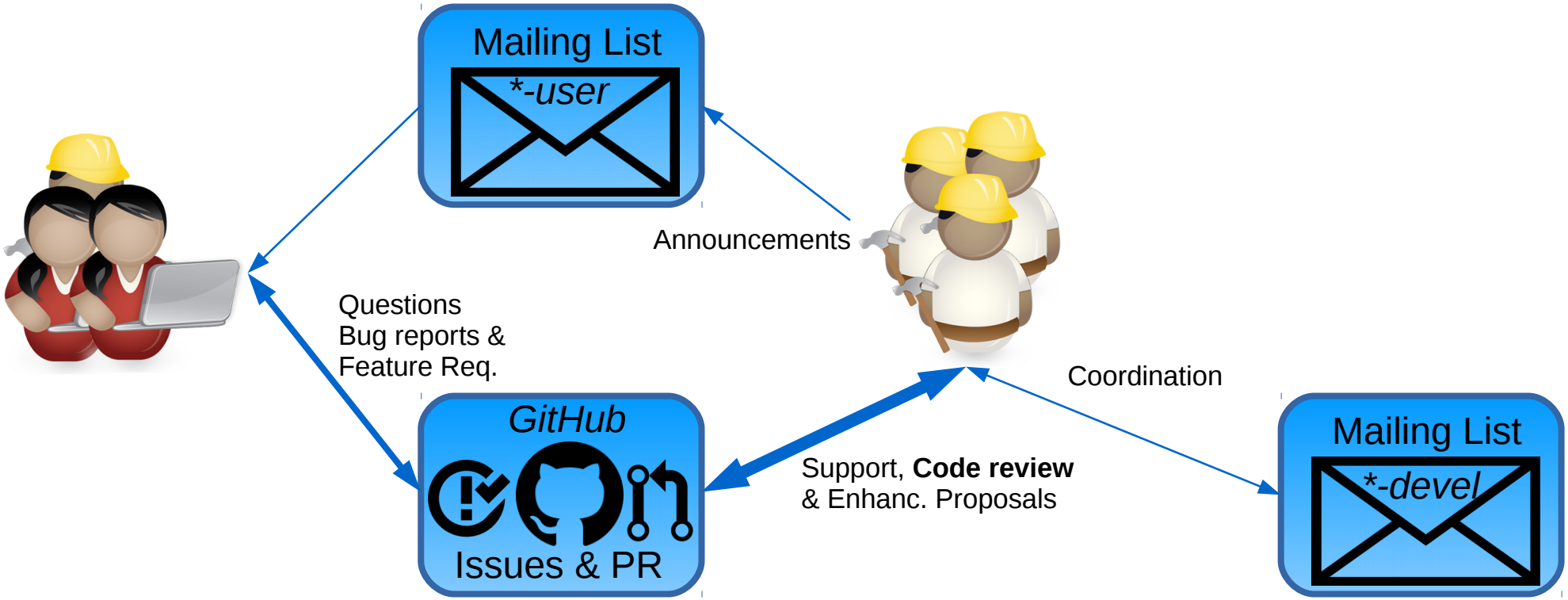
http://sf.net/p/sardana/wiki/SEP0

Communication channels (current)



- ✔ Everything ends in a mailing list archive
- ✖ SourceForge tickets do not integrate well with email (cannot email to tracker)
- ✖ Devel mailing list ends up **bloated** with admin email

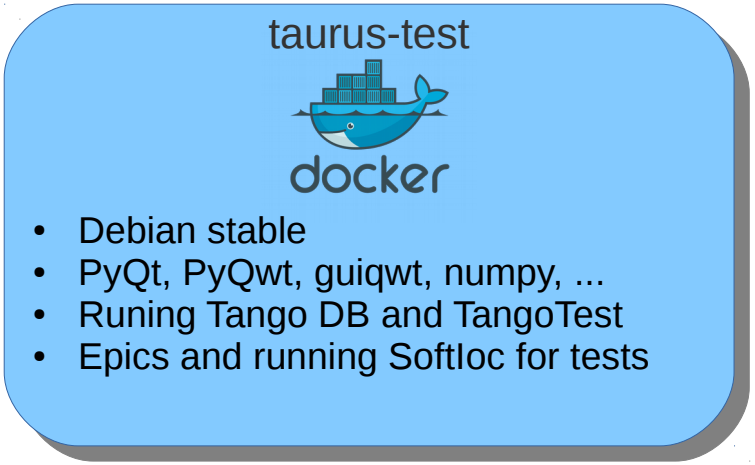
Communication channels (coming)

