ALBA, A TANGO BASED CONTROL SYSTEM IN PYTHON

D. Fernandez-Carreiras, F. Becheri, S. Blanch, T. Coutinho, G. Cuní, J. Klora, C. Pascual-Izarra, S. Rubio-Manrique, R. Suñé, CELLS, Bellaterra, Barcelona, Spain.

Abstract

Alba is a member of the Tango collaboration. We have focused on the development of support for Python in Tango. Now, most device servers and clients are based on Python. On the client side python is combined with Qt (Nokia) / PyQT (RiverBank) for graphical interfaces and IPython (open source) for command line interfaces. Python is fast and suitable for most device servers, and gives an enormous flexibility in terms of evaluation of expressions, and embedded on-line calculations. This paper describes the choices taken on controls software development

However, Alba has also developed specific servers in C++, which Tango integrates perfectly with Python servers and clients in the control system. Many others written in C++, Java and Python, which have been published by the members of the Tango collaboration [2], are also used at Alba.

Figure 1: View of the Accelerators' tunnel; on the left the storage ring, and on the right the booster. Picture taken on the 8th of September 2009 (J. Klora).

INTRODUCTION

The Tango collaboration counts now five members: chronologically, the ESRF (where Tango was initially developed), Soleil (The first synchrotron using Tango as the unique controls middleware), Elettra, Alba and DESY. The Python support for Tango (PytTango) has been developed at Alba by E. Taurel (now at the ESRF) and T. Coutinho. PyTango is now extensively used at Alba.

Tango can be described as a middleware built on top of CORBA[3], and a set of tools and services for developing a control system. It is based on the concept of "Device Servers" distributed in different computers, typically Linux and Windows Input Output Controllers (IOCs) having the hardware electronics for control and data acquisition. It is a client-server design having a central database for name services and configuration parameters. Tango fully supports C++, Java and Python for both servers and clients.

PYTHON AND TANGO

The Tango core is written in C++, but supports a number of bindings, for example, Labview, Matlab (Soleil), C (ESRF). Python is interfaced with the Tango Core libraries using boost [4] and therefore it is strongly linked to the C++ Tango Core. Java support is on the other hand independent, which leads sometime to different or outdated behaviors in the core. Both Tango clients and servers are supported in Python. Today more than two thirds of the device servers and almost all Graphical User Interfaces (GUIs) at Alba are developed in Python.

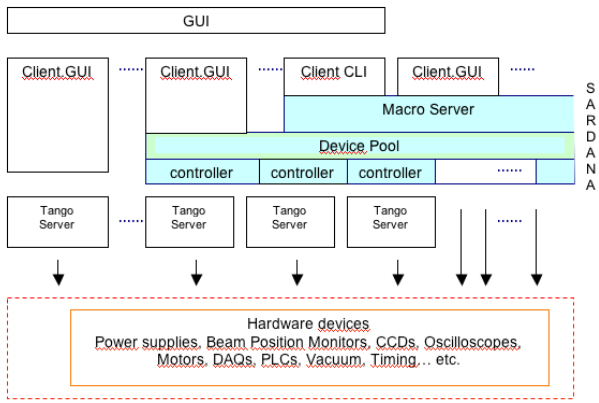

Figure 2: Software architecture of the Alba Control System.

Sardana Framework Package

Sardana [5] is the name given to the framework built on top of Tango, shown in blue in **Figure 1**. It is inspired on SPEC's [6] architecture. It provides a series of facilities for dynamically integrating new hardware components into the control system, and making them available for all clients. It also offers other functionalities like scripting (with a large library of standard macros), data storage, etc.

The device pool is a Tango device server, which offers an extra abstraction layer to the hardware, allowing new devices to be software "hot-plugged". It takes care of software synchronization between devices and adds functionalities, that the device might be lacking, for example "motor backlash". The device pool is a device server written in C++, however most controllers are written in Python (both Python and C++ are accepted). Controllers are created, deleted and edited in run time. Controllers can access directly the hardware or other Tango devices. Clients found always a standard interface for a device type independently from the underlying hardware.

The macroserver is a Python device server that manages the execution of scripts or procedures usually called macros. It has doors associated. A door is a Tango Server that manages connections from clients to the macroserver. Macros are python classes. A standard macro library stores the common procedures for scanning, moving motors, counting, etc. It is usually accessed from the macro executor GUI or the Command Line Interface (CLI).

Command Line and Graphical Interfaces

Spock is a IPython based CLI for Sardana. It communicates with the macroserver through one of its doors. Because it is IPython based, all CLI niceties (like word completion on Tab, command history on Up, Down arrows) are automatically available which makes it very easy for spock to mimic SPECs UI. The user executes 'magic' commands (like scans, alignment procedures) in the command line which spock translates to execution of macros inside the macroserver.

