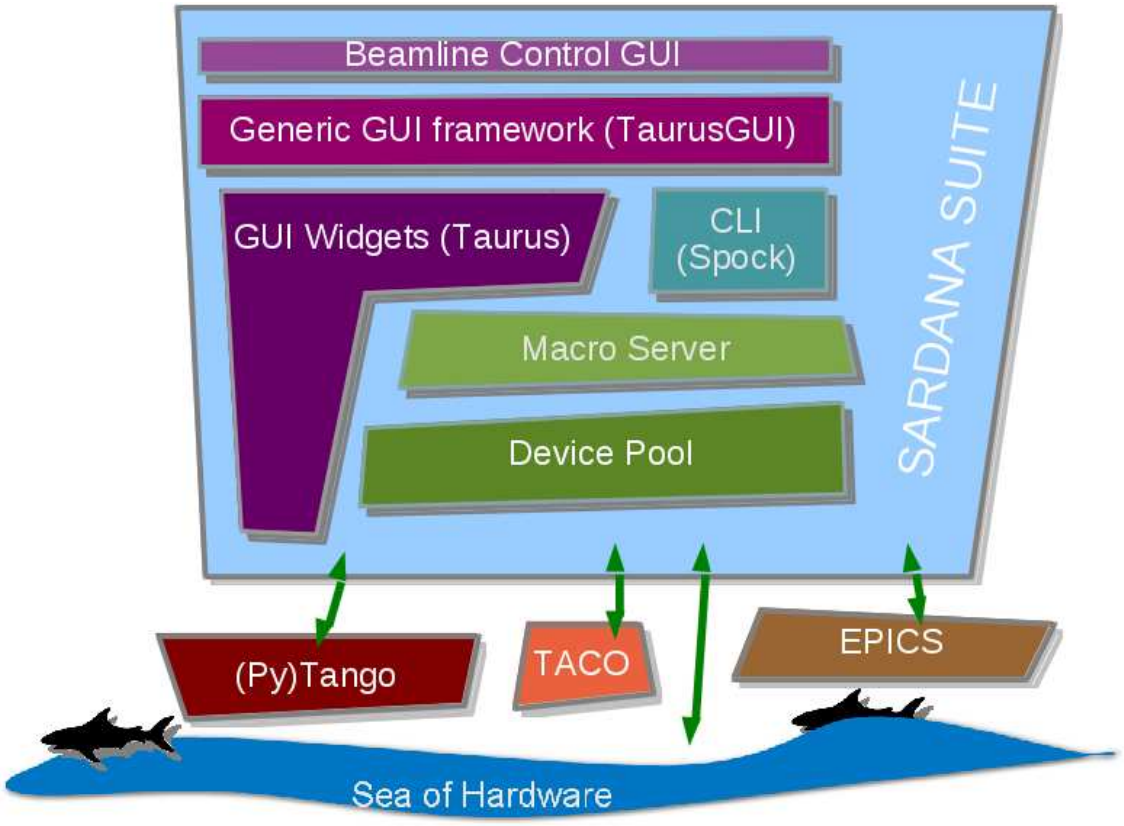


BEAMLINE CONTROL AND DATA ACQUISITION SYSTEMS AT ALBA

Guifré Cuní on behalf of the Controls Section. SSRF & ALBA Workshop
Dec 18th, 2013.

Outline

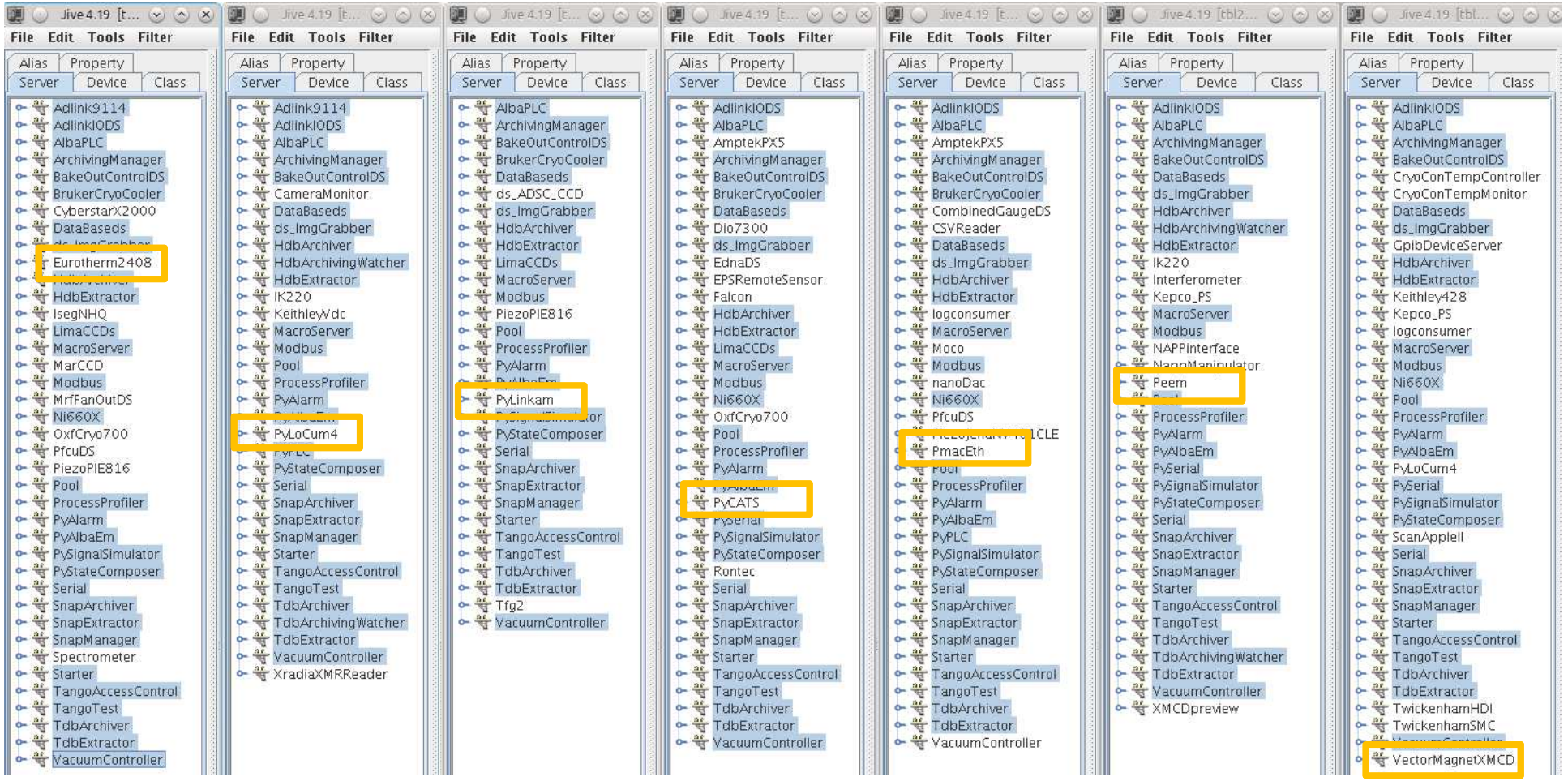
- Common Hardware Components
- Tango Device Servers
- Sardana
  - Device Pool
  - Macro Server
- Taurus
  - Command Line Interface
  - GUI Widgets
  - Generic GUI Framework
  - Beamline Control GUI

