BRINGING QUALITY IN THE CONTROLS SOFTWARE DELIVERY PROCESS

Z. Reszela, G. Cuní, C. M. Falcón Torres, D. Fernández-Carreiras, G. Jover-Mañas, C. Pascual-Izarra, R. Pastor Ortiz, M. Rosanes Siscart, S. Rubio-Manrique (ALBA-CELLS Synchrotron, Spain)

Introduction

Alba is a 3rd generation synchrotron located near **Barcelona, Spain**. It comprises accelerators and 7 beamlines and was successfully commissioned in 2012. Nowadays all its beamlines host user experiments regularly. At the same time 2 more beamlines are under construction and it is planned to expand the facility even more in the near future.

The **Alba Controls Section** develops and operates a diverse variety of controls software which is shared within **international communities** of users and developers. This includes: generic frameworks like **Sardana** and **Taurus**, numerous **Tango** device servers and applications where, among others, we can find **PyAlarm** and **Panic**, and specific experiment procedures and hardware controllers. A study has commenced on how to improve the **delivery process** of our software from the hands of developers to laboratories, by making this process more **reliable**, **predictable** and **risk-controlled**.

www.albasynchrotron.es

www.sardana-controls.org

www.taurus-scada.org

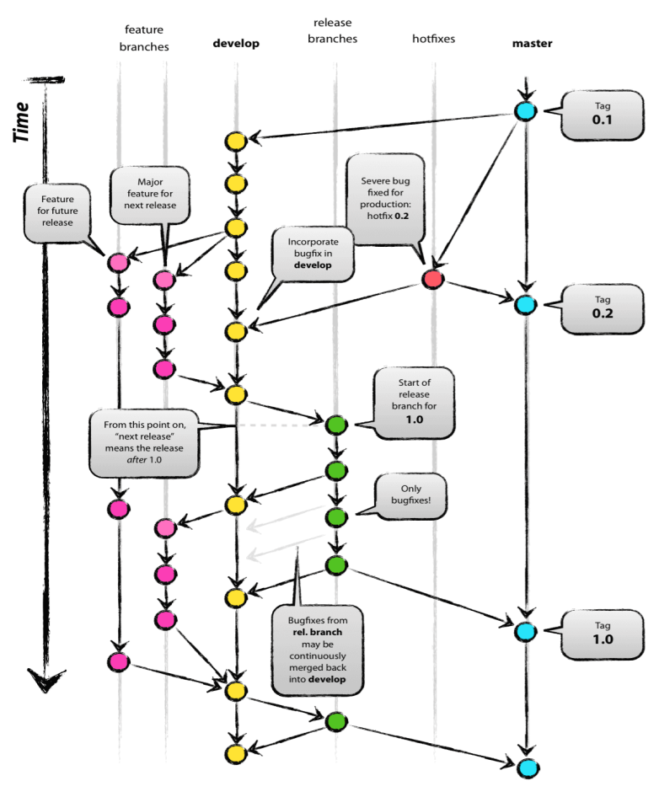
www.tango-controls.org

Package for Alarms and Notification of Incidences from Controls

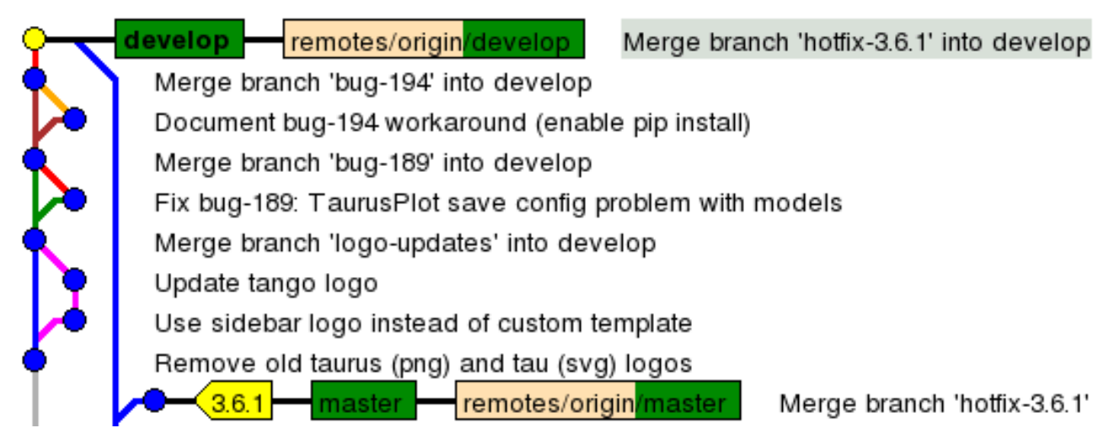
www.tango-controls.org/community/projects/panic

