THE TANGO COLLABORATION: STATUS AND SOME OF THE LATEST DEVELOPMENTS

E. Taurel[1], D. Fernandez[2], M. Ounsy[3], C. Scafuri[4]
[1]*ESRF, Grenoble, France,* [2]*ALBA, Barcelona, Spain*
[3]*Soleil, Orsay, France,* [4]*Elettra,Trieste, Italy*

INTRODUCTION

Tango is an object oriented, multi language control system framework developed to fulfil the needs of today's user of control systems. These needs are interfacing hardware in a reliable and performing manner, hiding its complexity, treating errors and alarms, distributing controls over the network, providing easy high level access, archiving long term data, user friendliness, interfacing some well-known commercial software and providing some kind of Web access. Performing all these tasks is a huge amount of work. With limited man power and amount of money, a solution to have these tasks done in a relatively short delay is to create (or enter) a collaboration between institutes. The development of the Tango control system is an example of this kind of collaboration including 4 institutes which are:

ALBA the new Spanish synchrotron actually under construction in Barcelona

ELETTRA the Italian synchrotron located in Trieste

ESRF (European Synchrotron radiation facility) the European synchrotron located in Grenoble (France)

SOLEIL the French synchrotron under construction in Paris

In this paper, we will first gives a general overview of what is Tango with its main features, then, we will detail some of the new developments done within the Tango community and finally, we will explain the day to day life within our collaboration.

WHAT IS TANGO?

Tango uses CORBA for doing network communication. CORBA is a language independent standard for implementing distributed objects on the network. Tango uses the omniORB [1] implementation of CORBA in C++ and JacORB [2] for Java. OmniORB and JacORB are freely available and open source software.

Tango supports two programming languages which are C++ and Java. It is supported on Linux, Solaris and Windows. Tango is free software and is available as a source distribution package for UNIX like operating system and as a binary distribution for Windows. You can download it from [3].

Philosophy

The philosophy of Tango is to provide users with an object oriented control system which is powerful and easy to use. Tango uses CORBA as network layer and offers the advantages of using CORBA but hiding its details. We have adopted the approach that all TANGO objects are derived from one class called Device. Device is the basic TANGO component. It defines all the network features of equipment controlled with Tango and is defined within a single CORBA network interface. Therefore, from the control system client point of view, there is no difference between controlling sophisticated equipment like a Radio Frequency cavity or a Magnet cooling water flow meter.

Device, Command and Attribute

Within Tango, every controlled equipment is a device. Each kind of equipment is described by one specific implementation of the Tango device pattern that we call a Tango class. Every time this equipment is installed and has to be controlled, a new instance (called a device) of this Tango class is created. A Tango device support command and attributes and has a state. Commands are used to implement "action" on a device while attributes are used to read/write data values. Attributes can be 0, 1 or 2 dimensional data. Each attributes has a set of parameters like description, units, limits, alarms…

Device Server process

A device server is an operating system process with one or several Tango class. Following a predefined *main()* or *winmain()* structure, the device access software programmer merges all the Tango classes s/he want to run within the same process. Device name and number for each Tango class is defined in the database and is retrieved during the process startup sequence. Multiple instances of the same device server process may run within a single TANGO system. Each instance has an instance name specified when the process is started. The couple process executable name/instance name uniquely defines a device server.

Polling features

Within each device server, a "polling" thread is created. This polling thread has two rules which are:

- Regularly read attribute or execute command and store the result in a cache buffer. When a client request is received by the device server, the default behavior is to first check if the requested data (command result or attribute reading) is already stored in this cache buffer. If this is the case, data are fetched from this buffer instead of accessing the real device thus allowing very good response time. This is particularly helpful for slow devices accessed via serial line.
- Be the source of event for the Tango event system (described more deeply in following chapter)

Client API

Tango clients could be programmed using only the CORBA API. Nevertheless, CORBA knows nothing about Tango and programming at this level means that clients have to know all the details about CORBA programming. The Tango philosophy is to hide these recipes in an API. The API is implemented as a set of C++ and Java classes. This API implements automatic reconnection between clients and server in case of server restart or front-end computer reboot. It also hides which release of the Tango network Interface Definition Language (IDL) Tango device implements.

Client/Server communication

For the most important API remote calls, and on top of the event system, Tango supports two kinds of requests which are the synchronous model and the asynchronous model. Synchronous model means that the client wait (and is blocked) for the server to send an answer. Asynchronous model means that the client does not wait for the server to send an answer. The client sends the request and immediately returns allowing the CPU to do anything else (like updating a graphical user interface). The client has the choice to ask to be informed by the server when the request is finished or to periodically check if the request if done.