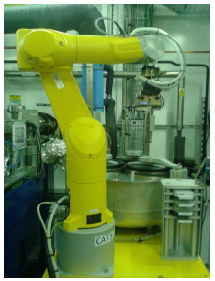
Common Hardware Components (i)

- Motion Controller
- Vacuum IP and Vacuum Gauges
- Equipment Protection System PLC
- Diagnostic Cameras, Electrometers, …
- AI, AO, DAQ (NI6602, Adlink 2005, …)
- Archiving System
- Software Alarm System
- 2 Industrial PCs + 3 Workstations + 1 Terminal

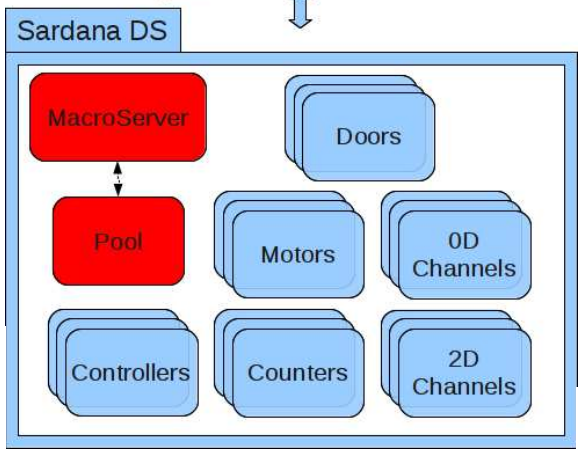
Common Hardware Components (ii)

