Good practices for Tango DeviceServers development

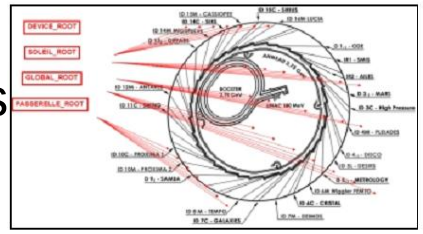
Alain BUTEAU – Synchrotron SOLEIL

Software for Controls and Data Acquisition group leader

On behalf the Tango team

SOLEIL and Tango

- SOLEIL started a collaboration with ESRF on TANGO in 2002
- Nowadays, SOLEIL operates 30 Tango Control Sys
  - 1 for accelerators
  - 1 for each beamline



- The ICA team (aout 20 people) is in charge of the development , maintenance and operation of software f Controls and Data Acquisition
  - Software engineering skills
  - From Tango controls
  - Up to GUI (See COMETE framework presentation on Friday)
  - Data management at the application levels (for data storage and data retrieval)

Writing a Tango Server is easy.

So what ?

Developing a DeviceServer in 5 minutes

- Use POGO to generate the code skeleton
- Implement the relevant methods to read/write attributes
- Implement the relevant methods to execute commands
- Compile and link with Tango libraries
- Declare the DeviceServers and its associated devices within jive
- Start the binary
- Run you client application (for instance ATKPanel)
- That's fine you can remotely control your equipement

It is easy !! No ?

That's how we started in 2002 at SOLEIL

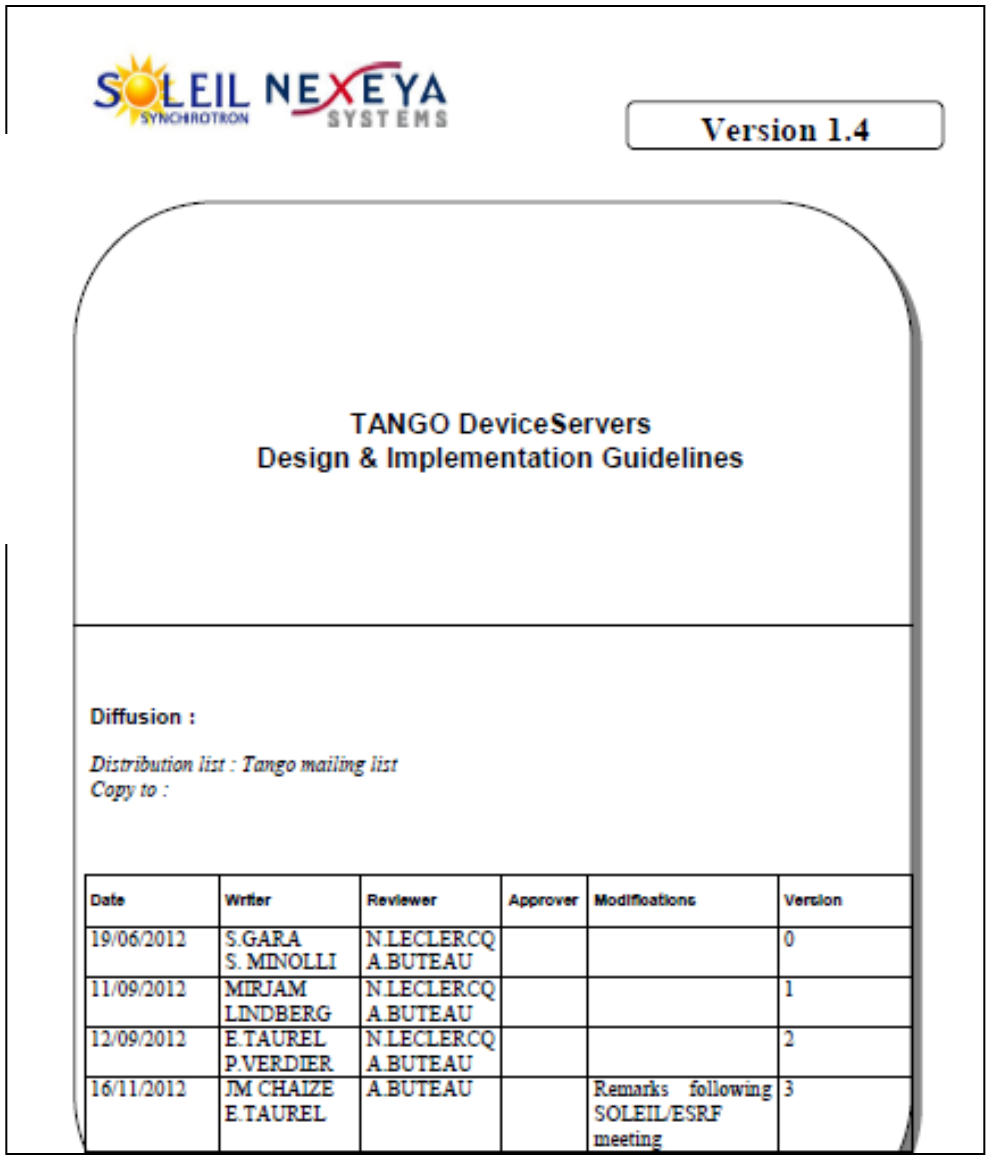
- We had to develop a Control system for our Accelerators and for 30 beamlines and no single line of code to do it
- We started developing our first Tango DeviceServers with the previous method
  - Because we had to learn how to do it properly
  - We were “regular” “young” “software developers” : (N.I.H syndrom, God vision when developing code, etc..)
  - Because we had a lot of pressure from our users to be able to control their equipments
- But we already knew this method is not sufficient when you plan to :
  - Use Tango devices as a “LEGO” bricks that will have to be reused and recomposed in various contexts
  - Maintain the code for 30 years
  - Obtain very high rates of software reliability (which is mandatory if you plan to deliver 6000 hours of beam per year)
  - Be able to automate user processes in a reliable and reproductible manner

