Dynamic Control Systems

Advantages, Challenges, Strategies
Topics

- Dynamic Control Systems
- Dynamic Devices and UI's
- Users as Developers
- Scaling
- Tools and strategies
Dynamic Control Systems

dynamic adjective
dy·nam·ic | \dī-'na-mik \n
Definition of dynamic (Entry 1 of 2)

1a: marked by usually continuous and productive activity or change
   a dynamic city
 b: ENERGETIC, FORCEFUL
   a dynamic personality

2 or less commonly dynamical \dī-'na-mi-kəl \n   a: of or relating to physical force or energy
   b: of or relating to dynamics (see DYNAMICS)

3 of random-access memory: requiring periodic table refreshment of charge in order to retain data

https://www.merriam-webster.com/dictionary/dynamic
Dynamic Control Systems

ALBA is a Synchrotron in its 8th year of operation, with 8+3 beamlines. Current changes and upgrades in our control system are:
Dynamic Control Systems

ALBA is a Synchrotron in its 8th year of operation, with 8+3 beamlines. Current changes and upgrades in our control system are:

- Adding or removing elements from the machine:
  - Adding new Insertion Devices / removing gauges or ion pumps
Dynamic Control Systems

ALBA is a Synchrotron in its 8th year of operation, with 8+3 beamlines. Current changes and upgrades in our control system are:

- Adding or removing elements from the machine:
  - Adding new Insertion Devices / removing gauges or ion pumps
- Expanding an existing installation:
  - Adding a new beamline or an experimental station in an existing beamline
Dynamic Control Systems

ALBA is a Synchrotron in its 8th year of operation, with 8+3 beamlines. Current changes and upgrades in our control system are:

- Adding or removing elements from the machine:
  - Adding new Insertion Devices / removing gauges or ion pumps
- Expanding an existing installation:
  - Adding a new beamline or an experimental station in an existing beamline
- Controlling an existing system in a completely new way:
  - Replace RF control by FPGAs / RF IOT plants by Solid State Amps
Dynamic Control Systems

How we deal with these changes in the Control System:

- Once new Hardware is available for testing, it's introduced in our Cabling and Equipment databases.

- Control Engineers develop TANGO Devices to access the new hardware.

- If needed, High level devices are adapted to User needs, using Python Dynamic devices, like Processor, Composer or Façade Devices.

- Old user interfaces are modified by Control Engineers, while new UI's are developed by users using the Taurus GUI framework.
Tools: Python

Main programming language in TANGO community

Provided to scientists to replace matlab UI's and scripts.

Dynamic Typed Language

Functional programming

Compiled on runtime

SciPy / Science ecosystem

Modules are mutable objects

```
eval("import my_code")!
```
Tools: Taurus

Python + Qt
Windows/Linux
Core + Library of widgets
Tango / Epics / Sardana
Design by wizard / drag & drop
Few/none lines of code
Tools: Dynamic Devices in TANGO

TANGO Control System can be adapted to change in different ways:

- **Dynamic Devices** create Attributes depending on Hardware channels (DAQ)
- **Processor Devices** allow to add/modify Attributes using existing attributes in Python formulas (PLC's)
- **Composer Devices** provide Attributes evaluated from formulas that access Attributes from other Devices (Alarm System)
- **Dynamic User Interfaces** load devices and create widgets on runtime to visualize all available information.
Device Attributes created on-the-fly
With Fixed or Variable number of attributes

User Interfaces generated on-the-fly
Composer Devices

SR01/VC/ALL

State: MOVING

Property Name
SR01/VC/IP.*
SR01/VC/IPCT.*
SR01/VC/VGCT.*
SR01/VC/SPBX.*
SR01/VC/CAG.*
SR01/VC/PIR.*
SR01/VC/EPS.*
SR01/VC/SPN.*
SR01/VC/SPN.*
SR01/VC/RGA.*

3 DevicesList

AveragePressure
SR01: 1.87e-10 mbar
CCG-01: 2.90e-10 mbar
CCG-02: 2.70e-10 mbar
CCG-03: 1.00e-12 mbar
PNV-01: 2.50e-10 mbar
SPN-01: 