Tango Tools

TANGO is delivered with a full set of graphical tools for building TANGO device servers, testing them, plotting, logging and archiving. All graphical tools are written in Java using the Swing graphical library. The following graphical tools have been developed so far:

- ***pogo*** - a graphical tool for generating device classes in C++ or Java. Removes a large part of the tedious work involved and makes device class development very rapid.
- ***logviewer*** - a graphical tool for viewing and filtering log messages.
- ***jive*** - a graphical tool for viewing and modifying the database and testing devices.
- ***devicetree*** - a generic graphical tool for testing devices and building simple monitor panels.
- ***astor*** - a graphical tool for supervising a TANGO control system e.g. starting device servers, checking if device servers are running etc. and testing devices. It uses a specific Tango device server called "Starter"

Tango Services

The following services are offered in Tango :

- ***database*** - a system wide database using MySQL is provided for persistent storage of device properties and device names

- ***naming*** - a naming service is provided via the database which allows devices to be found independent of which host they are running on
- ***event*** - TANGO uses the omniNotify implementation of the CORBA Notification service. This service is described more deeply in the following chapter
- ***logging*** - a logging service which uses log4j is provided for all devices.
- ***archiving*** - archiving of historical data in a database is provided for off-line analysis and retrieval
- ***groups*** - groups of devices can be constructed on the fly and controlled as a single device.

Tango Graphical User Interface

Tango ATK (Application Toolkit) is a collection of Java classes to help building applications based on Java/Swing which interact with Tango devices. It is developed using the design pattern called Model-View-Controller (MVC) also used by Swing. Tango ATK provides a set of graphical objects called "Viewers" adapted to each type of Tango attributes or commands and non-graphical objects. The rule of these non-graphical objects is the interface with the Tango device. The communication between the non-graphic and graphic objects is done by registering graphic objects as listener of the non-graphic object. The non-graphics objects emit Tango ATK events which are sent to all graphical objects registered as listeners. Tango ATK helps minimizing graphical application development time, avoid code duplication and provide a common look and fell between applications. A talk (and a paper) describing more deeply Tango ATK will be given in this conference.

Nevertheless, Java is not the choice of everyone. Therefore, another graphical layer based on C++ using the Qt library is now under development by the Elettra control group. Nevertheless, this will also used the philosophy adapted by Tango ATK.

Bindings

Physicists as well as engineers are using more and more commercial tools like LabView, Matlab or Igor to help them in their day to day work. A modern control system has to give them a way to talk to their equipment from these tools. Tango supports bindings for Matlab, LabView and Igor products. This means that it is possible to access Tango devices from a Matlab macro or from a LabView "virtual instrument". A binding also exists for the Python language but only on the client side allowing a python script to access any kind of tango device.

LATEST DEVELOPMENTS

The Event System

An event system has been recently added to Tango kernel features. Events are a critical part of any distributed control system. Their aim is to provide a communication mechanism which is fast and efficient. The standard CORBA communication paradigm is a synchronous or asynchronous two-way call. If the client has a permanent interest in a value he is obliged to poll the server for an update in a value every time. This is not efficient in terms of network bandwidth nor in terms of client programming. For clients who are permanently interested in values the event-driven communication paradigm is a more efficient and natural way of programming. In this paradigm the client registers her interest once in an event (value). After that the server informs the client every time the event has occurred. This paradigm avoids the client polling, frees it for doing other things, is fast and makes efficient use of the network. Tango events are only available on device attributes. The clients continue receiving events as long as they stay subscribed. Most of the time, one specific device server thread (the polling thread) detects the event and then pushes the device attribute value to all clients. Nevertheless, in some cases, the delay introduced by the polling thread in the event propagation is detrimental. For such cases, some API calls directly push the event. To propagate the event between the device and the registered clients and to add some filtering features, Tango uses a CORBA service called "The CORBA Notification service". We have chosen omniNotify [4] as our implementation of the Notification service. Five types of events are implemented in Tango:

- The change event: An event is triggered and the attribute value is sent when the attribute value changes significantly. The exact meaning of significant is device attribute dependant. For analog

and digital values this is a delta fixed per attribute, for string values this is any non-zero change i.e. if the new attribute value is not equal to the previous attribute value. The delta can either be specified as a relative or absolute change

