SARDANA, THE SOFTWARE FOR BUILDING SCADAS IN SCIENTIFIC ENVIRONMENTS

T. Coutinho, G. Cuní, D. Fernández-Carreiras, J. Klora, C. Pascual-Izarra, Z. Reszela, R. Suñé,
CELLS, Bellaterra, Barcelona, Spain
A. Homs, E. Taurel, V. Rey, E.S.R.F, Grenoble, France

Abstract

Sardana is a software package for Supervision, Control and Data Acquisition in scientific installations. It delivers important cost and time reductions associated with the design, development and support of the control and data acquisition systems. It enhances TANGO [1] with the capabilities for building graphical interfaces without writing code, a powerful python-based macro environment for building sequences and complex macros, and a comprehensive access to the hardware. Just as Tango, Sardana is Open Source and its development model is open to collaboration, which provides a free platform that scales well to small laboratories as well as to large scientific institutions. The first beta version has been commissioned for the control system of Accelerators and Beamlines at the Alba Synchrotron [2]. Furthermore, there is a collaboration in place, comprising Desy [3], MaxIV [4] and Solaris [5], and several other potential users are evaluating it.

THE DESIGN CHOICES IN THE SOFTWARE OF THE CONTROL SYSTEM

Alba is a third generation synchrotron located near Barcelona in Spain. The Installation of the Control System for the Accelerators finished at the end of 2010. The final functional tests took place during the first weeks of 2011. Currently we are carrying out the final stages of the installation and commissioning of the seven Beamlines included in the first phase.

The infrastructure implicated in controls includes more than 350 racks, 6300 equipments and 17000 cables. The controls architecture is highly distributed, comprising about 2500 network devices.

Tango as the Control System Middleware

Tango was chosen among the three options considered: EPICS [6], commercial SCADA (Supervisory, Control And Data Acquisition) and Tango. At that time, the commercial SCADAs were not adapted to the requirements, and although they presented some interesting features off the shelf (like the archiving, trending, etc), many applications needed to be developed anyway, and a non negligible effort needed to be dedicated to integrate motion, synchronization, sequencers, etc. In other words, they were not a solution per se, but to be combined with EPICS, Tango or other toolkits.

Sardana: the Scientific SCADA Suit

The other way around, both Tango and EPICS have various choices as graphical toolkits. However, they both lack in some way an integrated development and runtime environment as commercial SCADAs offer. But on the other hand, commercial SCADAs did not fulfil the requirements either. In several installations, like synchrotrons, we find a large control system for the particle accelerators with different subsystems such as vacuum, radio frequency, power supplies, diagnostics and protection systems, and many "smaller" control systems, one per experimental station. They usually have significantly different requirements.

We needed a flexible graphical interface, allowing multiple clients, with many specific capabilities such as diffractometers control, and above all, a powerful sequencer. Many of these characteristics are implemented in SPEC [7]. However, SPEC has some limitations in the Graphical interfaces and in the capability for managing multi-clients. Therefore at that point we decided to start a development of a SCADA for scientific institutions: *Sardana* [8]. In order to facilitate its adoption, Sardana is built on widely available Open Source technologies and it is itself distributed under the Lesser General Public License [9]. Nowadays it has already been exported outside Alba, and other institutes like Desy or MaxLab are participating in the effort.

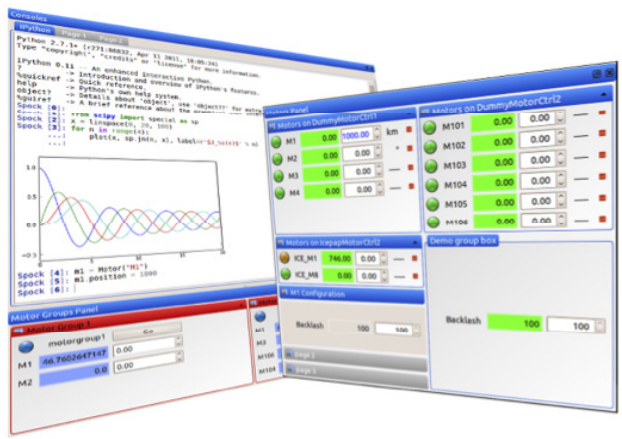

Figure 1: Few views and widgets of the Sardana´s Human-Machine interfaces.

Sardana provides optimized access to hardware, macro execution, software synchronization, generic graphical interfaces, and access to save/restore facilities (Fig. 1). It is mostly developed in Python. It has a Core in charge of managing the hardware: the "*Device Pool*". It has also a powerful sequencer which handles macros, scans, series, loops, etc: the "*Macroserver*" (Fig. 5). This is imperative in any experimental station in a synchrotron, and extremely useful in all cases.

HUMAN MACHINE INTERFACES

The Graphical User Interfaces (GUIs) are developed in a framework called Taurus [10]. It is written in python and built on top of PyTango [11] and PyQt4 [12]. The Command Line Interface (CLI) is based on IPython and makes extensive use of Taurus as well. It is known as "*spock"*(Fig.4), and it was highly inspired on SPEC. The appearance, standard macro names, and the syntax of SPEC command line have been adopted. This is an important point, because many Synchrotron users are familiar with SPEC, considered already a *de facto* standard.

