THE LMJ TARGET DIAGNOSTICS CONTROL SYSTEM ARCHITECTURE

S. Perez CEA/DIF, Bruyères le Châtel, 91297, Arpajon, France
T. Caillaud, CEA/CESTA, Le Barp, 33114, France

Abstract

The French Laser Megajoule (LMJ) is, behind the US NIF, the second largest inertial fusion facility in the World. More than 30 diagnostics will be installed and driven in a huge and complex integrated computer control system. The aim of this paper is to describe an architecture based on the TANGO open source software for the very low level control system, Python language for the development of drivers and the French commercial PANORAMA© software as the main high level SCADA.

This choice leads to guaranty the evolution of the middleware software architecture of this facility supposed to be operated during dozen of years with the capability of using many instruments including sustainability.

Maintenance and qualification tools will also be described. The use of Open Source Software like TANGO, Python and QT will allow the capability for diverse contractors to insure all future developments.

INTRODUCTION

Since it definitively abandoned nuclear testing, France has relied on the Simulation Program to guarantee the operational performance and safety of its nuclear deterrent weapons throughout their lifetime.

Successful simulation requires both:

- Qualified computer codes that integrate laboratory-validated physics models to simulate weapon functioning;
- Teams of qualified physicists to use these codes.

In this respect, the Megajoule Laser (LMJ [1]) plays a vital role, as it is used to validate the numerical codes and certify the skills of French physicists.

On October 23, 2014, French Prime Minister Manuel Valls declared the facility operational after starting up the first experiment.

Target diagnostics are a key for numerous physical data acquisition. CEA will develop dozen of these equipments during next twenty years. Each target diagnostic will be dedicated to one or several kind of measurements like X-ray, visible, UV or particles like neutron…

During the life cycle of the LMJ Facility, CEA needs to implement new kind of equipments compliant with a stabilized control command system.

This paper describes the command control architecture used for target diagnostics and the reasons why we insure sustainability for such a huge modular installation.

We will describe the 2 layers of the Target Diagnostics Control System (TDCS) and particularly the use of TANGO for Layer 0, which guaranty modularity and life time expectancy, and the French SCADA PANORAMA E² [2] for Layer 1 dedicated for every LMJ command control subsystem.

USING A TARGET DIAGNOSTIC

In 2014-2015, three different target diagnostics have been installed: two X-ray imaging systems (different ranges of energy and waveform) and a complex diagnostic used for Hohlraum temperature measurements including an absolutely calibrated broadband X-ray Spectrometer, a Gating Spectrometer and a time resolved Imaging System of the emitting area.

This paper will focus on a « simple » X-ray imaging system. This diagnostic can actually be compared to a giant microscope. Made of 4 parts (alignment beams, a telescopic motorized arm, filters and a framing camera), the command control configures the equipment, then focus it using alignment and the arm, in order to acquire data from an optical camera (Fig. 1).

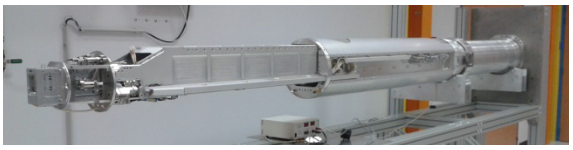
Figure 1: an X-Ray Target Diagnostic.

Each part of the diagnostic has to be driven by the command control and, as these functions should be reused in future target diagnostics, a modular command control architecture must be designed.

THE LMJ COMMAND CONTROL ARCHITECTURE

The LMJ command control architecture is driven by the 4 classical component layers, as shown in Figure 2 :

- Layers 2 and 3 are devoted to the common control system (administration, main supervisory, prediction and tuning system [4] [5], sequences [6]…)
- Layer 1 is set for the main subsystems command controls (target and laser diagnostics, synchronization [7]…) and interfaces between them.
- Layer 0 is the main layer for equipment communications. It includes drivers, communication protocols as well as maintenance and qualification tools.

