SARDANA – A PYTHON BASED SOFTWARE PACKAGE FOR BUILDING SCIENTIFIC SCADA APPLICATIONS

Zbigniew Reszela, Guifre Cuni, David Fernandez-Carreiras, Jorg Klora [on leave], Carlos Pascual-Izarra, CELLS-ALBA Synchrotron, Cerdanyola del Vallès, Spain
Tiago Coutinho, ESRF, Grenoble, France

Abstract

Sardana is a software suite for Supervision, Control and Data Acquisition in scientific installations. It aims to reduce cost and time of design, development and support of the control and data acquisition systems[1]. Sardana, thanks to the Taurus library[2], allows the user to build modern and generic interfaces to the laboratory instruments. It also delivers a flexible python based macro environment, via its *MacroServer*, which allows custom procedures to be plug in and provides a turnkey set of standard macros e.g. generic scans. Thanks to the *Device Pool* the heterogeneous hardware could be easily plug in based on common and dynamic interfaces. The Sardana development started at Alba, where it is extensively used to operate all beamlines, the accelerators and auxiliary laboratories. In the meantime, Sardana attracted interest of other laboratories where it is used with success in various configurations. An international community of users and developers[3] was formed and it now maintains the package. Modern data acquisition approaches guides and stimulates current developments in Sardana. This article describes how the Sardana community approaches some of its challenging projects.

SARDANA AND ITS COMPONENTS

Sardana is a distributed control system based on the client-server model. The communication protocol is Tango[4]. Different Sardana configurations are possible depending on the scale of the installation. As an example, a small laboratory could have a single Sardana server exporting one Device Pool and one MacroServer. If needed, multiple Device Pool servers could be distributed over different hosts. Configurations with many MacroServer servers are also possible. Multiple Graphical User Interface (GUI) and Command Line Interface (CLI) clients can communicate with the Sardana system, at the same time.

Device Pool

Scientific installations are characterized by multiple and heterogeneous hardware. The particle accelerators comprises power supplies, vacuum equipment, radio frequency stations, insertion devices and many diagnostics and actuators among others. The laboratories, e.g. a synchrotron beamline, usually consists of even more diverse instruments like, diffractometers, monochromators and sophisticated detectors full of moveable axes and experimental channels. These laboratories frequently require to modify their configuration depending on the experiment being performed. Sardana interfaces all the equipments via the Device Pool and its plugin *controller* classes. Sardana elements could be classified by their types in three main groups:

- moveables: physical motors e.g. stepper or servo motors, piezo actuators and logical pseudo motors e.g. the energy, composed translations
- experimental channels: counters and pseudo counters; 0D – scalar values based on mathematical operations e.g. averaging or integration of samples; 1D – one dimensional detectors e.g. Multi Channel Analyzers, Position Sensitive Detectors; 2D – two dimensional detectors e.g. CCD cameras
- other elements: communication channels, enumerated scalars called I/O registers

The controllers group the Sardana elements, which are organized and identified by the axis number. All these elements are represented by TANGO devices, and could be spread in many Device Pool servers or grouped in a single one. The Device Pool optimizes common control processes. The grouped acquisitions are handled by the M*easurement Group*, which configures and synchronizes the experimental channels. The grouped motions are implemented inside the Pool and if the hardware allows that, motion of all the axes can be started simultaneously by one single command. The API of the controllers and the programmed algorithms take into account all these particularities so access to the hardware is optimized.

MacroServer

A basic requirement for the scientific SCADA system is to provide a sequencer capable to plan and execute experiment procedures. MacroServer, together with its *Doors*, provide these and other functionalities (via Door different client applications access the MacroServer engine). The MacroServer allows the execution of multiple procedures (called *macros*) sequentially or even simultaneously if different Doors are used. A macro can accept input parameters and return a result or produce data, which might be interchanged between chained or nested macro sequences. It is possible to interrupt the sequence execution at any moment as well as temporarily pause and resume it. Features for adding, editing and browsing the available macros are accessible via the MacroServer at runtime. Sardana provides a miscellaneous set of standard macros. Their naming and parameters often follow conventions adopted by SPEC[5] what optimizes the users learning curve. Probably the most useful and sophisticated macros are the generic, n-dimensional scans. The scan process consist of data acquisition of the involved experimental channels while varying the scanning variable (typically a moveable

ISBN 978-3-95450-146-5

object). Scans in Sardana are available in various modes: step, hybrid and continuous, where distinct actions like motion, data acquisition or data storage are synchronized and optimized. Sardana allows the scan data to be stored in many different formats. This process is handled by one or more optional *recorders*. They receive the scan data and the experiment metadata from the scanning macro and despatch them to the destination in the correct format e.g. a file, console output, a data post-processing program.

