# TANGO - AN OBJECT ORIENTED CONTROL SYSTEM BASED ON CORBA

J-M.Chaize, A.Götz, W-D.Klotz, J.Meyer, M.Perez, E.Taurel
ESRF, BP220, Grenoble, 38043, FRANCE

*Abstract*

This paper presents TANGO[1] - an object oriented control system based on CORBA[2]. TANGO has been developed at the ESRF. All control points in TANGO are implemented as methods or attributes of CORBA network objects (servants). Control actions are executed by invoking methods on objects. Objects are served by device servers. TANGO device servers can be written in C++ or Java. Device servers can run on Linux, Windows/NT, Solaris, or HP-UX. TANGO is fully compatible with the ONC RPC based TACO [1] control systems. In this paper the TANGO idl definition, device pattern, database , naming service, event service and scripting languages are presented. The present status of TANGO and how it will be deployed in a TACO control system will be reviewed.

## 1 INTRODUCTION

The task of building a control system in today's world has been heavily influenced by the ever increasing choice of Commodity-Off-The-Shelf (COTS) products. Many of the control problems (hardware and software) have been solved and can be bought ready to use off-the-shelf. This has advantages in terms of price, functionality and time-to-be-ready. However the products have to be integrated in order to form a control system. System integration is therefore one of the main tasks of a control system builder today. TANGO has been developed with system integration as one of its main design goals.

In TANGO system integration is achieved by wrapping. Wrapping means inserting a layer of software between the product to be integrated and the system in which it has to be integrated. The wrapper layer runs on the product platform and communicates with the control system via the network. The wrapper software needs to be multi-platform, network based and language independent. TANGO has chosen CORBA as its COTS wrapper software.

## 2 WHAT IS CORBA ?

CORBA is a definition of how to write object request brokers. The definition is managed by the Object Management Group (OMG [4]). Various commercial and non-commercial implementations exist for CORBA for all the mainstream operating systems. Implementations which respect the CORBA 2.0 (and later) standard are inter-operable over the network using the IIOP protocol. CORBA uses a programming language independent definition language (IDL) to define network object interfaces. CORBA defines a number of common services for various commonly needed functions e.g. naming, events, trading. Language mappings are defined from IDL to the main programming languages e.g. C++, Java, C, COBOL, Smalltalk. For an excellent reference on CORBA with C++ refer to [2].

*Which ORB to use ?* At the ESRF we have tested various free and commercial ORB's. The commercial ORB's are very expensive in general and not all of them respect the CORBA norm. A number of free ORB's exist but they do not all offer full CORBA compliancy plus support for C++, Java and multi-threading. After trying out different products we chose Orbacus from OOC [5] as ORB. It is fully CORBA compliant, has C++ and Java support, multi-threading, is free for non-commercial use and comes with full source. In addition it is reliable and has good support.

## 3 TANGO PHILOSOPHY

*Isn't choosing CORBA enough ?* CORBA has been designed as middleware and therefore one could imagine that the choice to base a control system on CORBA is sufficient. Unfortunately not. CORBA is first and foremost a way of defining objects and accessing them i.e. it does not treat the problem of control systems specifically. Secondly it is very rich and offers a large number of possibilities and services. A control system has to limit itself to a subset of these in order to ensure inter-operability. What interfaces and services to use and how to use them is what makes up a local control system's philosophy and flavour. The TANGO philosophy and its justification can be summarised as follows:

1. A single type of network object - all control objects are of the Device type. This means only a single IDL file and only a single type of object to support. All control objects will be derived from Device. This ensures all objects support the same basic interface and functionality. Support for multiple versions of the Device object will be added by deriving new versions of Device e.g. Device_2.
2. Hide CORBA details from programmers - control system programmers need only know about their specific part of the system and not the details of network programming. This is achieved by providing programmers with a Device pattern for implementing new control classes. Clients access network objects via an API

[1] TANGO - TAco Next Generation Objects
[2] CORBA - Common Object Request Broker Architecture

which wraps the CORBA specific knowledge required to build up and maintain a connection to a server.
3. Support for control system communication paradigms - this means providing synchronous, asynchronous and event based communication.
4. Keep it simple and generic - simplicity and generality have been favoured in order to keep TANGO applicable to a wide range of problems and scaleable. Specificity is implemented at the Device level by deriving a new control object and implementing it in a device server e.g. ccd camera or insertion device
5. Use only freely available software - in order to collaborate with external groups expensive commercial ORB's and databases have been avoided.