Code control

- Code repositories were migrated from SVN to GIT
- Easier branching and merging
- Easier tools and workflows
- Distributed architecture
- Better performance
- Cleaner history of commits with less effort



feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0



```
develop  remotes/origin/develop   Merge branch 'hotfix-3.6.1' into develop
    Merge branch 'bug-194' into develop
    Document bug-194 workaround (enable pip install)
    Merge branch 'bug-189' into develop
    Fix bug-189: TaurusPlot save config problem with models
    Merge branch 'logo-updates' into develop
    Update tango logo
    Use sidebar logo instead of custom template
    Remove old taurus (png) and tau (svg) logos
3.6.1  master  remotes/origin/master   Merge branch 'hotfix-3.6.1'
```

SEMANTIC VERSIONING

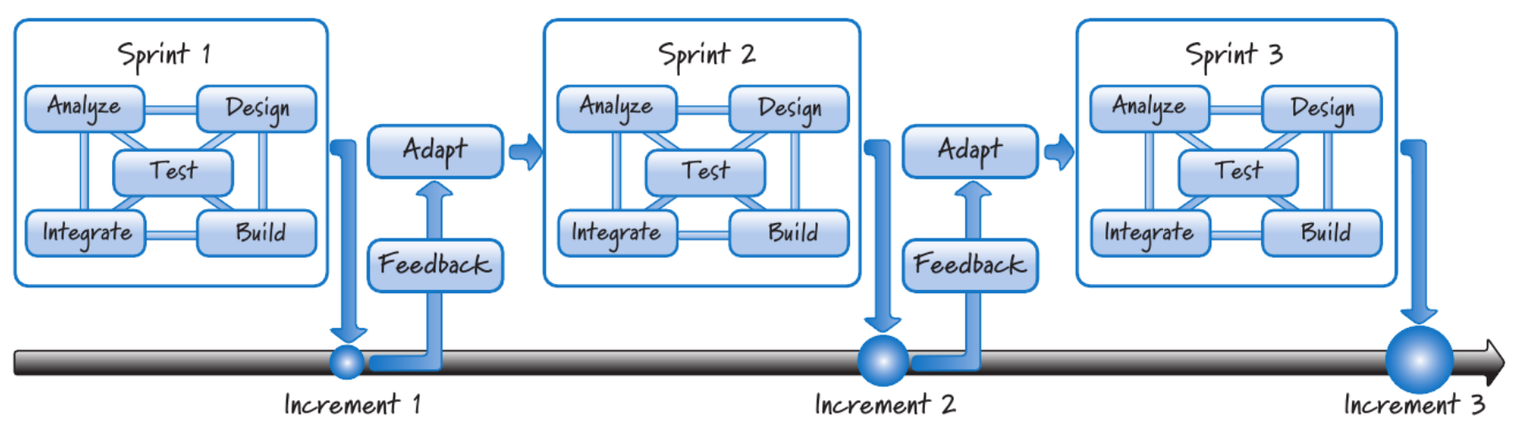
Given a version number **MAJOR.MINOR.PATCH**, increment the:

1. **MAJOR** version when you make incompatible API changes
2. **MINOR** version when you add functionality in a backwards-compatible manner
3. **PATCH** version when you make backwards-compatible bug fixes

Code design

Internal projects

- Single-person projects were passed to *Scrum* teams of 4-6 developers
- Mixture of senior and junior developers
- Scrum activities brought interesting design discussions
- Avoid upfront designs and plans
- Promote iterative and incremental developments