There is one Layer 1 for the all TDCS and as many as necessary drivers included in Layer 0 for each defined

ISBN 978-3-95450-148-9

target diagnostic. PANORAMA E² is the Framework used for Layers 1 to 3 (Fig. 2).

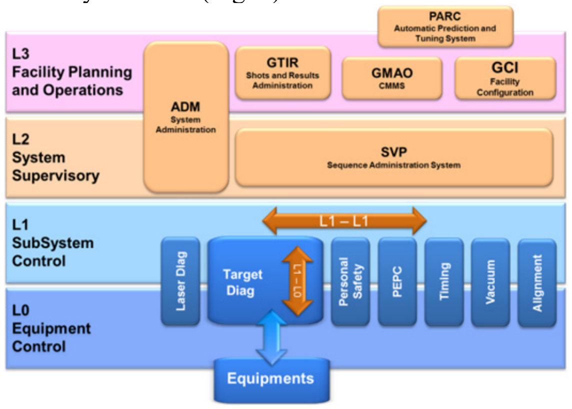

Figure 2: The LMJ Command Control Architecture.

Within that configuration it seems necessary to use a modular Layer 0 architecture and established PANORAMA Layer 1 software that accepts futur Target Diagnostics as new plugin elements.

LAYER 0

A way to make Layer 0 modular is to develop hardware to software abstraction layer between each equipment and the command control. This kind of architecture has been used by Microsoft with DirectX© or by National Instrument MAX© system. Using an abstraction layer allows equipment sustainability: Layer 1 software does not have to be modified even if the low level equipment changes. Only the equipment driver has to be updated but the low level interface will remain the same (Fig. 3).

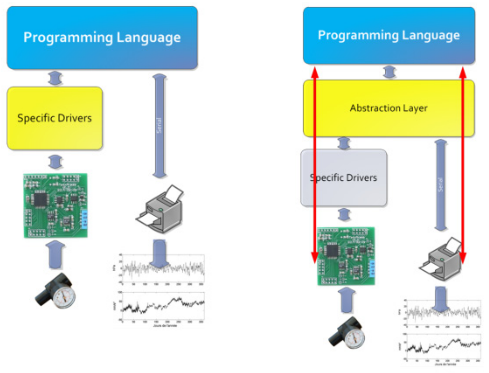

Figure 3: An Abstraction Layer Architecture.

TANGO as Layer 0

TANGO is a software architecture that have the following characteristics:

- Uses an abstraction layer,
- Has been used from several years in huge installations (mostly European synchrotrons),
- Supports different operating systems (Linux and Windows),
- Allows modularity and, by the way, includes a lot of instrument drivers,
- Can be programmed in different languages (C++, Java, Python),
- Part of a large community,
- Available as an open source architecture [3].

Next figure shows how Tango architecture matches our needs for modularity (Fig. 4).

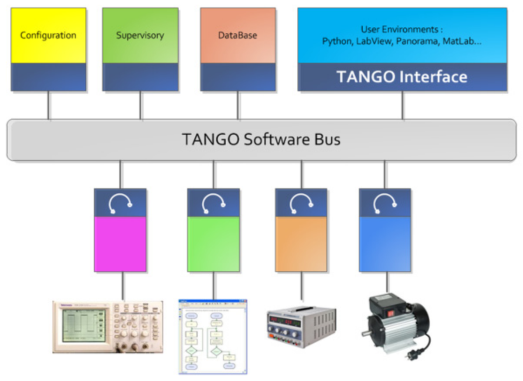

Figure 4: The Tango Architecture.

TANGO framework comes with several tools that simplify low level development phases.

In TANGO, a driver is usually called a Device Server (DS). The DS communicate with each associated equipment and is able to deliver information to the TANGO software bus in different ways (push, pull, event…).

A DS can also be a single process software whose methods are shared between other DS (barycenter and matrix computations for instance).