Taurus Based Human Machine Interfaces

Sardana client applications (CLI and GUIs) are developed with the Taurus library. Taurus was originally conceived for connecting client side applications to Tango device servers using the PyTango library. For the GUI part, Taurus is built on top of PyQt[6]. Taurus is designed with a Model-View-Controller (MVC) architecture and has currently transcended its Tango-centric origins by accepting *scheme* plugins. A scheme plugin provides the model objects for a certain source of data and/or of controllable variables. Schemes exist for several hardware access layers (Tango, Epics, SPEC) and even for interacting with a Python interpreter. Each model object has a unique URI based name and it can be transparently used by the higher levels of the library. This makes it possible to mix in one application variables from different control systems, values retrieved from the archiving systems or any other source of data.

Taurus comprises a complete set of widgets (forms, plots, macro execution and experiment configuration panels, etc.) which provide the View and Controller components of the MVC and which assure a standardized look-and-feel of the user interfaces. They are easy to program thanks to the user-friendly API and are even integrated in the standard graphical Qt designer. The TaurusGUI and its configuration wizard allows the creation of complete GUI applications with just few clicks. The GUI is typically configured with a synoptic view representing a number of instruments, and can be managed with different perspectives. All these features considerably speed up the application development process (to the point that users often create ad-hoc temporary GUIs to solve particular tasks).

Spock is a Command Line Interface built on IPython and makes extensive use of Taurus as well. Its main functionalities are macro execution and control of any Sardana element. Spock provides context tab completion and an easy access to the history of executed commands which makes it user-friendly and efficient.

CURRENT & FUTURE DEVELOPMENTS

Testing Framework

Many enhancements are currently being evaluated and developed. Some of them require refactoring of the Sardana and Taurus core concepts, which increases the risk of regression. The Sardana Testing Framework was developed in first order to mitigate this risk. The framework provides guidelines regarding test organization, naming conventions and the required test case documentation. It comprises a base test case and test suite classes as well as tools to ease tests development. From now on, many new developments follow the test driven development process. As the next step, it is planned to set up the continuous integration service, which will enable the real benefits of the Sardana Testing Framework.

Continuous Scans

The Generic Scan Framework was recently extended by the continuous mode, however present solutions still lack of generalization on various levels. The future scans must provide transparency between the different synchronization modes of the experimental channels and external attributes involved in the acquisition process. The precise timestamp distributed over the involved hardware and hosts could be used for software triggering and data timestamping. A new element type – trigger/gate and a corresponding controller will be added to the Device Pool. Their interfaces must be determined to allow configurations of equidistant or arbitrary sequences of events in time or distance domains. Modern devices could deliver multiple functionalities e.g. one device could control a moveable axis, generate trigger signal and provide an experimental channel of the position measurement. The current design of the controller concept needs to be enhanced to allow handling elements of different types by one controller class. Usually the continuous scan requires the acquisition process to be performed while the scanning variable changes with a constant rate. In the case of the pseudomotors with non-linear formulas this require control of the motion trajectories of the involved physical motors. Current implementation approximates this cases and maintains the constant velocity of the physical motors.

Other Enhancements

A diffractometer is a common instrument in the synchrotron beamlines and is available in various geometries e.g. Eulerian 4-circle or 6-circle, kappa 4-circle etc. Experiments involving diffractometers are based on scans in reciprocal space, which require complex diffraction calculations. A base diffractometer controller class (psudomotor) uses the HKL library[7] and encapsulates all the complex calculations. A set of derived classes just defines the geometric specific physical and pseudo motor roles. In addition, any diffractometer instrument, could be controlled with the specific Taurus based widgets e.g. diffractometer alignment or hkl scan.

High speed and high resolution detectors and cameras require advanced control and optimized image post processing. The Lima library[8] fulfils these requirements. It was used to develop an early version of a 2D experimental channel controller. Its functionalities allow scans involving 2D experimental channels. Further

ISBN 978-3-95450-146-5

improvements are already planned e.g. to access the Lima objects directly instead of accessing them via Tango or to allow passing references to image data instead of the full raw images.