Tango Device Servers



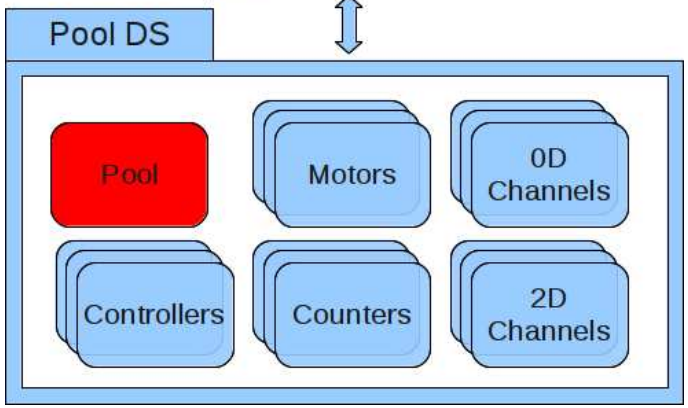


Pool: Objects related to motion control and data acquisition

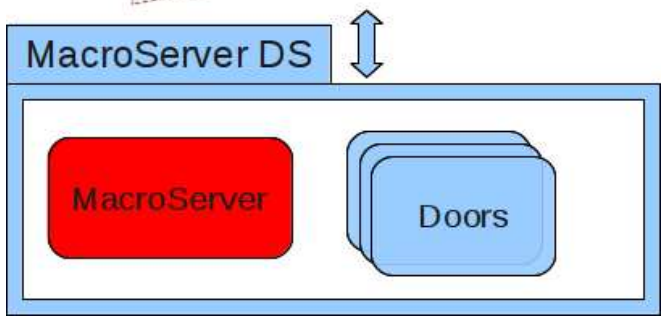


One Process

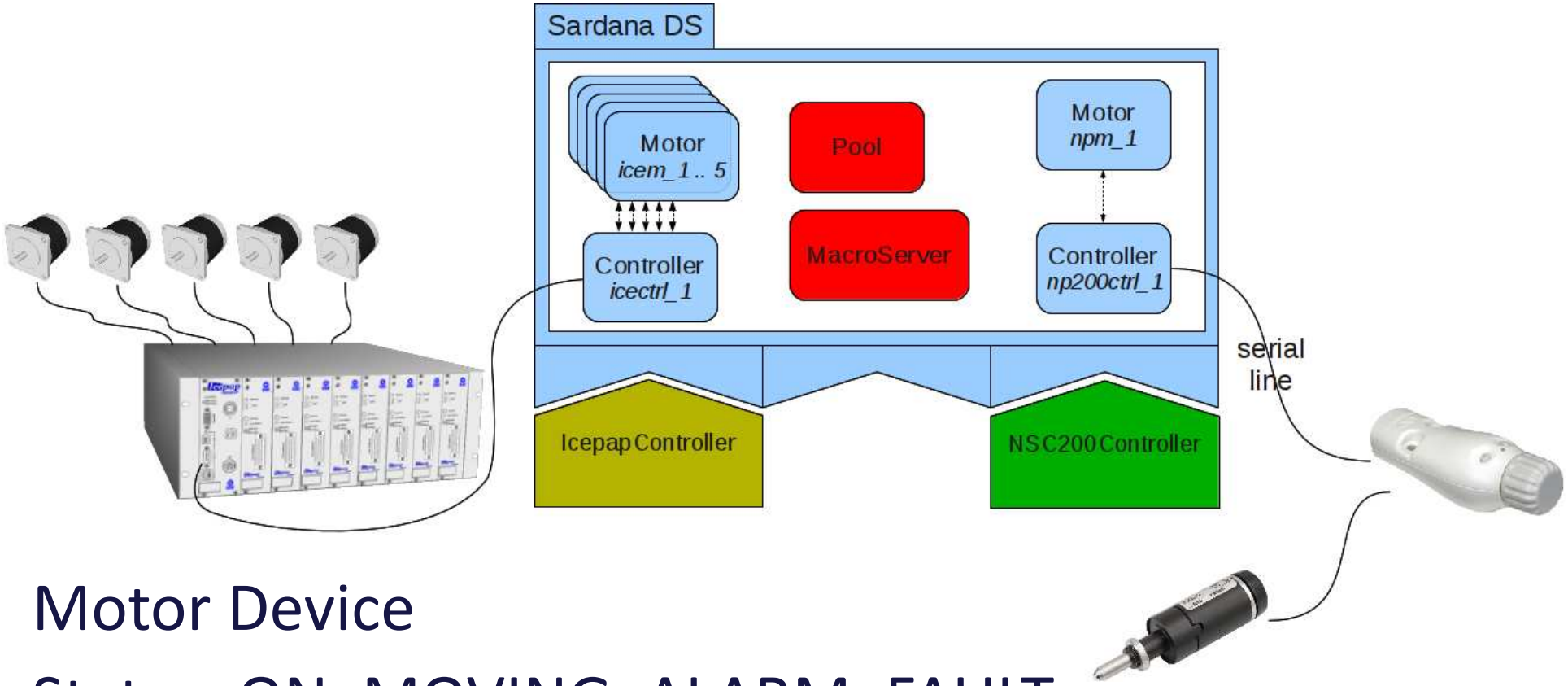
MacroServer: Manages procedures (macros) and client entry points (doors)



Two Processes



Sardana Device Pool (i)



Motor Device

States: ON, MOVING, ALARM, FAULT

Commands: Stop, Abort

Attributes: Position, Offset, Velocity, …

Sardana Device Pool (ii)

Writing a Pool Controller

- State of each Device
- Read Access
- Write Access
- Command Execution

```
class NSC200Controller(MotorController):
    """This class is the Tango Sardana motor controller for the Newport NewStep
    handheld motion controller NSC200.
    This controller communicates through a Device Pool serial communication
    channel."""

    ctrl_properties = \
        { 'SerialCh' : { Type : str,
                         Description : 'Communication channel name for the serial line' },
          'SwitchBox': { Type : bool,
                         Description : 'Using SwitchBox',
                         DefaultValue : False},
          'ControllerNumber' : { Type : int,
                                 Description : 'Controller number',
                                 DefaultValue : 1 } }

    def __init__(self, inst, props, *args, **kwargs):
        MotorController.__init__(self, inst, props, *args, **kwargs)

        self.serial = None
        self.serial_state_event_id = -1

        if self.SwitchBox:
            self.MaxDevice = 8

    def AddDevice(self, axis):
        if axis > 1 and not self.SwitchBox:
            raise Exception("Without using a Switchbox only axis 1 is allowed")

        if self.SwitchBox:
            self._setCommand("MX", axis)

    def DeleteDevice(self, axis):
        pass

    _STATE_MAP = { NSC200.MOTOR_OFF : State.Off, NSC200.MOTOR_ON : State.On,
                   NSC200.MOTOR_MOVING : State.Moving }

    def StateOne(self, axis):
        if self.SwitchBox:
            self._setCommand("MX", axis)

        status = int(self._queryCommand("TS"))
        status = self._STATE_MAP.get(status, State.Unknown
        register = int(self._queryCommand("PH"))
        lower = int(NSC200.getLimitNegative(register))
        upper = int(NSC200.getLimitPositive(register))

        switchstate = 0
        if lower == 1 and upper == 1: switchstate = 6
        elif lower == 1: switchstate = 4
        elif upper == 1: switchstate = 2
        return status, "OK", switchstate

    def ReadOne(self, axis):
        try:
            if self.SwitchBox:
                self._setCommand("MX", axis)
            return float(self._queryCommand("TP"))
        except:
            raise Exception("Error reading position, axis not available")

    def PreStartOne(self, axis, pos):
        return True

    def StartOne(self, axis, pos):
        if self.SwitchBox:
            self._setCommand("MX", axis)
        status = int(self._queryCommand("TS"))
        if status == NSC200.MOTOR_OFF:
            self._setCommand("MO","")
        self._setCommand("PA", pos)
        self._log.debug("[DONE] sending position")

    def StartAll(self):
        pass

    def AbortOne(self, axis):
        if self.SwitchBox:
            self._setCommand("MX", axis)
        self._setCommand("ST", "")
```

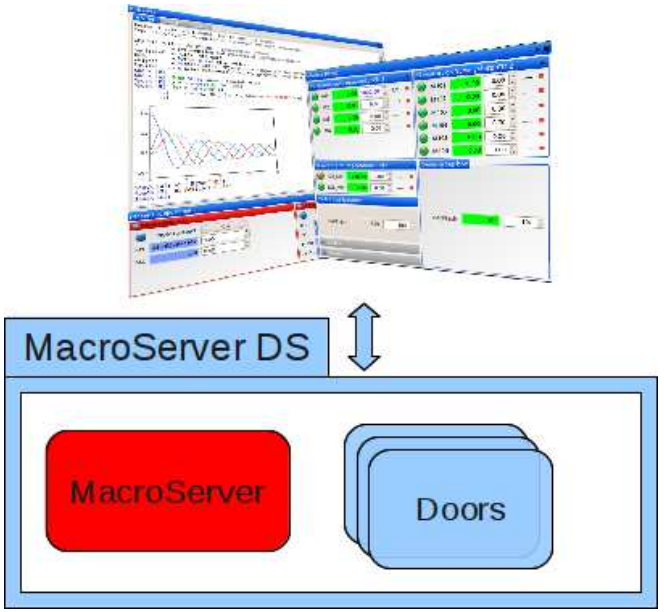
Sardana Macro Server (i)

