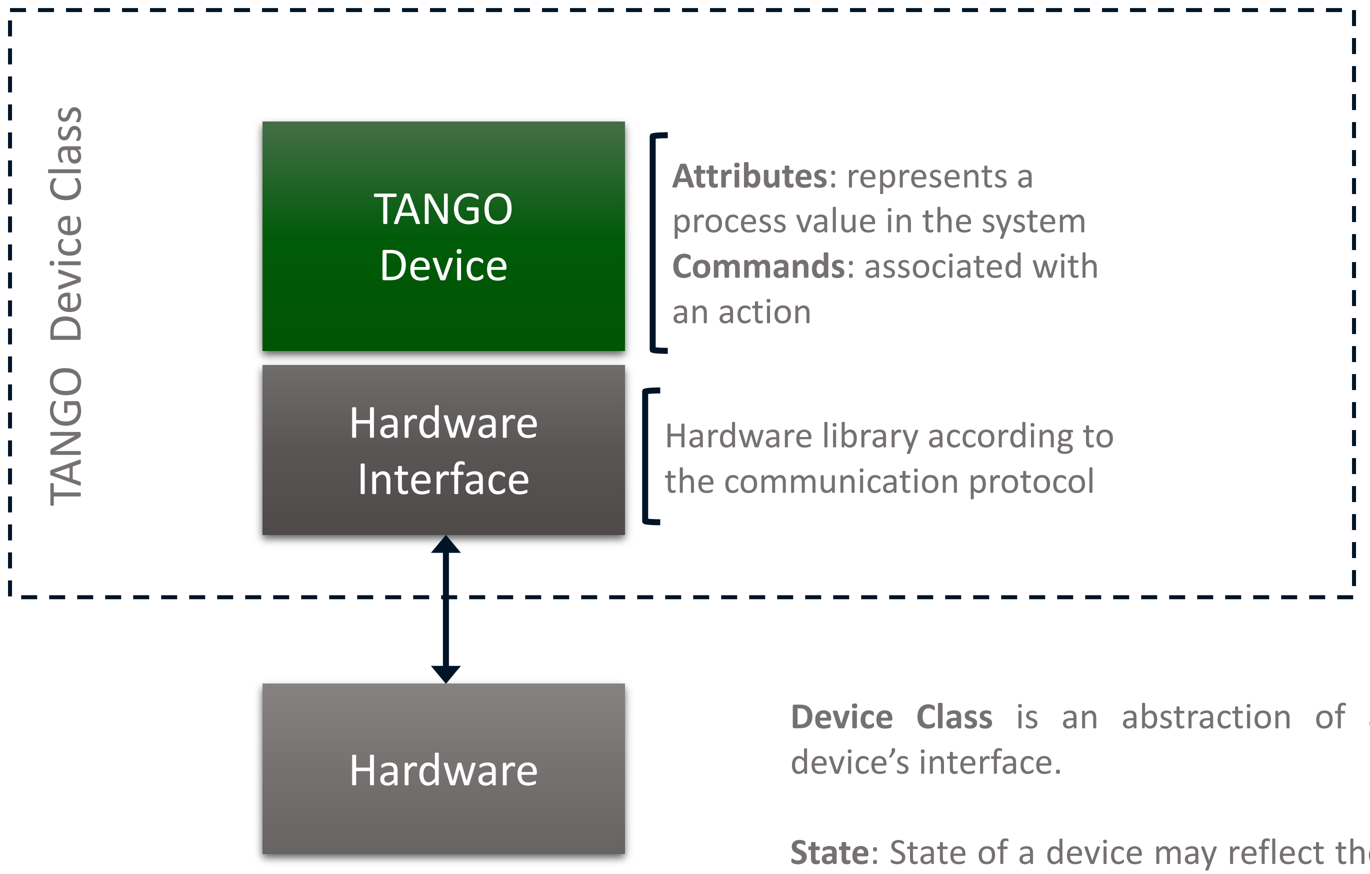
MXCuBE Code Camp
Tango and Sardana

Lund, 9-11 October 2023

Tango

- Distributed control system: bringing equipment into the network
- CORBA 😱
- Attributes, commands
- Events

https://www.tango-controls.org/



Attributes: represents a process value in the system
Commands: associated with an action

