Synchrotron SOLEIL

Tango: un système de contrôle Orienté Objet

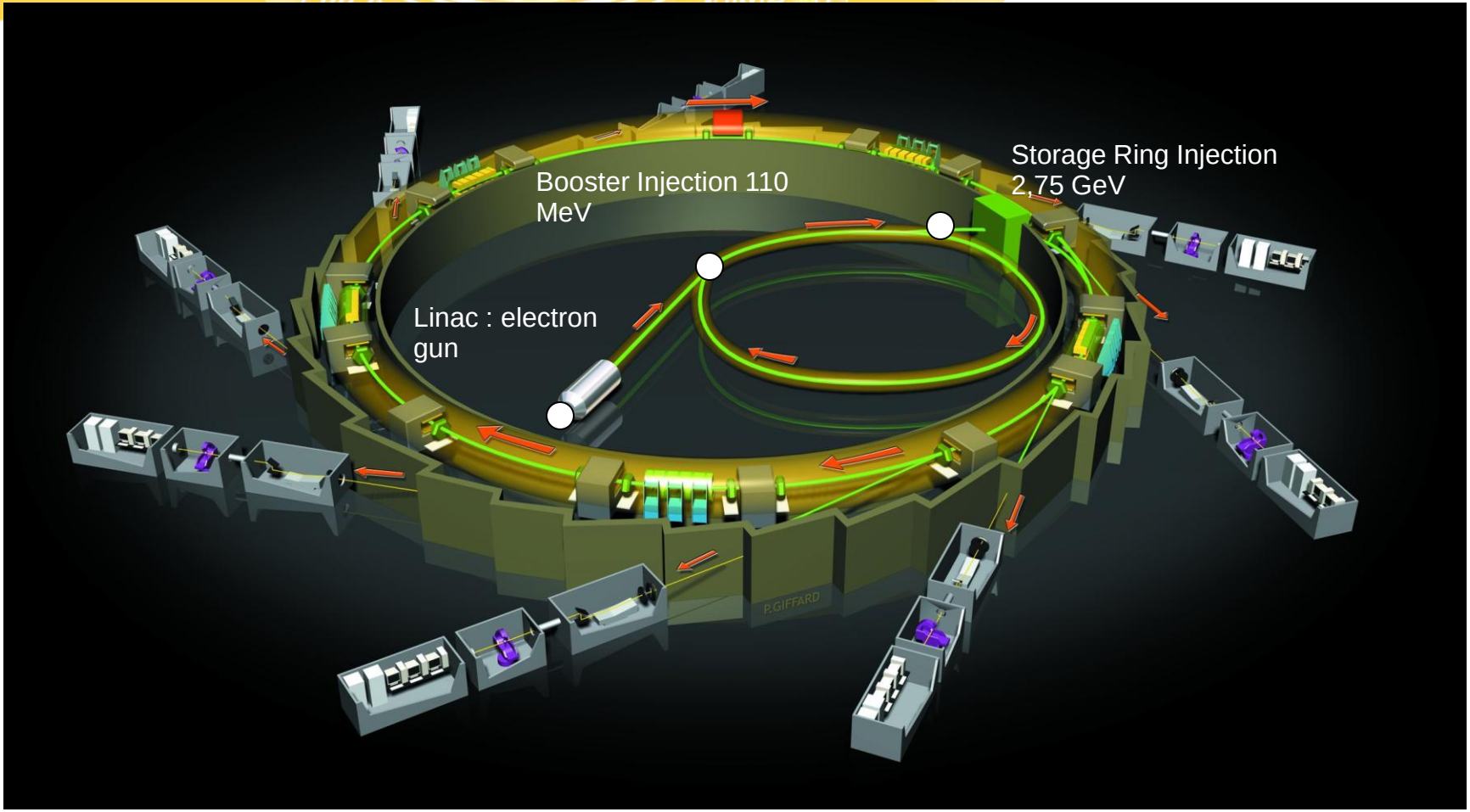
Alain BUTEAU :
Responsable du groupe Informatique de Contrôle/Acquisition

Agenda

- SOLEIL : quelques rapides rappels
- Tango : genèse du projet
- Concepts principaux
  - ✓ Le bus logiciel
  - ✓ La notion de Device
  - ✓ La base de données statique
  - ✓ Les DeviceServers
- Développement d'un Device tango
- Panorama des applications clientes
  - ✓ Applications prêtes à l'emploi pour :
    - Configurer et administrer le système de contrôle, monitorer les devices, archiver les données produites par les Devices dans des bases de données MySQL ou Oracle, construire des synoptiques.
  - ✓ Développements d'applications spécifiques : java, MATLAB, python, Labview, IGOR, SCADA
- Quelques chiffres opérationnels de Tango à SOLEIL
- La collaboration Tango
  - ✓ Histoire
  - ✓ Organisation actuelle
  - ✓ Les outils collaboratifs
  - ✓ L'élargissement de la communauté
- Conclusion
  - ✓ L'évolution de nos architectures logicielles

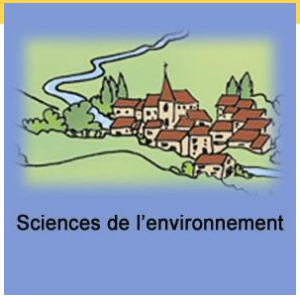
SOLEIL : Quesaco ?

Machine : linac, booster, storage ring



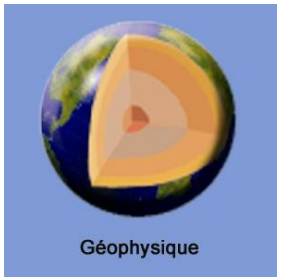
Storage Ring : 354 m circumference (113 m diameter)

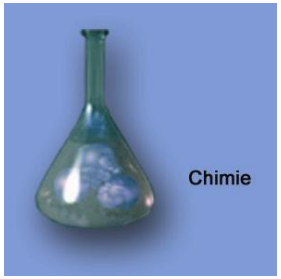
Domaines d'applications



Détection de substances polluantes, optimisation de pôts catalytiques, nouveaux matériaux…

Connaissance de la structure des matériaux du manteau terrestre…

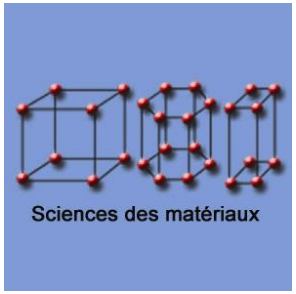


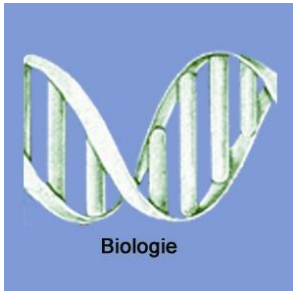


Procédés catalytiques, exploration de la matière et connaissance de ses propriétés électroniques, magnétiques
(ex: stockage magnétique haute densité)

Recherche de nouveaux médicaments, imagerie des tissus osseux, vaisseaux sanguins, étude de l'ADN...

Élaboration de nouveaux matériaux, (ex : semi et supra conducteurs, disque durs et mémoire magnétique,batteries, étude de la prise rapide de ciment)





Dans tous les domaines, un large accueil est prévu pour les industriels et acteurs des enjeux sociétaux

Archéologie, patrimoine, aéronautique, pharmacologie, microélectronique…

The site and budget

Total budget 2002 ⇨ 2010 :

451,7 M€

Budget repartition for the 2002-2010 period (conditions euros 2000)

- investments **227.3 M€**
  - buildings 64M€, machine 63M€, BL 3M€
- operations **58.3 M€**
- Salaries (2002-2010) **166.1 M€**

- la région IDF (53 %)
- le CNRS et le CEA (31%) *(2 actionnaires de la société SOLEIL à 72 et 28%)*
- le département de l'Essonne (12%)
- le ministère de la recherche (2%)
- la région centre (2% : 3 lignes de lumière)

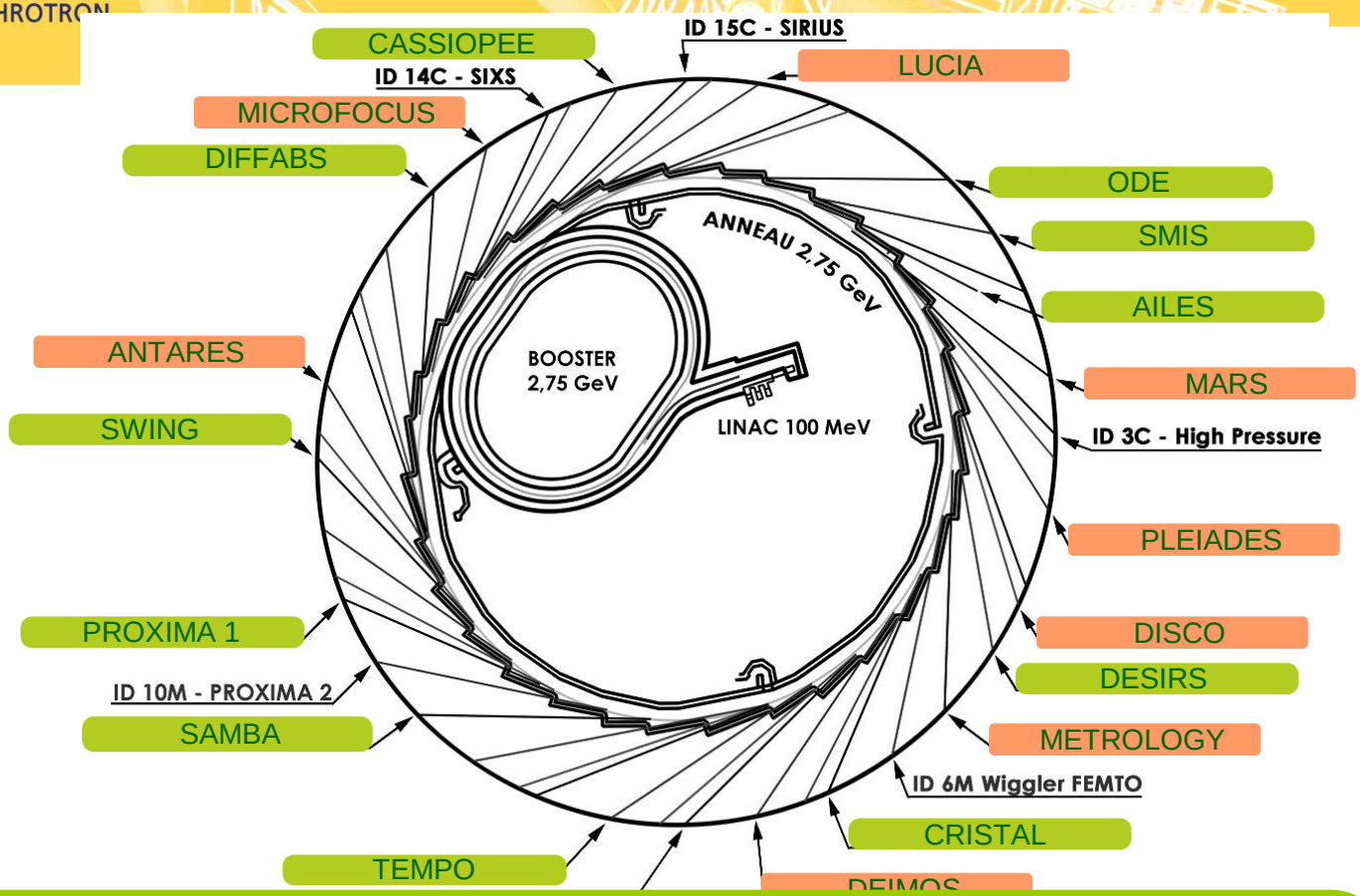
After 2010, the annual budget for operations will be ~ 48 M€/year

