ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

BUILDING S.C.A.D.A. SYSTEMS IN SCIENTIFIC INSTALLATIONS WITH SARDANA AND TAURUS*

D. Fernandez-Carreiras, J. Andreu, F. Becheri, S. Blanch-Torné, M. Broseta, G. Cuni, C. M. Falcon-Torres, R. Homs-Puron, G. Jover-Mañas, J. Klora (on leave), J. Moldes, C. Pascual-Izarra, S. Pusó-Gallart, Z. Reszela, D. Roldan, M. Rosanes-Siscart, A. Rubio, S. Rubio-Manrique, J. Villanueva, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Barcelona, Spain
T. Kracht, M. T. Nunez Pardo de Vera, DESY, Hamburg, Germany
T. Coutinho, A. Homs, E. Taurel, ESRF, Grenoble, France
V. Hardion, A. Milan, D. P. Spruce, MAX IV Laboratory, Lund, Sweeden
P. Goryl, L. Zytniak, L. Dudek, Solaris, Kracow, Poland

Abstract

Sardana and Taurus form a Python software suite for Supervision, Control and Data Acquisition (SCADA) conceived for scientific installations. Sardana and Taurus are open source and deliver a substantial reduction in both time and cost associated to the design, development and support of control and data acquisition systems. The project was initially developed at ALBA and later evolved to an international collaboration driven by a community of users and developers from ALBA, DESY, MAXIV and SOLARIS as well as other institutes and private companies. The advantages of Sardana for its adoption by other institutes are: free and open source code, comprehensive workflow for enhancement proposals, a powerful environment for building and executing macros, optimized access to the hardware and a generic Graphical User Interface (Taurus) that can be customized for every application. Sardana and Taurus are currently based on the TANGO Control System framework but also capable to inter-operate to some extend with other control systems like EPICS. The software suite scales from small laboratories to large scientific institutions, allowing users to use only some parts or employ it as a whole.

INTRODUCTION

ALBA is a third generation synchrotron located near Barcelona in Spain. The Beamlines started the operation in 2012. The ALBA installation relies on Ethernet as the standard fieldbus. This makes the installation homogeneous, easy to maintain and guarantees a good longevity of the components. About 1200 Ethernet connections, 5000 identified instruments, and 10000 variables stored in the permanent database form the infrastructure of the control system of the particle accelerator. Each Beamline comprises about 40 Ethernet connections, 200 instruments and 300 process variables stored in the permanent database. In addition, Beamlines have dedicated detectors, producing a throughput of 300 MB/s increasing every year with newer technologies [1]. Beamlines of the same institute or across different installations share the basic requirements, although depending on their particular purpose they may have specific requirements; detectors, sample environments, synchronization, etc.

One of the first decisions on the control system design was TANGO[2] [3] as a framework. TANGO was chosen among the three options considered. The other two were EPICS [4], and a commercial SCADA. 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 or the alarm handling, many applications needed to be developed "ad-hoc" and a significant effort was required to integrate motion, synchronization, the sequencer, and the scientific data formats. In other words, they were not a solution *per se*, but to be combined with EPICS, TANGO or other toolkits. This is the usual case, found for example at CERN, where PVSS (nowadays integrated in WinCC) is combined with other frameworks such as JCOP and UNICOS [5].

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, motion, timing and protection systems. Besides, many "smaller" control systems, one per experimental station, coexist, interact and in some aspects share information and resources with the central system. They usually have significantly different requirements. We needed a flexible graphical interface, allowing multiple clients, with a number of specific capabilities such as the control of diffractometers, and above all a powerful sequencer. Many of these characteristics are found in SPEC [6]. SPEC is a complete and powerful software tool, which for many years has been and still is, the *"de-facto"* standard control system for X-ray and neutron experimental stations. Still, SPEC has few limitations, when aiming for multiple graphical interfaces and in general managing multi-clients or using operating systems other than Unix. Therefore at that point ALBA decided to start a development of a SCADA for scientific institutions: It was named Sardana [7] [8].