4 DynamicAttributes

[CCGPressures=||lambda p,d=dev.|=DEVICES,x=XAttr.pp=PIRPressures,f=|lambda c:
PIRPressures=x[XAttr(dev+)'Pressure':1e+12] for dev in DEVICES if 'pri' in dev]
IfPressure=DevVarDoubleArray(XAttr(dev+)'Pressure') or 1e-12 for dev in DEVICES
MaxPressure=DevDouble(max(CCGPressures))
AveragePressure=DevDouble(sum(CCGPressures)/len(CCGPressures))
SR01_VC_CCG-01_Pressure=DevDouble(CCGPressures[0])
SR01_VC_CCG-02_Pressure=DevDouble(CCGPressures[1])
SR01_VC_CCG-03_Pressure=DevDouble(CCGPressures[2])
ThermoNames=DevVarStringArray(sorted([%s,%2.1f]%(a,XAttr[‘SR01d’]/‘EPCS-PLC’
Thermocouples=DevVarDoubleArray(float(s.split(‘:’))[1]) if float(s.split(‘:’))[1]~3000
CCGAxis=DevVarDoubleArray([[1+2*i]*float(len(Thermocouples))/24]len(CCGPres
PAxis=DevVarDoubleArray([[1+2*i]*float(len(Thermocouples))/24]len(IFPressures -1)) for
ThermoAxis=DevVarDoubleArray(range(len(Thermocouples))))

5 DynamicCommands

#Write here your Command formulas

6 IgnoreList

*RGA*
*MBS* 
*SERIAL*
*SPBX*
*PIR*
*IP*

7 KeepAttributes

yes

8 KeepTime

1000
Composer Devices

Sergi Rubio Manrique, ALBA Synchrotron
User Interfaces generated on-the-fly

Either from searches or alarm formulas
Users as developers: codeless applications

Operators develop their own app using drag and drop widgets.

Thus freeing control engineers for harder tasks

But, apps are stored locally:

- Hard to keep track of versions
- Hard to test after upgrades
- Solved enforcing a common shared folder for tools (with snapshots) and **git**
Scaling issues

Increase of cpu/memory problems on the client side

- huge applications generated automatically
- too many connections, high cpu/memory usage

cpu/memory problems on server side, caused by clients

- timeouts
- excessive polling, too many clients accessing the same devices
- deadlocks by interrupted commands (uncoherent state)
Users as developers: Dynamic Attributes

Many users are experimented programmers, although not worried about performance or scaling issues.

To allow them to write code freely, we provide Dynamic Devices for calculations or building prototypes.

Calculations results become available in Control System tools (User Interfaces, Archiving, Alarms).

<table>
<thead>
<tr>
<th>Property name</th>
<th>Value</th>
</tr>
</thead>
</table>
| DynamicAttributes   | TIMES=DevVarDoubleArray([b.time.tv_sec+b.time.tv_usec*1e-6 for b in VAR('BUFF')])  
                      | VALS=DevVarDoubleArray([b.value for b in VAR('BUFF')])  
                      | AVG=numpy.average(VALS)  
                      | DI=abs(VALS[-1]-VALS[0])  
                      | DT=TIMES[-1]-TIMES[0]  
                      | Lifetime=ATTR('AVG') * ATTR('DT') / ATTR('DI') / 3600.0 if (DI > 0 and AVG > 0) else 0.0  
                      | LifeAvg=numpy.average(VAR('LT',VAR('LT')[-10:]))  
                      | Current=ATTR('VALS')[-1]  
                      | Product=Lifetime*Current |
| DynamicCommands     | GET_BUFF=str(VAR('BUFF'),XDEV('sr/di/dcct').attribute_history('averagecurrent',10))  
                      | ADD_LT=str(VAR('LT'),default=[]).append(ATTR('Lifetime')) |
| SubDevices          | sr/di/dcct |
Users as Developers: Controlled Scripts

Running an script within a TANGO DynamicDS Device helps us to "Control" how the script is executed