Why using Guidelines for design and implementation ?

Then software engineering practices are mandatory to achieve such results

- At SOLEIL we developed about 400 different Tango DeviceServer
  - Interfacing all kind of equipment's (power supply , motion systems, beam diagnostics, etc.)
  - Interfacing “pure software” devices (such as Scanning engine, Calculation routines, etc..)
- This development work has been done by more than 10 different developers
  - with different skills and knowledge on Java and C++ programming
- We also had an operational feedback on what we did
  - Being in daily contact with Machine and Beamline users
  - Doing also “On call” duties 6000 hours per year
- We accumulated an important “in house experience” of developing “Good” TangoDeviceServer
- In 2012 we had the opportunity thanks to our collaboration with MAX-IV to hhire external experienced software engineers to summarise this experience in a document for the Tango Community

Tango Guidelines



SOLEIL SYNCHROTRON NEXEYA SYSTEMS

Version 1.4

TANGO DeviceServers
Design & Implementation Guidelines

Diffusion :

Distribution list : Tango mailing list
Copy to :

Date	Writer	Reviewer	Approver	Modifications	Version
19/06/2012	S.GARA S. MINOLLI	N.LECLERCQ A.BUTEAU			0
11/09/2012	MIRJAM LINDBERG	N.LECLERCQ A.BUTEAU			1
12/09/2012	E.TAUREL P.VERDIER	N.LECLERCQ A.BUTEAU			2
16/11/2012	JM CHAIZE E.TAUREL	A.BUTEAU		Remarks following SOLEIL/ESRF meeting	3

Standardizing design and coding practices is a win-win approach

- At the community level:
  - To be able to reuse « good quality » DeviceServers developed by other institutes
    - ✓ Avoiding the cost of « redoing » everything on our own
  - To ease Tango usage for newcomers
    - ✓ Which helps enlarging then the community
- At the institute level
  - To be able to easily integrate new software developers resources
- The « Tango design guidelines » is quite a dense document which you must take time to understand and read carefully

Overview of the "Tango DeviceServers design and implementation Guidelines"