## 4 IDL FILE

Seeing as there is only one interface to support there is only one IDL file. The IDL file contains the following network interfaces :

- Device - the basic interface of all control objects including the database. Each Device has state. Actions are performed on devices by executing commands which pass one input parameter and return one output parameter (one of the TANGO predefined data types). Commands can be executed synchronously or asynchronously. Asynchronous commands have to supply a Callback object to receive the answer. Devices support a list of attributes which can be read or write. A device can return general information about itself or its state. Every device has a black box of the last n commands and implements security.
- Callback - client object which will be called by the server to return an asynchronous response. The callback has a handler method which is called when the response is unpacked.
- Monitor - a system network object for monitoring devices or attributes. Clients register their interest.
- Consumer - a client object for receiving events from a monitor.

TANGO also supports some pseudo network interfaces. Pseudo network interfaces are implemented only on the client side and not in the server. The following pseudo network interfaces are supported :

- GroupDevice - a client object for grouping Devices and executing commands on a group of Devices
- GroupAttributes - a client object for grouping Device Attributes of different Devices and reading and writing them.

A copy of the TANGO idl file can be found on the TANGO web page.

## 5 DEVICE PATTERN

Device servers are written using the Device pattern (see figure 1). The aim of the Device pattern is to provide the control programmer with a framework in which s/he can develop new control objects derived from the Device class. The Device pattern uses other design patterns like the Singleton, Command and Factory patterns (cf. [2]). The Device pattern creates the following hierarchy of classes :
Command –> MyCommand : a class for each command to implement based on the Command pattern. Each class must implement the *is_allowed()* and *execute()* methods.
DeviceClass –> MyDeviceClass : a Singleton class per device class which creates a list of commands and stores them in a vector. The derived MyDeviceClass has Factory methods for creating the list of commands and devices (retrieved from the database).
Device_impl –> MyDevice : device class implementing the hardware access necessary for each command and all device attributes in its methods and stores all device specific information.
Device_impl –> DServer : a special instance of Device_impl which exists only once per server and implements commands necessary to stop, restart and administer the server.

## 6 MULTITHREADING

Multithreading is an efficient way of implementing concurrency. TANGO supports multithreading at two levels - at the ORB level and at the server/client programmer level. CORBA 2.2 distinguishes between single and multithreaded ORBs however it does not specify the underlying threading policies in the case of multithreading. It is up to each ORB implementation to define and provide its own multithreading support. Orbacus provides a rich set of multithreading models. Servers can use the blocking, reactive, one-thread-per-client, one-thread-per-request or thread pool model. Which model to use is specified when the server starts up (via a command line option for example). TANGO uses the reactive model for servers which implement hardware access, and thread pool for servers like the database which need to serve a large number of clients simultaneously and do not have concurrency conflicts. At the programmer level TANGO is thread-safe i.e. server and client programmers are free to create threads as they need them and make calls to other servers in these threads.

## 7 ATTRIBUTES + PROPERTIES

*What are TANGO attributes and properties?* In addition to commands TANGO devices also support normalised data types called attributes and properties. Properties can be device, class or attribute specific.
*Why do we need attributes ?* Commands are device specific and the data they transport are not normalised i.e. they can be any one of the TANGO data types with no restriction on what each byte means. This means that it is difficult to interpret the output of a command in terms of what kind of value(s) it represents. Generic display programs need to know what the data returned represents, in what units it

