TANGO FOR EXPERIMENT CONTROL

J.Meyer, L.Claustre, S.Petitdemange, O.Svensson, A.Götz, ESRF, Grenoble, France
T.Coutinho, J.Klora, ALBA, Barcelona, Spain
F.Picca, M.Ounsy, A.Buteau, SOLEIL, Paris, France

Abstract

The Tango control system framework allows you to control an accelerator complex as well as single equipment. The framework contains the communication bus with the standard communication modes (synchronous, asynchronous, event driven) as well as the basic hardware access modules, GUI tools and development kits, bindings to commercial products (LabView, Matlab, IgorPro) and services (administration, archiving, access control) to set up a control system.

Tango was mainly developed by several synchrotron light sources that have to support not only the accelerator complex but also a lot of experimental end stations. For synchrotron experiments we have to control the whole process from basic hardware access over data taking to data analysis.

This paper describes in the first part the special features of Tango allowing flexible experiment control. The dynamic configuration, the rapid hardware interface development and the sequencing and scanning framework are some examples.

The second part gives an overview of some packages developed in the Tango community for experiment control: A HKL library for diffraction computation and diffractometer control, a library to control 2D detectors and a data analysis workbench with workflow engine for on-line and off-line data analysis. These packages are not part of Tango and can be used with other control systems.

TANGO BASICS

What is Tango?

Tango [1] is a control system tool kit developed by a community of institutes. It is object oriented with the concept of devices (objects) for each piece of hardware or software to be controlled. Tango classes are merged within operating system processes called Device Servers. Three types of communication between clients and servers are supported (synchronous, asynchronous and event driven).

But Tango is not only the software bus which handles the communication between device servers and clients. The Tango tool chain offers software from the hardware interface to the graphical user interface for several programming languages as shown in Table 1.

Tango utilities are available, with the basic installation, for code generation, device configuration and testing and for administration and survey of a whole Tango control system.

An archiving and a configuration snapshot system usable with Oracle or MySQL are also available.

Table 1: Available Tango Modules.

Module	Description
Core Libraries	Client/Server communication libraries for C++, Python and Java
Device Classes	About 300 hardware interface classes are available to download [2]
GUI Frameworks	Available for C++ and Python using QT, for Java using Swing and a web interface written in PHP
Client Bindings	LabView, Matlab and IgorPro
Tools	Pogo – Code generator for device classes in C++, Python and Java Jive – Configuration and testing tool Astor – Administration and survey of the Control system
Archiving	Archiving and snapshot system with GUIs and web interface. Usable with Oracle and MySQL
Alarm System	Event driven alarm service
Sardana	Framework for experiment control : Interface standardization, configuration, sequencing, command line interface

Rapid Interface Development

Important for flexible experiment control is a short development cycle to interface and integrate new hardware into the control system. Tango with its code generator (Pogo) offers a very comfortable way of implementing new hardware interfaces. The graphical interface builder allows the user to design the network interface and the main functionality (state machine, refreshing rate, memorization, etc.) of a device server.

The device server code can be generated in three programming languages: C++, Python and Java. The user has only to fill-in the code necessary to access the underlying hardware. For any change of the interface or for example the state machine, Pogo can be used to modify the existing code (Fig. 1).

Dynamic System Configuration

For experiment control systems distributed over several hosts it is very handy to have a centralised way to survey and to reconfigure running software. Tango offers such an administration system which is part of the core package. Two parts are necessary, the Starter server which is running on the distributed hosts and the graphical administration interface Astor (Fig. 1).

ISBN 978-3-95450-124-3

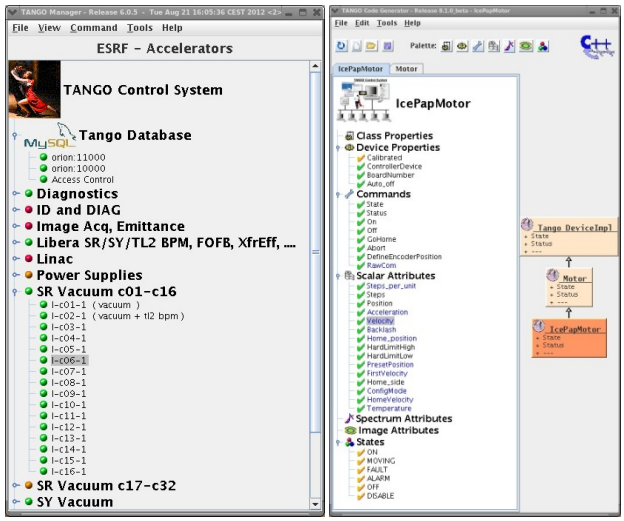


Figure 1: Astor and Pogo.