- The attribute quality factor change event: Each Tango attribute has a quality factor describing the quality of the attribute data. An event is triggered and the attribute value is sent if the attribute quality factor changes e.g. from *valid* to *alarm* or vice versa.
- The periodic event: An event is sent at a fixed periodic interval.
- The archive event: This event is a mix of the change event and the periodic event. It has been implemented to simplify the development of the Tango archiving system.
- The user event: The criteria and configuration of these user events are managed by the device server programmer who uses a specific method of one of the Tango device server kernel class to fire the event

Another Graphical User Interface for Tango Based on Qt

A C++ graphical user interface library is being developed at ELETTRA, based on the Qt widget set. It is composed of 3 modules: Ttk (Tango tool kit), QtControls and QTango. Ttk is a non graphical library which transparently manages event subscribing, polling threads, device proxy creation and caching, and client side error logs. Since it depends only on Tango, it can be also used to write complex Tango applications (e.g. middle layer servers) or to bind Tango to other C++ graphical toolkits (for example wxWidgets). QtControls is a small set of custom Qt widgets used to display (view) and set (control) data in a format suitable for controls. QTango is built on top of Ttk, QtControls and Qt widgets. It automatizes the most common usage patterns used for writing graphical control applications. Specifically associate a device attribute or command with a viewer (scalar or spectrum) or a controller. This library allows the inexperienced user to write graphical control applications with a very basic knowledge of Tango; it is sufficient to know the fully qualified name of the commands and attributes of interest. At the same time QTango provides experienced programmers with full access to the underlying Tango objects instantiated and managed by Ttk.

Tango Web access

The Tango to Web interfaces is composed by two PHP tools: one for visualizing the current data and the second one to display data coming from the Tango archiving database. The requirements for the web server are PHP configured with a few common modules and for the client a browser with cookies and popup enabled. No plugging or jvm are necessary. The user interface is composed by several pages called panels which display the information coming from a particular device (or device aggregation). Each panel is composed of a certain number of "widgets" and some HTML tags; a widget is a graphic element which represent a single attribute (or command or property, yet not implemented). There are also some pages which allow to browse all the available devices, select a whole device or a single attribute, select and customize the widget associated to each attribute The communication to Tango is implemented through a raw text socket. There is a Python script which includes the Python to Tango binding (PyTango) and act both as a Tango client and socket server. The PHP socket client is encapsulated in a buffering system in order to limit the network bandwidth used to get the data out of the control system. There is also a little PHP extension which implements an experimental SOAP web service.

The whole Tango to web project is in its start-up phase and many developments are foreseen in the next months. For further details see [5] and [6]

The Tango Alarm System

The alarm server, named Alarm Collector, is a particular Tango device server, written in the C++ programming language.

A double client/server architecture allows the Alarm Collector to gather the necessary input values from the involved device servers as a client, whereas as a server will then provide alarm notification to any client interested in receiving alarms. It heavily relies on the Tango event system to collect the input values as well as to provide the alarm notifications.

On a per-event basis, the server will then evaluate all the relevant alarm formulas, i.e. those that contain the changed attribute, update the list of active alarms, and send it to the alarm clients. A talk (and a paper) describing more deeply the Tango alarm server will be given in this conference.

THE TANGO COLLABORATION

The history

The development of Tango started in 1999 at the ESRF. In 2002, the new French light source project SOLEIL studied several solutions to implement the control of its accelerator complex plus beam lines and selected Tango. A collaboration agreement has been signed allowing both institutes to share the development workload and take together the best strategic decisions. In January 2004, the machine control team of the Italian light source ELETTRA joined the collaboration and participates actively in its development. In December 2004, the new Spanish synchrotron called ALBA also decided to join the collaboration and to use Tango for their control system.

How it works?

Two or three meetings are organized every year to take decision and to follow-up the action plan. These meeting are hosted by each institute in a round-robin manner. A mailing list is also dedicated to Tango problem, questions and sharing. Nowadays, up to 60 people are members of this list. This is a moderate traffic mailing list (in average one/two mails per day). A tango coordinator has been nominated within each institute member of the collaboration. Its rule is mainly to keep other institute aware of what's happening in its own institute (concerning Tango), to organize the regular meeting and to follow what happen on the mailing list.