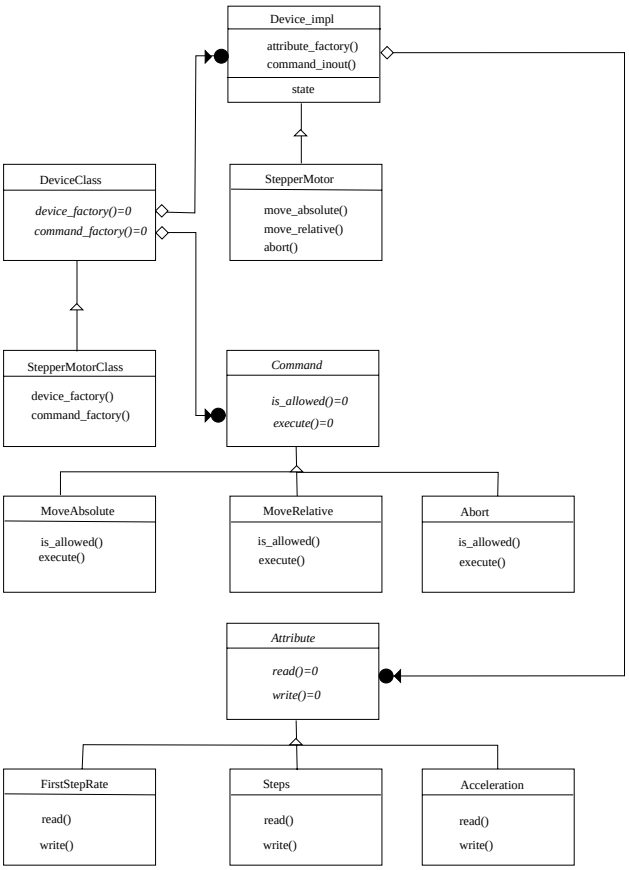


Figure 1: **Device Pattern Class Diagram** *(refer to [3] for class diagram notation)*

is, plus additional information like minimum, maximum, quality etc. Attributes solve this problem.
TANGO attributes[3] are zero, one or two dimensional data which have a fixed set of attribute properties e.g. quality, minimum and maximum, alarm low and high, engineering units, description etc. They are transferred in a specialised TANGO type and can be read or read-write. A device can supports a list of attributes. Clients can read one or more attributes from one or more devices.
TANGO device properties represent device specific information like device description and device configuration information like hardware addresses or for example first-steprate, speed or acceleration for a stepper motor. Properties are stored in the database and can be retrieved, updated or inserted via the database device server. Properties can be any simple type or sequence of simple types.

## 8 DATABASE

TANGO uses MySQL [6] as the database for storing permanent information. Permanent information can be device names and aliases, network addresses (IOR's), list of devices and their classes per device server, and properties. MySQL is a relational database which implements a subset of SQL.
The following tables have been defined for TANGO :

- device - contains device names, aliases, IOR's, class, version, start and stop timestamp
- device properties - contains device specific properties e.g. hardware addresses, device minimum and maximum
- class properties - contains class specific properties e.g. default minimum and maximum
- attribute properties - contains list of predefined attribute properties e.g. engineering units, attribute minimum and maximum

The database is accessed via a device server. The device server sends SQL requests to the MySQL server to interrogate or update the database. MySQL runs on many platforms. Performance is not a problem (MySQL is one of the fastest relational databases around). MySQL is free for non-commercial use and comes with full source.

## 9 NAMING

Naming and finding network objects is a fundamental service in any distributed system. CORBA offers a naming service which is hierarchically organised. TANGO on the other hand uses a 6 field naming scheme - [//facility/]domain/class/member[/attribute.property]. Where facility refers to the control system instance, domain refers to the subsystem, class the class and member the instance of the device. Attribute and property provide fine grained access to device attributes and properties. Because TANGO has its own database it has its own repository for names. Device names and network addresses (in the form of a stringified CORBA IOR) are stored in the device table in the database when the device server starts up. Clients only need to connect to the database device server in order to retrieve any device name. The database device server is started on a known port and host as a named servant. Clients connect to the database device server using an Orbacus extension which converts the port and host and name into a CORBA network object[4]. Once the client has connected to the database device server it uses the TANGO naming service to retrieve the device IOR from the database and build and maintain a connection to it. The TANGO API hides the details of this two step bootstrapping mechanism. Reconnection is managed as follows : if a server is restarted the client gets a CORBA communication exception the first time, from this point on every time it accesses the device it requests the new IOR from the database and tries to rebuild the connection until it succeeds. If the device is restarted immediately then at most one request is lost. If the database server is restarted then the named servant automatically reconnects.