With Astor it is possible to survey all running servers on the distributed hosts, start or stop servers, run generic test tools, start the configuration interface (Jive) or set-up the logging service on devices. Astor is designed as the central entry point to the distributed control system.

Scanning and Sequencing

The Tango framework offers two ways to prepare scans and sequences for an experiment.

1. The SARDANA project [3], developed at ALBA, offers a generic user environment for scanning and sequencing in Python.

   Today the user environment consists of a highly configurable standard graphical user interface (Fig. 2), a command line interface understanding SPEC [4] commands, and a way to compose new applications either by programming or with a graphical interface builder. It further consists of a Python macro executer, a standard set of macros, a range of common hardware types (like motors, counters, cameras, etc.) and a configuration editor to set all this up.

Figure 2: Sardana Configuration GUI.

2. At SOLEIL a Tango scan device server [5] was developed which offers sophisticated scanning and data acquisition functionality. To implement experimental procedures the graphical work flow editor Passerelle [6], was customized (Fig. 3). This allows even non programmers to prepare sequences to run an experiment.

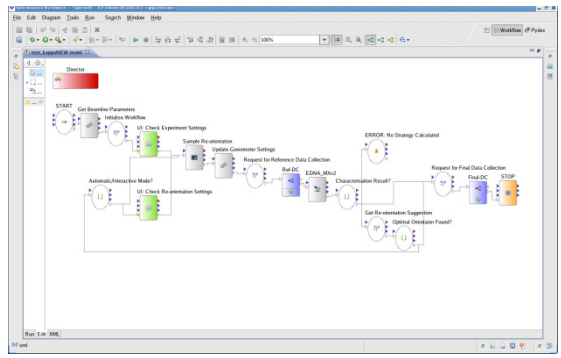


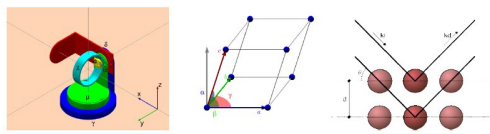
Figure 3: Passerelle Work Flow Editor.

PACKAGES FOR EXPERIMENTS

On top of the basic Tango control system several packages are available, inside the collaboration, which can be helpful for experiment control. When using Tango as control system the package integration is immediate. But they are Tango independent and can be adapted to other underlying systems.

Diffractometers

The Soleil synchrotron in Paris, France developed a C-library for reciprocal space transformations [7]. The purpose of the library is to factorise single crystal diffraction angles computation for different kinds of diffractometer geometries (Fig. 4).



THE equation

$$RUB\vec{h} = \vec{Q} = \vec{k}_f - \vec{k}_i$$

Figure 4: Reciprocal space transformation.

The main features are:

- Mode computation
- UB matrix computation
- Crystal lattice refinement
- Pseudo axes (psi, eulerians, q, ...)

Today the HKL library can handle 5 different geometries:

- 2 circles.
- Eulerian 4 circles
- Eulerian 6 circles
- Kappa 4 circles
- Kappa 6 circles

and supports operation modes as:

ISBN 978-3-95450-124-3

- Bisector
- Constant omega, chi, phi

For a usage within the Tango control framework a configurable device server based on the HKL library and the corresponding graphical user interface (Fig. 5) are available for download [8].

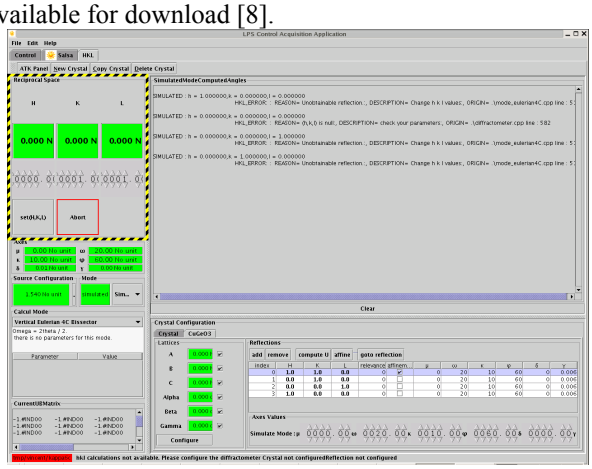


Figure 5: Diffractometer GUI.