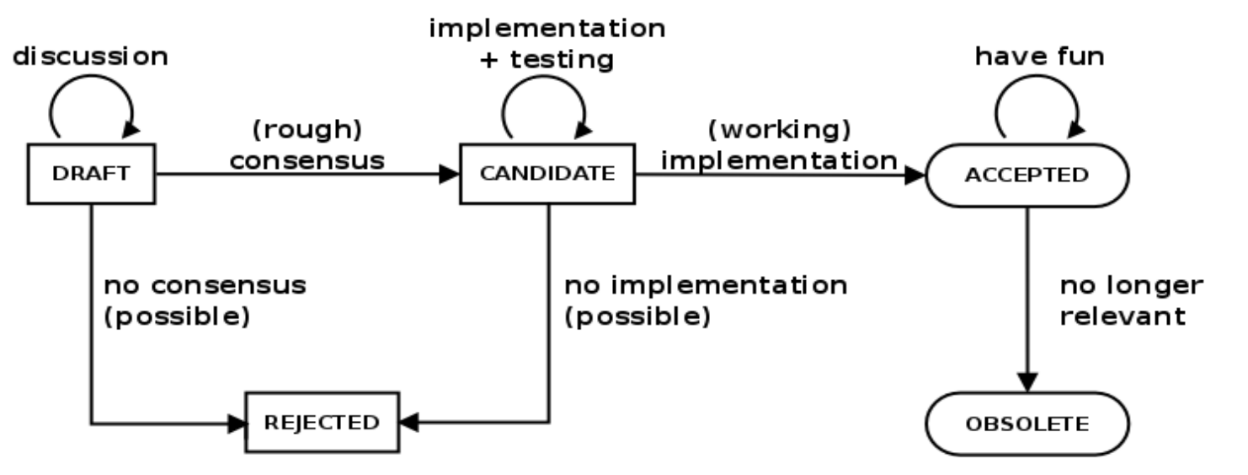


Community projects

- Sardana & Taurus decision-taking was opened to a community
- Sourceforge remote collaboration tools e.g. mailing list, wikis, issue tracker helps in the code design
- Critical improvements and modifications are organized and formalized around public processes:

SEP Sardana Enhancement Proposal

TEP Taurus Enhancement Proposal



Configuration Management

- Bliss system (by ESRF) was compared with Software Configuration Management (SCM) tools in terms of automated packaging and deployments
- Bliss pros: intuitive for non-packaging-experts, offline access to hosts' configurations
- Bliss cons: restricted to bliss users only, poor automation of building and deploying packages, not for Windows