The MacroExecutor is a widget offering a user-friendly interface to the macro repository. Standard and user defined macros are selected from a graphical user interface, prompting for input arguments and offering results and plots. A snapshot is shown in the **Figure 3**.

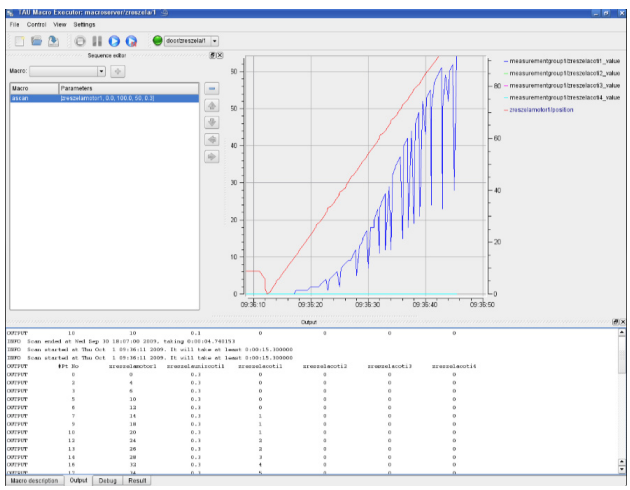

Figure 3: Snapshot of the MacroExecutor.

Tau

Tau is the graphical interface layer at Alba. It is built on PyTango and uses Qt [7]. It implements a model view controller pattern. The models are provided by the TauCore layer, which is independent of the widgets. Normally GUIs are generated by the Qt Designer, but not exclusively. A library of Tau Widgets is available also from the designer. Written mostly in Python although, some of them in C++ and integrated with sip, these set of widgets constitute the basic bricks for creating graphical applications or new widgets. A model property gets the model from the factory in the TauCore and subscribes to events typically fired by changes in the Tango Attributes.

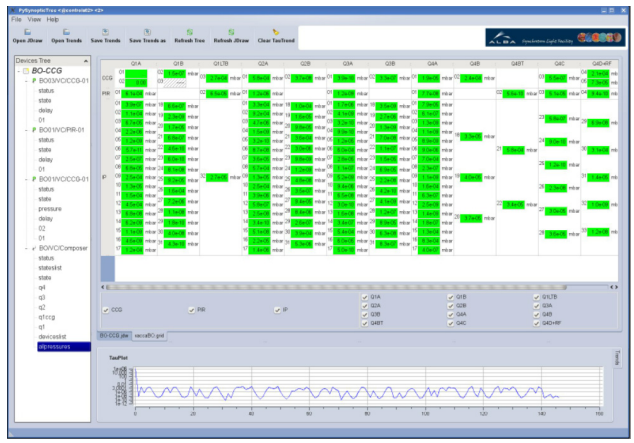

Figure 4: Snapshot of the GUI for the vacuum control of the booster. It has a TauTree on the left, a TauGrid on the top and a TauTrend on the bottom.

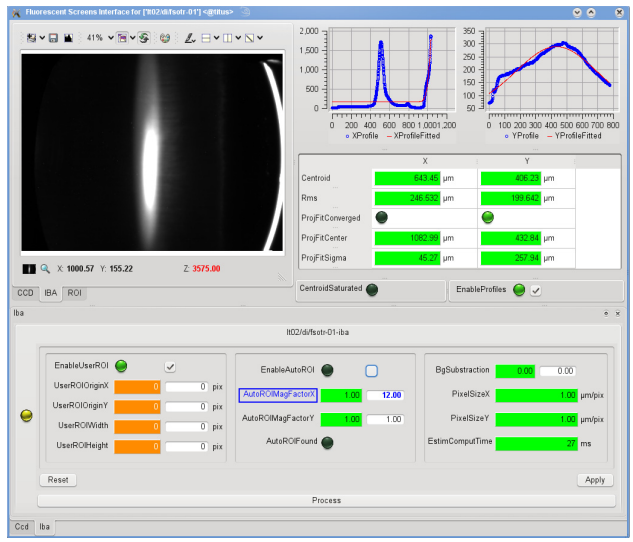

Figure 5: Fluorescence screen GUI. It combines, Qub, showing the image, TauPlot, showing the vertical and horizontal sections and TauForms.

Device Servers in Python

Tango device servers written in Python are very suitable for many applications. The macroserver itself is a good example. When a large number of attributes is needed, for example, the case of PLC device servers, dynamic attributes show many advantages and Python is the best choice [8]. Many attributes can be dynamically created depending on "live" information. Besides, they can be configured in properties from information coming from controls and cabling database [9].