Tango concepts revisited

Device : Concept

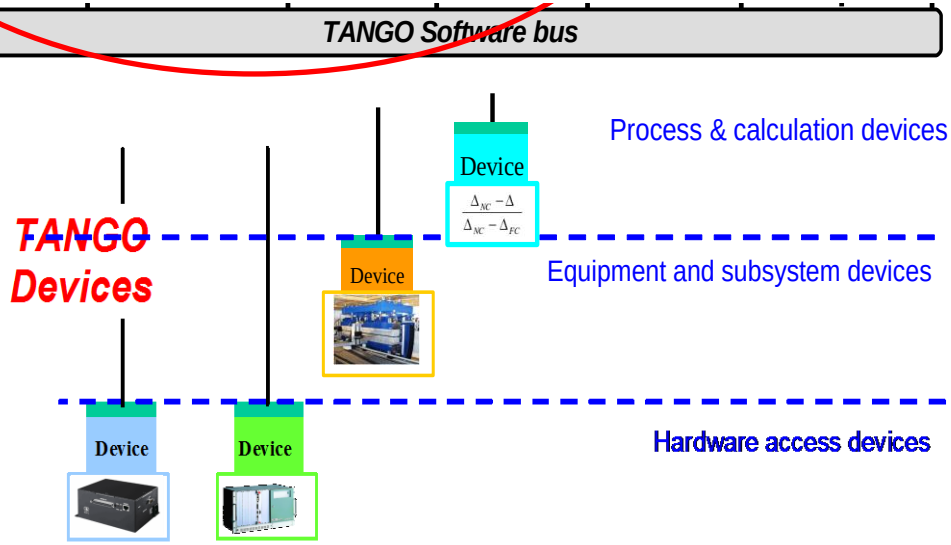
- Historically a device is an entity to control (Hardware or software)
- device = 1 polymorphous object
  - 1 equipement (ex: 1 power supply)
  - 1 collection of equipment (ex: 1 motor + 1 coder)
- A pure software component e.g :
  - A scan device
  - A data fitter device

Recommandations :

- ***A Tango device allows to make an abstraction of the equipment nature and implementation details***
- ***1 Device = 1 service = 1 element***
- ***of the system***

Device : Hierarchy

- A Tango control system can be logically hierarchically organized
  - At higher level the devices are logical to :
    - ✓ Manage and represent subsets of the control system
    - ✓ To perform sequences of actions on low level devices
  - These higher level of device must evolve regardless of real hardware

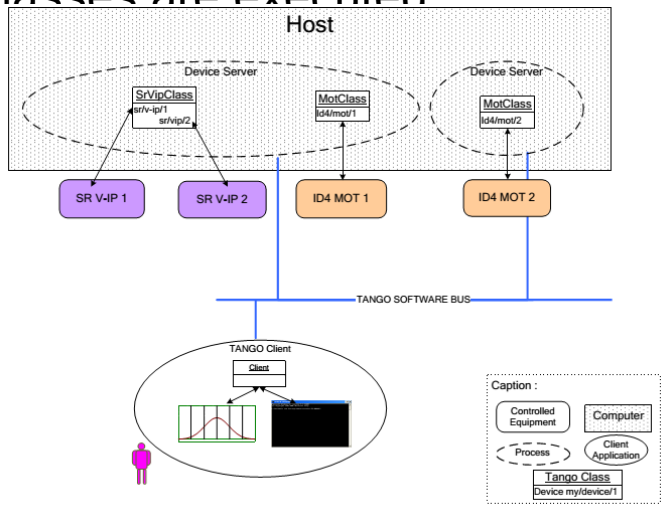


Recommandation : Design your Tango Control system with a hierarchical view in mind (even if technically Tango doesn't impose this structure)

Vocabulary misunderstandings

- Sometimes, there are misuses of language regarding the concepts of: device, device server and TANGO class.
- TANGO class:
  - a class defining the interface and implementing the device control or the implementation of a software treatment.
- Device:
  - An instance of a TANGO class giving access to the services of the class.
- Device Server:
  - A process in which one or more TANGO classes are executed

Recommandation : These three concepts are closely related, but they express different and important concepts of Tango. Take time to clearly understand them!