Sardana was early presented to the TANGO community, where in particular DESY and the ESRF showed interest and contributions. Later it became a

* On behalf of the Sardana community.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

community with four formal members (light sources), DESY in Germany, MAXIV in Sweden, SOLARIS in Poland and of course ALBA in Spain, with a growing number of users coming from commercial companies and public institutes.

THE SCOPE OF SARDANA

Sardana provides optimized and standardized access to the hardware, generic graphical interfaces, save/restore of configuration and settings, software synchronization and macro execution. A powerful sequencer, which manages the edition and execution of macros and sequences, is mandatory in any experimental station in a synchrotron, and extremely useful in all cases. It forms a scientific SCADA suite, covering the weaknesses industrial SCADAs have for scientific applications, and enlarging the scalability of scientific packages [9].

Sardana was conceived as a solution for a synchrotron, but extendable to other scientific laboratories, like neutron sources. It aims to be flexible and scalable. It has been designed for suiting large installations like a particle accelerator, or smaller such as experimental stations, or small labs. Particle accelerators need synchronization, control and data acquisition for vacuum, power supplies, radio frequency stations, insertion devices and the multiple diagnostics. Besides, traversal systems, like equipment and personnel protection, monitor and control the overall installation. Experimental stations share some aspects, like vacuum and certain diagnostics. However, optics are different from accelerators'; usually motorized mirrors and crystals. Thus, the control system includes a number of motors, typically about a hundred per Beamline, mostly stepper and few servomotors, different kinds of experimental channels, detectors, and often spectrometers and diffractometers.

Sardana is developed in Python, which is a generally established language in the scientific community. It uses TANGO as a middleware. It is modular and open source, with components having a clear and documented interface, making it extendable by many programmers of different institutes. It is based on a client-server model where typically the server has two main entities: The Device Pool and the Macroserver, which will be detailed in the following sections.

Sardana focuses in the particular controls needs of experimental stations. It provides an environment to manage macro edition and execution, handling the different combinations of hardware and software. It manages multiple clients, and access to the hardware and storage of experimental data. It provides standard graphical and command line interfaces customizable and configurable, with no need of programming. This is the case for the creation of the control system and graphical interfaces. Macros are a special case that is covered in the next section.

SEQUENCER, MACROS, MOTORS AND SCANS

At the heart of the Sardana system is the standard catalogue of reusable procedures (macros) and sequences, which offers also templates and tools for creating and maintaining a user repository of macros. Examples of standard macros are *ct*, for acquiring with all channels defined during the specified time, *ascan, a2scan, dscan, mesh*, etc, for the different n-dimensional types of scans. Typically a scan is a sequence of motions of arbitrary movables and acquisitions of the experimental channels during a time and intervals defined.

The names of the standard macros and the syntax have been designed to follow SPEC conventions. SPEC is well known in the scientific community, and this is crucial to ease the learning process of users. Sardana and SPEC could coexist and interact in the same installation, mitigating this potential shortcoming.

As we introduced in the previous section, the macros are executed in a central process, called Macroserver, implemented as a TANGO device. Typically there is one Macroserver in a Sardana installation, although there can be more than one if the application requires it.

Macros are Python classes (Figure 1). They are executed and sequenced in the Macroserver, and can be also edited and debugged under its supervision. Macros can be executed from a Command Line Interface (CLI) built on top of IPython, or from the graphical interface. It is also common to start development as a Python script, which progressively incorporates more functionality. At any point, this is easily converted into a macro for a better integration into the control system.