Each DS is made of:

- Methods (a typical list of functions that will be available for Layer 1 or for other DS)
- Properties (items configurations like motor characteristics, IP addresses…)
- Attributes (values changed by the equipment like arm positions, motor currents, specific acquired datas…)

One main TANGO development tool is Pogo, used to generate both DS framework and DS documentation with a choice of 3 different languages for writing the drivers.

The whole driver skeleton and interfaces are generated; the developer « only » have to fill up the communication protocol between hardware equipment and the corresponding methods.

14 specific « States », are used for states machines in Pogo. They are used to define the real condition of the equipment (for instance « MOVING » for an arm, « OPEN » and « CLOSED » for an obturator…). These states are also used in the Layer 1 control software or by other DS (Fig. 5).

ISBN 978-3-95450-148-9

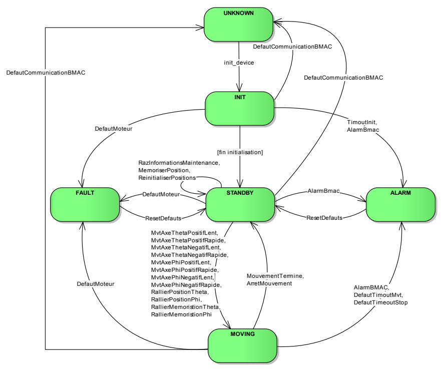

Figure 5: State machine example.

The simplest way to get information from an equipment is to :

- Use the specific method :
  - The method speaks to the equipment,
  - The equipment answers and the method fills up one or several attributes,
- Layer 1 or another DS client reads the attribute value.

This works. But, actually, when more than one client try to access the equipment before the end of each request and if the equipment (or the DS) is not compliant with some design rules, a deadlock can occurs and timeout messages are sent by to the clients (Fig. 6).

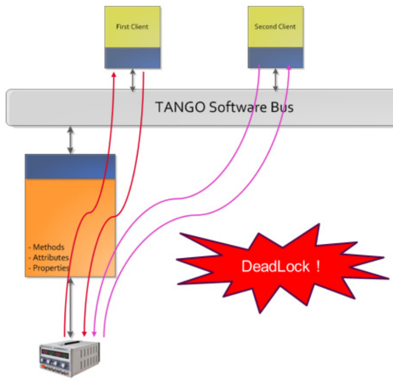

Figure 6: Deadlock.

One way to avoid this situation is to configure the TANGO polling mode.

This specific configuration (which also can be used for computer hardware architecture evaluation) can be applied to each single method.

When activating the polling mode, the DS is, by the way, in charge of calling the method at a configured period and filling up the attributes in a buffer.

Depending on the configuration, when the attribute changes or exceeds a specific range, an event is sent to the clients. Each client gets directly the information from the buffer and not using the method.

If a client tries to call a DS'method, the method is not used, but the reading directly comes from the attribute's buffer (Fig. 7).

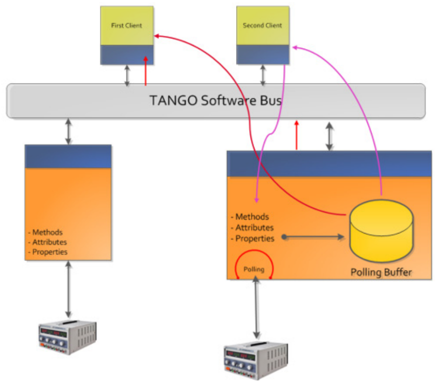

Figure 7: Polling activation.

Using a Modular Architecture for Optical Cameras and Siemens PLCs

Usually, complex instruments (optical cameras, motor controllers with slots) are made of sub-equipments.

The optical camera used in our first target diagnostics uses 5 sub-equipments whose drivers can also be shared with future cameras.