<table>
<thead>
<tr>
<th>Device properties [SR06/CT/CALC-A]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Property name</td>
<td>Value</td>
</tr>
<tr>
<td>B</td>
<td>2.701</td>
</tr>
<tr>
<td>CheckDependencies</td>
<td>True</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>450</td>
</tr>
<tr>
<td>ExtraModules</td>
<td>PyTangoArchiving.Reader as HDB as oriol</td>
</tr>
<tr>
<td>IgnoreList</td>
<td></td>
</tr>
<tr>
<td>KeepAttributes</td>
<td>True</td>
</tr>
<tr>
<td>KeepTime</td>
<td>2000</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>/control/user-scripts/composers/sr_rf_calc_common.py</td>
</tr>
<tr>
<td>LogLevel</td>
<td>WARNING</td>
</tr>
<tr>
<td>Machine_Pforw</td>
<td>PLANT</td>
</tr>
<tr>
<td>PollingCycle</td>
<td>1000</td>
</tr>
<tr>
<td>SortLists</td>
<td>False</td>
</tr>
<tr>
<td>UseEvents</td>
<td>False</td>
</tr>
<tr>
<td>UseTaurus</td>
<td>False</td>
</tr>
</tbody>
</table>
Users as Developers: Controlled Scripts

Running an script within a TANGO DynamicDS Device helps us to "Control" how the script is executed

# Flag to control if the IDs are open. True when open; False when closed
idsOpen = bool(VAR(idsOpen,VALUE) if WRITE else VAR(idsOpen) or False)
U = Uo if idsOpen else Uc

machinebeam = XATTR(srdi/dcct/averagecurrent)
userBeam = float(VAR('UserCurrent',VALUE) if WRITE else VAR('UserCurrent',default=130))
setUsrBeam = bool(VAR('usrBeam',VALUE) if WRITE else VAR('usrBeam', default=False))

Scripts versions can be managed by git
Scaling issues

Example of unexpected performance issues in the TANGO Database:
Scaling issues

Example of unexpected performance issues in the TANGO Database:

- Archiving the CPU usage of the TANGO Database, we correlated the problem to shutdowns and machine testing periods.
Scaling issues

Example of unexpected performance issues in the TANGO Database:

- Archiving the CPU usage of the TANGO Database, we correlated the problem to shutdowns and machine testing periods.
- No long query was producing the issue (typical search for removed devices/attributes).
Scaling issues

Example of unexpected performance issues in the TANGO Database:

- Archiving the CPU usage of the TANGO Database, we correlated the problem to shutdowns and machine testing periods.
- No long query was producing the issue (typical search for removed devices/attributes).
- Analyzing archived data with HDB++, we linked it to devices accessed by Matlab scripts (corrector magnets) during tests.
Scaling issues

Example of unexpected performance issues in the TANGO Database:

- Archiving the CPU usage of the TANGO Database, we correlated the problem to shutdowns and machine testing periods.
- No long query was producing the issue (typical search for removed devices/attributes).
- Analyzing archived data with HDB++, we linked it to devices accessed by Matlab scripts (corrector magnets) during tests.
- Finally, we found that, in the lack of settings, there were 88 devices checking their event configuration at 10Hz, doing short and fast queries but frequent and constant.
Tools: Process Profiler device

ProcessProfiler device provides cpu/ram stats.

Stats are provided as TANGO attributes, so it can be archived or used as alarm triggers.

It not only analyzes designed processes, but any process in the machine that takes too many resources.

Tools: PANIC Alarms
Tools: Using archiving for Diagnostics

TANGO HDB++ event-based Archiving provides diagnostic tools for evaluating the Control System Load. PyTangoArchiving API configures new archivers on demand.
Future: from Virtual Machines to Containers

Dynamic Control Systems is a problem of scaling an ever-increasing need of resources:

faster cpu's -> more data -> bigger memory usage -> slower performance

Then, as we already create Devices on-the-fly:

- Should we explore the creation of new VM's or Containers on-the-fly?
- If devices and applications run on independent containers, how do we manager their growth?
- Using containers, limits between Control Engineering and System Administration get blurry.
Summary

- Enabling dynamicity on the Control System allow to adapt faster to change.

- Enabling users as developers reduces the gap between groups:

- Scaling requires to have the proper tools:
Summary

- Enabling dynamicity on the Control System allow to adapt faster to change.
  - if data repositories (Cabling/Control Databases) are updated accordingly
  - if IT resources are continuously monitored and scaling strategies are adopted

- Enabling users as developers reduces the gap between groups:
  - but gap must be further reduced through training and best practices enforcement
  - Users must be conscious of the effect of their applications in terms of performance

- Scaling requires to have the proper tools:
  - integration of IT resources with Control System tools (Archiving/Alarms)
  - new diagnostic Attributes must be needed on devices (MemUsage, Blackbox)
Thanks/Gràcies for your attention