2D Detectors

A significant number of 2D detectors are used in large scale facilities' control systems for quantitative data analysis. Common control parameters and features can be identified in these devices, but most of the manufacturers provide specific software control interfaces. A generic image acquisition library, called LImA [9], has been developed at the ESRF for a better compatibility and easier integration of 2D detectors to existing control systems.

Detector	Interface
Basler	GigE
Prosilica	GigE
Ueye (IDS)	GigE
Pilatus (Dectris)	300K, 1M,2M,6M, 6M-F
PCO Edge	Camera-Link
Pco Dimax	GigE + Camera-Link
PhotonicScience 4022	USB
RoperScientific	PVCAM SDK
Andor I-Kon	USB
PerkinElmer Flat-Panel	Proprietary board
ADSC 315r	Proprietary board
MarCcd 165	Proprietary board
Mythen (strip detector)	Proprietary board
XPAD	Proprietary board
Maxipix (ESRF)	Espia
Frelon 2K (ESRF)	Espia

Figure 6: LImA Detector Plug-ins.

The main design goals are:

- Control system independence
- A rich set of common functionalities, providing alternative software solutions when features are not implemented by the hardware.
- Intensive use of multi-threaded algorithms, synchronised by events for ensuring the performance of high throughput detectors.

Today we have already a big number of available detector plug-ins for LImA developed by the Tango community (Figs. 6 and 7).

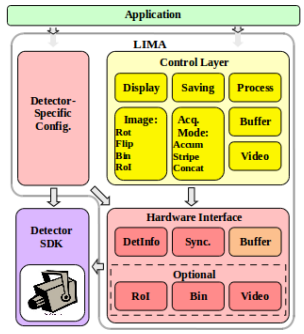


Figure 7: LImA Internal Structure.

Data Analysis Work Bench

The Data Analysis WorkbeNch (DAWN) [10] is an eclipse [11] based workbench for doing scientific data analysis (Fig. 8). Developed in collaboration by Diamond and the ESRF, it implements sophisticated support for:

- Visualization of data in 1D, 2D and 3D
- Python script development, debugging and execution
- Workflows for analyzing scientific data calling Python and binary codes (implemented with the Passerelle work flow editor [6] : see Figure 3)

DAWN is not restricted to one scientific domain. It is developed by and for the synchrotron community foremost but has strong overlap with other communities like neutron scattering, photon science and any scientific communities with the above or similar needs.

DAWN is being used at the ESRF and Diamond for automating online and offline data analysis.

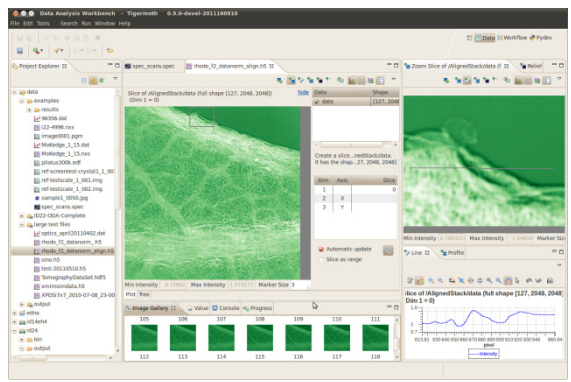


Figure 8: Data Analysis Workbench.

REFERENCES

[1] http://www.tango-controls.org
[2] http://www.tango-controls.org/device-servers
[3] http://www.tango-controls.org/static/sardana
[4] http://www.certif.com
[5] http://tango-ds.cvs.sourceforge.net/cvsroot/tango-ds/InputOutput/Scan
[6] http://code.google.com/a/eclipselabs.org/p/passerelle
[7] http://people.debian.org/picca/hkl
[8] http://tango-ds.cvs.sourceforge.net/cvsroot/tango-ds/Instrumentation/Diffractometer
[9] http://lima.blissgarden.org/
[10] http://www.dawnsci.org
[11] http://www.eclipse.org

ISBN 978-3-95450-124-3