Hardware library according to the communication protocol

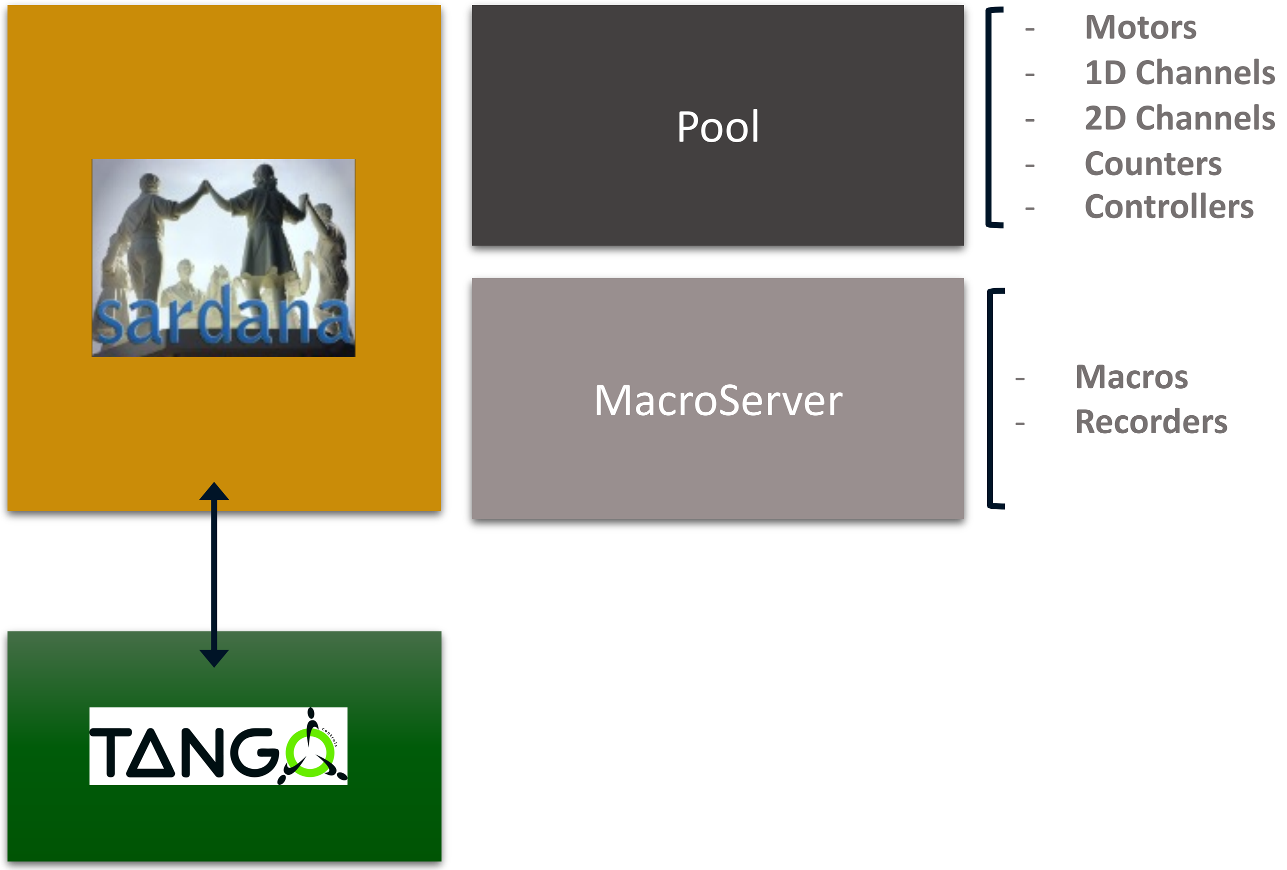
Device Class is an abstraction of a device's interface.

State: State of a device may reflect the equipment state. State machine defines available operations in different states of the device.