SALTSTACK

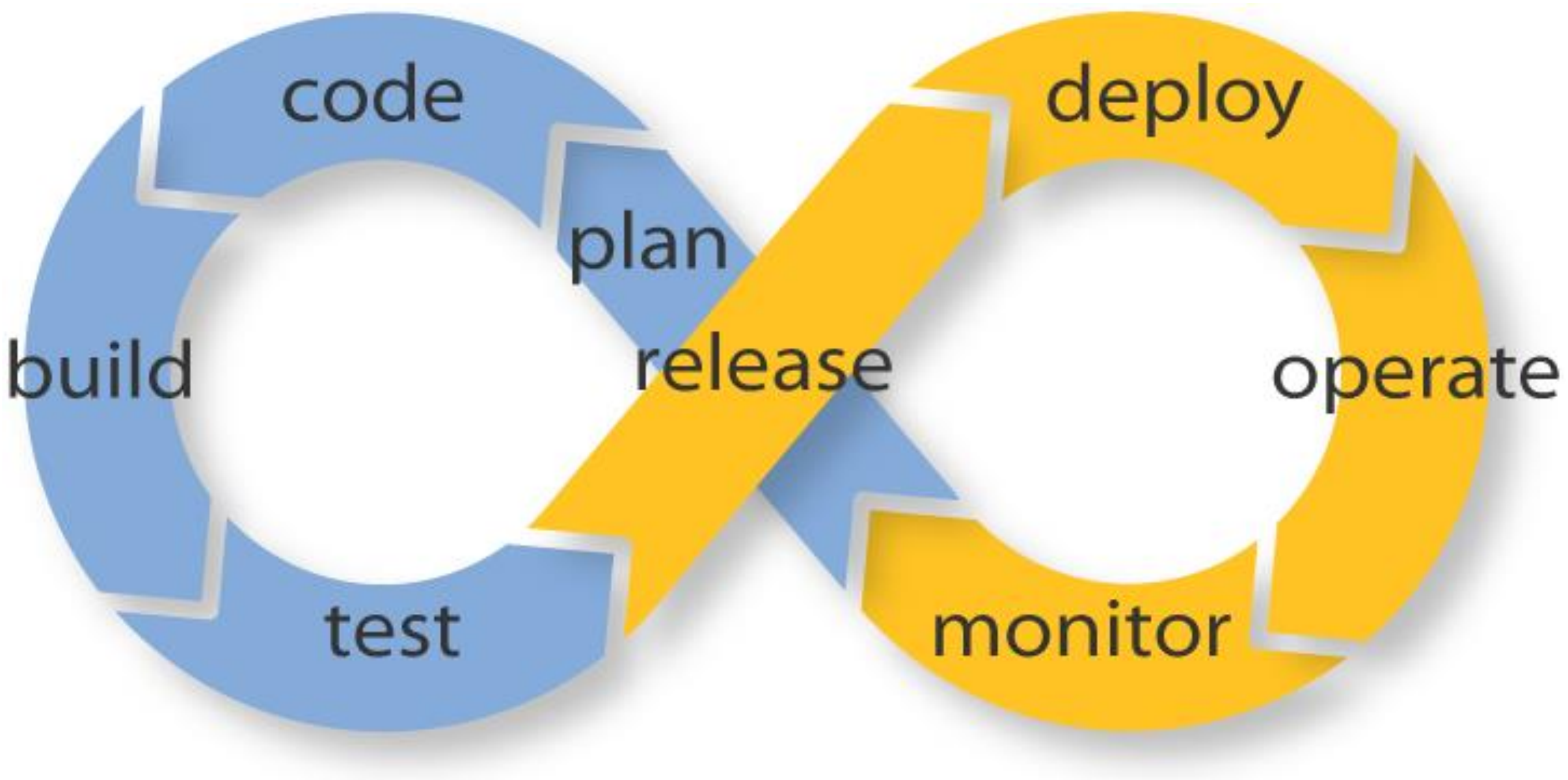
- Salt pros: repos integration modules e.g. apt, yum allows operating system agnostic configurations, either master-minion or ssh-based architecture, supports Windows, easy integration Continuous Delivery tools, high scalability
- New SCM cons: require new packaging from projects, and setup of repos

```
$ salt '*' package.install sardana
$ salt '*' package.latest_version sardana
```

Operating system agnostic installation of packages using Salt.

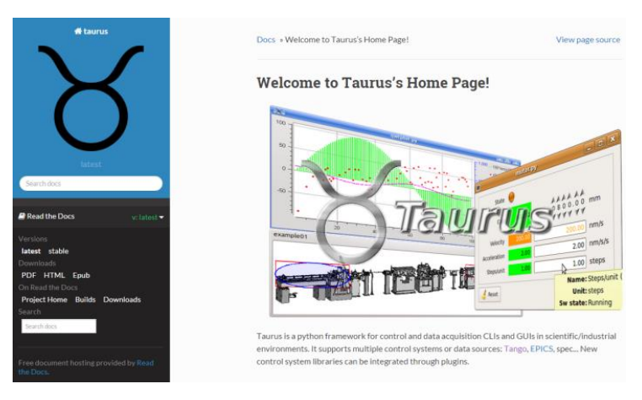
Packaging

- Python packaging libraries e.g. distutils builds rpm, deb, msi packages
- msi allows unattended installations necessary for SCM
- Package repositories (repo) e.g. yum for rpm, needs to be setup
- Proof-of-concept Taurus CD pipeline: builds rpm and msi, uploads them to the repo and SCM pulls from the repo