A macro is a specific procedure that can be executed at any time

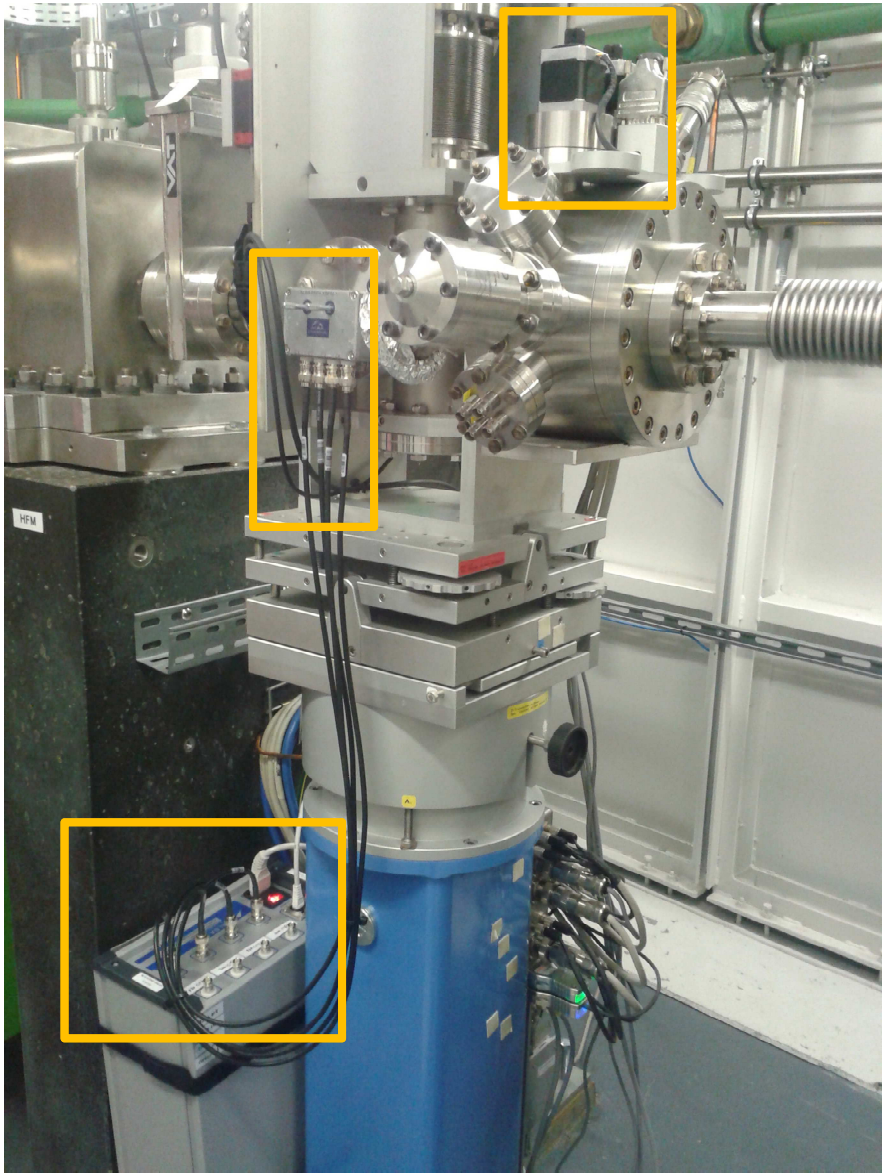
- Written in Python (function or class)
- The Door provides access for GUIs and CLIs
- Can have parameters and result
- Scientists can add macros

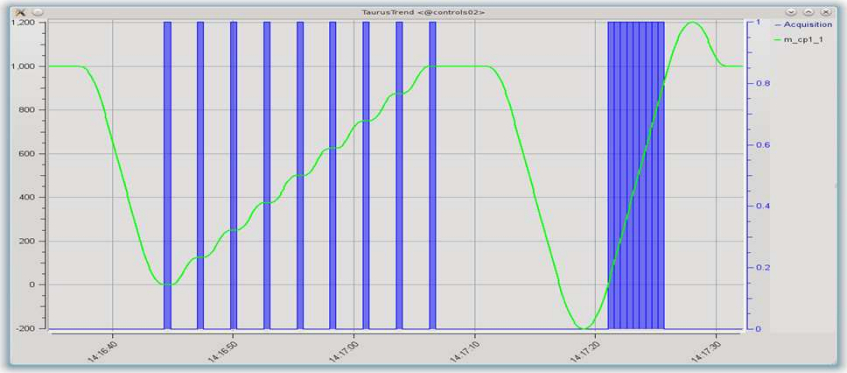
```python
from sardana.macroserver.macro import macro

@macro()
def hello_world(self):
    self.output("Hello, World!")
```

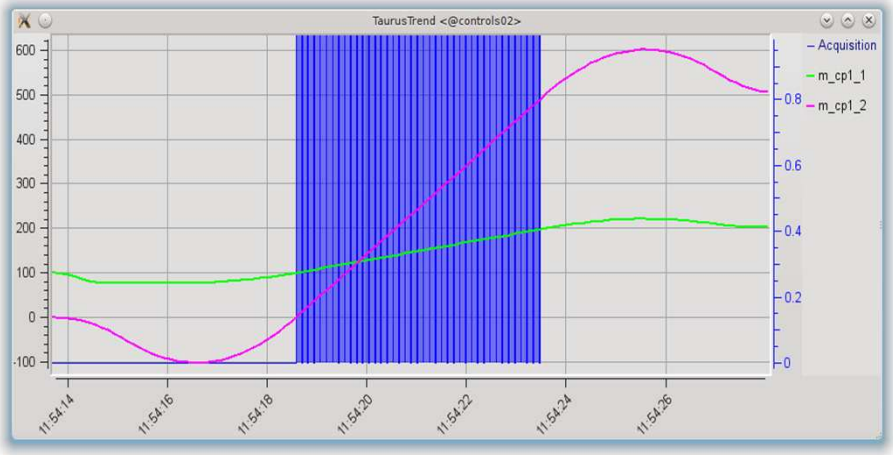
```python
from sardana.macroserver.macro import macro, Type

@macro([ ["moveable", Type.Moveable, None, "moveable to move"],
         ["position", Type.Float, None, "absolute position"] ])
def my_move(self, moveable, position):
    """This macro moves a moveable to the specified position"""

    moveable.move(position)
    self.output("%s is now at %s", moveable, moveable.getPosition())
```