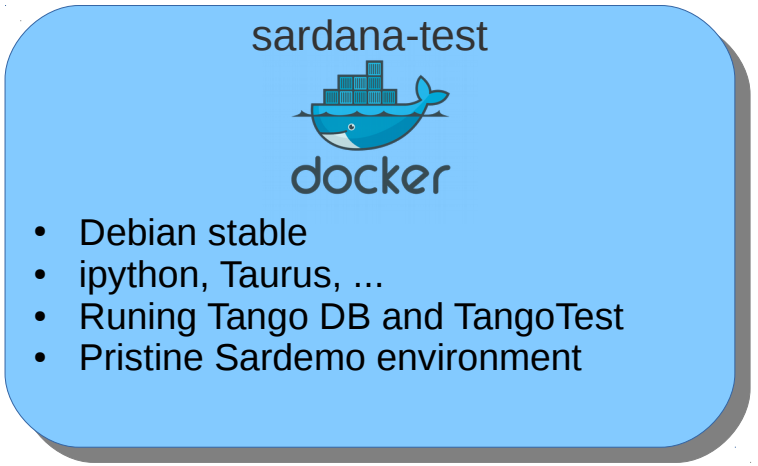


- Most activity concentrated and recorded as GitHub Issues and Pull Requests
- Less useless email (watch/unwatch PR & issues). Use **mention** tags
- Code review discussion in same place as original issues / Pull Requests

Automated tests

- Taurus and Sardana provide:
  - a set of utilities* based on **PyUnit** (*unittest* module)
  - a Test Suite (~500 tests in Taurus and ~100 test in Sardana) formed by:
    - happy-path tests for the key features
    - exhaustive tests for new features that were developed using Test-Driven Development
  - Docker Containers ready for tests



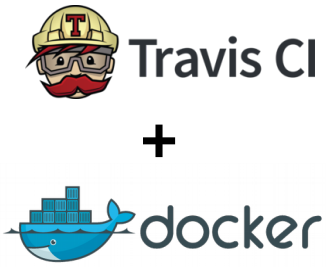


* http://sf.net/p/sardana/wiki/**SEP5**

Containerized tests

The docker containers are useful for:

- test isolation in CI
- multiple-environment test (different distros can be easily added)
- communication-related tests (we can run more than one container simultaneously)
- controlled-environment for support and debugging



