UNIFYING ALL TANGO CONTROL SERVICES IN A CUSTOMIZABLE GRAPHICAL USER INTERFACE

S. Rubio-Manrique, G. Cuní, D. Fernández-Carreiras,
C. Pascual-Izarra, D.Roldán, ALBA Synchrotron, Cerdanyola del Vallés, Spain
E. Al-Dmour, Max-IV, Lund, Sweden

Abstract

TANGO is a distributed Control System with an active community of developers. The community features multiple services like Archiving or Alarms with an heterogeneous mix of technologies and look-and-feels that must be integrated in the final user workflow. The Viewer and Commander Control Application (VACCA) was developed on top of Taurus to provide TANGO with the user experience of a commercial SCADA, keeping the advantages of open source. The Taurus GUI application enables scientists to design their own live applications using drag-and-drop from the widget catalog. The VACCA User Interface provides a template mechanism for synoptic-driven applications and extends the widget catalog to interact with all the components of the control system (Alarms, Archiving, Databases, Hosts Administration). The elements of VACCA are described in this paper, as well as its mechanisms to encapsulate all services in a GUI for an specific subsystem (e.g. Vacuum).

INTRODUCTION

Tango and Taurus

ALBA[1] Synchrotron is a third generation Synchrotron lightsource in Barcelona, Europe, providing synchrotron light since 2012 to users in its 7 beamlines, with 2 more under construction. ALBA institute has been an active member of the Tango Collaboration[2][3] since the very beginning of its design and construction phase. Tango is an open source object-oriented control system, done in collaboration between ESRF and a growing community of institutes and companies developing new device servers and tools using either C++, Python or Java.

Our Human Machine Interfaces to the Tango Control System are developed using Taurus[4][5], a framework for creating both GUIs and command-line tools to interact with scientific[6] and industrial control systems and related data sources. Originally developed in-house at ALBA, Taurus opened its development to the members of the Tango community, becoming popular among the new members of the Tango Collaboration. Some of the causes of this success are the technologies involved: Qt, Python[5] and its integration with SciPy[6], the popular stack of scientific libraries for python.

Tango as SCADA

The concept of a Supervisory Control and Data Acquisition system (SCADA) is largely used in both industrial and scientific worlds to define a control system that remotely controls large installations, using multiple communication channels and providing an application to control large processes distributed on many remote equipments or stations

Tango fits perfectly within SCADA definition, providing the communication channels and software tools needed to manage large particle accelerators and other facilities, along with additional control services such as Archiving, Alarms and User Access. These services are managed by a collection of applications developed either by the core team or by other members of the community: Jive for database configuration, Astor for control host management, JDraw for Synoptics, Mambo for Archiving, ATK/Taurus/QTango for GUI development, Sardana for experiment control as well as several web toolkits and multiple alternatives for Alarm systems[7] like PANIC[8].

Those applications provide a rich functionality and a full-featured control system, but also a diverse collection of look & feels and workflows that may be inconsistent and interfere with user interaction (Fig.1). This paper will address this issue providing a Taurus-based solution.

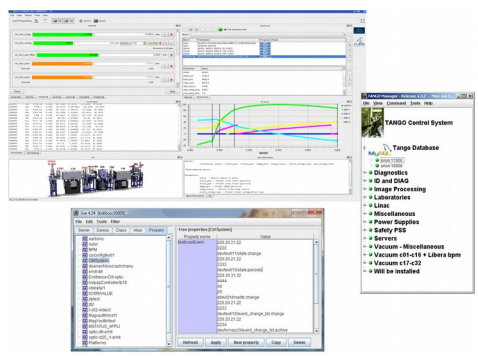

Figure 1: Tango look & feels: Taurus, Jive, Astor

State of the Art, Taurus and CSS

GUI consistency issues are not unique to Tango or to open source control systems, as they may apply to any collaborative project on which each member needs to tune the generic tools to its own context.

At ALBA, these problems were mostly solved once Taurus[5] became the default GUI framework for all our applications, thus creating the opportunity to unify the user interaction with the whole control system. Taurus does not enforce control-system specific conventions but is open instead to include multiple schemas and allows

ISBN 978-3-95450-148-9

the final user to configure what must be shown and how for all the catalog widgets: attribute and command forms, synoptics, data source trees, plots and trends, image viewers and the drag-and-drop application builder, the TaurusGUI.