Sequences are batches of macros. They can be created from the graphical user interface and are also administered in the history and favourites.

```
class set_lm(Macro):
    """Sets the dial limits on the specified motor"""
    param_def = [
        ['motor', Type.Motor, None, 'Motor name'],
        ['low',   Type.Float, None, 'lower limit'],
        ['high',  Type.Float, None, 'upper limit']
    ]

    def run(self, motor, low, high):
        name = motor.getName()
        self.debug("Setting dial limits for %s" % name)
        motor.getDialPositionObj().setLimits(low,high)
        self.output("%s limits set to %.4f %.4f (dial units)" % (name, low, high))
```

(a)

```
LAB-01-D01 [1]: ascan?

Syntax:
        ascan <motor> <start_pos> <final_pos> <nr_interv> <integ_time>

Do an absolute scan of the specified motor.
    ascan scans one motor, as specified by motor. The motor starts at the
    position given by start_pos and ends at the position given by final_pos.
    The step size is (start_pos-final_pos)/nr_interv. The number of data points collected
    will be nr_interv+1. Count time is given by time which if positive,
    specifies seconds and if negative, specifies monitor counts.

Parameters:
        motor : (Motor) Motor to move
        start_pos : (Float) Scan start position
        final_pos : (Float) Scan final position
        nr_interv : (Integer) Number of scan intervals
        integ_time : (Float) Integration time
```

(b)

Figure 1: (a) Definition of a macro to set limits of a given motor. (b) Execution of a macro from the command line interface. Question mark shows the documentation.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

The concept of scan and the extension to continuous scan is a key feature in a Beamline control and data acquisition system. Traditionally, a continuous scan needed a configuration "*ad hoc*" for one particular axis, with a lot of hardware involved to count encoder or motor pulses and generate external triggers for the channels involved. The generalization of this type of scans is strategic in all synchrotrons and potentially in any other laboratory. A generic continuous scan framework provides the flexibility of step scans in a continuous and synchronized data acquisition, allowing time-resolved experiments, or in any case speeding up the experiments and reducing the software and motion overheads. The complexity resides in using any arbitrary combination of movable elements, experimental channels, and detectors, generating the required triggers and evidently, allowing the coexistence of slow interpolated software-triggered channels, pseudocounters and pseudomotors with hardware triggered fast channels and readout encoders [10] [11].

HUMAN MACHINE INTERFACES

The graphical layer is Taurus[12]. It is written in Python and based on Qt. The Command Line Interface is built on IPython and makes extensive use of Taurus as well. It is known as SPOCK. As mentioned, in order to facilitate the learning process and although they do not share any code, the appearance, syntax, and the standard macros names, have been chosen similar to SPEC.

Sardana is truly multi-client, so several user interfaces are managed simultaneously; graphical interfaces, text interfaces running in different processes or locations, or even the command line interface embedded in the GUI.

Fully functional GUIs for controlling equipment can be quickly created in Sardana without having to write any code, but just configuring the different components of a standard generic GUI [13]. In the cases where a more specialized GUI or widget is required, it can be created using the Qt designer, for which a plug-in providing a catalogue of Taurus widgets is available.

The principal example is the user interface to the control system of a Beamline, which is typically Sardana installation with at least a Device Pool and a Macroserver. The generic GUI provides of-the-shelf panels for managing macros and sequences, as well as standard panels associated with the instruments defined in the Device Pool. Often the GUI includes also custom widgets and synoptics for visualizing the system. The layout of the GUI can be configured in different perspectives, allowing switching from different panel arrangements suited for different applications.

DATA MANAGEMENT

Data management covers several aspects, as shown in Figure 2. Data is the result of an experiment, which must be complete, and must contain the necessary metadata for the data analysis to succeed. Besides, data sets could eventually be completed on-line or modified off-line to contain analysed and processed data. Sardana uses a modular system for producing outputs. It natively supports the HDF5 – NeXus data format for the experimental data, but it can also write to other formats such as SPEC files or DESY´s FIO format for backwards compatibility. Data sources are typically the Macroserver and the Device Pool, although it can be any component present in the control system. The control system offers also other possibilities, like relational databases for infrastructure and historical data, although this is not yet fully integrated in Sardana.

SPEC files have a plain-text format (ASCII). They store the scan results, with some metadata such as motor positions, and the set of motors and experimental channels intervening in the scan. One-dimensional (1D) data could also be archived in ASCII format. This is not the case for two-dimensional (2D) data, which require extra binary files, like EDF (ESRF Data Format). HDF5-NeXus binary files on the other hand, can contain all data and metadata regardless of dimensionality in a single compact file, which complies with well-documented standard application definitions and which can be accessed with generic and widely supported HDF5 libraries and tools. The data files in both HDF5 or SPEC can be then analyzed with different software packages such as PyMCA [14] or fit2d [15].