Developing software within the collaboration

Several way of developing shared software within the collaboration has already been used. The more common way is to assign a complete sub-system to one institute. The long term archiving system is an example of this kind of development which has been assigned to the Soleil institute. The features requested to the sub-system as well as it main architecture are discussed within our collaboration meeting. Then, people in charge of the development are doing their work in their home institute. It also happens that a sub-system developed in one institute is taken in charge by another institute due to work load issue. An example of such a case is the Tango client Python binding which has been initiated by Soleil and which since its release 2 is now taken in charge by our Italian colleagues. Finally, it is also possible that people in two institutes work at the same time in the same sub-system. This implies that developers in both institutes have a good knowledge of the sub-system in order that their work does not interfere with the work done by the other developers in the other institute. Obviously, in such a case, CVS is a mandatory tool. One of the features of the Tango kernel library (called "database on file") has been developed this way. It was added to the C++ library by Italian colleagues while at the same time work on the same C++ library was done by ESRF developers.

Sharing software

To practically share the software, we are mainly using two tools which are the Web and CVS server through SourceForge facility [7]. Each institute has in their WEB server, some pages dedicated to Tango where you can find useful information like documentation or download code feature. The URLs are given in [3], [8] and [9]. Two different projects have been created on SourceForge to share software using SourceForge CVS servers. The first one is called **Tango-cs** and is mainly dedicated to the Tango core source code (Java and C++ API sources, kernel device server source code, Tools source code…). The second one called **Tango-ds** is dedicated to Tango device servers source code. The rule of this second SourceForge project is to store source code for all device servers used to interface commercial hardware (and then possibly bought by the other institute) and general purpose device server not linked to a specific hardware. Actually, source codes for 45 differents Tango device classes are stored within this SourceForge project. The source code for device server used to interface hardware developed in house is also available as a kind of example in the WEB pages of each

institute. Finally, more than 200 Tango device classes are actually available. In order to know that someone is developing a device server class for a new kind of equipment, as far as this new class is stored in the CVS repository, a mail is sent on the mailing list to inform Tango users that a Tango class for this kind of equipment is under development and will be ready soon.

Using software pattern to improve collaboration between institute

As an object oriented control system, Tango uses object oriented language (C++ or Java). These two languages support the notion of abstract classes in C++ or interface in Java. Using theses features, Tango implement what is called Abstract Device class. Within a Tango abstract device class for one kind of equipment, you define what its basic interface is (in term of Tango command and/or attributes) but you don't define how these commands or attributes are effectively coded. Each concrete class developed for this kind of equipment will inherit from this abstract device class (or interface) and will effectively add the code necessary to implement the already defined command and/or attributes. This gives the assurance to application programmer that any equipment of this kind will support this command and/or this attributes and he can base his software on this assumption. One poster presented in this conference describe more deeply this way of developing Tango device classes.

This abstract device class pattern could be used to extend collaboration between institutes on the application part. One application using only command and/or attributes defined by abstract classes can be shared between institutes whatever the concrete class are.

What we share and what we don't share

From the experience we gained in our collaboration since several months, it is now possible to set up a list of what is effectively shared and what is not shared in our collaboration. The kernel part of Tango is something which is effectively shared by our institutes. Device classes are also shared but obviously only between institutes using the same hardware or with the same kind of requests. Bindings are also something which is easily shared. What is not actually shared is the graphical layer above Tango. Elettra's colleagues prefer using C++ graphical layer like Qt to interface their Tango devices, ESRF uses pure Java for graphical application and Soleil uses a Java based Scada system but re-using some part of the Tango ATK layer. It's actually too early to know what Alba will use as graphical layer. A reason of these divergent choices could be that the market in this area is so large in term of languages and in terms of already existing tools and libraries that it is difficult to reach an agreement between institutes. On top of that, the drawing quality of one system could be judged differently from one country to another due to some cultural habits.

CONCLUSION

To solve all user requests which always increase, it is nowdays necessary to enter a control system collaborative development or to choose one which is already developed. Tango is an example of a collaborative development. Only collaboration has allowed us to give all these functionalities to the kernel and to have a Tango device classes catalogue with more than 200 classes. Even if by some aspects collaboration could be seen as a heavy task (finding a date for collaboration meeting for instance), it is the only way to develop a modern full featured control system within a reasonable delay and with limited manpower and money.

REFERENCES

[1] http://omniorb.sourceforge.net/
[2] http://www.jacorb.org/
[3] http://www.esrf.fr/Infrastructure/Computing/tango
[4] http://omninotify.sourceforge.net
[5] http://www.elettra.trieste.it/~tango/Canone
[6] http://www.elettra.trieste.it/~tango/E-Giga
[7] http://sourceforge.net
[8] http://www-controle.synchrotron-soleil.fr:8001/collaboration
[9] http://www.elettra.trieste.it/~tango/index.html