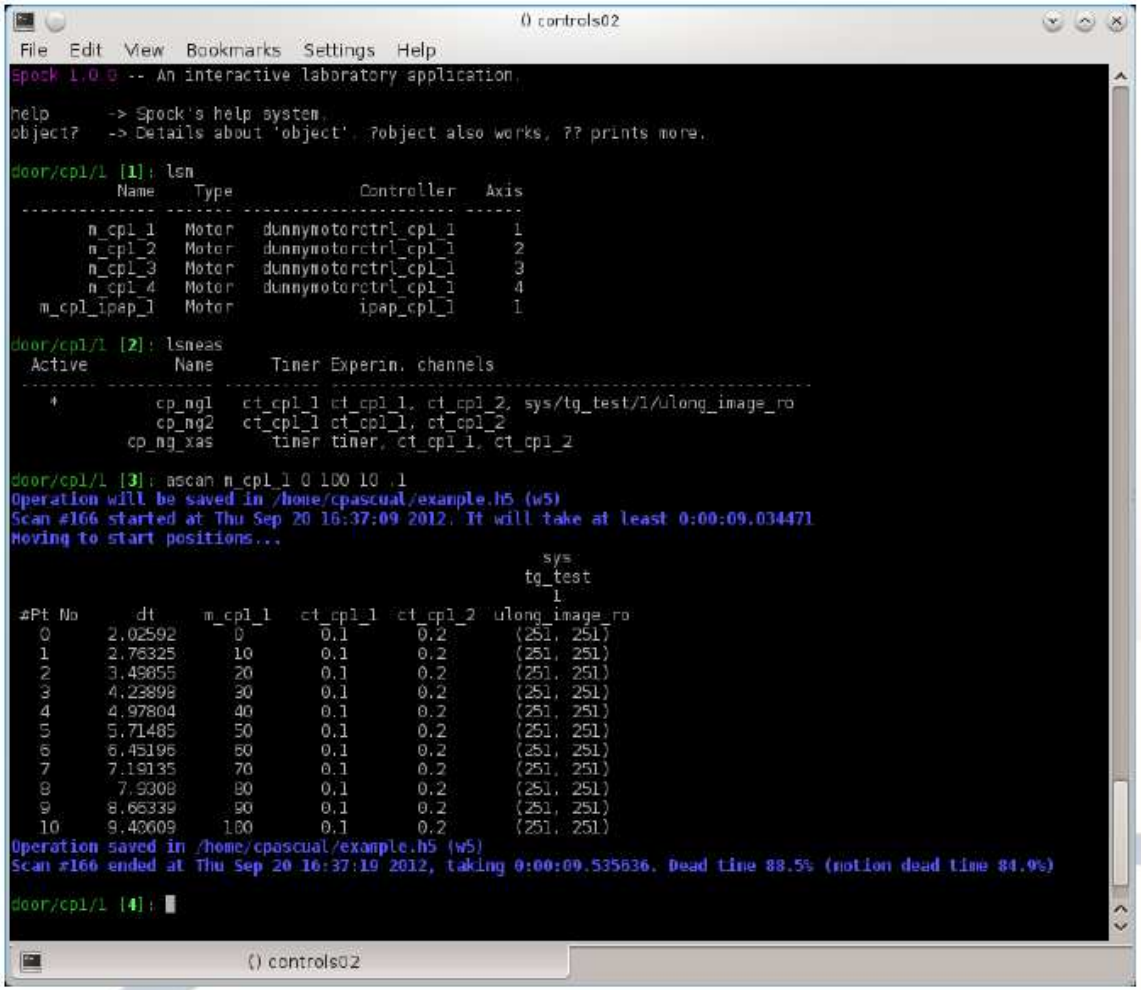
Sardana Device Server Example







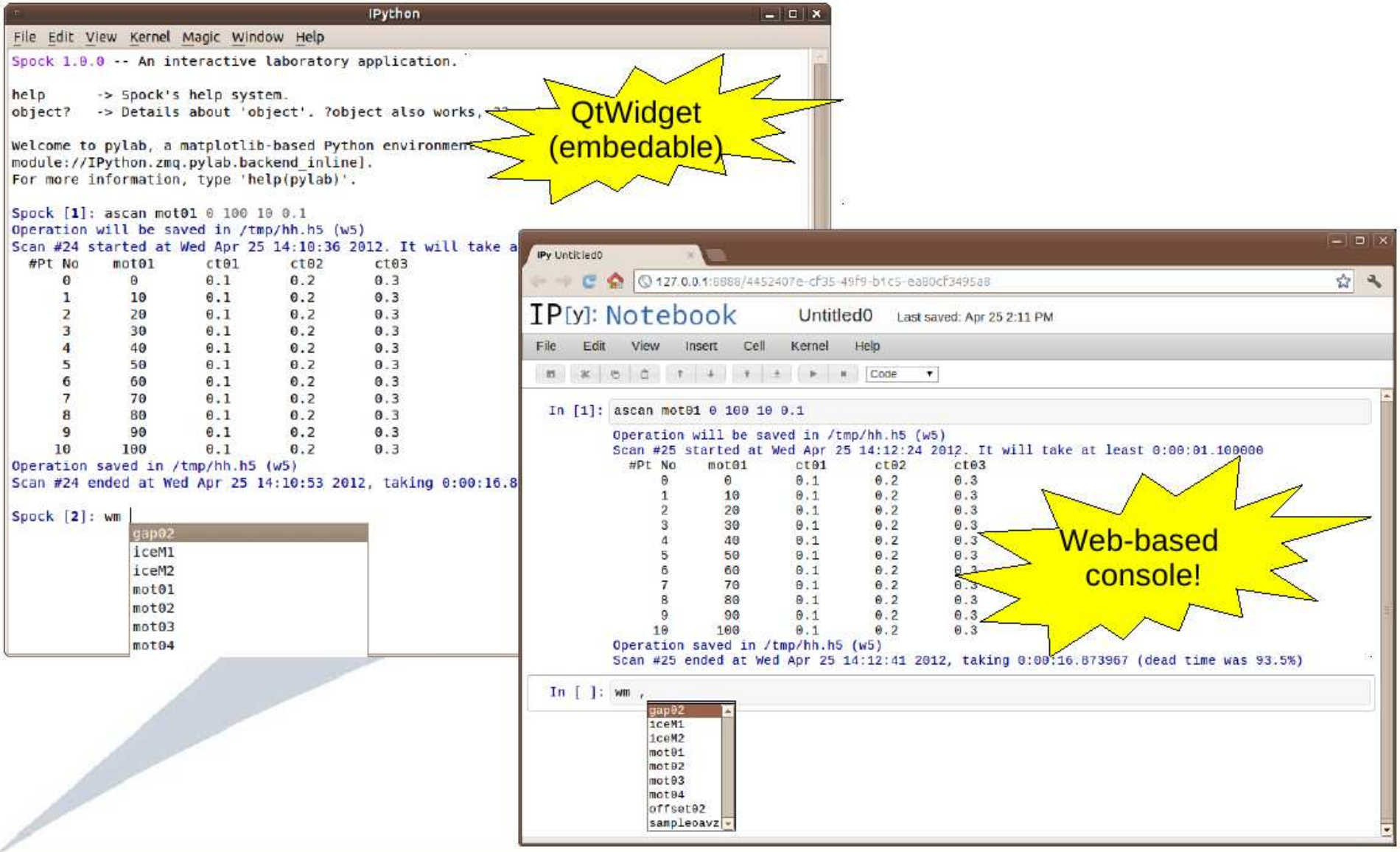
Taurus Command Line Interface (i)



- iPython based
- SPEC-like CLI

	Standard Macros
listing information	lsenv, dumpenv, lsa, lsm, lspm, lsexp, lsior, lsmeas, lsct, ls0d, ls1d, ls2d, lspc, lsctrl, lspc, lsi, lsctrllib, lsa, lsmac, scanhist
