MOPV037

ALBA CONTROLS SYSTEM SOFTWARE STACK UPGRADE

G. Cuni, F. Becheri, S. Blanch-Torné, C. Falcon-Torres, C. Pascual-Izarra, Z. Reszela, S. Rubio-Manrique
(ALBA-CELLS, Barcelona, Spain)

ICALEPCS 2021

ALBA, a 3rd Generation Synchroton Light Source located near Barcelona in Spain, is in operation since 2012.

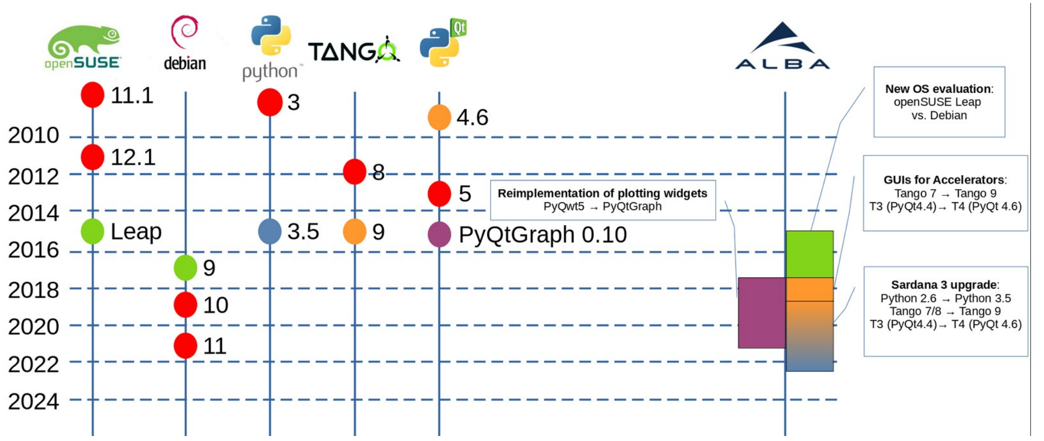
During the last 10 years, the updates of ALBA's Control System (CS) were severely limited in order to prevent disruptions of production equipment, at the cost of having to deal with hardware and software obsolescence.

The construction of the second phase new beamlines accelerated the renewal and upgrade of the software stack.

We are in the process of switching to
- Debian OS, Tango 9, HDB++
- Python 3, PyQt5 and PyQtGraph, etc.

In order to ensure the project quality and to facilitate future upgrades, we try to:
- automate testing, packaging with CI/CD pipelines
- configuration management

using, among others, the following tools:
- pytest, Docker, GitLab-CI and Salt...



pytest

GitLab CI

SALTSTACK

docker

ALBA Control System Obsolescence

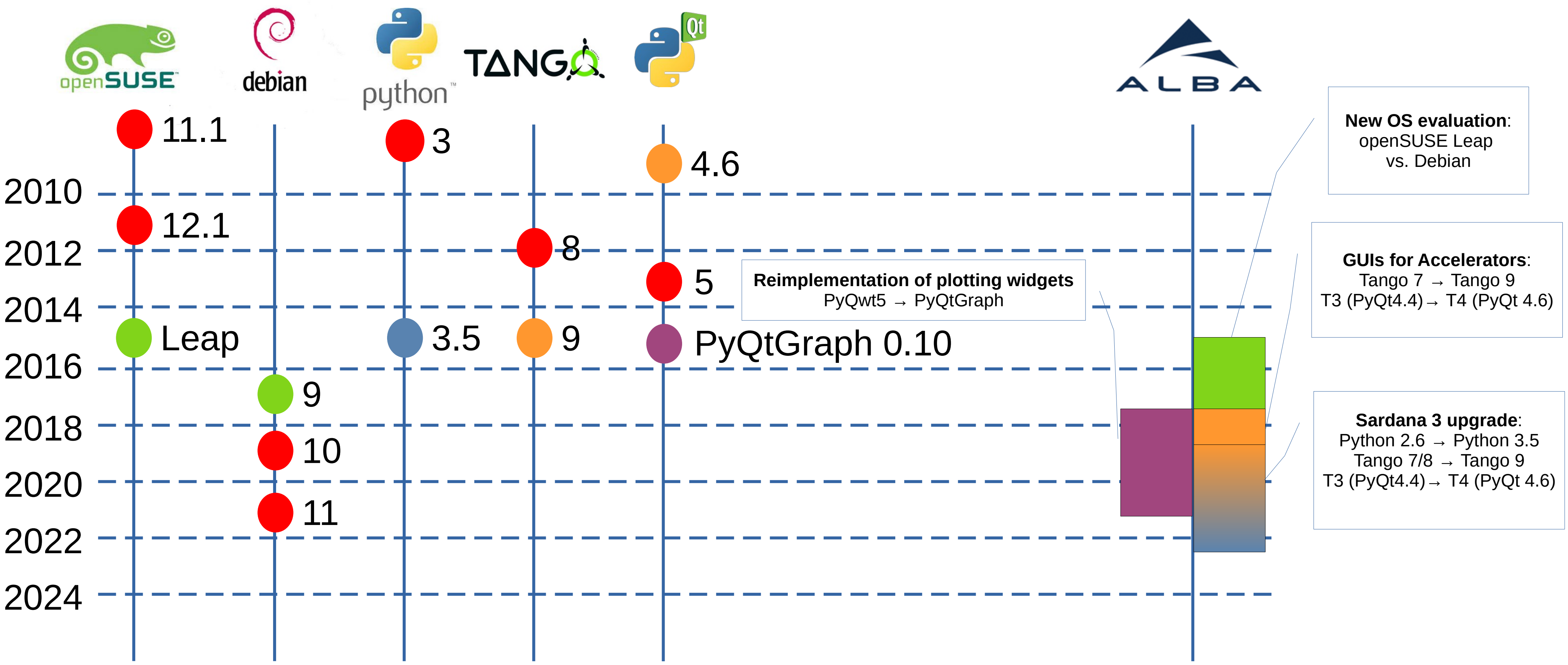
ALBA Control System (CS)