Developers can create Taurus applications from the standard Qt designer benefiting from a catalogue of Taurus widgets which makes the task easier (Figs. 2 and 3).

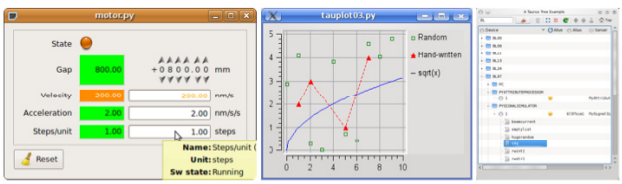

Figure 2: Taurus provides a complete catalogue of widgets.

But the aim is skipping the programming part if it is not needed. Sardana can be setup by "only configuring" the different components. Taurus provides an utility called TaurusGUI which produces standard GUIs from simple configuration files. This utility is typically used to generate standard interfaces for Beamlines. By just configuring a few parameters, a GUI is created for a given Beamline, including widgets for managing and launching macros and sequences, a synoptic view of the system,

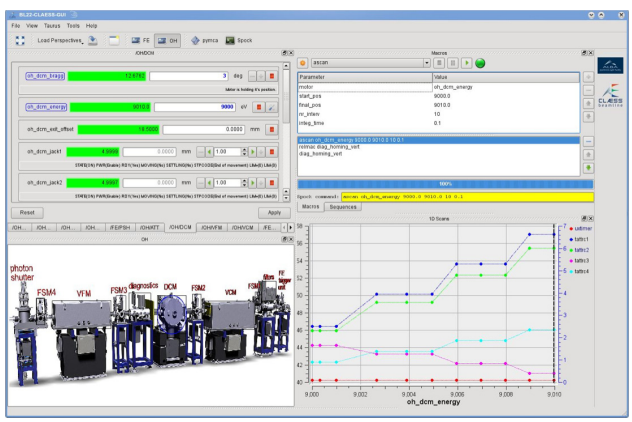

Figure 3: Taurus a highly configurable and customizable standard GUI for Sardana.

panels for controlling the instruments of the Beamline, graphical elements for monitoring values, etc. All these elements are presented in a highly customizable interface which allows the user to create different "perspectives" for different tasks. The elements of the GUI can be re-arranged by drag&drop, and new elements can be added just as easily without leaving the running GUI.

SEQUENCER, MACROS, MOTORS AND SCANS

Sardana provides a standard catalogue of reusable procedures (macros) and sequences for performing specific tasks such as scanning, acquiring data, controlling motors, etc. Besides, it offers the templates and tools for creating and maintaining a user repository of macros.

Although the names of the macros and the syntax might look compatible with SPEC, unfortunately, the differences in the technologies, parsing, grammars and hardware interfaces, made impossible to keep the compatibility with the existing SPEC macros. Hence, SPEC and Sardana are not compatible and macros can not be interchanged between them.

Macros are executed in a central process, called Macroserver. Typically there is a Macroserver in a Sardana installation, although there can be more than one if the application requires it.

Macros are python classes. They are executed and sequenced in the Macroserver, and can also be edited and debugged under its supervision.

Both Taurus based GUIs and CLIs (like spock) connect to the Macroserver through a TANGO object called the "Door". It is through this access point that macro execution can be controlled from outside the Macroserver. A Macroserver can have multiple doors but each door can only run a single macro at a time.

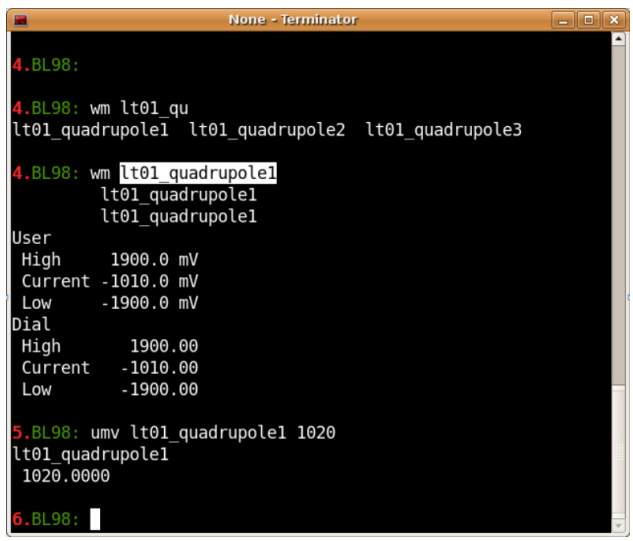

Figure 4: spock: The Command Line Interface of Sardana

When connected to a Macroserver, spock is fed with meta-information about the known macros and elements that are part of Sardana. This way, it can provide features like context sensitive word completion, command history, macro error handling and debugging. Graphical interfaces also receive this information enabling powerful widgets to be constructed .