moving & acquiring	wa, wm, pwa, pwm, set_lim, set_lm, set_pos, mv, umv, mvr, umvr, ct, uct, settimer
scans step continuous	ascanc, a2scanc, a3scanc, a4scanc, amultiscan, dscanc, d2scanc, d3scanc, d4scanc, dmultiscan, meshc, fscan
other	(u)senv, (u)defmeas, (u)defelem, (u)defctrl, prdef, relmac, relmaclib

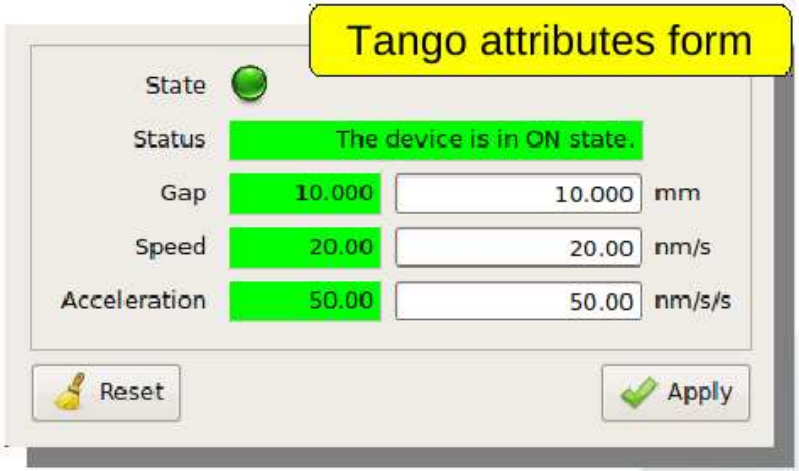
Taurus Command Line Interface (ii)



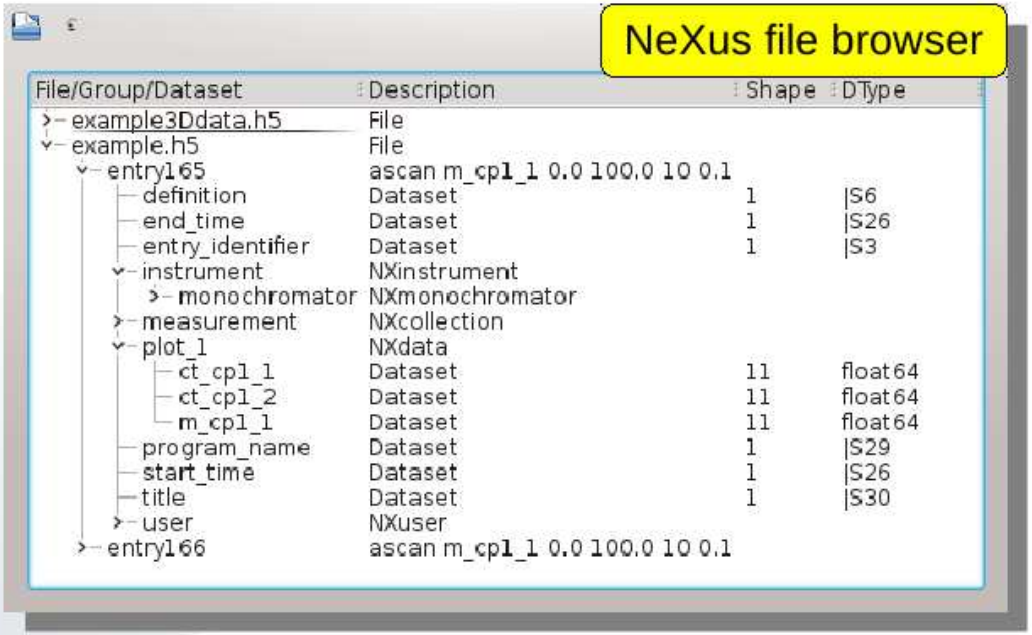
Taurus GUI Widgets (i)