- Uses **Tango** as middleware, a distributed control system framework based on **CORBA**.
- Python is the main programming language.
- GUIs at ALBA are developed using **Taurus**, a library for building desktop apps in **PyQt**.
- **Generic** and **transversal services** are integrated in Tango ecosystem and used across the facility:
  - **Sardana**, a scientific SCADA suite for experiment control on beamlines (BLs).
  - **Panic**, an IEC62682 compliant Alarm Handling suite.
- Every sub-system has its **specific applications**, e.g. MXCuBE control application for the macromolecular crystallography experiments (BL13), TXM application used for the tomography experiment (BL09), or the accelerator timing system controls stack.
- Software run on diskless compact PCI and industrial PCs, and VMs with **Linux** distribution

Obsolescence

- OpenSuSE 11.1, a Linux distro used during the ALBA initial years reached its end-of-life in 2011. This forced us to use core libraries and modules (e.g. libc, Python, numpy, PyQt4,...) that dated back to 2008.
- Use of OpenSuSE 11.1 tied us to use Python 2.6.
- The Bliss packaging system was limited in terms of automatic package creation and deployment. Software was unpackaged in non-standard paths and it was not possible to properly define package dependencies
- Tango 7 and its event system was unmaintained since Tango 8 was released in 2012.
- PyQt4 reached its end of life in 2018 and PyQwt5 was by then already unmaintained (its latest release from 2011) and was never ported to PyQt5 or Python 3 leaving us without a plotting library.

ALBA Control System Upgrade



Reimplementation of plotting widgets
PyQwt5 → PyQtGraph

New OS evaluation:
openSUSE Leap
vs. Debian

GUIs for Accelerators:
Tango 7 → Tango 9
T3 (PyQt4.4) → T4 (PyQt 4.6)

Sardana 3 upgrade:
Python 2.6 → Python 3.5
Tango 7/8 → Tango 9
T3 (PyQt4.4) → T4 (PyQt 4.6)

Applied Practices & Tools

Testing

- We switched from unittest to PyTest for developing automatic tests (it provides better API and many useful features out-of-the-box).
- We employ hardware simulators in our tests. We plan to employ automatic tests with real hardware into CI and run nightly stress tests to discover non-easily reproducible bugs and performance degradations.

pytest

Automation and Reproducibility

- Executing tests as part of the CI reduced risk of regressions in the Taurus and Sardana projects and we will promote it in other developments.
- Creating packages and uploading them to Debian, PyPI and conda repositories enables going one step further towards CD
- We also build and publish docs as part of CI.

GitLab CI

Configuration

- We use Salt to configure and manage software (Debian, Git, pip, conda) on remote nodes.
- We define a catalog of ALBA Services e.g.: Tango, Taurus, Archiving, etc. The application of the Salt recipes allows us to automate the installation and configuration of the same Services in parallel in different machines and in a reproducible way.

SALTSTACK

Containerization

- Our use of (Docker) containers has been mainly limited to the context of software development as:
  - providing the environments for the CI jobs
  - providing a pre-configured a clean environment for manual testing or packaging of software
  - Developing or debugging in a reproducible environment

docker

Towards ALBA II & Conclusions

ALBA II project will consist of an upgrade to the **4th generation class of synchrotron** light source and is planned to happen in 2028.

The ALBA Computing Division started preparing for the future requirements and plans to start exploratory projects. We will at least evaluate use of:

- **containers** or **isolated environments** in order to facilitate software upgrades by achieving inter-apps isolation and isolation from the host OS.
- **web technologies** for operators GUIs, which in comparison to desktop applications are cross-platform compatible and more manageable
- **Tango 10** which code will be refactored/rewritten in order to make it immune to the obsolescence of libraries and technologies e.g. CORBA.

We believe that it is crucial to keep the OS updated. In our case, the lack of update in the OS conditioned many other updates and, in the long term, generated more efforts in workarounds than the effort of keeping it up to date. Also, it generated a huge effort when it had to be finally updated.

Keeping the controls system up-to-date is a collective responsibility of the whole team. Decisions at different levels, from day-to-day to strategic, should be taken considering the long-term maintainability of the control system.

It is highly recommended to continuously follow and explore emerging technologies in order to propose improvements, feel self-confident and determined in proposing and conducting upgrade projects.