PYTHON PACKAGES

In the above pieces of software and frameworks several standard open-source packages are used. Here are some of them: IPython for the command line interface providing a useful interactive python shell, Qwt providing graphics for 2D plotting, SciPy and NumPy for numerical data analysis. QtControls, a Qt based C++ widget library developed by Elettra for control widgets with a python

binding using SIP and QUB (a package for fast 2D and 1D visualization developed at the ESRF) are also used.
SPS is a C based shared memory library developed at the ESRF with a python binding written in pure C. PyMca [10] and NewPlot (ESRF) are very convenient for online and offline data analysis and visualization.

USE CASES

Our experience concludes that step-scans, sequences and rapid integration of new hardware for a particular experiment are crucial requirements for all Beamlines (and in most cases for accelerators too).

Python cares of plotting, macro execution, graphical interfaces, command line, and slow data acquisition, which are a big part of the controls software. For particular data acquisition needs, like Beam Position Monitors, or fast X-Ray Detectors, another technology might be more suitable, like coding C modules in the kernel or C++ servers in user space.

Data recorders for slow acquisition, NeXus [11], SPEC files, log files and even 2D data are also generated in python. Currently Alba uses the ESRF data format for two-dimensional images, and the SPEC format for scans. NeXus support is also available. At the moment we are working on compatibility with other synchrotrons, which are also adding Nexus support to their data storage systems (Soleil, Diamond and the ESRF).

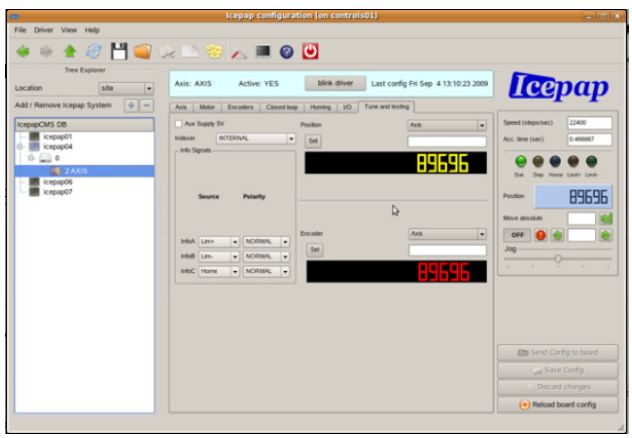

Figure 6: Snapshot of the Icepapcms.

Although most software is written on the Tango Framework, few stand-alone applications are needed. It is the case of the Icepapcms, the configuration software for Icepaps. Icepap is the standard motor controller at Alba. The hardware has been developed at the ESRF, and the configuration software at Alba. Many parameters, some of them critical, must be configured before using a motor, like current, encoders, limits, etc. It manages a database with historical configurations; supports motor catalogs and has a tool for online tests. Icepapcms is built with the same standard tools.

CONCLUSION

Alba has focused the software developments on Python. Most device servers and all graphical user interfaces are written in Python. It has proven to be suitable for GUIs (combined with Qt), and for most device servers. It is also very convenient for writing simulations and dynamic attribute expressions. Moreover it is an excellent choice for data analysis. Only the cases where very fast data acquisition, storage or processing is needed complementary solutions are proposed.

CONTRIBUTIONS

Many developers are working on this project. We would like to thank all of them. E. Taurel is the main Tango Core developer and the first author of the Sardana Framework and the Tango binding for Python. Besides the ones named in the Author Tab, there are many other people working in the project: L. Krause (Power Supplies, Linac), Z. Reszela (Tau widgets, beamlines), J. Moldes (Libera BPMs, Timing, beamlines), A. Milán (RF, beamlines), M. Niegowski (Radiation Monitors, PLC GUIs), We would also like to thank other contributors from different institutions, in particular A. Homs, V. Rey, S. Petitdemange, G. Berruyer, L. Claustre, M. Guijarro and the whole Bliss group at the ESRF, for their great collaboration. Not forgetting T. Nuñez and T. Kracht at DESY, and all our Partners in the Tango Community.

REFERENCES

[1] Alba Synchrotron, http://www.cells.es
[2] Tango Control System, http://www.tango-controls.org
[3] CORBA, http://www.corba.org
[4] Boost C++ Libraries, http://www.boost.org
[5] J. Klora, T. Coutinho et al. “The architecture of the Alba Control System”, Proceedings NOBUGS 2008.
[6] SPEC, Certified Software, http://www.certif.com
[7] Qt, http://qt.nokia.com
[8] S.Rubio-Manrique et al, “Dynamic Attributes and other functional flexibilities of PyTango”. These proceedings.
[9] D.Beltran et Al, “Alba Controls and Cabling database”. These proceedings.
[10] PyMCA, http://pymca.sourceforge.net
[11] Nexus data Format, http://nexusformat.org