While some laboratories (such as Alba) use Sardana and Taurus as a complete scientific SCADA suite, other users are interested in using some components integrated in their running control system. The installation can be tailored and/or extended to fit different needs. Taurus and Sardana currently provides several extension points e.g. schemes, domain specific widgets, core extensions, macros, controllers, recorders, but these are implemented in different ways. A generic plugin system is one of the future goals which would not only make currently mandatory dependencies optional, but would also simplify and unify the extension of Taurus and Sardana.

SARDANA COMMUNITY

The origins of Sardana reach 2005, when early decisions about the ALBA Control System were taken. First the TANGO was chosen as the control system framework for ALBA. Common requirements of the beamlines sketched the first Sardana specification. The development process started immediately, and the core of the system was mostly maintained by one developer. Sardana was successfully commissioned during the installation and commissioning of the accelerators and beamlines which was completed in 2012. It grew to a mature product and other institutes, mainly synchrotrons, selected it as their control system. Sardana is free and open source (LGPL) which attracts many new users. The big interest of current and potential users together with many ongoing enhancement projects lead Alba to open also the project management by pushing for the formation of the Sardana Community in 2013. Sardana is currently used in DESY – Germany, MaxIV – Sweden and Solaris – Poland. These three institutes, together with Alba, form the actual collaboration. To a lesser extend, Taurus is also used at the ESRF and within Tango collaboration (by most official and unofficial participants). All these institutes actively participate in the community activities. Commercial support is given by the Nexeya and the Cosylab.

The first decision of the community was to formalize discussion process about the proposals of improvements and modifications. The Sardana Enhancement Proposal (SEP) process was introduced, and up to now 12 SEPs are defined (some of them already accepted). All of the Sardana repositories, the core and the third-party macros and controllers were migrated from SVN to GIT, which fits better to Sardana Community profile. The code contribution workflow, the branching model and naming conventions were defined. In order to provide the highest quality of the Sardana core code, all its contributions must pass through an open and transparent public code review and integration process. Another organizational change, which is currently in progress, is the Taurus library separation. Sardana and Taurus codes were isolated and soon the Taurus code will be migrated to its own GIT repository and project. Sardana is hosted on the Souceforege platform and uses a number of offered tools e.g. an issue tracker, mailing lists, wikis etc. They are widely used by the Sardana Community (both users and developers) in their daily life. The Sardana Workshop meetings, often organized as satellites to the Tango Meetings, take place once a year and the virtual conferences are organized according to the needs.

The Sardana and Taurus release cycle includes two official releases per year: in January and in July. The latest installers, for Linux and Windows platforms, are available to download from the PyPI repository or could be installed directly with pip or easy_install. Sardana and Taurus could also be installed directly from sources, which could be obtained from their GIT repositories, with the setuptools installation. Debian users have also access to the official debian packages.

ACKNOWLEDGEMENT

Many people have worked in this project. This is a work achieved with the effort of the whole ALBA controls group. The ALBA scientists took part in the specification refinement process, what has to be recognized, together with the beam time offered for the acceptance tests. Important contributions were received from T. Nuñez and J. Kotanski from DESY, F. Picca from Soleil and V. Valls, E. Taurel (who as the Tango expert guided the initial steps of the Sardana developments), A. Homs and V. Rey from the ESRF. The authors would also like to thank the experts from other institutes, for their valuable feedback and ideas, in particular to T. Kracht from DESY, N. Leclerq from Soleil, D. Spruce and A. Milan from MaxIV and P. Goryl and L. Żytniak from Solaris.

REFERENCES

[1] T. Countinho et al. “Sardana, The Software for Building SCADAs in Scientific Environments”, ICALEPCS2011, Grenoble, WEAAUST01
[2] The Taurus library web page: http://www.taurus-scada.org
[3] The Sardana project web page: http://www.sf.net/projects/sardana
[4] TANGO web page: http://www.tango-controls.org
[5] SPEC web page: http://www.certif.com/spec.html
[6] The PyQt library web page: http://www.riverbankcomputing.com/software/pyqt
[7] The HKL library web page: http://www.synchrotron-soleil.fr/portal/page/portal/Instrumentation/EnvironnementInstrumental/hkl
[8] The Lima library web page: http://lima.blissgarden.org

ISBN 978-3-95450-146-5