For framing cameras and streak cameras, we developed the following drivers:

- Andor CCD
- Agilent power supply
- Kentech high voltage pulse generator
- One specific electronic board for each camera

No doubts that there will be no needs for Layer 1 to access some very low level methods of sub-equipment. To make this point easier for level 1 developer, we defined the following naming convention[1] :

- Each instance of sub-equipment DS will be called BN_NameOfDriver_xxx
- Each instance of the Layer 1 connected DS will be called *HN_NameOfDriver_yyy*

For framing cameras, BN driver instances will be :

- BN_CCD_001
- BN_Agilent_001
- BN_GXD_001
- BN_ElecCIIX_001

The high level DS connected to Layer 1 will be :

- HN_CIIX_001

Actually, the *HN* driver does the main job as the *BN* driver does the more complex and dirty one!

The following Figure shows the interface organization for framing cameras (Fig. 8).

[1] Where *BN* stands for "Bas Niveau" i.e. "Low Level" and *HN* stands for "Haut Niveau" i.e. "High Level" ; *xxx* and *yyy* are numbers

ISBN 978-3-95450-148-9

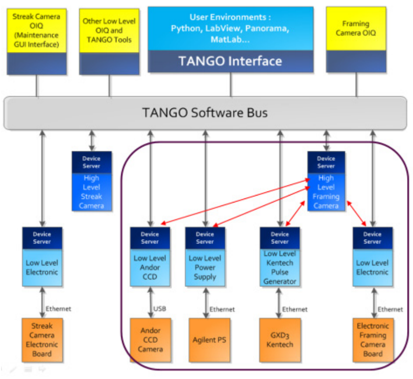
Streak Camera OIQ (Maintenance GUI Interface)
Other Low Level OIQ and TANGO Tools
User Environments : Python, LabView, Panorama, MatLab...
TANGO Interface
Framing Camera OIQ
TANGO Software Bus
Device Server High Level Streak Camera
Device Server High Level Framing Camera
Device Server Low Level Electronic
Device Server Low Level Andor CCD
Device Server Low Level Power Supply
Device Server Low Level Kentech Pulse Generator
Device Server Low Level Electronic
Ethernet
USB
Ethernet
Ethernet
Ethernet
Streak Camera Electronic Board
Andor CCD Camera
Agilent PS
GXD3 Kentech
Electronic Framing Camera Board

Figure 8: High level and Low level drivers.

Managing PLCs is quite a bit more complicated.

S7-300 Siemens PLC's are used to manage pumps, valves and gauges. These systems are necessary for the hardware security equations implementations which are very closed to the sensors.

Nevertheless, as Layer 1 and Layer 2 need to access to individual components state (in order to forbid non wanted actions), *HN* and *BN* conventions have been used to write the driver in the following way:

- Low level driver for PLC access (controller, manager and look up tables)
- High level access for individual sensors components (valves, gauges, pumps…)

This configuration, even if it allows full access to individual components, warranties that by the use of PLC equations; methods can or cannot be applied. Level 1 should be able to open a valve, but, if pressure equations written in the PLC do not complain, the *HN_Valve* DS method will reject the command and will reply a verbose error (Fig. 9).

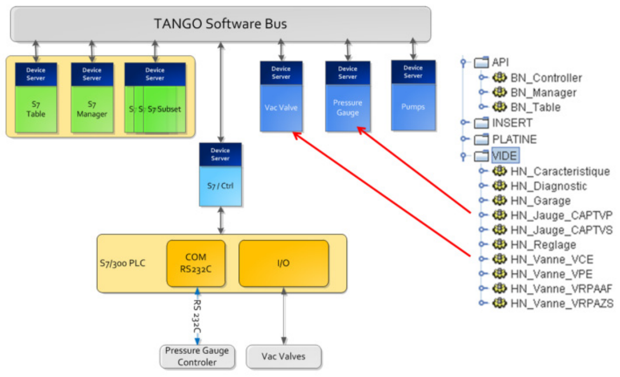

Figure 9: PLC driver configuration.