Specific data recorders can be added for different data files or applications. In this sense, specific data recorders manage specific application definitions in NeXus files. Moreover, Sardana can use several recorders simultaneously, and therefore write the data in different formats at the same time (Figure 2).

The intercommunication between the different components is assured by TANGO itself, in few cases JSON-encoded (JavaScript Object Notation) for complex structures.

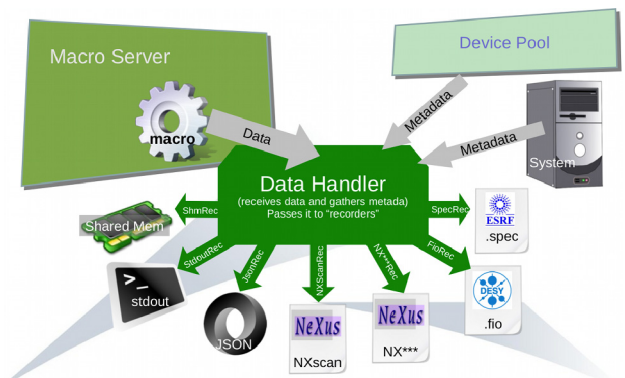

Figure 2: Sardana is driven by the data acquisition, managing data and metadata for data analysis and on-line information for control.

THE CORE

The core of Sardana manages the access to the hardware. It is implemented as a TANGO device, called "Device Pool". It normally runs altogether with the

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

"Macroserver device" in the same computer or even in the same process, although this is not mandatory. Large Installations may have several Device Pool instance that can be accessed from the Macroserver. The Device Pool provides access to the hardware in an efficient way. Sardana provides different abstractions. The first distinction comprises two kinds: Movable elements such as motors or pseudomotors and experimental channels, like, counters/timers, 0D (zero dimensional), 1D, or 2D detectors. Movable elements are typically motors associated to a physical axis, but they can also be associated to other devices (able to scan), like power supplies or temperature controllers. Counter/timers are scalars that implement the concept of integration or accumulation over a defined interval, typically corresponding to a detector exposure. 0D channels represent the type of scalars whose magnitude is intrinsically normalised by time (electric current, photon flux, pressure, temperature, etc.), so their values are averaged over the measurement interval. The Device Pool provides automatic averaging operations for controllers that only implement signal sampling, as well as the optional parameter integration over time in order to simplify arithmetic with accumulating counters. 1D devices generate data that can be represented by vectors, usually coming from detectors such as Position Sensitive Detectors (PSD), Multi Channel Analyzers (MCA) or other dispersive techniques, which measure in many cases the spectral dependence (or decomposition) of a magnitude. 2Ds [16] or two-dimensional data is typically associated with imaging detectors, such as CCD/CMOS cameras, pixel array detectors or frame grabbers. Besides movable and experimental channels, there are other additional types, useful for building interfaces and in general control systems. Those are for example, communication channels, enumerated scalars called I/O registers, or complex objects such as sample changers.

The elements are organized in controllers. A controller has one or more elements. The elements in a controller typically share some functionality. For example, a motor controller has often many motors or axes associated. Those axes can be started synchronously by one single command. So the controller can exploit this feature with two main goals: to optimize the access to the hardware and to push multi-device synchronisation towards its minimum latency and jitter. A movable can be of many types, such as a stepper motor, a DC motor, a piezo linear actuator, etc. Different kind of movable elements, different manufacturers or different models offer different functionalities. The device pool handles this diversity, provides homogeneous interfaces, optimizes the communication and handles the synchronization.

Sardana offers also complex controllers and elements, which combine physical elements and calculations. For example, the energy is often calculated as a result of Grating pitch or Bragg angles combined with the position of mirrors and crystals, the gap and offset of the slits are often the result of the position of four physical blades, the height, and pitch of a table are a function of the position of the legs. These axes are called pseudomotors. In the same way, occasionally, a calculation from several channels is needed: for example, a combination of two scalar counters with one encoder position. The result would be another channel integrated in plots, and data files as any other experimental channel. This is called a pseudocounter.