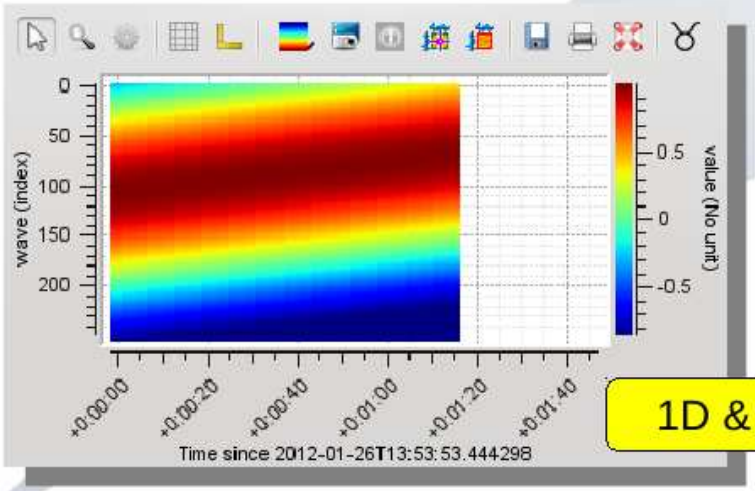
Taurus provides General-purpose widgets

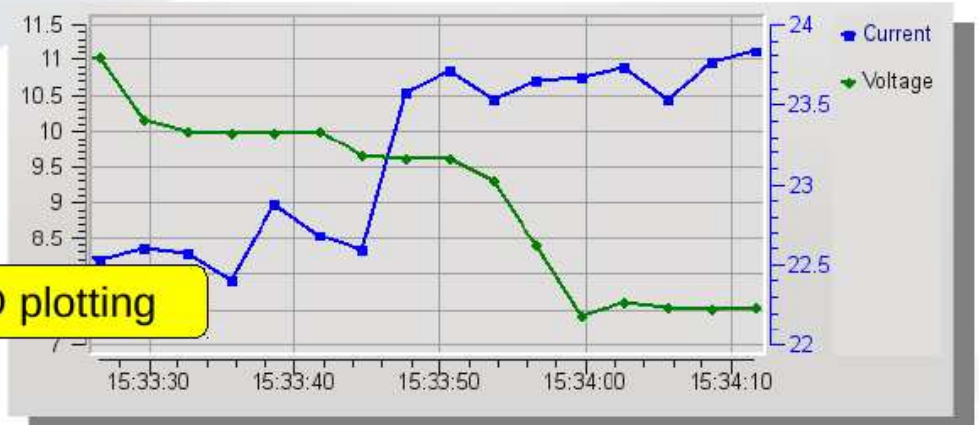
Tango attributes form



NeXus file browser

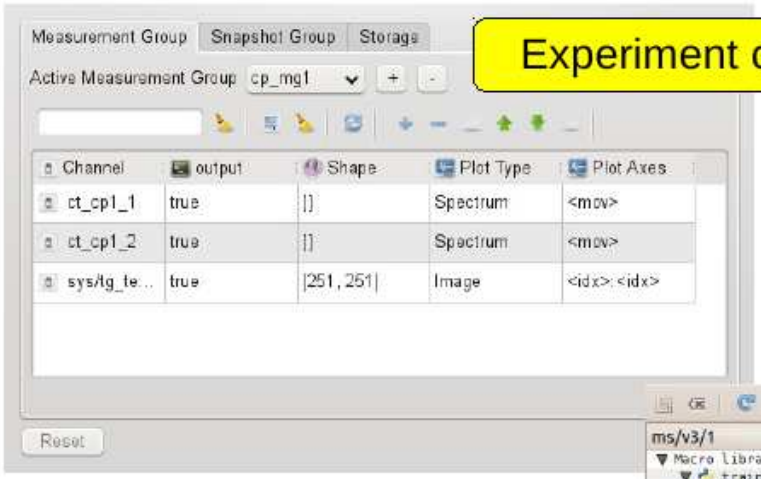






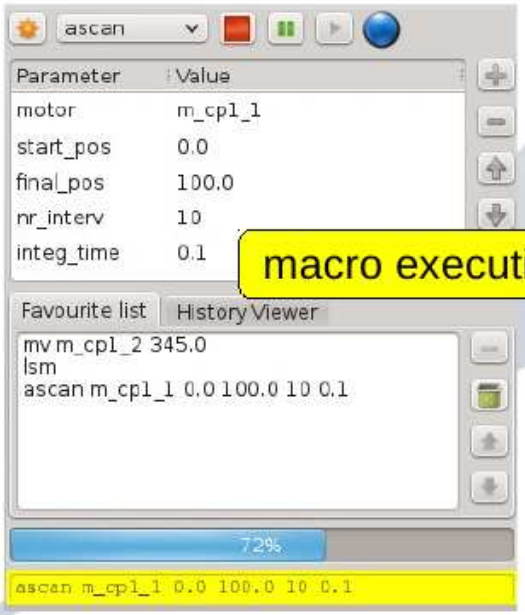
1D & 2D plotting

Taurus GUI Widgets (ii)