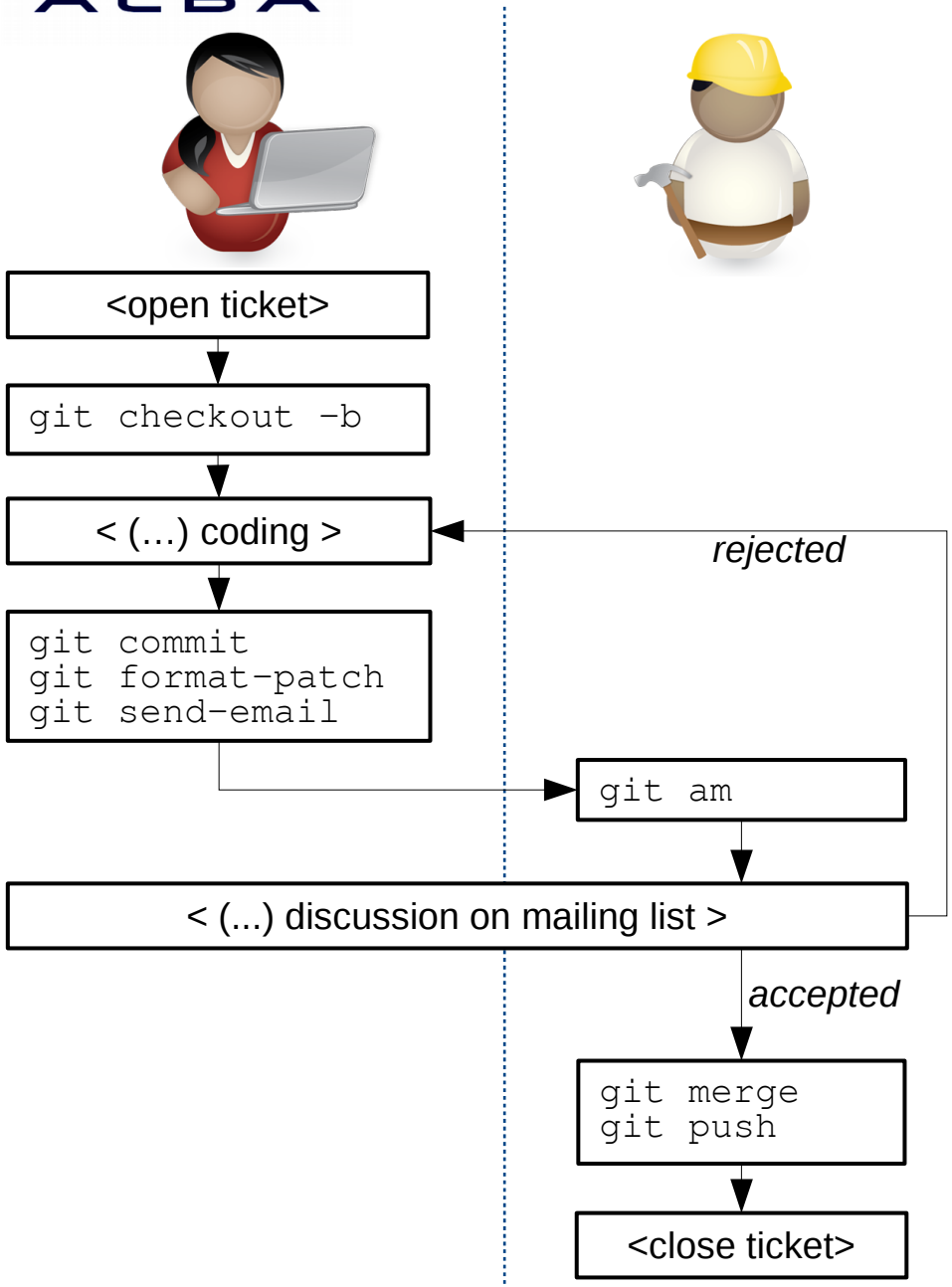
taurus/travis.yml

```
language: python
sudo: required
services:
  - docker
python:
  - "2.7"
before_install:
  - docker pull cpascual/taurus-test
  - docker run -d --name=taurus-test -h taurus-test --volume=`pwd`:/taurus cpascual/taurus-test
  - sleep 10

script:
  - docker exec taurus-test /bin/bash -c "cd taurus ; python setup.py install"
  - docker exec taurus-test /bin/bash -c "taurustestsuite"
```


```
5628 ----------------------------------------------------------------------
5629 Ran 560 tests in 79.604s
5630
5631 OK (skipped=11)
```

Code review workflow (current)



