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 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 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 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
Taurus & Sardana Communities

Community Driven SciSoft Projects

C. Pascual-Izarra et al.

11th NOBUGS - Copenhagen - Oct 2016
Taurus & Sardana Communities

Community Driven SciSoft Projects

C. Pascual-Izarra et al.

11th NOBUGS - Copenhagen - Oct 2016
Last year's contributors

Lines contributed to Sardana and Taurus during Jul15-Jul16

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

Privative vs Free/Open Source

Licensing
openness
Open Development vs Open Source

Development openness

Community development (the Bazaar)

Closed-group development (the Cathedral)

Licensing openness

Privative

Free/OpenSource

RISC OS

debian

Windows

red hat
Why Open Development?

Community development (the Bazaar)

- Openness
  - 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

Closed-group development (the Cathedral)

Free/OpenSource

Development

C. Pascual-Izarra et al.
11th NOBUGS - Copenhagen - Oct 2016
Our walk to the Bazaar

- Community development (the Bazaar)
- Closed-group development (the Cathedral)

Free/OpenSource

- Development openness

In-house development
- Code was “Free”, but license unclear
- Code in **internal** ALBA SVN repository
Our walk to the Bazaar

Community development (the Bazaar)

Closed-group development (the Cathedral)

Development openness

Free/OpenSource

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

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 decision-making** and **good documentation**

http://sf.net/p/sardana/wiki/SEP0
Communication channels (current)

- **Mailing List**
  - *-user
  - Questions
  - Bug reports & Feature Req.
  - Support

- **SourceForge**
  - Ticket tracker
  - Big changes

- **Enhancement Proposal**
  - Wiki
  - Interaction

- **Mailing List**
  - *-devel
  - Support
  - Coordination & Code review
  - Automated notifications

✅ 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)

- Mailing List
  - *-user
  - Announcements
- GitHub
  - Issues & PR
  - Support, **Code review** & Enhanc. Proposals
- Mailing List
  - *-devel

- 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](http://sf.net/p/sardana/wiki/SEP5)
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

```yaml
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"
```

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
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)

- fork develop & code
- create PR
- commit changes to PR
- check and propose changes
- automated tests
- merge PR to develop

Discuss and commit changes on PR

Pull-request based Continuous Integration

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

- 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)

 Mariners Wanted

 ✓ 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)

Sphinx +

Travis CI

docker

github:pages

- 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, ...)

Community Driven SciSoft Projects

C. Pascual-Izarra et al.

11th NOBUGS - Copenhagen - Oct 2016
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!
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