Attributes and commands

- Attribute : Definition
  - Physical unit produced or administrated by the device
    - ex: motor position, power emitted by a power supply, …
  - T**he main purpose of an attribute is to replace getters and setters.**
- Command: Definition
  - **A command is associated with an action. *On, Off, Start, Stop* are commons examples**

Recommandation : To get an information produced by a device prefer Attributes (that can be archived or monitored in trends) than Commands

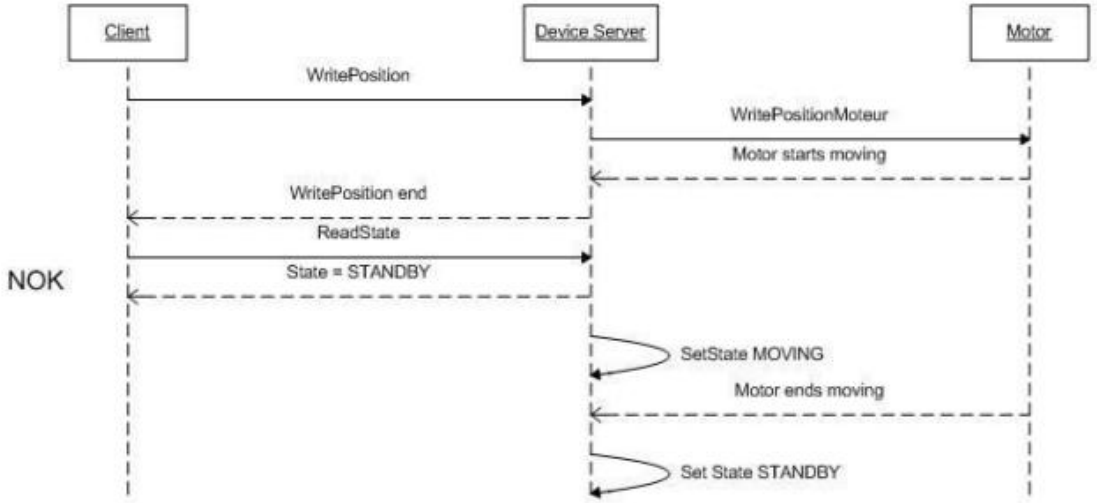
States

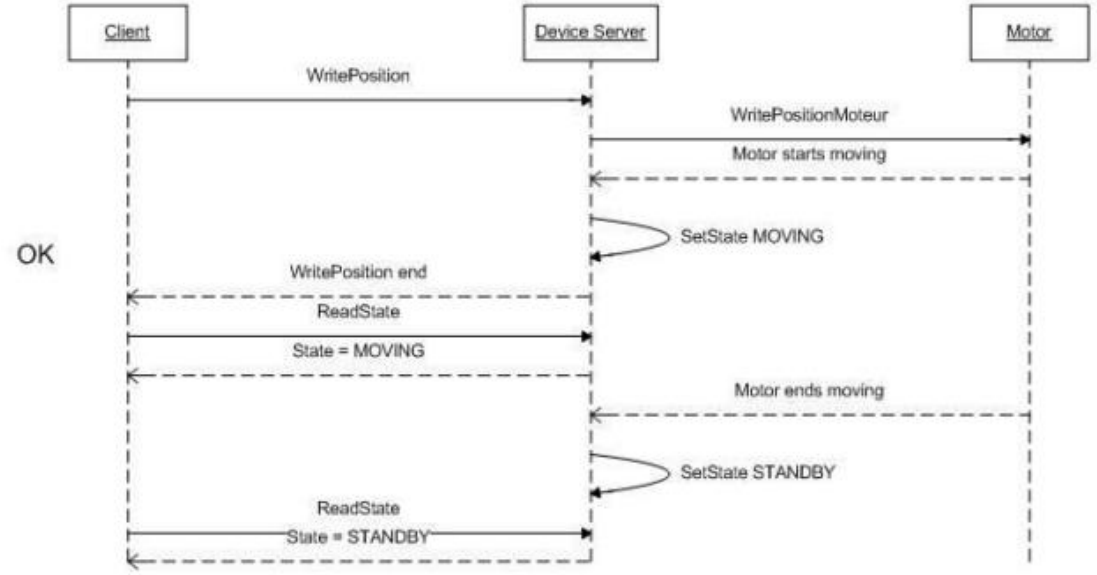
- A TANGO device has a UNIQUE state (meaning a finite state machine).
- The device state is a key element in its integration into the control system.
- Therefore, you should be very careful in the management of state transitions in the device implementation