Maintenance and Qualification GUI Tools

In order to test, control and maintain Layer 0, we will be using Jive (a TANGO native tool for DS implementation usage) and some specific GUI developed in QT/Taurus TANGO framework. Main screen gives access to the Layer 0 TDCS and configuration menus driven by *HN* DS. These tools are used for both qualification and maintenance in locations where Layer 1 is not available.

A very low level interface let the maintenance operator analyze specific configurations like motors interfaces or optical cameras. In LMJ, these actions are granted by Layer 1 (Fig. 10).

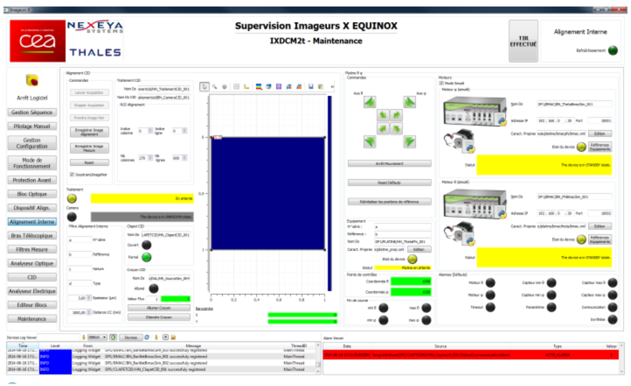

Figure 10: Maintenance GUI Tool.

Real and Virtual Modes

Every command control software is test in our ICSS integration platform [8]. Each subsystem uses specific software simulators that reproduce the equipment responses to methods.

For target diagnostics, we developed two DS *BN* modes by using the same Pogo architecture:

- *Real mode* is directly connected to the equipment,
- *Virtual mode* gives responses from specific files.

Two properties for the *HN* DS are used :

- A Boolean that tags mode type,
- A property that gives the name of the *BN* driver mode.

In both cases, the interface between *HN* Layer 0 and Layer 1 remains the same.

This configuration gives also the developer, the ability to test a full TDCS without the need of all real equipments. For each virtual mode, XML files deliver data and for some specific equipments, like motor controllers, an external software can generate defaults states. Actually, each *BN* DS exists in both real and virtual mode but only the real mode is loaded inside the facility command control.

For PLCs, the virtual mode is located inside the controller. This software emulates PLC's cards at the low level interface. There is no change for *HN* DS but the configuration.

LAYER 1

PANORAMA for Layer 1

Layer 1 for TDCS is developed using the French SCADA PANORAMA E² [2]. This LMJ requirement is necessary to insure all subsystems interfaces and protocol access to Layers 2 and 3.

PANORAMA comes with a generic graphical and VbScript editor. A generic framework and libraries

ISBN 978-3-95450-148-9

written for LMJ are delivered for developments but, as PANORAMA is mostly appropriate for automation, there was a need to interface this high level SCADA to a low level instrumentation architecture like TANGO.

Codra developed this new TANGO to PANORAMA interface (*binding*) for CEA. This interface creates a bi-directional link between each TANGO DS object and PANORAMA objects. In that way, a PANORAMA software developer does not have to know anything on the low level TANGO architecture (Fig. 11).

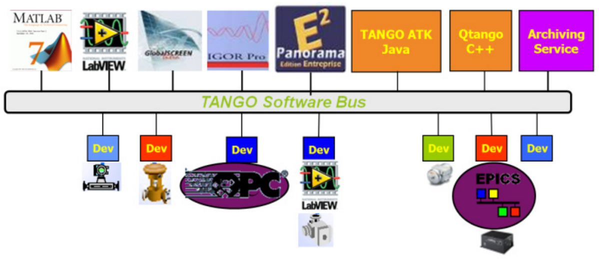

Figure 11: PANORAMA Binding.

A specific editor manages target diagnostics TANGO DS libraries and scripts written in VbScript to respond to Layer 2 sequences orders.

This binding is open source and available for free at www.TANGO-controls.org [3].