[3] not to be confused with CORBA attributes which can represent any data type

[4] in the future this might be replaced by a CORBA compliant bootstrapping mechanism

## 10 DATA TYPES

*What data types does TANGO support?* TANGO supports a fixed set of data types for transferring data with commands and for attributes. All simple types and sequences of simple types are supported. In addition TANGO supports sequences of strings and longs, sequences of strings and doubles and sequence of TANGO attributes. The CORBA Any type is used to pack the different TANGO types and pass them over the network.

## 11 MONITORS

Monitors keep registered clients informed of device events (e.g. state changes) without the clients having to poll. Clients register their interest in an event by sending a request to the monitor service. The clients have to provide a Callback object which will be called when an event occurs. System wide events i.e. available for all TANGO devices, are state change, value changed, and alarms. Device server programmers can add their own device specific events e.g. counting stopped or buffer overflow. The monitors will rely on internal polling and the device cache to generate events. Monitors dispatch events to clients using CORBA oneway calls. For the moment we have decided against using the new CORBA Notify service for distributing events because there is no free implementation available. This might change in the future.

## 12 DEVICE CACHING

In a large control system (e.g. of 10 000 devices) running on a large number of hosts it is necessary to provide fast access to a large number of devices simultaneously to clients. With the normal device access paradigm this is not possible because accessing the hardware of hundreds of devices takes time even if all accesses are started in parallel (as is the case for asynchronous calls). The solution to this is to use cached values. For many clients a cached value which is guaranteed not to be older than a certain time is perfectly acceptable. TANGO has a device cache which is filled by system pollers. Clients can choose to read the cached or real value by toggling the source flag of a device.

## 13 API

*If CORBA is a high level object broker why do we need an API still ?* While it is true TANGO clients can be programmed using only the CORBA API, CORBA knows nothing about TANGO. This means clients have to know all the details of retrieving IORs from the TANGO database, additional information to send on the wire, TANGO version control etc. These details can and should be wrapped in a TANGO API. The API is implemented as a library in C++ and as a package in Java. The API also implements the pseudo-network objects like Groups and switches automatically between real, cached and other e.g. TACO, device sources. The API is what makes TANGO clients easy to write.

## 14 SCRIPTING

Scripting is still one of the most efficient and powerful ways of doing rapid code development. TANGO proposes to support scripting at two levels - at the device level and at the client level. Scripting at the device level means downloading scripts to the device server which will be activated and executed locally e.g. automating a startup sequence or monitoring a slow device. Tcl will be supported as script language. At the client level a number of well known scripting languages will be supported e.g. Tcl, Python, LabView, Matlab. All the scripting languages will have the same generic interface to TANGO.

## 15 PLATFORMS

TANGO is supported on 4 platforms presently - Linux, Windows NT, Solaris and HP-UX. All features of TANGO are supported on all platforms. This means device servers, the database and clients can run on all platforms. Frontends run Linux (on VME or PCs) or Windows (on PCs). Clients run on PCs, workstations or server machines.

Table 1: **Performance** - *TANGO performance figures measured on Windows/NT on a Pentium III @ 450 MHz, Linux on a Pentium @ 200MHz, HP-UX on an HP9000/735, Solaris on an UltraSparc 1, network was Ethernet 10baseT. Note the times presented here represent the minimum overhead to trigger an action, the time to execute the action in the server has to be added to this.*

| from - to | platform | transferred | time |
|---|---|---|---|
| client - device | Win/NT | 8 bytes | 0.9 ms |
| client - device | Linux | 8 bytes | 1.7 ms |
| client - device | HP-UX | 8 bytes | 3.0 ms |
| client - device | Solaris | 8 bytes | 3.7 ms |
| client - device | Linux | 1 Mbyte | 1.5 s |
| build connection | Linux | 1 device | 10.0 ms |

## 16 STATUS

TANGO is still being actively developed therefore not all parts of TANGO described above are implemented. The first device servers controlling simulated and real hardware are running. The database device server is available and the first simple clients (without the API) are working. The TANGO gateway which provides TANGO clients with access to the old TACO [7] device servers is running. The next step is to implement device attributes, asynchronous calls, monitors and interfaces to scripting languages. We will evaluate using the Notify service for events and Asynchronous Messaging for asynchronous calls.