Recommandations :

- ***The device state must, at any time, reflect the internal state of the system it represents.***
- ***The state should represent any change made by a client's request.***
- ***Use always the same set of states for similar conditions .At SOLEIL:***
  - ***all Actuators are MOVING (when moving)***
  - ***all Sensors are "RUNNING" (when acquiring data)***
- ***Be careful to "non desired" transient state that will raise problem on higher level***

State


Client
Device Server
Motor
WritePosition
WritePositionMoteur
Motor starts moving
WritePosition end
ReadState
NOK
State = STANDBY
SetState MOVING
Motor ends moving
Set State STANDBY


Client
Device Server
Motor
WritePosition
WritePositionMoteur
Motor starts moving
SetState MOVING
OK
WritePosition end
ReadState
State = MOVING
Motor ends moving
SetState STANDBY
ReadState
State = STANDBY

Design recommendations

Reusability

- Estimates systematically, before coding a device, the possibility of reusing a device available in the code repositories
- Design the device as reusable/extensible as possible because it may interest the others developers in the community.

Recommandations :

- ***Configurable: (e.g.: no port number "hard coded", but use of a parameter via a property)***
- ***Self-supporting: the device must be usable outside the private programming environment (eg: all the necessary elements to use the device (compile, link) must be provided to the community).***
- ***The use of the GPL should be considered, and the use of proprietary libraries should be avoided if possible***
- ***Portable: the device code must be (as much as possible) independent of the target platform unless it depends on platform specific drivers***
- ***Documented in English***

Generic interface programming

- The device must be as generic as possible which means the definition of its interface should reflect the service rather its underlying implementation.
  - ✓ For example, a command named “WriteRead” reflects the communication service of a bus (type: message exchange), while a command named “NI488_Send” reflects a specific implementation of the supplier.
- Show the general characteristics (attributes and commands) of a common type of equipment that it represents.
  - ☞ For example, a command ”On” reflects the action of powering on a PowerSupply while a command named “BruckerPSON” reflects a specific implementation which must be avoided.

SOLEIL SYNCHROTRON

TANGO Connecting things together

Performances related design recommendations

- Do not make assumptions about the nature of the clients!
  - The behavior of the Device must the same if I use a client that reads a single attribute or a client that reads them all
- Do not make assumptions on the number of clients connected to a Device
  - At the beginning 1 client is connected
  - But in real operation after a few months or years , you may end with tens of clients on a single device

Recommandation : The response time of the device should be minimized and in any case lower than the default Tango timeout

Recommandations for the specifications/design/development phases

- During the specification process define the attributes, commands, states of your Device
- Write test cases focusing on error scenarios
- Use usual UML diagrams :
  - To clarify data relationship within the internal device software objects
  - Use sequence/state diagrams when state transitions are complex
  - Make a pair code review and cross check with the Recommendations Guidelines

General Code implementation Consideration

- Don‟t forget that the TANGO interface is only a mean to insert a service in a control system.
- Therefore, it is necessary to think the device internal design like any other application and just add the TANGO as an interface on top of it.
- As a rule of thumb if the code implemented within the POGO markers is too long, a good practice is to move it to another class.
  - ☞Then Pogo generated methods will be only a few lines of code long

Recommandations :

- ***In practice, it is necessary to avoid mixing the generated code by POGO and the developer‟s one.***

Language

- The TANGO community is international and the developments could be shared with the community, so it is recommended to use ENGLISH for a device development.