By using TANGO for layer 0, PANORAMA for layer 1 and this new binding between these 2 layers, CEA has now the 3 main tools for developing LMJ TDCS GUI.

A generic GUI is devoted to exchange with top layers. In that way, future target diagnostic Layer 1 GUIs will be like new plugins for the main interface (Fig. 12).

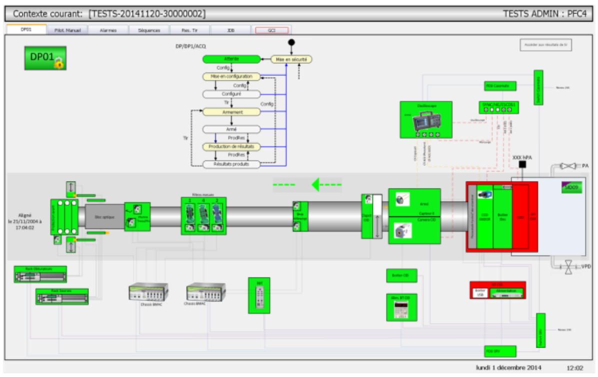

Figure 12: XRay Layer 1 interface.

MANAGING CONTRACTORS

Each part of a target diagnostic has been developed by an external contractor upon CEA's requirements. In order reduce interface problems, we chose to give full responsibility for the hardware contractor to take in charge DS developments, based on the full knowledge of his materials.

If this worked in some cases, it was not sufficient in others, due to the misunderstanding of TANGO architecture (which was clearly new challenge for some industrials) and the knowledge of interfaces between sub systems of a target diagnostic.

We wrote a "TANGO design rules guide" for new developers which is, by now, used in the community but was quite fresh during the beginning of our project.

Managing contractors was mostly a race between requirements, interfaces and time scheduling.

The best advice should be to keep a developing team that works on a low level DS skeleton, quite functional, and make it fully industrialized once tests are sufficient. Externalizing process should only be efficient at this very moment.

SUMMARY

This paper describes the TDCS architecture based on TANGO for the low level equipment driving, PANORAMA for the high level SCADA and the new TANGO to PANORAMA binding that makes the bidirectional glue between both layers.

The hardware to software abstraction layer insures the installation to keep software interfaces while equipment upgrade. TANGO makes it.

Layer 1 architecture uses a common interface which allows future target diagnostics software to be plugged in.

All these specifications have been chosen to guaranty modularity and life time expectancy for :

- adding new target diagnostics
- managing heterogeneity
- using new kind of equipment for target diagnostics
- helping upgrade computers and equipments

Reference [9] gives a full explanation on the LMJ facility control system status report.

REFERENCES

[1] www-lmj.cea.fr
[2] uk.codra.net/panorama
[3] www.tango-controls.org/download/binding
[4] MOC3O06 : PARC : How to computerise Laser Setting on the Megajoule Facility, ICALEPCS 2015, by S. Vermersch, CEA/CESTA, Le Barp, 33114 France.
[5] MOPGF075 : PARC : An Automated Laser Setup System for the Laser Megajoule Facility, ICALEPCS 2015, by J.P. Airiau, CEA/CESTA, Le Barp, 33114 France.
[6] TUB3O04 : The LMJ System Sequences Adaptability, ICALEPCS 2015, by Y. Tranquille-Marques CEA/CESTA, Le Barp, 33114 France.
[7] THCOCA05 : Laser Megajoule Timing System, ICALEPCS 2013, by P. Raybaut, V. Drouet, J.J. Dupas, J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91680 France.
[8] MOCOBAB03 : The Laser Megajoule ICCS Integration Plateforme, ICALEPCS 2013, by J.P. Arnoul, J. Fleury, A. Mugnier, CEA/CESTA, Le Barp, 33114 France.
[9] FRA3O02, The Laser Megajoule Facility Control System Status Report, ICALEPCS 2015, by J. Nicoloso, CEA/DIF, Bruyères le Châtel, 91680 France.

ISBN 978-3-95450-148-9