Thus, the Device Pool manages comprehensively the access to the hardware, handling the interdependencies between logical and physical channels, synchronization, and concurrent access of the different components.

Although the Device Pool and the Macroserver can be in different process and run in different computers, for small installations they are usually two devices in the same process. They are complementary to each other. Graphical interfaces can have direct access to the Device Pool, but other user interfaces, for example SPOCK, go only through the Macroserver. A Macroserver has a number of client connection points called "Doors" which are also TANGO devices. Each client uses a dedicated Door to request the execution of macros by the Macroserver. They were designed to run the macros in independent threads. This feature allows the parallel execution of macros on separate Doors, provided that no conflict exists due to macros modifying (that is, writing on) the same hardware device.

A small installation has a Sardana server, which has only one Device Pool device, only one Macroserver device and one or more Door devices associated to it.

Sardana has run up to now integrated in a TANGO installation. Interfaces with other systems are under development and tests. EPICS, SPEC, or even TACO (the predecessor of TANGO), can be interfaced from Sardana, either from the Device Pool as an integrated interface to the hardware in order to be accessed from macros, or on the upper level as a scheme in Taurus for simple channel connections from graphical interfaces.

COLLABORATION

The Sardana code was migrated to a public SourceForge repository in 2011. The Taurus code, very popular in the Tango community, was moved to a separated repository to simplify those numerous uses that did not need the Device Pool-Macroserver parts. In 2016, Sardana and Taurus migrated to GitHub. The pull-requests and code-review workflows proved to be more convenient for a growing collaborative community[17]. Inspired on the Debian Enhancement Proposals (DEP), the SEPs and TEPs (Sardana and Taurus Enhancement Proposals) define the workflows for promoting new features. A number of SEPs have been successfully completed such as the generic continuous scan (SEP6) or the integration of HKL reciprocal space coordinates (SEP4). Automated tests, test-driven-development and continuous integration / continuous delivery paradigms were as well introduced. The documentation is continuously updated and available on the Read-The-Docs platform. Two releases per year are scheduled. All these efforts on standardization and openness allowed

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

Sardana/Taurus to become a collaboration with four members-contributors, a growing user community –some of them spontaneous contributors as well – and where up to now, the memorandum of understanding was not needed[18].

CONCLUSIONS AND FUTURE PERSPECTIVES

Sardana was initially developed at ALBA to overcome some of the limitations of the existing SCADA solutions. The industrial solutions miss scientific hardware and functions, and the scientific solutions like SPEC, Labview or Matlab, may lack to some extend of scalability, concurrency and arbitration. Sardana, is installed and running in the particle accelerators and Beamlines at ALBA. It is in operation in nine Beamlines at DESY, and all accelerators and Beamlines in MAXIV and SOLARIS. There are still a considerable number of features pending, and a great effort is still needed in this field. But Sardana is following its schedule, and several other labs and private companies are using it or have already expressed their interest. Due to the open source nature of the project, the Sardana development is likely to gain considerable momentum. The latest release, published in of 2017, improves its performance, as well as the experiment configuration, plotting and access to data files. It also natively integrates continuous scans. Looking ahead to the upcoming versions, the efforts of the community will be focused among others in the configuration tools, full integration of 2D detectors, and continuous scans frameworks with complex movements and trajectories.

The configuration tools are crucial for gaining users among small labs, making the configuration and access to hardware catalogues simpler and intuitive.

ACKNOKLEDGMENTS

Many people made important contributions to this project. In particular the authors would like to thank J. Ribas, R. Suñé, on early versions of Taurus and PyTango, F. Picca at Soleil for his contribution on the H.K.L library, packaging and beta tests. In addition, the authors would like to thank A. Götz, from the ESRF and N. Leclercq, from Soleil for their comments and valuable feedback. Thank you also to all members of the Computing Division team at ALBA, all four sections, Controls, Electronics with O. Matilla, IT-Systems with A. Pérez and Management Information Systems with D. Salvat. Finally the authors would like to thank the user community for their understanding, their patience, continuous feedback and contributions to the project.