Recommandations :

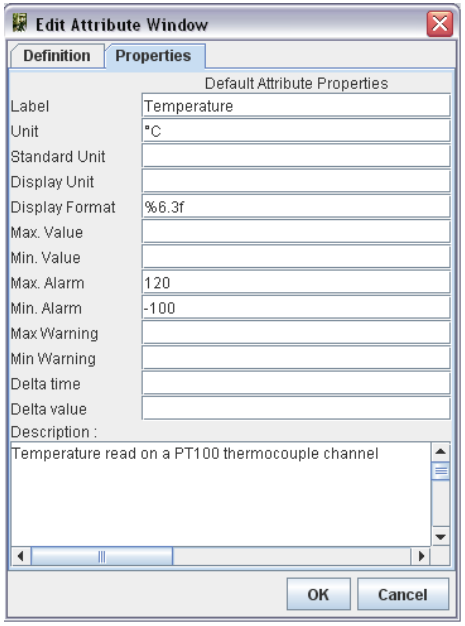
English will be used for:

- ***The interfaces definition (attributes and commands)***
- ***The device documentation (online help for command usage and attributes description)***
- ***The comments inserted in the code by the developer***
- ***The error messages***
- ***The name of variables and internal methods added by the developer.***

Code generation : POGO is mandatory

Recommandations :

- ***The use of POGO is mandatory for creating or modifying the device interface.***
- ***Every command, attribute, property or device state must be fully documented; this documentation is done via the POGO tool.***
- ***The automatically generated code by POGO must not be modified by the developer.***
- ***The developer must include its own code in the "PROTECTED REGION" specified parts.***



Tango interface : naming rules

- Having homogeneous conventions for naming attributes, commands and properties is a good way to promote DeviceServers reuse inside the Tango collaboration (or in your institute)

Recommandations :

- ***The Tango class name is obtained by concatenating the fields that compose it – each field beginning with a capital letter:***
  - ***Eg : MyDeviceClass***

Tango interface : naming rules

The attribute naming recommendations are:

- ***Name composed of at least two characters***
- ***Only alphanumeric characters are allowed (no underscore, no dashes)***
- ***Start with a lowercase letter***
- ***In case of a composite name, each sub-words must be capitalized (except the first letter)***
- ***Prohibit any use of vague terms (eg: readValue).***

The command naming recommendations are:

- ***Name composed of at least two characters***
- ***Only alphanumeric characters are allowed (no underscore, no dashes)***
- ***Start with a uppercase letter, In case of a composite name, each sub-words must be capitalized***
- ***Prohibit any use of vague terms (eg: Control).***

Tango interface : naming rules

- It is a good practice that a particular signal type is always named in a similar way in various DeviceServers.
  - For example the intensity of a current should always be name "***intensity***" (and not "***intens***", "***intensity***", "***current***","***I***" depending on the DeviceServers).
  - This allow the user to quickly make the link between the software information and the physical sensor and reciprocally.
- The choice between the *Expert* or the *Operator* level for an interface must be thoughtful.
- All basic commands must be accessible at the OPERATOR level

Implementation

Guidelines

Tango polling mechanism

- TANGO implements a mechanism called *polling* which alleviates the problem of equipment response time
  - ☞ (which is usually the weak point in terms of performance).
- From the perspective of the device activity, the polling is in direct competition with client requests. The client load is therefore competing with polling activity
  - ☞ This means that polling activity has to be tuned in order to keep some device free time to answer client requests.

Recommandations :

- ***The recommendation for device polling tuning is to keep the device free 40% of time.***

Tango threading mechanism

- Threading is a possible solution for the load problem
  - a thread (managed by the device developer) supports communication with the material (polling or other) and the data obtained are put in the “cache”.
  - You can now produce the “last known value” to the client at any time and optimize the response time..
  - This approach, however, has a limit where it is necessary to reread the hardware to assure clients that the returned value is the system “current state”.

Recommandations :

- ***When the design of the Tango class requires threading:***
- ***• in case of simple thread usage, in C++ the recommendation is to use a C++11 thread***
- ***• In case of acquisition thread with messages exchange in C++ the recommendation is to use Yat4TANGO::DeviceTask class..***

Use Yat and Yat4Tango libraries

- YAT library offers many utilities for :
  - Portability
  - Threading
  - Active Objects features
  - Timers
  - Date/Time
  - Etc..
- YAT4Tango offers utilities linked with Tango
  - logging , exception helpers, etc ..
- Both are :
  - hosted within the tango-cs repository
  - actively developed and supported by SOLEIL

Recommandation :Use Yat and YAT4Tango when portability (Windows /linux) is required or to benefit from advanced features for threading , etc ..