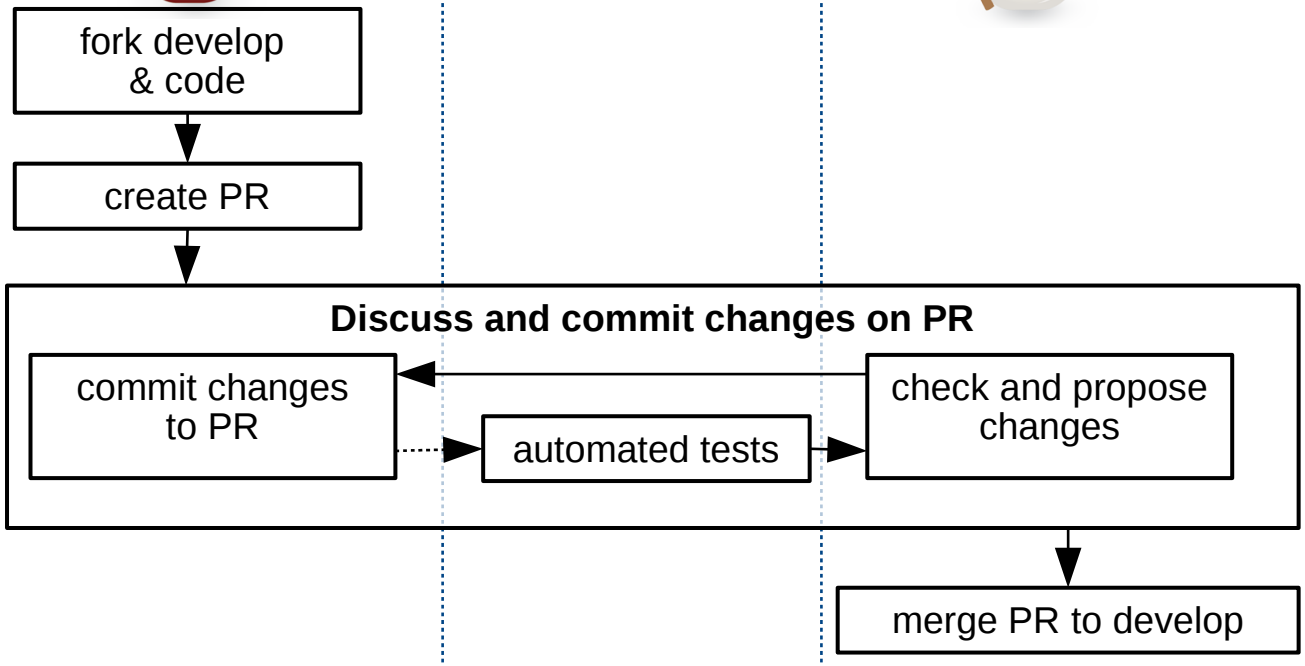
Mailing-list + patch code review:

- ✔ no tools required (just email client)
- ✔ no login required
- ✔ all discussion logged (in mail archives)
- ✖ contributor needs to learn conventions
- ✖ many tedious steps for integrator
- ✖ bad integration with SourceForge tickets
- ✖ saturates devel mailing list

Code review workflow (PR-based)

Travis CI

AppVeyor



Pull-request based Continuous Integration

- Easier for contributors
- Lighter for integrators
- All logged in PR discussion
- Every iteration is auto-tested