Sardana

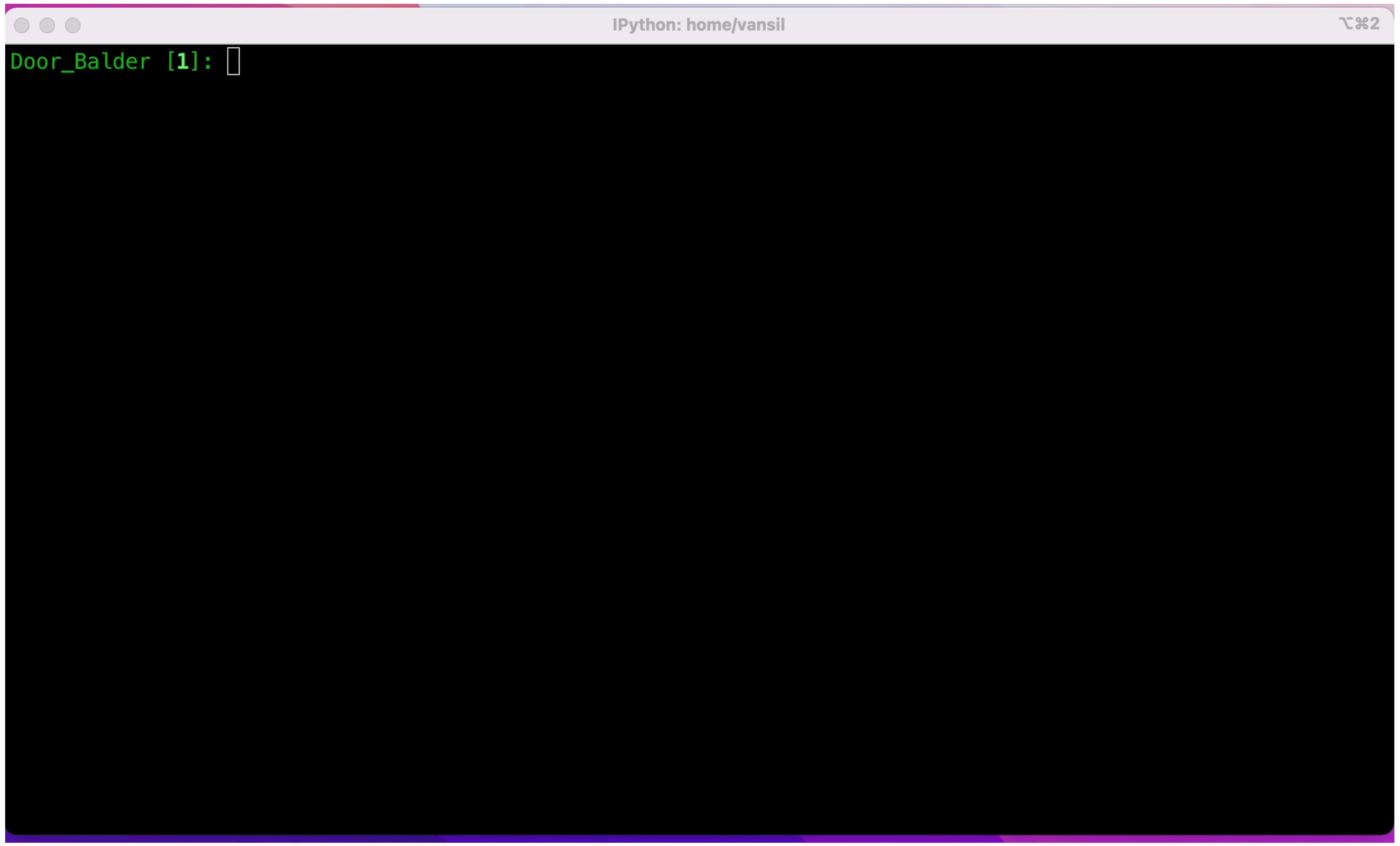
- Experiment Orchestration
- Tango based
- Macroserver runs macros
- Pool of equipment available
- Multiple languages supported

https://www.sardana-controls.org/



Sardana

- Spock is the CLI
  - Run macros, set parameters…



IPython: home/vansil ⌥⌘2

```
Door_Balder [1]: 
```

Tango in mxcube

This is not the standard tango API, but with modified internals to be able to work cooperatively with other greenlets

```
from tango.gevent import DeviceProxy

class TangoCommand(CommandObject):
      def __init__(self, name, command, tangoname=None,
                   username=None, **kwargs):


           .
           .
           .
      def __call__(self, *args, **kwargs):
           .
           self.device = DeviceProxy(self.device_name)
           tango_cmd_object = getattr(self.device, self.command)
           ret = tango_cmd_object(*args)
           .
```

e.g. Executing sys/tg_test/1/Status command

Tango in mxcube

```
from tango.gevent import DeviceProxy

class TangoChannel(ChannelObject):
      .
      .
      self.device = DeviceProxy(self.device_name)
      self.attribute_name = attribute_name
      .
      .
      .
      def get_value(self):
            self.device.read_attribute(self.attribute_name).value

      def set_value(self, new_value):
            self.device.write_attribute(
```

e.g. reading sys/tg_test/1/ampli attribute

Tango in mxcube: events

- Two options:
  - Polling:
    - Using mxcubecore.Poller callback is update()
      which emit("update", value)
  - Events:
    - Subcribing to CHANGE_EVENT events for the given attribute
    - Upon event reception
      - push_event(evt) is called
      - event data is queued and callback to same update() as before

Sardana in mxcube

```
class SardanaChannel(ChannelObject, …)
      Similar as TangoChannel … but with Taurus.Attribute

class SardanaCommand(CommandObject)
       Similar as TangoCommand … but with Taurus.Device



class SardanaMacro(CommandObject, SardanaObject, C[illegible]nelObje
      self.door = Device(self.doorname)

      def __call__(self, *args, **kwargs):
            # formatting the command
            fullcmd = self.macro_format + args_stuff
            .
            self.door.subscribe_event("Result",CHANGE_EVENT,result_callback)
            .
            self.door.runMacro(fullcmd.split())
```

Everything in sardana is a tango device
The “door” is the tango device which runs macros

Sardana in mxcube

```
class SardanaChannel(ChannelObject, …)
      Similar as TangoChannel … but with Taurus.Attribute

class SardanaCommand(CommandObject)
       Similar as TangoCommand … but with Taurus.Device



class SardanaMacro(CommandObject, SardanaObject, ChannelObject)
      self.door = Device(self.doorname)

      def __call__(self, *args, **kwargs):
            # formatting the command
            fullcmd = self.macro_format + args_stuff
            .
            self.door.subscribe_event("Result",CHANGE_EVENT,result_callback)
            .
            self.door.runMacro(fullcmd.split())
```

```
emit("macroResultUpdated",value)
```