## 17 EXAMPLES

*What are examples of TANGO device servers ?* They can range from simple digital I/O, serial lines, stepper motors to ccd cameras and plc subsystems. The first TANGO device servers are an Oregon stepper motor controller for VME and PC/104 on Linux, a serial line device server for PC/104 or PC under Linux, an OPC[5] device server for talking to PLC's from Windows.

## 18 BACKWARDS COMPATIBILITY

*How to deploy TANGO in the existing ESRF control systems ?* The ESRF control systems are based on the predecessor of TANGO - TACO. There are over 30 instances of the TACO control system running the ESRF accelerators and beamlines. The accelerator control system has almost 10 000 devices belonging to almost 200 classes and hundreds of clients. Porting all the classes and clients to TANGO is out of the question. In addition it must be possible to integrate TANGO servers and clients in a running TACO system without shutting down the TACO control system. Fortunately TANGO is very similar to TACO in its basic concepts (device oriented access) and it is easy to map TACO to TANGO and vice versa. By providing gateways which translate from TANGO to TACO and from TACO to TANGO it is possible to integrate new servers and clients into the running system smoothly. The respective APIs switch automatically to use the correct protocol and gateway.

## 19 ADDED VALUE

*What have we gained by rebuilding TACO using CORBA?* Here is some of the added value brought by rebuilding TACO using CORBA :

1. support for system events thereby providing faster client access and reducing the network load
2. generic data access via attributes
3. a modern protocol (IIOP) which supports web-based solutions
4. support for C++ and Java
5. immediate reconnection between client and server
6. support for scripting servers and clients
7. rebuilding TACO enables us to improve it based on our experience e.g. implementing scanning in frontends

*What have we lost by rebuilding TACO with CORBA?* Here is some minus value brought by rebuilding TACO using CORBA :

1. servers and clients require more memory e.g. the shared libraries require a few megabytes compared to hundreds of kilobytes in TACO
2. the new control system does not run on OS-9, the commercial OS we are presently using on VME

[5]OLE Process Control

## 20 OPEN SOURCE

The TANGO project (like TACO) is an *Open Source* project. All code will be available free of charge and warranty from our ftp site (follow link on web site [8]). Anyone can download it, use it, and even collaborate on improving it. Any improvements or bug fixes made will be incorporated into the next release.

## 21 CONCLUSION

Although CORBA has a steep learning curve and has a rich set of services it is easy to use for building simple types of network objects like Device which do not rely on any of the CORBA services. The high-level of abstraction and the C++ bindings succeed in hiding all details of network programming. Performance of CORBA (overhead of a few ms per call) is more than enough for an object oriented control system. The paradigm of device oriented access has again proved to be very powerful and adapted to the problem of control systems. Although TANGO is not finished yet it is already possible to write TANGO device servers and clients and deploy them in the existing control systems. TANGO offers significant improvements compared to TACO e.g. its support for modern protocols (IIOP) and languages (Java, C++), immediate reconnection, scripting. In the future new developments and improvements e.g. scanning on frontends, will take place only in TANGO and not in TACO in order to encourage TACO users to move to the 21st century.

## 22 REFERENCES


[1] "Object Oriented Programming Techniques Applied to Device Access and Control" by A.Götz, W-D.Klotz, J.Meyer (ICALEPCS '91, Japan 1991)

[2] "Advanced CORBA Programming with C++" by M.Henning and S.Vinoski (Addison-Wesley 1999)

[3] "Design Patterns" by E.Gamma, R.Helm, R.Johnson, and J.Vlissides (Addison-Wesley 1998)

[4] OMG home page - http://www.omg.org

[5] OOC home page - http://www.ooc.com

[6] MySQL home page - http://www.mysql.com

[7] TACO home page - http://www.esrf.fr/computing/cs/taco

[8] TANGO home page - http://www.esrf.fr/computing/cs/tango