Continuous documentation

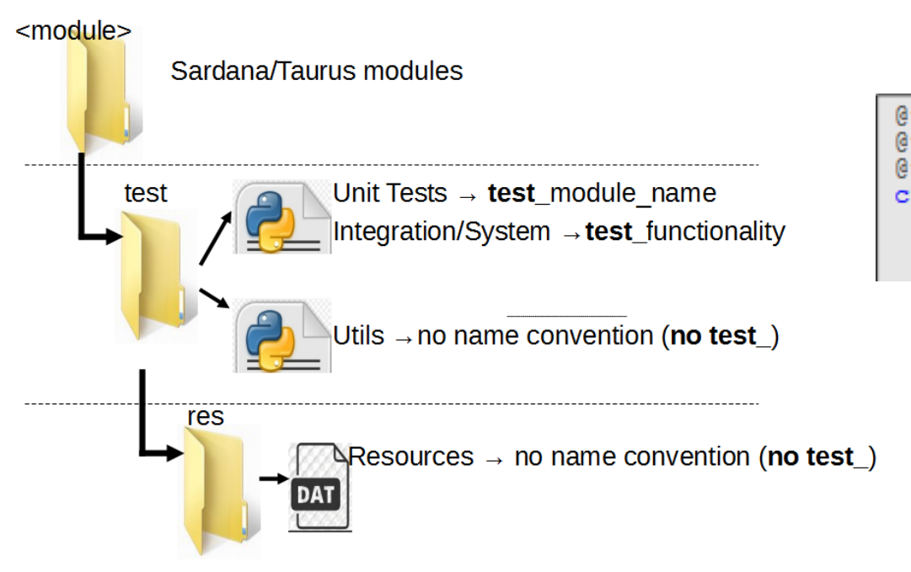
- Sardana and Taurus migrated their documentation to **Read The Docs** (RTD)
- Documentation is build on every commit
- Maintenance of the servers and the necessary software is outsourced to RTD
- Several version of the docs e.g. *stable* or *latest* are available
- Docs are available in different formats: html, pdf, epub



Taurus documentation on RTD.

Testing

- SEP5 established the common testing strategy for Sardana and Taurus.



<module>
Sardana/Taurus modules
test
Unit Tests → test_module_name
Integration/System → test_functionality
Utils → no name convention (no test_)
res
Resources → no name convention (no test_)
DAT

```
@testRun(macro_params=['mot1', '0', '3', '10', '.1'])
@testRun(macro_params=['mot2', '3', '0', '10', '.1'])
@testStop(macro_params=['mot1', '0', '5', '3', '.1'])
class MotorNGAscanctTest(RunStopMacroTestCase, TestCase):
    macro_name = 'ascanct'
    door_name = 'door1'
```

Inserting test scenarios to test cases using Python decorators.

- Tests should be written before developers start work on the features that they test.
- The automated test suite should be run by the **Continuous Integration** service on every commit.

Jenkins

```
$ python sardana/test/testsuite.py
$ python taurus/test/testsuite.py
```

- Mid or legacy projects should start automating the most common and important use cases.
- The rest of the scenarios should initially be tested manually.

Continuous Delivery

- Agile & Continuous Delivery (CD) aims to transform a concept into a working software as fast as possible.
- CD is based on fully **automated**, reliable, repeatable and constantly improving software delivery **pipelines**.
- Start with pipelines with just 3 stages: commit, acceptance and user acceptance (UAT).
- Jenkins works as the pipeline orchestrator.
- **Commit**: unit tests, code analysis, build packages
- All subsequent stages use packages built in commit stage
- **Acceptance**: deploy with SCM to production-like environment, execute automated acceptance tests
- Use **Docker** to prepare & run lightweight, reliable and isolated acceptance test environments
- **UAT** (on demand): deploy with SCM to production-like environment, execute manual acceptance tests

Proof-of-concept Taurus continuous delivery pipeline.

Next steps

- Sardana and Taurus projects could already apply the CD strategy to their biannual releases.
- Ideally their pipelines should be accessible by the whole community of developers.
- Try **online** providers for the **continuous integration/delivery** tools.
- Try **online code review** platforms. It may bring new quality to the current review processes, making them more accessible to the developers and reducing the workload on the integration managers.

Acknowledgement

- Alba Controls Section and especially: R. Homs, D. Roldan, J. Andreu, F. Becheri and S. Blanch
- Sardana and Taurus community members
- Alba IT Systems Section and especially to S. Pusó, M. Rodríguez, M. Díaz and A. Pérez
- Alba MIS Section and especially to D. Salvat

ICALEPCS 2015
17-23 OCTOBER 2015 MCEC MELBOURNE