The SOLEIL Staff

357 permanent positions

Direction 19 people
Experiments 144 people
Machine 61 people
Administration 29 people
Technical Services 65 people
Computing and Controls 41 people

The SOLEIL BeamLines



43 Beamlines potentielles
26 BLs sont financées

Aujourd'hui plus de 20 beamlines accueillent des utilisateurs (environ 2000 par an)

Missions équipe Informatique Controle Acquisition

- Définir, développer et maintenir en condition opérationnelle
  - ✓ l'ensemble des systèmes informatiques nécessaires aux Lignes et à la Machine
  - ✓ En assurant astreintes 24H/24H, pendant 6700 heures par an
- Etre garant de la pérenité des logiciels de Contrôle/Acquisition
  - ✓ Pendant la durée de vie des installations (30 ans)
- Notre périmètre couvre:
  - ✓ L'interfaçage logiciel des équipements au controle
  - ✓ Les acquisitions de données et les outils logiciels de stockage/restitutions des données (i.e DataManagement)
  - ✓ La conduite du process, et les interfaces Homme Machine,

Le groupe Informatique de contrôle/acquisition

Informatique et Contrôle/Acquisition

- **14 permanents**
- **+ renforts externes**

(prestations en régie ou au forfait sur les projets les plus matures)

- Domaines d'expertise
  - ✓ Ingénierie logicielle
  - ✓ Java, C++
  - ✓ Système de contrôle et acquisition de données
  - ✓ IHM

Tango : La genèse du projet

La problématique du Contrôle/Commande à SOLEIL

- Mettre en œuvre :
  - ✓ 1 système de contrôle pour les accélérateurs : LINAC, Anneau, Booster
  - ✓ 1 système de contrôle pour chaque ligne de lumière
  - ✓ Besoin d'échanger des informations entre ces différents systèmes en parlant le même langage

Soit au total au moins 27 systèmes de contrôle différents

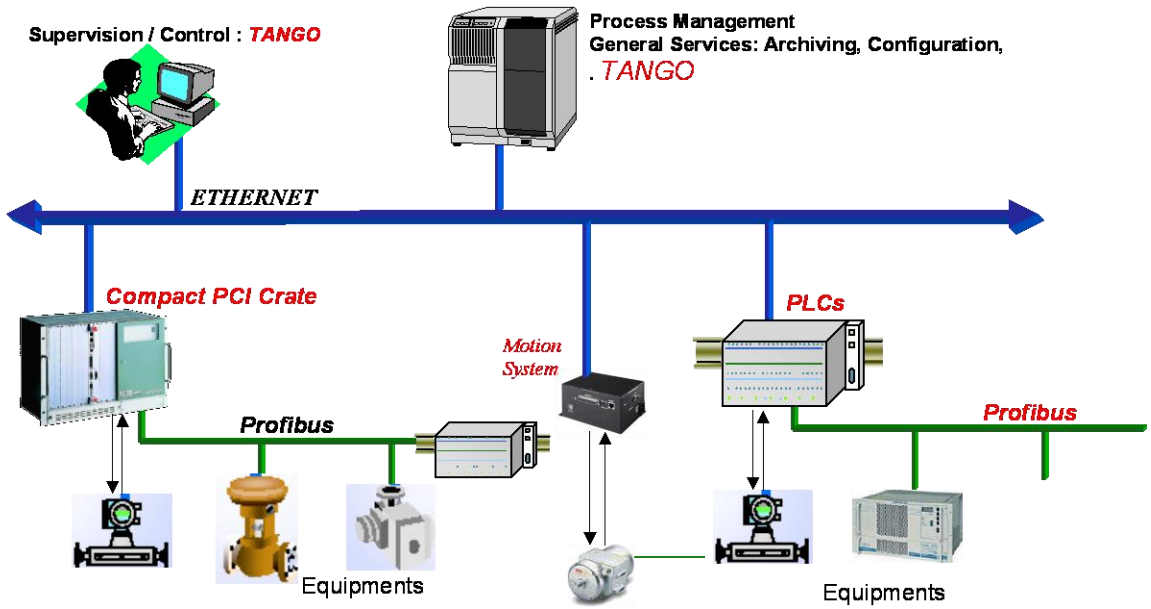
Vue du site juillet 2003

La démarche retenue pour faire notre choix en 2001/2002

Problème 1: Systèmes distribués

Les composants du système sont géographiquement répartis sur des machines interconnectées par un réseau informatique:

- Plus de 100 frontaux Compact PCI
- Plus de 100 automates
- Des dizaines de postes opérateurs
- Des dizaines de serveurs



Problème 1 : Comment faire communiquer tous ces systèmes ?

Problème 2: Object Oriented Paradigm

- Object oriented approach is now commonly recognized as the best methodology for software construction
- ***« A control system is a sea of networked objects ; every part of the control system should be a network object whether it is a low-level, high-level or application-oriented function »***
- Montrer à l'utilisateur ces objets systématiquement sous la même « forme »

Problème 2 : Comment obtenir des objets logiciels à l'échelle d'un réseau ?

Problème 3: Systèmes hétérogènes au niveau matériel et logiciel

- Des systèmes industriels lents
  - ✓ Automates
- Des systèmes d'acquisitions rapides :
  - ✓ Beam Position Monitors
- Des systèmes vendus clés en main :
  - ✓ Contrôle/Commande LINAC
- Des systèmes d'exploitation hétérogènes
  - ✓ Applications scientifiques sous Linux
  - ✓ Des drivers/librairies sous WIN32 ou sous Linux

Problème 3: Comment obtenir un tout cohérent à partir de matériels et de logiciels hétérogènes ?

Notre analyse des solutions possibles en 2002

- EPICS :
  - ✓ Pas orienté objet
  - ✓ Protocole d'échange des données « propriétaire »
  - ✓ Modèle de données « à plat » par échange de variables
  - ✓ A l'époque pas disponible sous Win32
- ACS
  - ✓ Basé sur CORBA
  - ✓ Encore embryonnaire
- TANGO
  - ✓ Basé sur CORBA
  - ✓ Embryonnaire
    - ❑ Mais basé sur le retour d'expérience de l'ESRF avec TACO
    - ❑ Une solution de terrain, simple à comprendre et à mettre en œuvre
  - ✓ Au delà de la technique , possibilité de démarrer une collaboration avec une autre équipe partageant les mêmes besoins et ayant une culture technique proche (O-O, C++, java)

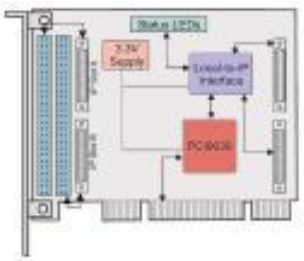
Tango : Concepts principaux

Tango : un bus logiciel

Analogie avec un bus electronique

- Chaque carte remplit une fonction bien définie
- Chaque carte “pas ou faiblement” couplée aux autres cartes
- Les développements de chacune des cartes peuvent être découplés

- ***Mais*** chaque carte doit respecter une interface stricte et bien définie pour pouvoir se connecter au système complet



TANGO software Bus

interface

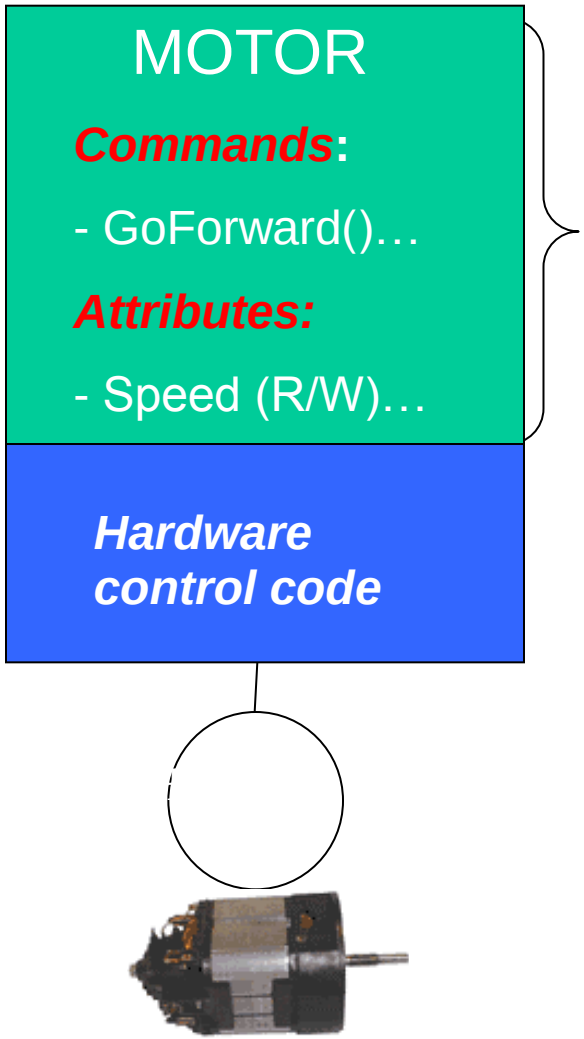
Device

Hardware (motor…)