Continuous delivery workflow

Travis CI

AppVeyor

fork develop & code

create PR

Discuss and commit changes on PR

commit changes to PR

automated tests

check and propose changes

merge PR to develop

merge to master and tag

staged release

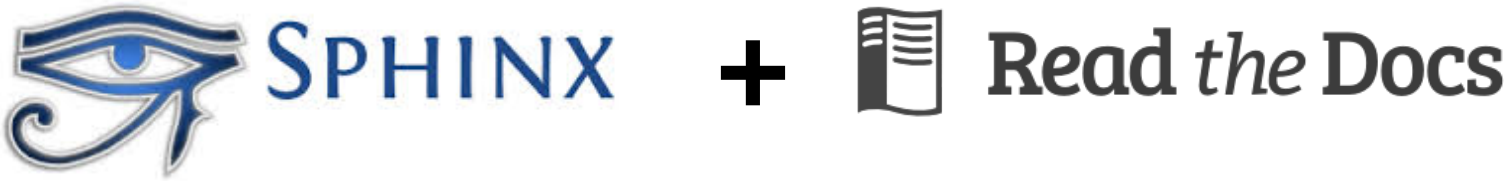
manual release tests

official release

Continuous Delivery

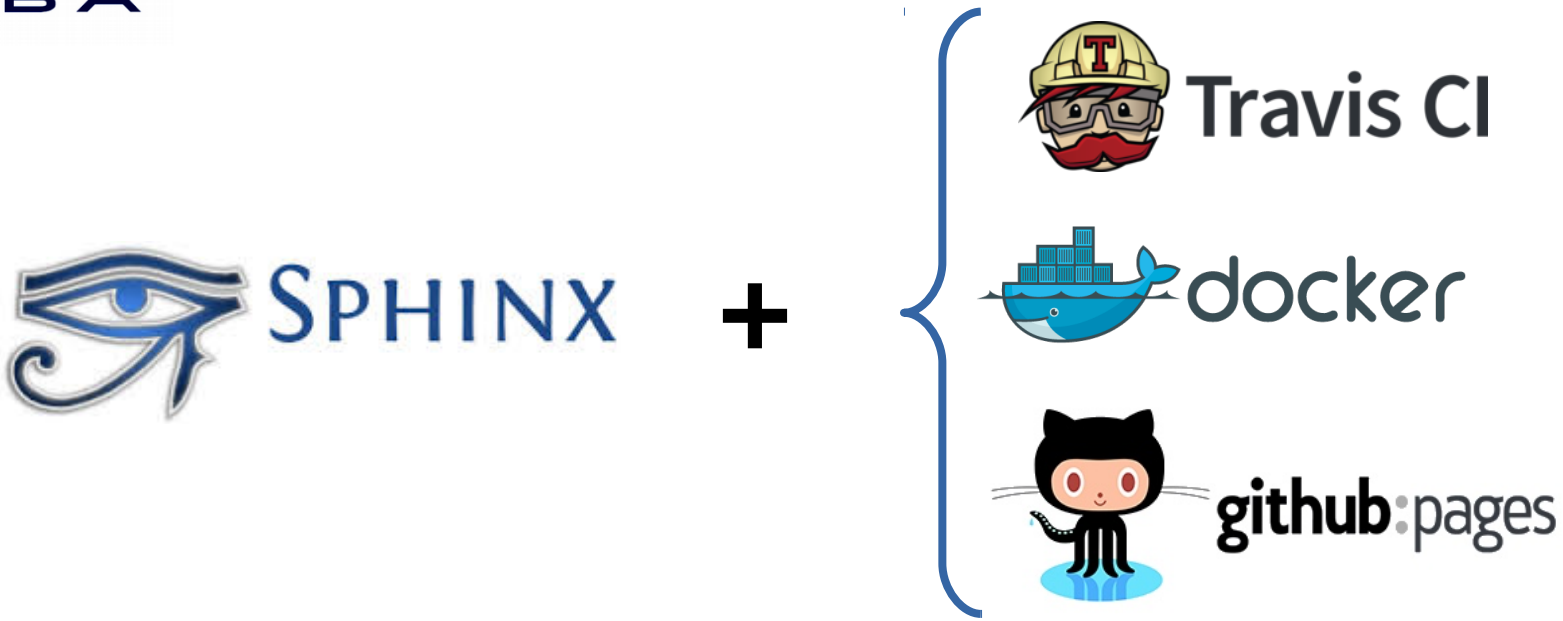
- ✔ Public and transparent
- ✔ Not tied to any institution
- ✔ Agile

Continuous Documentation (RTD-based)



- ✔ integrates well with GitHub
- ✔ out-of-the-box support for multiple doc versions
- ✔ out-of-the-box generation of pdf, epub, etc.
- ✖ mocks required for some dependencies (unfortunately, RTD does not support Docker)
- ✖ Difficult to debug (the environment is difficult to replicate)

Continuous Documentation (CI-based)



- ✔ docs can be built and deployed by Travis (just another artifact from each CI build)
- ✔ all dependencies already available in our docker containers
- ✔ reproducible, modular and transportable environment
- ✖ some configuration required (versions, pdf generation,...)

Beyond the Technologies

- **Be responsive and encouraging**

  prioritize external user requests even over your own institution's

- **Have a *"gradual strictness"* policy with contributors**

  forgive policy violations by new-comers, teach them as they get involved

- **Use well-known and free tools / workflows**

  prefer non-optimal well-known solutions over customized-but-unknown ones

- **Be transparent during the design discussions**

  use public channels even when discussing with your next-office neighbor

  document everything: APIs, roadmaps, proposals,...

“Barcelona is great in April”

...but even better in **October**!

ICALEPCS2017

Barcelona · Spain, October 8-13 · Palau de Congressos de Catalunya

Credits & legalese

- The Logos in this presentation are used for purposes of citation or identification, and do not imply endorsement by (or to) their respective owners.
- The “Proposal” Icon by Subhashish Panigrahi is Licensed under CC BY 3.0
- The "Bug report" Icon by Lemon Liu is Licensed under CC BY 3.0
- The ALBA picture is copyrighted by CELLS
- The “Man in the Tree” Picture is in the Public Domain
- The rest of this presentation (by C. Pascual-Izarra) is licensed under CC BY 3.0