REFERENCES

[1] D. Fernández-Carreiras, D. Beltran, T. Coutinho, G. Cuní, J. Klora, O. Matilla, R Montaño, C. Pascual-Izarra, S. Pusó, R. Ranz, A. Rubio, S. Rubio-Manrique, R. Suñé. “The design of the ALBA Control System; A cost-effective distributed hardware and software architecture”, in *Proceedings of ICALEPCS’11*, p1318. Grenoble. France.

[2] J-M. Chaize, A. Götz, W-D. Klotz, J. Meyer, M. Perez, E. Taurel, “Tango. An object oriented control system based on Corba”, in *Proceedings of ICALEPCS’99*, Trieste, Italy.

[3] TANGO website, http://www.tango-controls.org/.

[4] EPICS website: http://www.aps.anl.gov/epics

[5] H. Milcent, E. Blanco, F. Bernard, P. Gayet, “UNICOS: An open framework”, in *Proceedings of ICALEPCS’09*, Kobe, Japan.

[6] SPEC website, http://www.certif.com/content/spec

[7] T. Coutinho *et al.*, “SARDANA: The software for building SCADAS in Scientific Environments”, in *Proceedings of ICALEPCS’11*, Grenoble, France.

[8] Sardana doc: http://www.sardana-scada.org.

[9] Z. Reszela, G. Cuni, D. Fernández-Carreiras J. Klora, C. Pascual-Izarra, T. Coutinho, “Sardana- a python based software package for building scientific SCADA applications”, in *Proceedings of PCaPAC’14,* Karlsruhe, Germany.

[10] D. Fernández-Carreiras, F. Becheri, T. Coutinho, G. Cuní, R. Homs, G. Jover-Mañas, J. Klora, O. Matilla, J. Moldes, C. Pascual-Izarra, Z. Reszela, D. Roldan, S. Rubio-Manrique, X. Serra. “Synchronization of motions and detectors and continuous scans as a standard data acquisition technique”, in *Proceedings of ICALEPCS’13*, San Francisco, USA.

[11] Z. Reszela, F. Becheri, G. Cuní, C. Falcón-Torres, D. Fernández-Carreiras, R. Homs-Puron, J. Moldes, C. Pascual-Izarra, R. Pastor-Ortiz, D. Roldán, M. Rosanes-Siscart, “Sardana based continuous scans at ALBA. Current Status”, these proceedings. *Proceedings of ICALEPCS’17*, Barcelona, Spain.

[12] C. Pascual-Izarra, *et al.*, “Taurus big and Small: From particle accelerators to Desktop labs”, these proceedings. *Proceedings of ICALEPCS’17*, Barcelona, Spain.

[13] C. Pascual Izarra et al. “Effortless creation of Control & data acquisition graphical user interfaces with Taurus. *Proceedings of ICALEPCS’15*, Melbourne, Australia.

[14] V.A. Solé, E. Papillon, M. Cotte, Ph. Walter, J. Susini, “A multiplatform code for the analysis of energy-dispersive X-ray fluorescence spectra, Spectrochim”, Acta Part B 62 (2007) 63-68.

[15] A. P. Hammersley, “FIT2D: An Introduction and Overview", ESRF Internal Report, ESRF97HA02T. 1997.

[16] A. Homs, L. Claustre, E. Papillon, S. Petidemange, “LIMA: A generic Library for High throughput

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA169

image acquisition", in *Proceedings of ICALEPCS'11*, 676, Grenoble, France.

[17] Z. Reszela, *et al.,* "Bringing Quality in the control software delivery process", in *Proceedings of ICALEPCS'15*, Melbourne, Australia.

[18] C. Pascual-Izarra, et al. "Community driven scientific software projects: Lessons Learned on Tools and Practices", in *Proceedings of NOBUGS 2016*, Copenhagen, Denmark.