Board

An example of a Device Interface : A motor

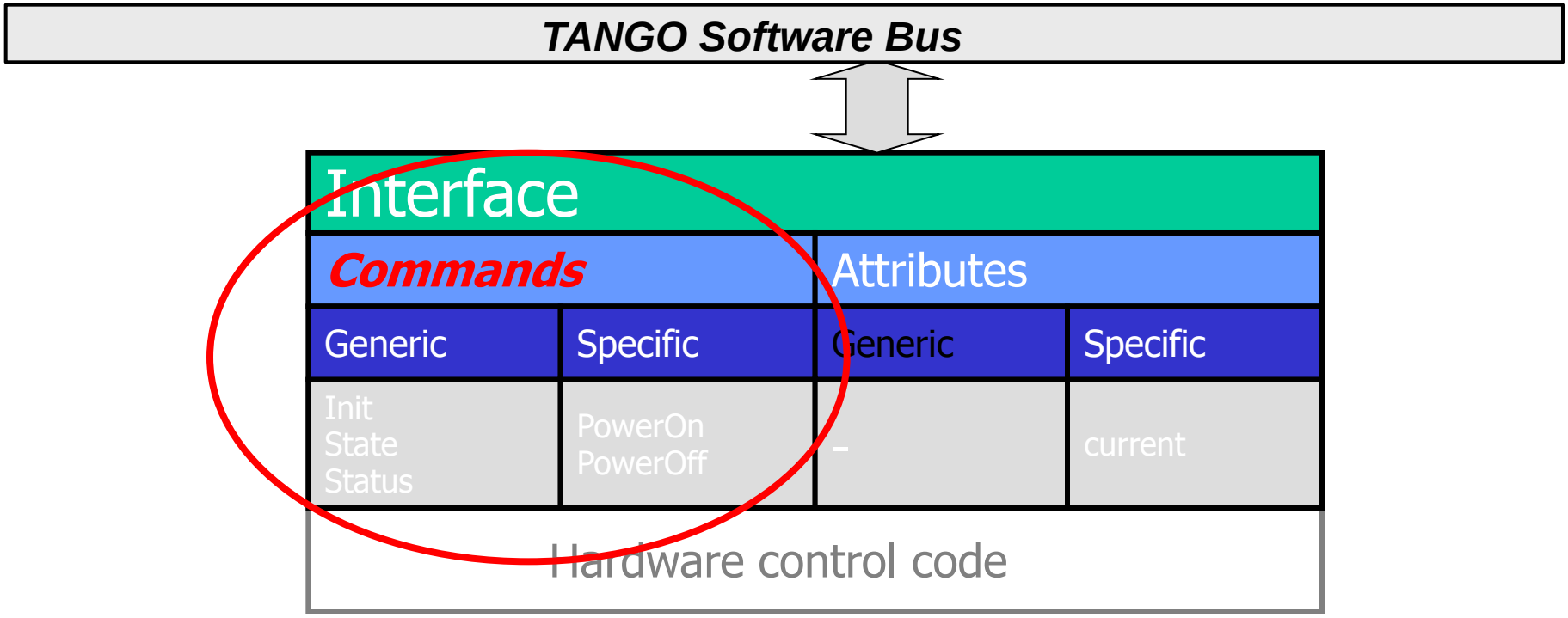
- The Interface :
  - ✓ describes what the Device is supposed to do
  - ✓ It's only a promise of the services you may expect from the Device
- But there isn't any magic !!!
  - ✓ Code has to be written to fullfill the promised services



- Historically a device is an entity to control
  - ✓ Hardware or software
- device = 1 polymorphous object
  - ✓ 1 equipment (ex: 1 power supply)
  - ✓ 1 cluster of devices :
    - ❑ a monochromator
    - ❑ a complete subsystem (a LINAC !)
  - ✓ A pure software component e.g :
    - ❑ A scan device
    - ❑ A data fitter device, FFT calculations,etc

But have in mind that a device is a real and good « Software Component » which respects O-O paradigm (encaspulation, reusability, ..)

- Has a communication interface
  - ✓ interface = ***commands*** + attributes
    - ❑ commands ≈ actions
    - ❑ attributes ≈ physical units



TANGO: device: interface: command

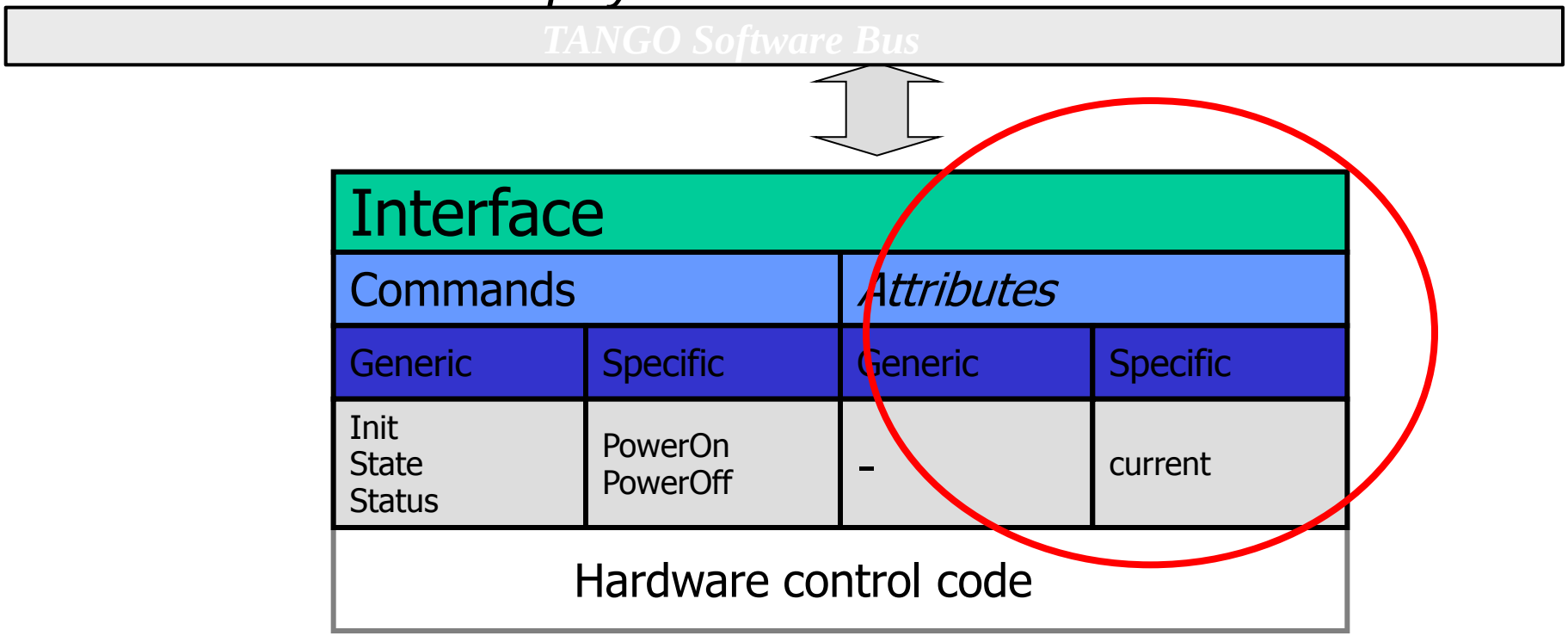
- Generic commands exist for EVERY Device
  - ✓ **Ping, Init, State, etc …**
- Specific commands
  - ✓ Are defined by the Device developer
  - ✓ Have : 0 or 1 in argument (*argin*)
  - ✓ Give : 0 or 1 out argument (*argout*)
- argin & argout = 1 of the 20 TANGO types
- Command are started from any client always in the same way :
  - ✓ call the ***command_inout*** method on the Device with the following arguments :
    - ❑ ***command name***
    - ❑ ***In parameter of the command***
    - ❑ ***Output:***
      - – ***result of command execution***
  - ✓ Exemple from Python :

```
>> my_motor_device.command_inout("AxisBackward")
>> my_motor_device.command_inout("GoToPosition",100)
```

- About argin & argout types…

TANGO	Description
DEV_VOID	no argin and/or no argout
DEV_STATE	device status
DEV_STRING	characters array
DEV_BOOLEAN	boolean
DEV_SHORT	Signed 16 bits integer
DEV_USHORT	Unsigned 16 bits integer
DEV_LONG	Signed 32 bits integer
DEV_ULONG	Unsigned 32 bits integer
DEV_FLOAT	32 bits floating point
DEV_DOUBLE	64 bits floating point
DEVVAR_CHARARRAY	table of bytes (i.e. characters)
DEVVAR_ SHORTARRAY	table of signed 16 bits integers
DEVVAR_ USHORTARRAY	table of unsigned 16 bits integers
DEVVAR_ LONGARRAY	table of signed 32 bits integers
DEVVAR_ ULONGARRAY	table of unsigned 32 bits integers
DEVVAR_ FLOATARRAY	table of 32 bits floating points

- Has a communication interface
  - ✓ interface = commands + attributes
    - commands ≈ actions
    - *attributes ≈ physical units*



- Definition
  - ✓ Physical unit produced or administrated by the device
    - ❑ *ex: motor position, tension emitted by a power supply, …*
- Format : from 0 to 2 dimensions
  - ❑ *SCALAR*
  - ❑ *SPECTRUM (i.e. vector)*
  - ❑ *IMAGE (i.e. matrix)*
- Type
  - ❑ *DEV_SHORT, DEV_LONG, DEV_DOUBLE*
    - – scalar, spectrum or image
  - ❑ *DEV_STRING*
    - – scalar only
- Accessibility
  - ✓ *READ*
    - ❑ read-only access
  - ✓ *WRITE*
    - ❑ write-only access
  - ✓ *READ_WRITE*
    - ❑ Read-and-write access
    - ❑ *set values* vs actual *value*