Error handling : Use Tango logging system

- The introduction of logging in the device code enables easy development, bug research and the user
- understanding of the device operations
  - ✓ DEBUG_STREAM : for software developpers
  - ✓ INFO_STREAM : user info (start/stop of a measurement, a process)
  - ✓ WARN_STREAM : warning messages (degraded mode ,etc)

It is not mandatory, but highly recommended to add an attribute named "log" in the device interface, which tracks all the internal activity of the device (as defined in TANGO Logging).

Exception handling

- The developer has to ensure:
  - That any exception is caught, completed (TANGO allows it) and spread (use of the rethrow_exception method),
  - If an error occur it must be logged using the Tango Logging Service
  - That the return code of a function is always analyzed,
  - That the device Status is always coherent with the State,
  - That the error messages are understandable for the final user
- The Status is the indicator that will help the user to find the error reason.
- But do not throw exception in the init_device method !! (the device must be kept alive in all cases)
- Of course , imagine all errors scenarios (cabling problem with the equipment, ..)

Exception handling

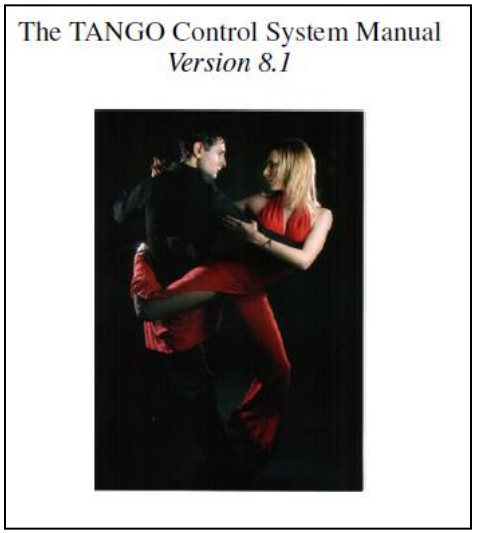
- It is a good practice to standardise the Reason Field of the Tango exception

Nom standardisé pour les types d'erreur
OUT_OF_MEMORY
HARDWARE_FAILURE
SOFTWARE_FAILURE
HDB_FAILURE
DATA_OUT_OF_RANGE
COMMUNICATION_BROKEN
OPERATION_NOT_ALLOWED
DRIVER_FAILURE
UNKNOW_ERROR
CORBA_TIMEOUT
TANGO_CONNECTION_FAILED
TANGO_COMMUNICATION_ERROR
TANGO_WRONG_NAME_SYNTAX_ERROR
TANGO_NON_DB_DEVICE_ERROR
TANGO_WRONG_DATA_ERROR
TANGO_NON_SUPPORTED_FEATURE_ERROR
TANGO_ASYNC_CALL_ERROR
TANGO_ASYNC_REPLY_NOT_ARRIVED_ERROR
TANGO_EVENT_ERROR
TANGO_DEVICE_ERROR
CONFIGURATION_ERROR
DEPENDENCY_ERROR
NO_DEPENDENCY

Resources available to work within the Tango community

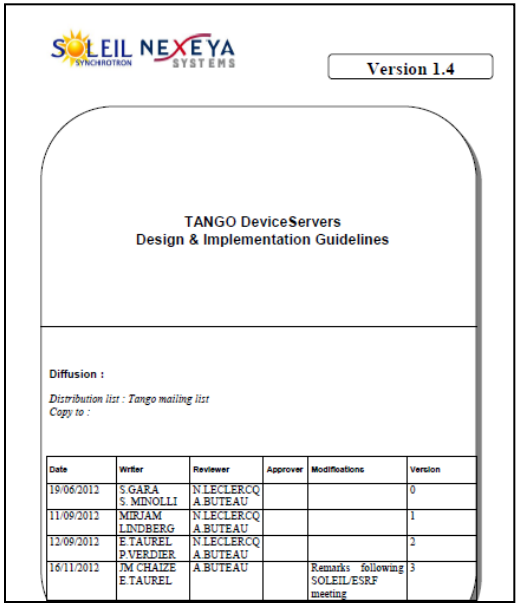
Documentation

- The Tango reference manual is the best entry point !!
  - Please RTFM (“Read this Fucking Manual”)