But, although Taurus provides a consistent UI framework for all applications developed at ALBA, it still leaves uncovered those control services that are either completely Tango specific, not developed by ALBA or do not fit well in the current model URI's of Taurus.

The same problem has been approached before by the EPICS community, developing a common UI framework to manage all the aspects of the control system. The EPICS-based Control System Studio (CSS)[9][10], provides a consistent GUI that integrates Forms (BOY), Alarms (BEAST), Archiving (DataBrowser), Synoptics and other community tools to deliver a single control application that deals with all control system services.

Both TAURUS and CSS Studio are powerful GUI toolkits with strong communities and full-featured SCADA behind, Tango and Epics respectively. Being both leading projects on its field, this paper presents the implementation of a Taurus-based application as consistent as CSS but with a higher versatility.

IMPLEMENTATION OF VACCA

VACCA was originally developed by the ALBA Control Section (ACS) as the Vacuum Control Application for Accelerators. It was designed as a synoptic based application capable of summarizing the state of hundreds of devices and provide navigation tools to locate and plot any increase in pressure or temperature readings from the vacuum chambers (typical usage and target of commercial SCADA applications).

The development of VACCA has been an iterative process:

- SynopticTree (2007) Java/ATK Based tool based on JDraw and DeviceTree applications (ESRF) and Mambo Archiving browser (Soleil). Used during the commissioning of ALBA linac and transfer lines.
- VACCA/PySynopticTree (2009): First Python release, using the TAU library and integrating Astor/Jive functionality in a single tool as an Alarm toolbar. It did heavy use of composer devices [11] to deal with hundreds of devices.
- VACCA3.0 (2012): First Taurus release, developed in collaboration with the ESRF. Used only for beamlines and backwards incompatible with TAU. Lacking the Astor/Jive/Alarms functionality but providing the versatility of Taurus GUI.
- VACC4 (2015): First release to be deployed on both small and large control systems. All previous features are provided and all widgets become interconnected with all Tango services.

VACCA and Taurus GUI

The current implementation of VACCA (fig. 2) is based on the TaurusGUI framework[5]. This framework goes beyond WYSIWYG and embeds the application design within the application itself, empowering the user to create new panels on running GUI just using drag & drop from the Taurus catalog widget. Initial widgets may be setup in a python module, while additional panels are added on runtime and stored when saving the current layouts as Perspectives: user specific views of the application that record both look & feel and configuration.

The VACCA core is just a TaurusGUI configuration module, that builds a default Perspective with pre-loaded widgets for Synoptic, Archiving Trends, Searches, Properties and a Device Panel. VACCA supports widgets that can be part of the Taurus Catalog, loaded from Vacca or other PyTango modules (like PANIC[8]) or just added by the developer on his custom setup file.

Figure 2: VACCA default perspective

The files that store VACCA customization at facility / system / user levels are:

- vacca.config : the core of vacca python module, it loads and interconnects the most basic widgets.
- vacca.default : overrides certain specific variables of the configuration (e.g. institute logo, devices to be hidden in searches, add/disable alarms widgets) and expands the widget catalog with custom panels.
- [your_config.py] : optional python module passed as command argument, used to do specific setups of synoptics, panel models and filters for tree and attributes widgets (e.g., show only BL vacuum).
- [~/.config/VACCA/.ini] : where user Perspectives, Panels and Plot settings will be saved.

To a Consistent User Interface Catalog

As stated in the introduction, mixing tools coming from different institutes and developer contexts results in common inconsistencies like:

- Displaying different names for same information, e.g. showing or not alias or labels for Tango devices.
- Tools forcing to use the conventions of an specific institute (lower/upper case, spaces/dots in names).

ISBN 978-3-95450-148-9

- Not providing always the same behavior for user actions (draggable text, context menus, keyboard shortcuts, usage of user/expert levels).
- Too many solutions to the same problem (e.g., at least 5 different syntax exist to evaluate string formulas in Tango).

As Taurus is currently used for all user Interfaces at ALBA , it solved all these issues except for Tango control services: the Tango Database, Alarms, Archiving, Snapshoting and Starter services.