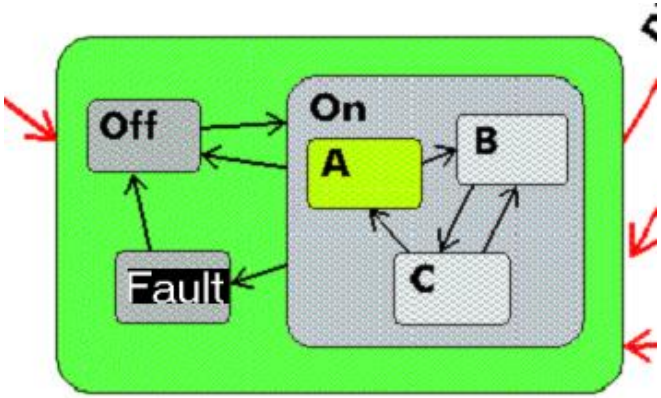
- Obtain/modify the current value of an attribute from any client ?
  - ✓ Prog. Env. OO (C++, Java, Python)
    - *dev.read_attribute(s) (attr_name_list)*
    - *dev.write_attribute(s) (attr_name_list)*

- Python Example:

```
>> x = my_motor_device.read_attribute (”position")
>> print x.value
>> x.value=100
>> my_motor_device.write_attribute (x)
```

TANGO: device: state

- Tango internally manages a state Machine
- Each Device has an associated state
  - ✓ *Device behavior = fonction (internal state)*
    - ❑ query -> internal state-> execute or error
    - ❑ internal status generated by the device developer
  - ✓ **14 predefined status**
    - ❑ ***ON, OFF, CLOSE, OPEN,***
    - ❑ ***INSERT, EXTRACT, MOVING,***
    - ❑ ***STANDBY, FAULT, INIT, RUNNING,***
    - ❑ ***ALARM, DISABLE, UNKNOWN***
    - ❑ a **color code** defined by Tango is associated to each status



- How to be more precise than one of these 14 predefined states ?
  - ✓ To inform operator of an abnormal condition
  - ✓ To give the expert a detailed diagnostic
  - ✓ Or to say that everything is OK
- Use the status information
  - ✓ which is a free text field
  - ✓ which must be managed by the device developper
- ***" Motor has reached backward limit switch"***
- ***"Power unit is in default (internal driver error 435-ER)"***
- ***"Ready to move"***

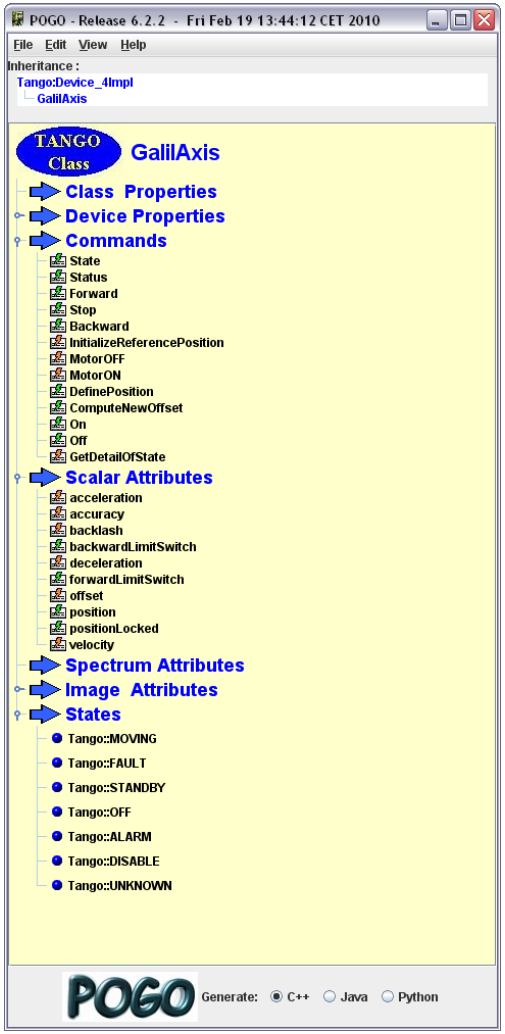
En résumé

- A tango Device is a networked object
- All Devices share the same Interface for clients applications

Device
State Status
read_attribute(s) write_attribute(s) command_inout()

Le développement d'un device Tango

Writing a Tango class



- Writing Tango device class need some glue code.
- We are using a code generator with a GUI called **POGO: P**rogram **O**bviously used to **G**enerate **O**bject
- Following some simple rules, it's possible to use it during all the device class development cycle (not only for the first generation)
- POGO generates
  - C++, java, Python Tango device class glue code
  - Makefile (C++)
  - Basic Tango device class documentation (HTML)

GalilAxis.cpp

(Unknown Scope)

```
//
//-------------------------------------------------------------------------
void GalilAxis::read_offset(Tango::Attribute &attr)
{
    DEBUG_STREAM << "GalilAxis::read_offset(Tango::Attribute &attr) entering... "<< endl;
}

//+-------------------------------------------------------------------------
//
// method :         GalilAxis::write_offset
//
// description :    Write offset attribute values to hardware.
//
//-------------------------------------------------------------------------
void GalilAxis::write_offset(Tango::WAttribute &attr)
{
    DEBUG_STREAM << "GalilAxis::write_offset(Tango::WAttribute &attr) entering... "<< endl;
}

//+-------------------------------------------------------------------------
//
// method :         GalilAxis::read_position
//
// description :    Extract real attribute values for position acquisition result.
//
//-------------------------------------------------------------------------
void GalilAxis::read_position(Tango::Attribute &attr)
{
    DEBUG_STREAM << "GalilAxis::read_position(Tango::Attribute &attr) entering... "<< endl;
}

//+-------------------------------------------------------------------------
//
// method :         GalilAxis::write_position
//
// description :    Write position attribute values to hardware.
//
//-------------------------------------------------------------------------
void GalilAxis::write_position(Tango::WAttribute &attr)
{
    DEBUG_STREAM << "GalilAxis::write_position(Tango::WAttribute &attr) entering... "<< endl;
}

//+-------------------------------------------------------------------------
```

Il reste évidemment à mettre le code nécessaire!!