Macros can be executed from the spock command line, which provides tab completion, history, etc. or from a graphical interface, having, favourites, recently used, etc.

Sequences are batches of macros. They can be easily created from the graphical user interface and also managed as favourites. One does not need to have notions of programming to create a sequence.

THE CORE

The core of Sardana is the so called "Device Pool". It is not only responsible to abstract specific hardware access but also to make sure this access is done as efficiently and as synchronized as possible. Hardware abstraction is possible by providing a set of interfaces to the outside world: Motors (discrete, continuous or pseudo), experimental channels (scalar, 1D and 2D or pseudo) and communication channels. Communication with specific hardware is achieved through the implementation of plug-ins known as "controllers". They can be written in Python or C++. A controller type exist for each interface supported by the Device Pool. When writing a controller, one must obey a specific interface in order for the controller to be considered valid. Controllers can be as simple as a mapping to another tango device server or as complex as a 4C diffractometer.

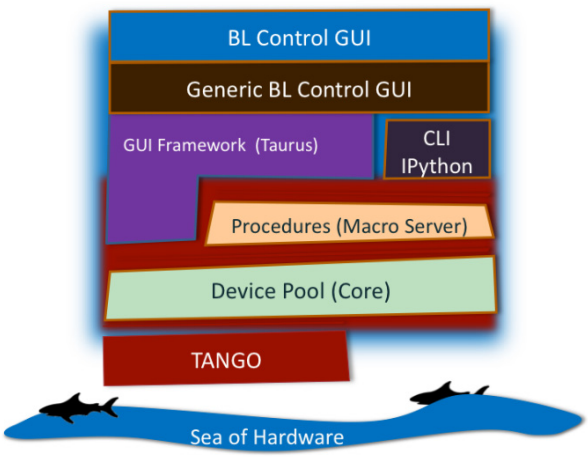

Figure 5: Conceptual design of Sardana.

NEXT STEPS AND STRATEGIC PLAN

In terms of strategy, we could publish the following mission statement: "*Produce a modular, high performance, robust, and generic user environment for control applications in large and small installations",* and the vision: *"Make Sardana the generic user environment distributed in the TANGO project and the standard basis of collaborations in control".*

Although Sardana has been used for the commissioning at Alba, and is installed and running in all Beamlines, is still in an early stage. There is a new release foreseen by the end of 2011, which will improve the experiment configuration, plotting, access to data files and overheads. Looking ahead to the version of 2012, our efforts will be focused among others in the configuration tool, full integration of 2D detectors, and continuous scans frameworks.

CONCLUSION

SCADAs are extensively used in industrial applications. They are optimized for access to Programmable Logic Controllers (PLCs), and popular field buses like Siemens Profibus, etc. Today there is no such a thing for scientific applications for which requirements are drastically different. Labview from National Instruments is a closer example, but it still lacks many important features, like complex "procedures". For example, Beamlines have devices such as diffractometers, combined with detectors, which needed to be scanned with monochromators, which are not well managed by Labview. SPEC manages all these cases, but it has poor Graphical Interfaces, and it does not allow concurrency and arbitration. There is place for improvement in this domain, and Sardana has included all these features in the requirements. There are still a considerable number of features pending and a great effort is still needed in this field. But Sardana is following the correct way, and several other labs (mostly synchrotron) have already expressed their interest.

CONTRIBUTIONS

Many people have worked in this project. In particular J. Ribas, who had a great contribution to the original "pytauico" and "pytauiwi", which was the seed for the current Taurus package. T. Nuñez and T. Kracht at Desy, D. Spruce and K. Larssen at MaxLab for the early tests and contributions to the Sardana device pool and Macroserver.

REFERENCES

[1] http://www.tango-controls.org. The TANGO official website.
[2] http://www.cells.es. The synchrotron light source ALBA. Cerdanyola del Vallès, Barcelona. Spain.
[3] http://www.desy.de. Desy: Deutsches Elektronen-Synchrotron. Hamburg, Germany.
[4] http://www.maxlab.lu.se. MaxLab. National Electron Accelerator Laboratory for Synchrotron Radiation Research, Nuclear Physics and Accelerator Physics. Lund, Sweden.
[5] Piotr Pawel Goryl et al. "Solaris Project Status and Challenges". Solaris Synchrotron, Krakow, Poland.
[6] http://www.aps.anl.gov/epics. EPICS official webpage. Experimental Physics and Industrial Control System.
[7] http://certif.com. SPEC official web page.
[8] http://www.tango-controls.org/static/sardana/latest/doc/html/users/index.html The official Sardana webpage.
[9] http://www.gnu.org/licenses/lgpl.html GNU Lesser General Public License.
[10] http://www.tango-controls.org/static/taurus/latest/doc/html/users/index.html. The official Taurus Documentation
[11] http://www.tango-controls.org/static/PyTango/latest/doc/html/users/index.html. The official PyTango Documentation
[12] http://www.riverbankcomputing.com/software/pyqt/intro PyQt: the official website.