Those services were originally managed at ALBA using their own tools (Jive, Panic, Mambo, Bensikin, Astor), but gradually replaced by python tools and widgets during the commissioning of ALBA. This was possible thanks to the device server nature of all Tango services, that physically splits each service on client and server side and allows to replace each element independently.

The next step was to embed these service specific widgets in a generic Taurus application, a work done by the VACCA python module. Although these widgets already existed in Taurus (TaurusPropTable) or other ALBA packages (PANIC.AlarmGUI) they have been subclassed to interact between them in a consistent application (Table 1).

Table 1: VACCA Widget Catalog

Widget	Parent	Features
VaccaTree	TaurusDevTree	Provides embedded search bar, device info, Start/Stop of devices, attribute dragging and device selection
VaccaPropTable	TaurusPropTable	Provides device property edition and drag & drop
VaccaSynoptic	taurus.qt.graphic	Provides device selection, graphic element highlighting, context menus, SVG synoptics
VaccaTrend	TaurusTrend	Archiving enabled by default, draggable legends
VaccaPanel	TaurusDevicePanel	Drag & drop of device labels, custom icons for each device class
VaccaBrowser	ArchivingBrowser	Drag & drop to device panel and trends
AlarmGUI	panic.gui	New signals for highlighting synoptics, accept drag & drop on search bar an editor

Other advantages of creating subclasses are:

- To provide a fix interface to each class, so existing Perspectives can handle updates in the upper libraries.
- To keep backwards compatibility with Taurus, PANIC and other dependencies.
- To avoid polluting the upper libraries with ad-hoc features only used within VACCA.
- To enable system-wide configuration, using Class Properties in the Tango Database to customize the appearance of every widget for a given Device Class.

Widget Interaction

All the default widgets loaded at VACCA startup exploit an essential feature of TaurusGUI: the Shared Data Manager (SDM). This mechanism allows the creation of Reader / Writer channels within the application to send information between widgets, enabling translation elements to be inserted in the dialog chain. This feature enables any PyQt widget to be used as device / attribute selector as long as it provides a Qt signal or slot to connect with.

Every widget instantiated from the VACCA catalog (Table 1) is aware of the sources of information available on its own application SDM, subscribing to the available channels and sending its own signals to the crowd. Widgets like Synoptic, Tree or Alarm GUI become enabled as model selectors. VACCA widgets are not only aware of signals at GUI creation time, but also when later added from the catalog into a running application, becoming instantly connected and interactive with the rest of widgets (see Fig.3).

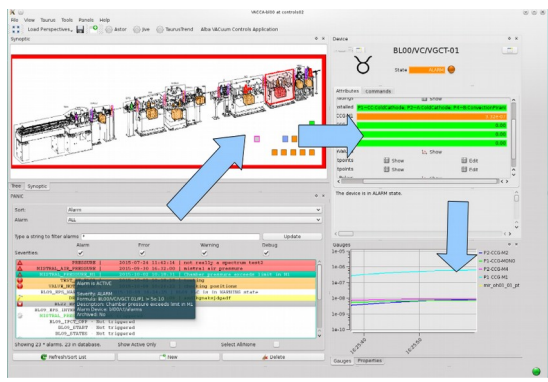

Figure 3: example of widget interactivity, selection of an active alarm will highlight in red the related devices in the Synoptic, where they can be selected for the Device panel or dragged into the Taurus trend.

Drag & Drop

Interaction between widgets that have multiple models have been implemented using drag & drop, a mechanism that is inherent to Taurus and used as the standard method to populate trends and attribute forms. VACCA extends the same approach to any text that matches Taurus models, enabling dragging from/to specific editors like the Properties or Alarms widgets.

When already existing, the drag & drop methods have been extended to use all available Mime types in Taurus: Device, Attribute, Model and PlainText for alias/label. It is used to reflect the multiple ways in which information may be referred-to in taurus, allowing the widgets to translate between the internal control naming

ISBN 978-3-95450-148-9

(Device/Attribute) and the user-defined naming (Alias/Label) in a way completely transparent to the user.

To provide a more generic solution when the full name of a Device is not known, the Archiving Browser have been added to VACCA to act as a universal search tool for Taurus. Developed to correlate user naming between databases it is both capable of searching on new data, archived data or even devices already deleted. Thanks to drag & drop capabilities it's possible to just drag the known alias into the browser, expand the matching attribute list and then drag the desired attribute to a trend, the alarm editor or wherever is needed.