Panorama des applications clientes : Applications prêtes à l'emploi

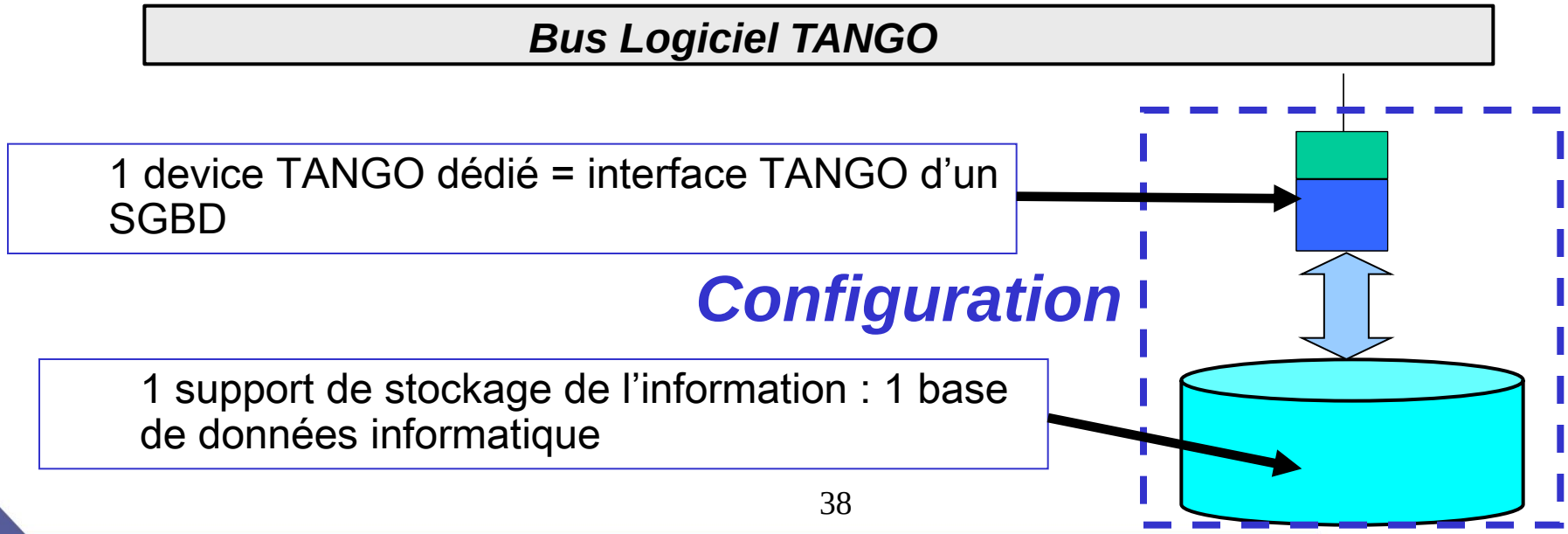
Configurer le système

Configuration : propriétés

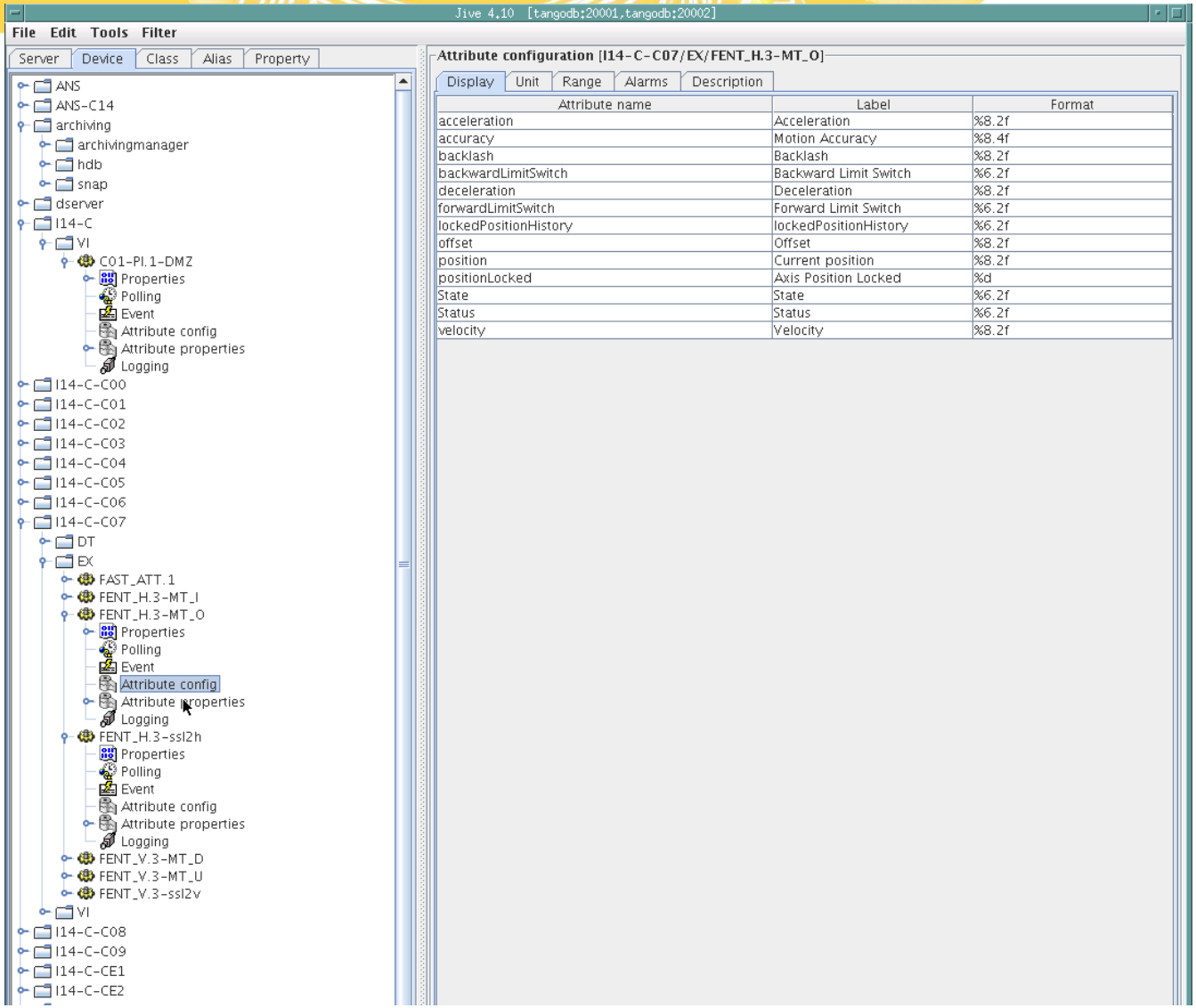
- Définition
  - ✓ données de configuration
  - ✓ concept généralisé à l'ensemble des entités TANGO
    - ❑ Device, attribut
  - ✓ propriété de ***device***
    - ❑ propre au device
    - ❑ définie(s) par le développeur
    - ❑ ex: adresse GPIB d'un périphérique
  - ✓ propriété ***d'attribut***
    - ❑ 18 propriétés TANGO prédéfinies + …
    - ❑ … propriétés définies par le développeur
    - ❑ ex: valeur initiale d'un attribut

Base des données de configuration

- élément critique du système
- source unique d'infos pour les devices et les clients



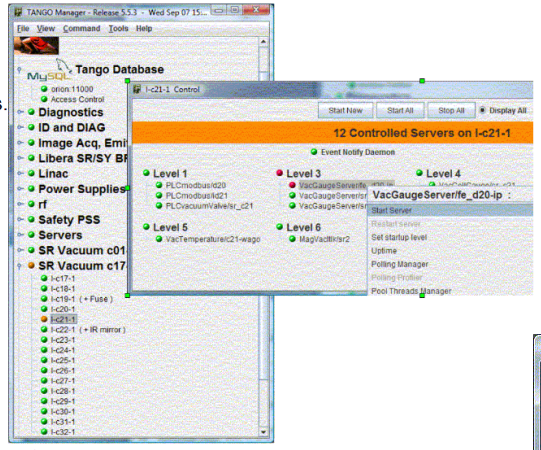
Jive : application de configuration

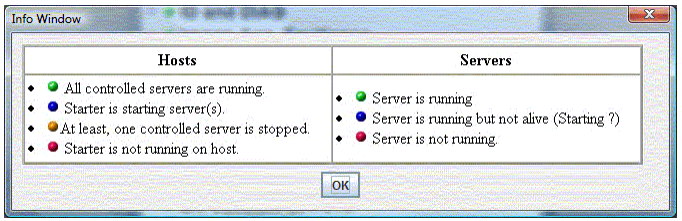


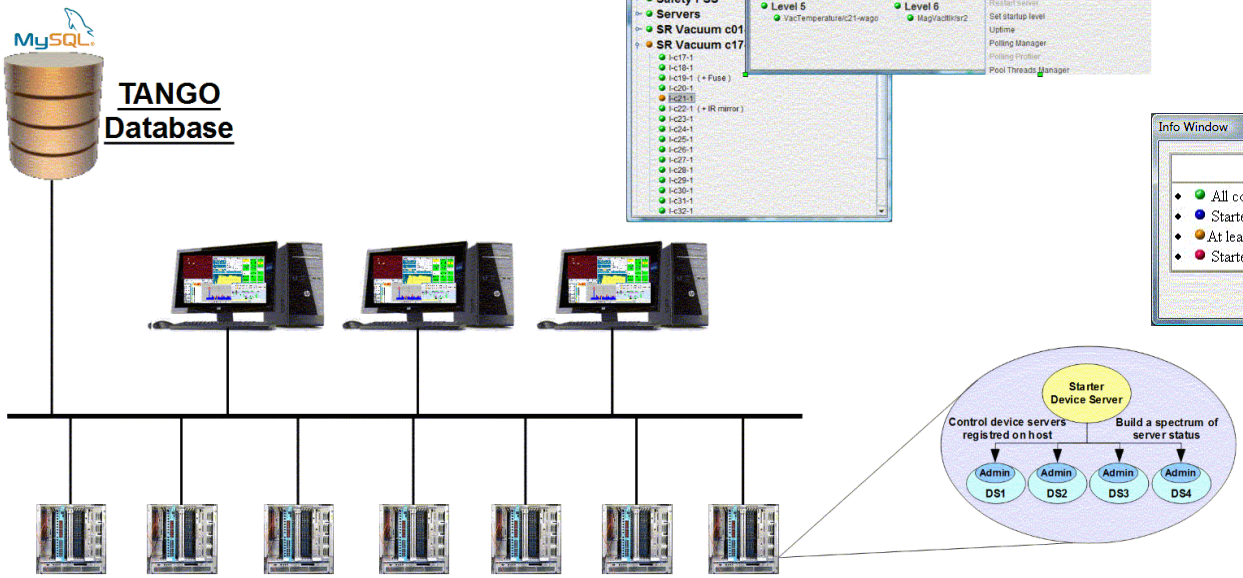
Administrer le système de contrôle

- Principle - Starter/Astor a couple to dance Tango :
  - Starter device server running on each controlled host.
  - A graphical client on all Starter devices.

Astor Client :
- display status of controlled hosts
- display information on CS and servers.
- Start/Stop servers
- ……

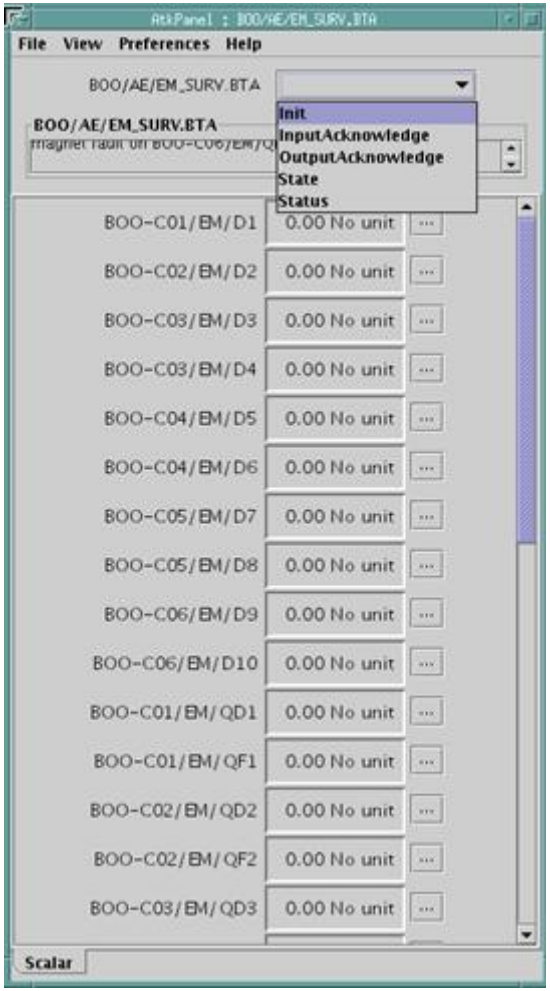






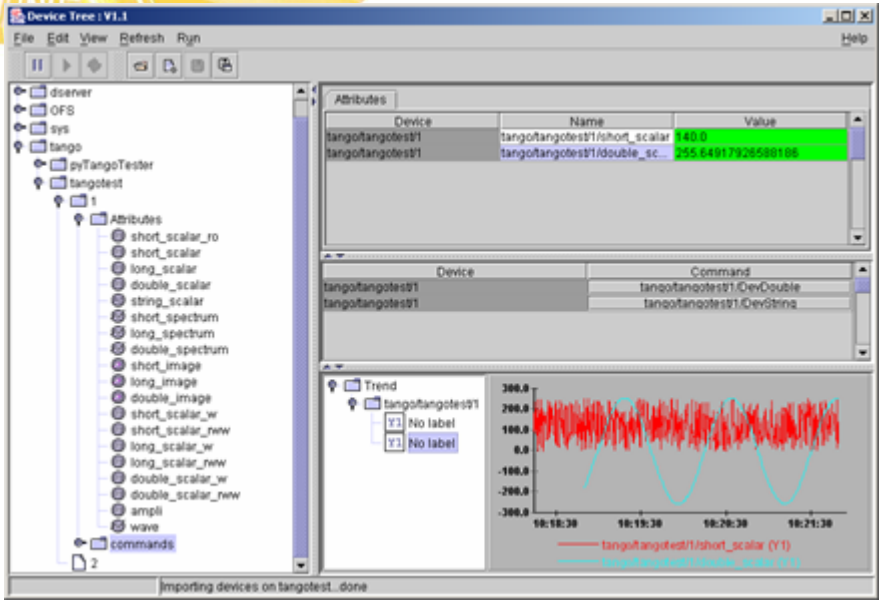
Interagir avec un Device quelconque

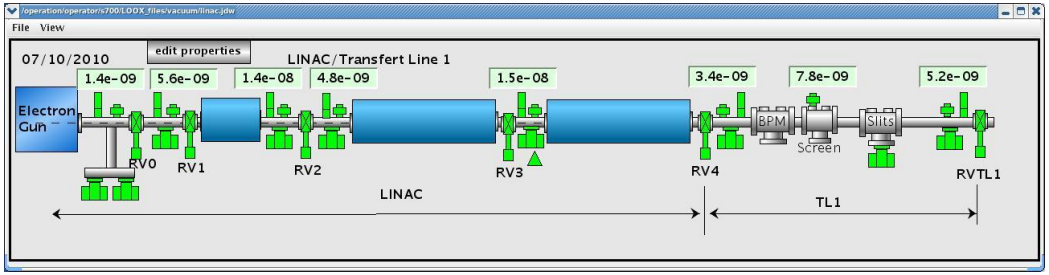
- ATKPanel
  - ✓ Permet d'agir avec 1 Device et montrer tous ses attributs et toutes ses commandes