- There are tutorials on the “Tango www site”

- A document “ Tango DeviceServers design consideration”
  - Tries to summarize all the question you must think about before coding
  - This document tries to summary years of mistakes and unanswered questions on the “How To” develop “good” Tango



Community tools

- The Tango “Pink” site is the central place to find documentations, examples, etc ..
  - It is unfortunately managed on a “best effort” basis and it may be sometimes outdated
  - Moreover the navigation on this Web site is sometimes strange : Use Google !!
- The mailing list (tango@esrf.fr) is the best way to have fast responses to technical problems
  - It is an active mailing list (a few hundreds of post per year)
  - And the spirit is quite professional and avoid the (usual ?) religious war on software language
  - So feel free to post . Tango dancers are proud of Tango and usually help
- The Tango meetings
  - Are important because you physically meet the Tango dancers
  - They last 1,5 days and the program is usually the following
    - ✓ Status on each institute Tango activity , resources, projects ,etc .
    - ✓ Status on Tango work packages
    - ✓ New technical ideas or projects

Source Code Management

Release policy and download site

- Tango Core
  - There is a major release each 18 months
  - The technical roadmap were defined by the E.C following the proposal of the Collaboration Coordinator
  - This major release is downloadable on the official Tango site
    - ✓ and integrates new versions of most the workpackage new releases
- Other Tango workpackages
  - Have their own lifecycle mostly driven by the institute in charge requirements constraints
  - For exemple the “Tango Archiving system” is released 5 times per year
    - ✓ Because a new version is deployed on the SOLEIL accelerator 5 times per year
  - They should be (in theory) accessible through the official Tango site
    - ✓ Or on SOLEIL Maven repository
    - ✓ Or on F.S.RF http

Source code repositories : 2 different SVN repositories

- For each software module an SVN structure has been defined
  - ☞in trunks, tags, branches “directories”
- The tango-cs http://sourceforge.net/projects/tango-cs
  - ✓ hosts Tango Core system and the workpackages defined previously
  - ✓ Write access is granted only to Tango projects contributor
  - ✓ Repository organization mimics Tango workpackages organization
- Tango-ds (http://sourceforge.net/projects/tango-ds/) hosts Tango DeviceServers which may be of common interest
  - ☞because they do not rely on particular memory mapping or specific electronics
  - ☞Most of them control commercial instruments
    - ✓ Ex : motor controllers, Agilent instruments, ADC boards,etc ..
  - ☞Or generic software processes or calculations
    - ✓ Data fitting, image analysis, etc ..

Tango-ds repository organization

- AcceleratorComponents
- Acquisition
- BeamDiagnostics
- BeamlineComponents
- Calculation
- Communication
- CounterTimer
- InputOutput
- MagneticDevices
- MeasureInstruments
- Motion
- OtherInstruments
- PowerSupply
- RadioProtection
- SampleEnvironment
- Security
- Simulators
- SoftwareSystem
- StandardInterfaces
- Temperature
- Training
- Vacuum

List is available :

http://www.tango-controls.org/device-servers

Tango-ds resources

- There are hundreds of Tango DeviceServers stored there
- With a large variability in terms of quality and reusability
- Nevertheless please store your project on this repository if it may be reused elsewhere
  - ☞i.e not direcly linked to a specific hardware
  - ☞Interface a commercial instrument
  - ☞Or a generic “software” component

Conclusion

Conclusion

- Tango is a very intuitive system from the user point of view
- It is also easy to understand for software developers
- Nevertheless , to get high quality results , software engineering methods are mandatory at all phases (specifications, design, implementation, code management,etc.)
- Entry cost in the Tango community decreases in the last years because of :
  - Documentation efforts (Tango Book and Guidelines)
  - Community Help
  - Development of the eco system of applications, libraries , development tools (POGO), code repositories, etc..
- But you still have to learn and make efforts