VACCA and the Control System Studio

Following the classification of CSS Compatibility Levels wrote down by the CSS Studio team at DESY [12], the Widgets in VACCA are integrated at "Advanced" and "Integrated" levels within the framework. Taurus and VACCA cover all together both the “Common Use” and “Selected Use Components” of CSS, thus becoming a full-featured control system navigator.

Performance of a Full-Control System GUI

Performance issues were faced by a GUI like VACCA at the beginning of its design, as it had to be deployed to manage hundred of devices during installation and commissioning, having to deal with dead times and timeouts, as well as avalanches of events in case of sudden incidents that may saturate the whole application.

These issues have been solved using two different approaches. In the case of timeouts, PyStateComposer devices were used to summarize the information on relevant attributes for each accelerator sector (typically 30 to 50 devices). This cached summaries (updated every 3 seconds) allowed to reduce the number of proxies to be opened by the application, and allowed to be resilient against hardware timeout by adding a layer in between and a hook in SDM to translate composed attributes to its real devices.

In case of high event rates coming from the Tango Control System, the solution applied was the event filtering mechanism of Taurus. Event filters can be introduced both at Attribute and Widget level, and executed on either python or PyQt threads.

Both setups, composers and event filtering, allow to deal with most common performance issues. In both cases, however, performance improvements were only needed when managing very large systems and were done just at vacca.default level, with no need to modify Taurus library sources.

CONCLUSION

Open Source control systems may compete in the Scientific world at the same level than renowned commercial SCADAS; leveling or exceeding the performance and features of those private systems.

But, in comparison, the integration of the different subsystems sometimes lack of a unified look & feel due to the multiple teams working on different institutions. This situation is changing with the commitment of new communities of developers that are working on improving both user workflow and performance. VACCA, as a Taurus tool developed originally to manage only vacuum and protection systems, has evolved to get his own role as control system navigator and central hub to interact with all Tango services.

Future integration of additional control systems and features in VACCA will be based in Taurus Schemas and Plugins mechanisms still under development, trying to keep backwards compatibility with existing Perspectives as much as possible.

ACKNOWLEDGEMENT

Most of the early development was done in collaboration with former ALBA members Ramón Suñé and Tiago Coutinho. I must also thank Carlos Falcón, Daniel Roldán (ALBA) and Antonio Milán (MaxIV, Lund) for their work on testing and deploying VACCA on different platforms and institutes. I would like also to thank Andy Götz (ESRF) for its support to the project and Eshraq Al-Dmour (MaxIV) for its role in the evolution of the first releases of VACCA at ALBA.

REFERENCES

[1] ALBA website: http://www.albasynchrotron.es
[2] A.Götz, E.Taurel et al, “TANGO V8 – Another Turbo Charged Major Release”, ICALEPCS'13, San Francisco, USA (2013)
[3] Tango website: http://www.tango-controls.org
[4] Taurus website: http://www.taurus-scada.org
[5] C. Pascual-Izarra et al., “Effortless Creation of Control & Data Acquisition Graphical User Interfaces with Taurus”, ICALEPCS'15, Melbourne, Australia (2015).
[4] Z. Reszela et al. “Sardana – A Python Based Software Package for Building Scientific Scada Applications”, PcaPAC'14, Karlsruhe, Germany (2014)
[5] D. Fernandez-Carreiras et al., “ALBA, a Tango Based Control System in Python”, ICALEPCS'09, Kobe, Japan (2009)
[6] SciPy website, http://www.scipy.org
[7] S.Rubio et al., “Extending Alarm Handling in Tango”, ICALEPCS'11, Grenoble, France (2011)
[8] S. Rubio-Manrique et al., “PANIC, a suite for visualization, logging and notification of incidents”, PCaPAC'14, Karlsruhe, Germany (2014)
[9] J.Hatje et al., "Control System Studio (CSS)", ICALEPCS'07, Knoxville, USA, (2007)
[10] CSS website: http://controlsystemstudio.org/
[11] S.Rubio et al., “Dynamic Attributes and other functional flexibilities of PyTango”, ICALEPCS'09, Kobe, Japan (2009)
[12] CSS at Desy: http://css.desy.de/content/e760/e761

ISBN 978-3-95450-148-9