Applications multidevice/multi attributs

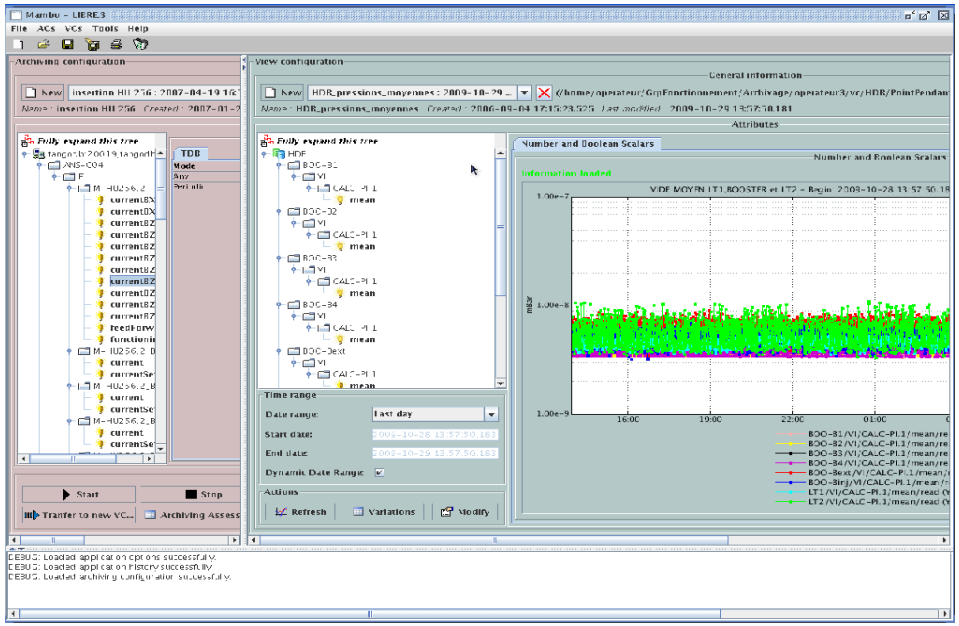
- DeviceTree permet de:
  - ✓ composer rapidement des panneaux de contrôle/simple
  - ✓ Monitorer l'évolutions des attributs en fonctions du temps
- Idem pour les applications atktables , atktuning, atktrend
- Construire un synoptique (jdraw)





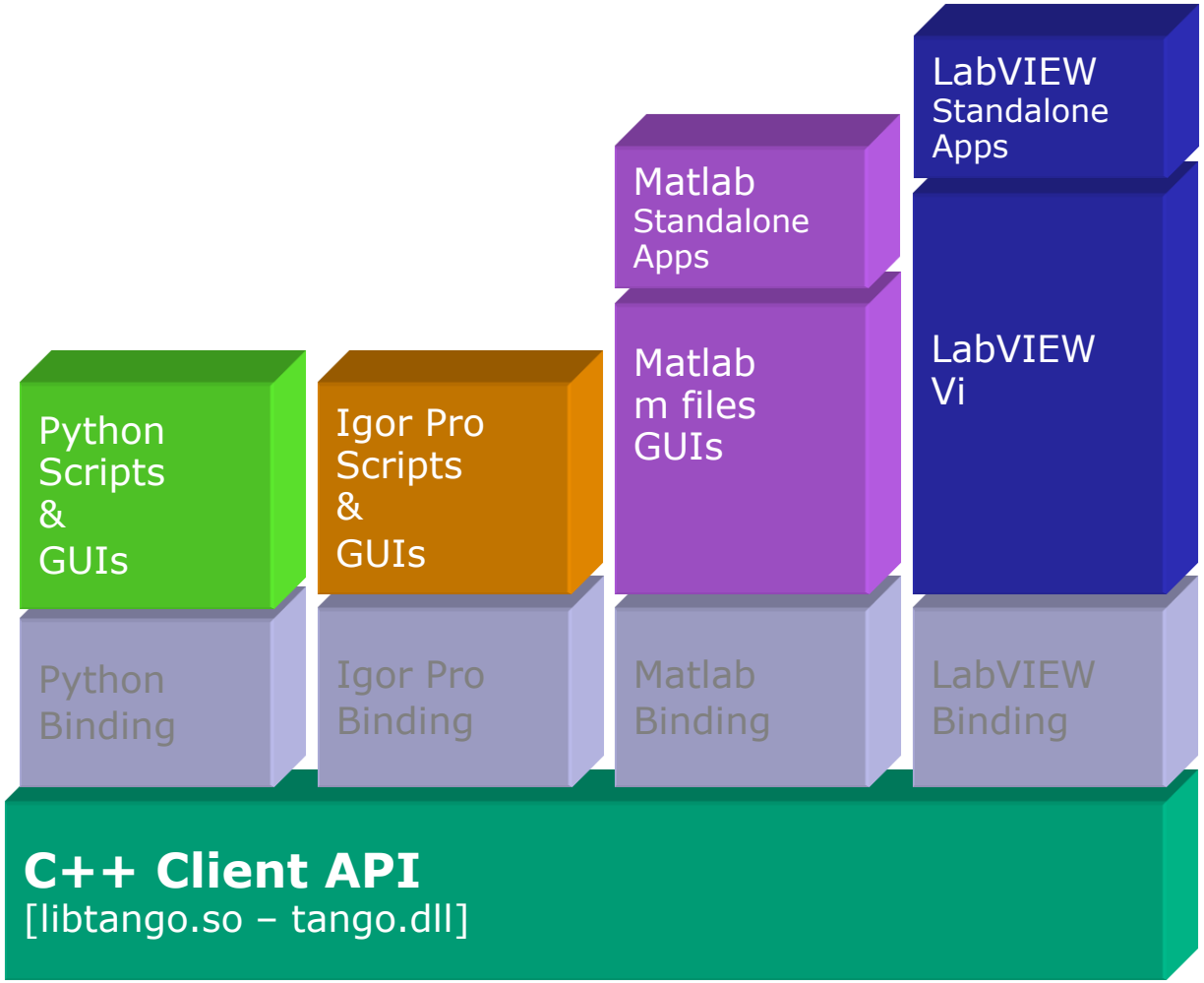
Archiver des attributs dans une base de données

- Le système d'archivage Tango a pour rôle de :
  - ✓ stocker les valeurs d'attributs dans une base de données (MySQL ou Oracle
    - ❑ Sur différents modes (période, sur critère de variation, etc .)
  - ✓ Les restituer graphiquement à l'utilisateur



Développer des applications clientes

The TANGO Client Platforms : C++ based



- Accéder à un device depuis Matlab…

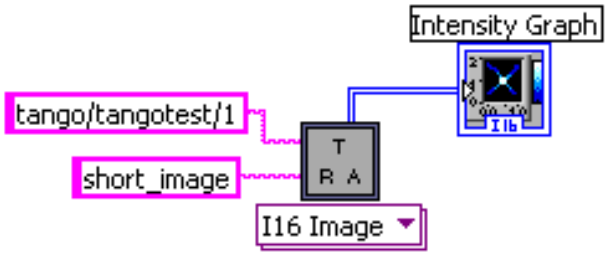
```
>> si = tango_read_attribute("tango/tangotest/1","short_image");

>> plot_image(si.value);
```

- Depuis python

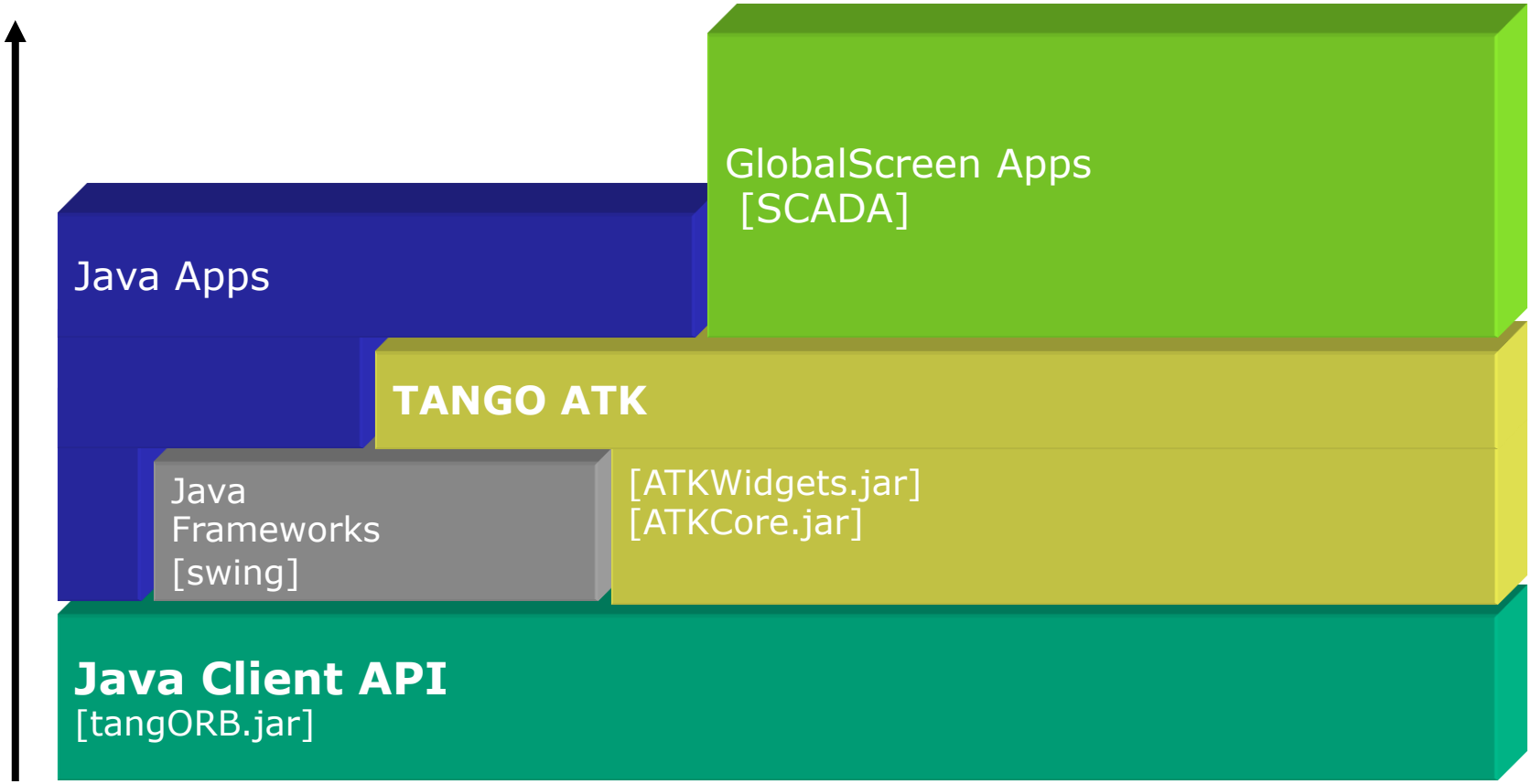
```
>> x = motor.read_attribute ("positionX")
>> print x.value
>> x.value=100
>> motor.write_attribute (x)
```

- ou LabVIEW…



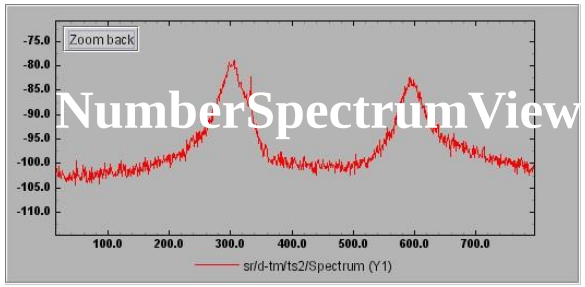
The TANGO Client Platforms

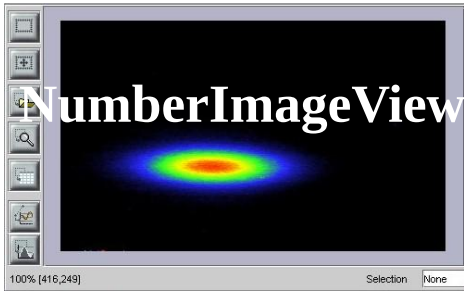
- The Java side…



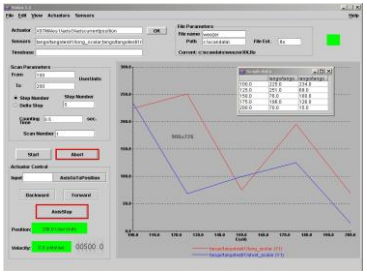
ATK Components library

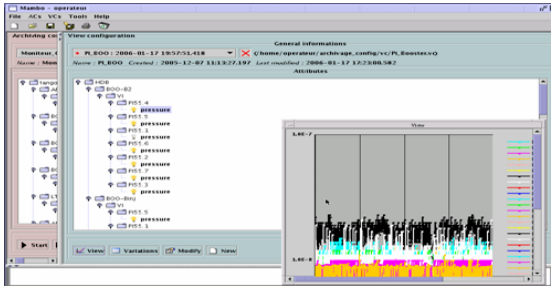
- **A Java GUI development framework**
  - ✓ Help standardize the look and feel of the applications
  - ✓ Implements the core of “any” Tango Java client



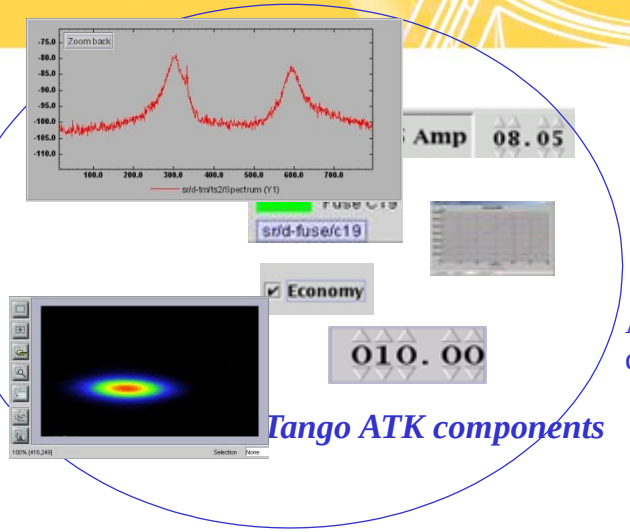


Designed for Java developers



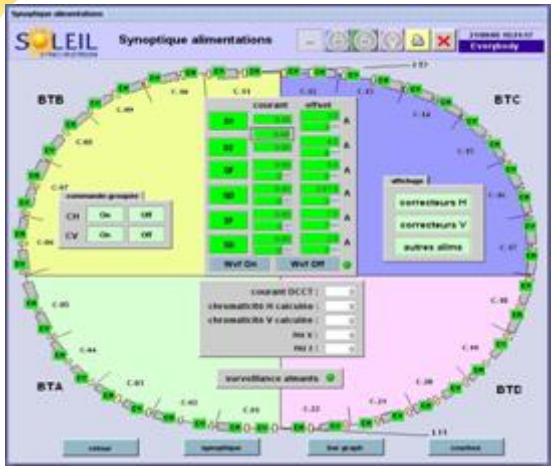


GUI development based on the GlovbalSCREEN SCADA system



Tango ATK components

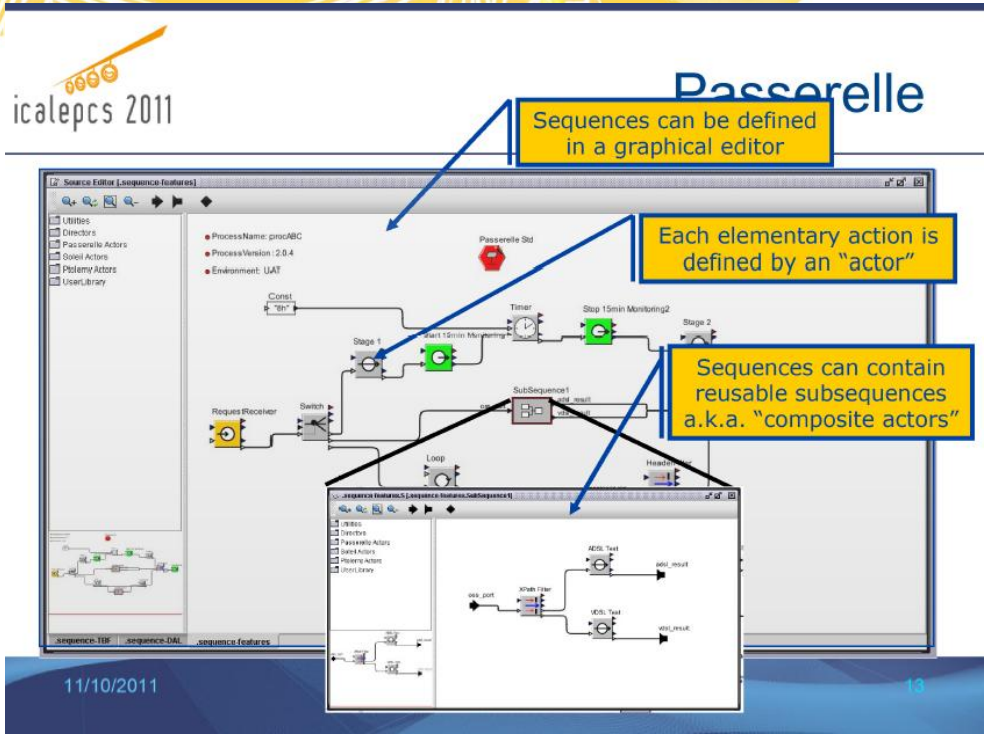
Assembly by Accelerator/Experiment divisions people with GlobalSCREEN



Booster Control Application

- Our Vision :
  - ✓ ***"To give Machine and Beamline groups the possibility to build their own GUI applications"***
  - ✓ ***"To keep a coherent look & feel for applications"***
- GlobalScreen industrial supervision software based on java
  - ✓ It allows integration of existing **TANGO ATK java** widgets by drag and drop
  - ✓ But is only used on the **presentation** layer
  - ✓ GlobalScreen provides many features :
    - ❑ access right management
    - ❑ **Remote access for maintenance**

Process Automation

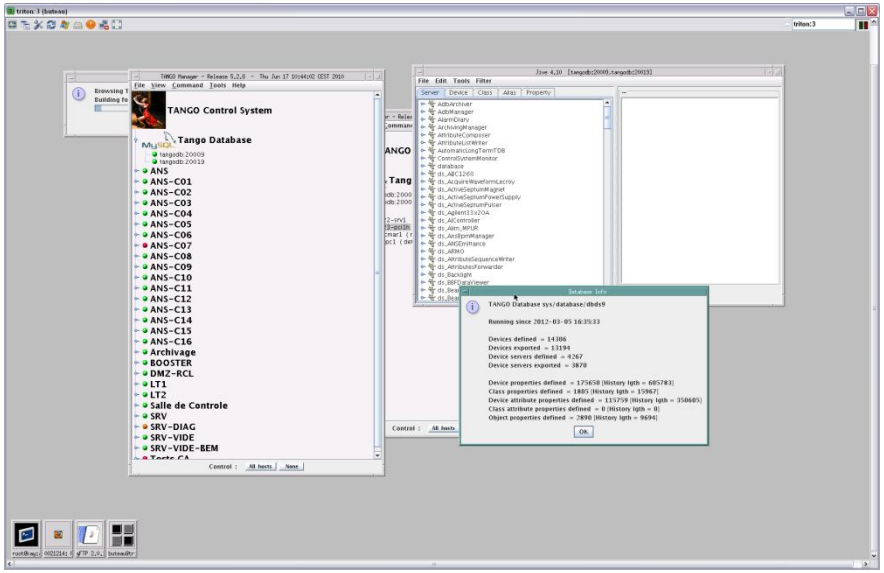


- Passerelle is a process automation tool based on the Ptolemy/Kepler framework
  - ✓ It is targeting "scientific workflows " development
- SOLEIL developed a set of Tango actors

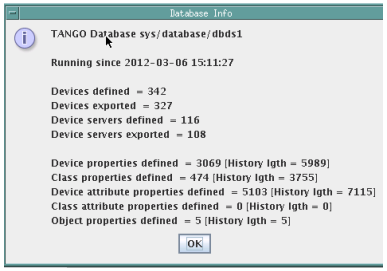
Tango@SOLEIL
Quelques chiffres sur le déploiement

Tango Devices

- Accélérateurs:



- 1 beamline



- Plus de 350 « Tango Classes »
  - développés et dans notre système de build automatique

Total Tango devices@SOLEIL: Entre 25 000 et 30 000

La collaboration TANGO
OpenSource,
Internationale,
Vivante et animée..

Collaboration ESRF-SOLEIL qui s'étend

- **Accord de coopération** pour le développement de TANGO entre **SOLEIL** et **ESRF** signé en 2002
- **ELETTRA** and **ALBA** ont rejoint la collaboration en 2004 et **DESY** in 2007
- Puis **FRM-II ,Laser MegaJoule, MAX-IV** (synchrotron suédois)
- Et plein d'autres : **LULI, ANKA, CEA –LIONS, etc ..**
- Beaucoup de synchrotrons pour des raisons historiques
- Mais TANGO n'est pas un framework orienté "métier synchrotron"

TANGO Collaboration

SOURCEFORGE.net

SOLEIL SYNCHROTRON — 2002

ESRF

elettra Synchrotron Trieste Laboratory — 2004

- 2/3 meetings par an
- Suivi des développements
- Choix des évolutions

- **Le socle commun: responsabilité des instituts par modules**
  - Noyau Tango C++ ESRF
  - Noyau Tango Java SOLEIL/ESRF
  - Framework ATK ESRF/SOLEIL
  - Framework ATKWEB SOLEIL
  - Système d'archivage SOLEIL
  - Bindings MATLAB, IGOR, Labview SOLEIL
  - Binding python :ALBA
  - Serveur python : ALBA

Collaboration : prise de décision

- Un commité exécutif se réunit lors des Tango meeting
- Les votants sont les instituts engagés dans la collaboration
- Le poids lors du vote dépend du statut de l'institut
  - *Contributeur*
  - *Collaborateur*
- Les points traités à l'ordre du jour sont :
  - *Les arbitrages sur les futurs développements en fonction des ressources de développement disponibles*
  - *Les décisions sur les outils et moyens communs (dépot de code Source,etc..)*
- Un interlocuteur Tango est désigné dans chaque institut
- Le commité exécutif vote pour accepter de nouveaux entrants

Les points forts de la collaboration

- Un engagement soutenu pour la pérennisation du noyau Tango
- Une grande réactivité inter-instituts des équipes de développements pour la maintenance des modules d'intérêts communs
- Le "pink site" : http://www.tango-controls.org
  - *WEB site to download code, get documentation, search the mailing list history, read collaboration meeting minutes, …*

- La mailing liste : tango@esrf.fr
  - *très active (une dizaine de mails par semaine)*

How to dance the Tango?

- A list of all Tango classes is available with their documentation under

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

- Common interest class sources are stored on a CVS server hosted by SourceForge
  - ✓ Project name = tango-ds
  - ✓ http://sourceforge.net/projects/tango-ds/
- On y trouve par exemple :
  - ✓ *Instrumentation Technology Libera beam position monitor, EPICS link, Modbus protocol, LabView data socket interface, Fire Wire camera, ADLink boards interface, GPIB interface, miscellaneous power supply or vacuum equipment interface, contôleur d'axes DeltaTau, Galil, XPS, etc…..*
- Local class sources are stored in a local CVS repository at each institute

- You can download Tango from

  http://www.tango-controls.org/download

  - ✓ As a source package for UNIX like OS
  - ✓ As a Windows binary distribution
- Use the (quite) abundant documentation:
  - ✓ The tutorials
  - ✓ The reference documentation
  - ✓ The mailing list

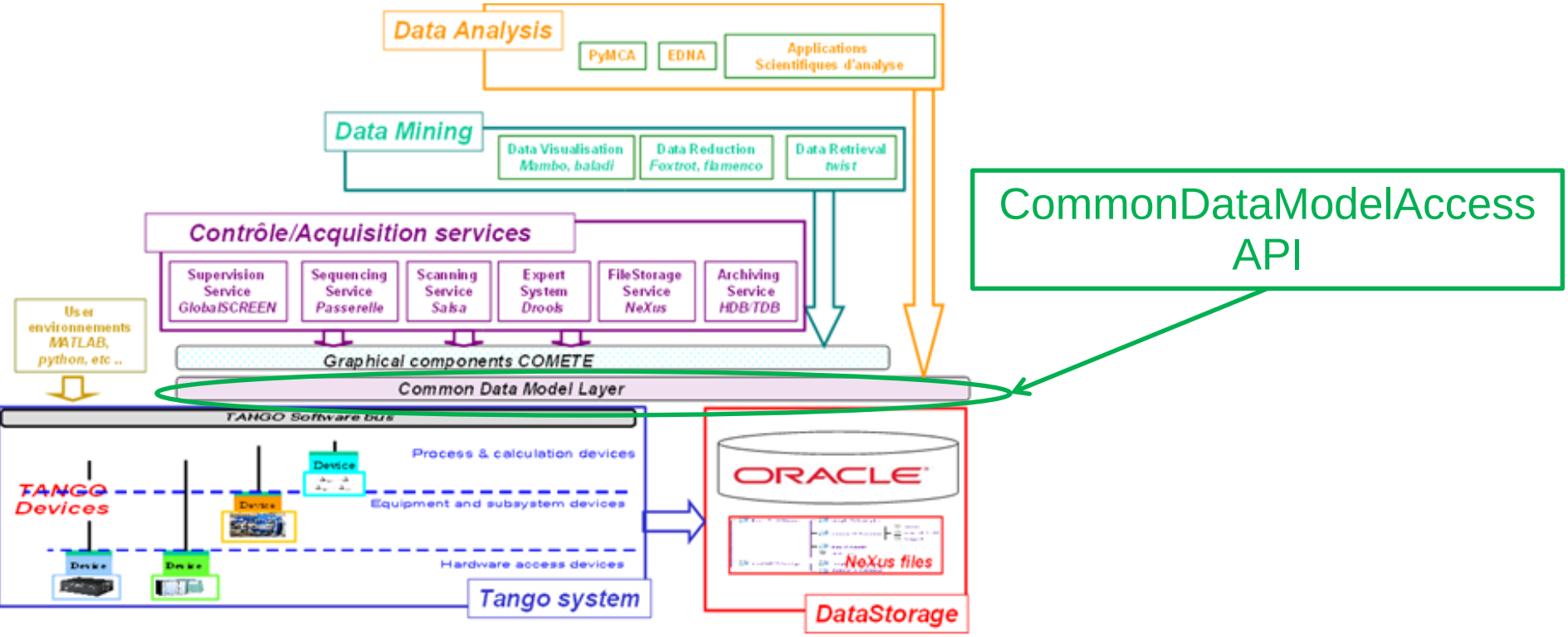
Conclusion et perspectives

L'évolution de nos architectures logicielles

- Tango est le middleware informatique de SOLEIL, vital pour son fonctionnement quotidien
  - ✓ *Le noyau est tehcniquement mature et fonctionne 24H/24 à une grande échelle de déploiement*
  - ✓ *SOLEIL participe et suit les évolutions du noyau (1 upgrade majeur tous les 18 mois)*
- Nos architectures logicielles évoluent pour prendre en compte le challenge que pose "l'avalanche de données"
  - ✓ *Les systèmes d'acquisition en continu sur des détecteurs 2D vont produire des PetaOctets par an et par ligne de lumière*

L'évolution de nos architectures logicielles

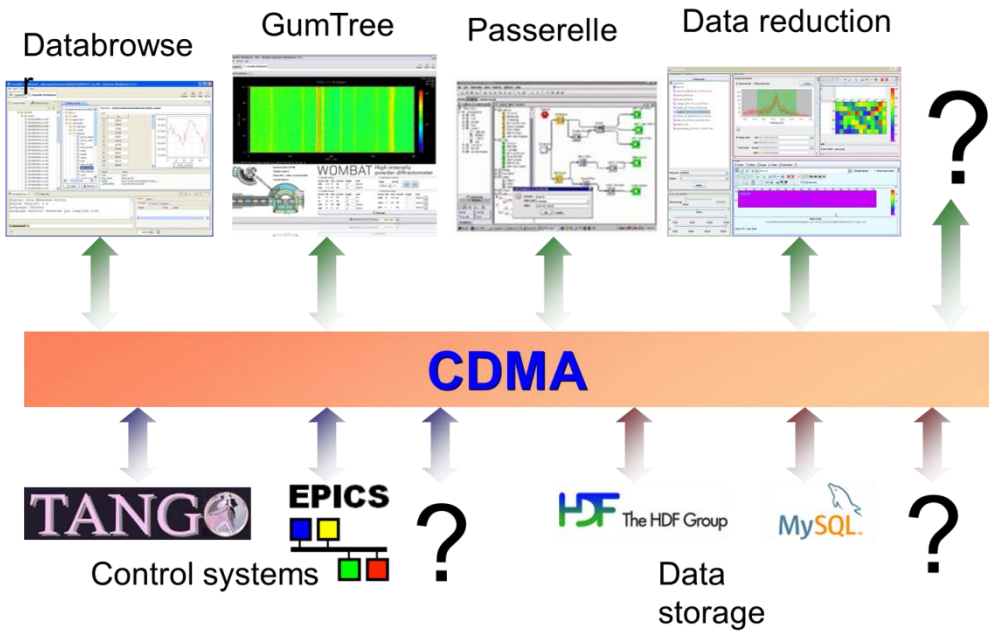
- Découpler les applications de haut niveau de la source des données :
  - ✓ Tango , archivage Tango, fichiers de mesure (HDF5)
- En unifiant les différentes API via le projet CommonDataModelAPI



L'évolution de nos architectures logicielles

- Peut être un prochain webinaire sur le CommonDataModelAccess ?
- D'ici la : le projet (en commun avec ANSTO,DESY) est ici :

https://groups.google.com/group/common-data-model?msg=subscribe&hl=fr



Questions…