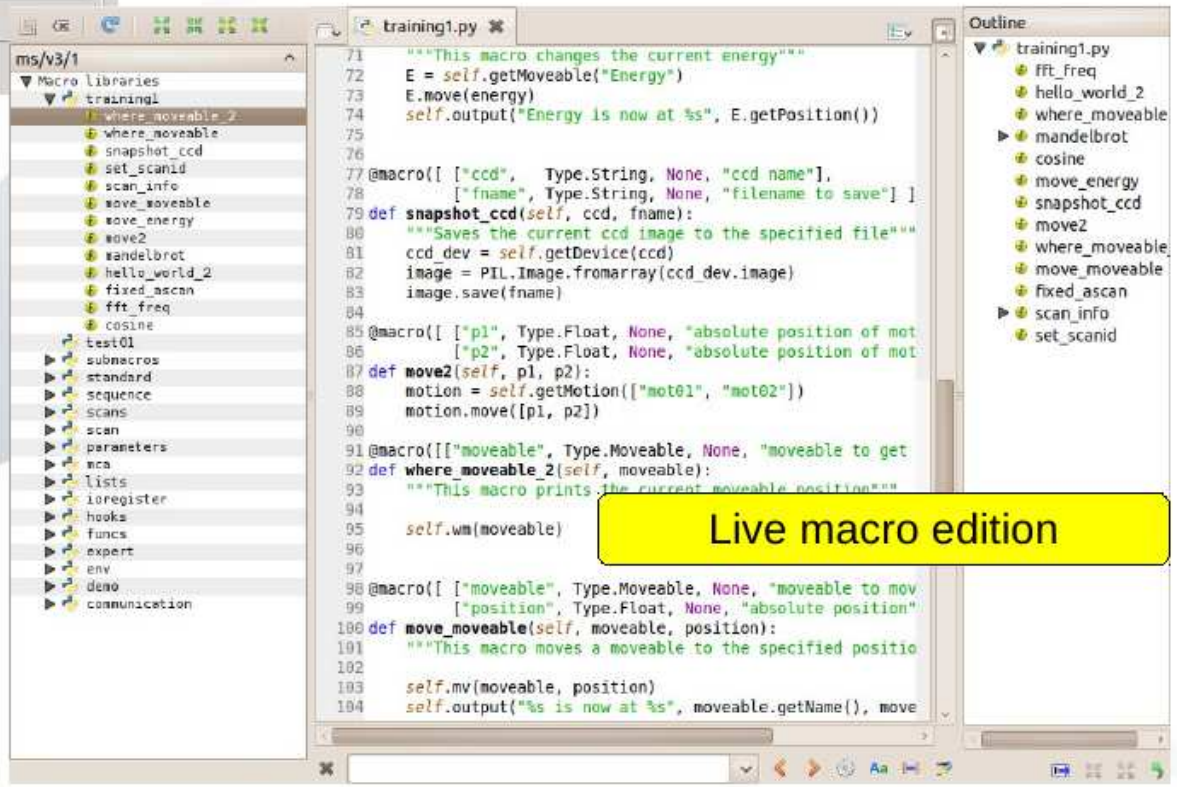


Experiment configuration

And also some specific widgets for dealing with macros

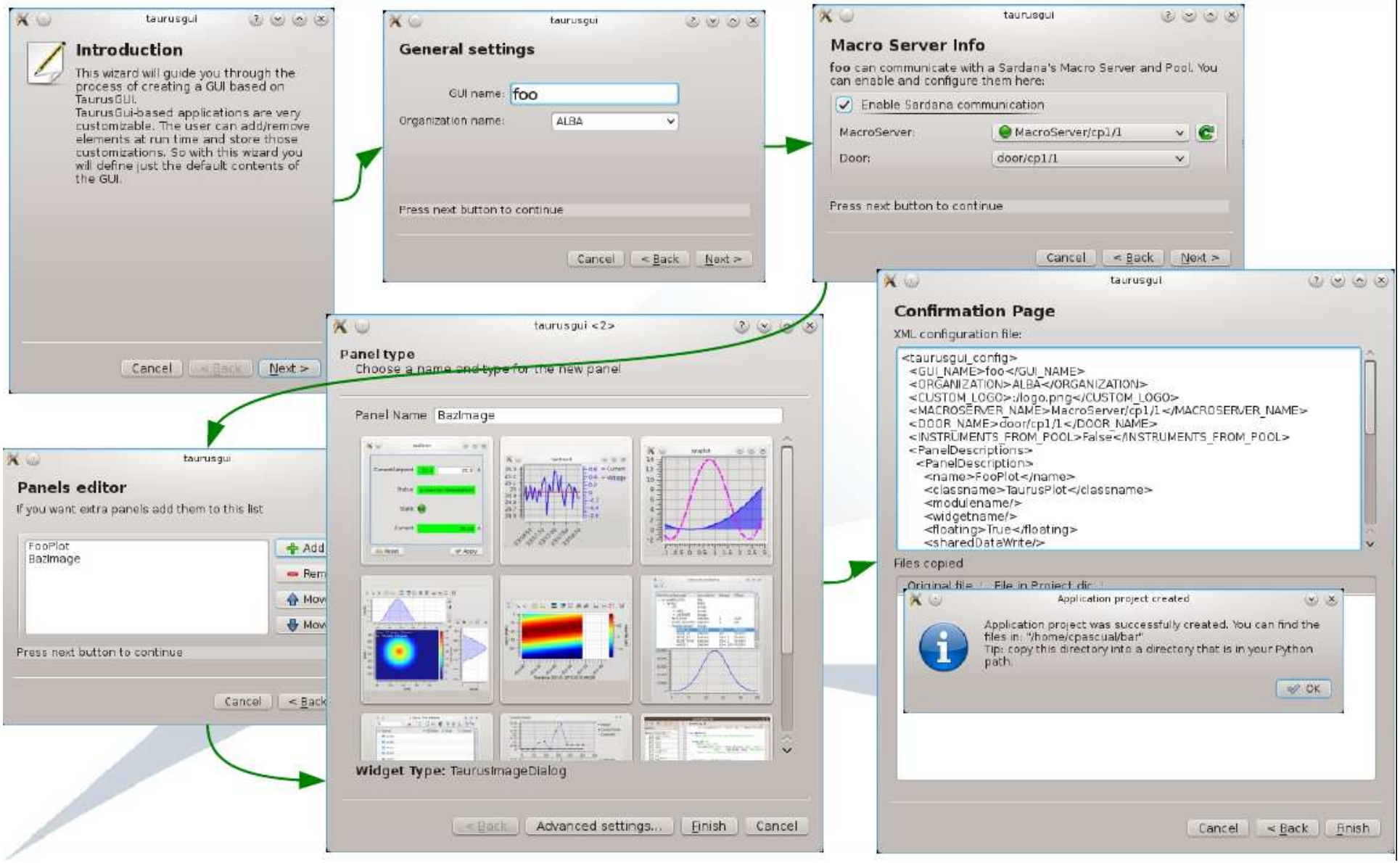


macro execution

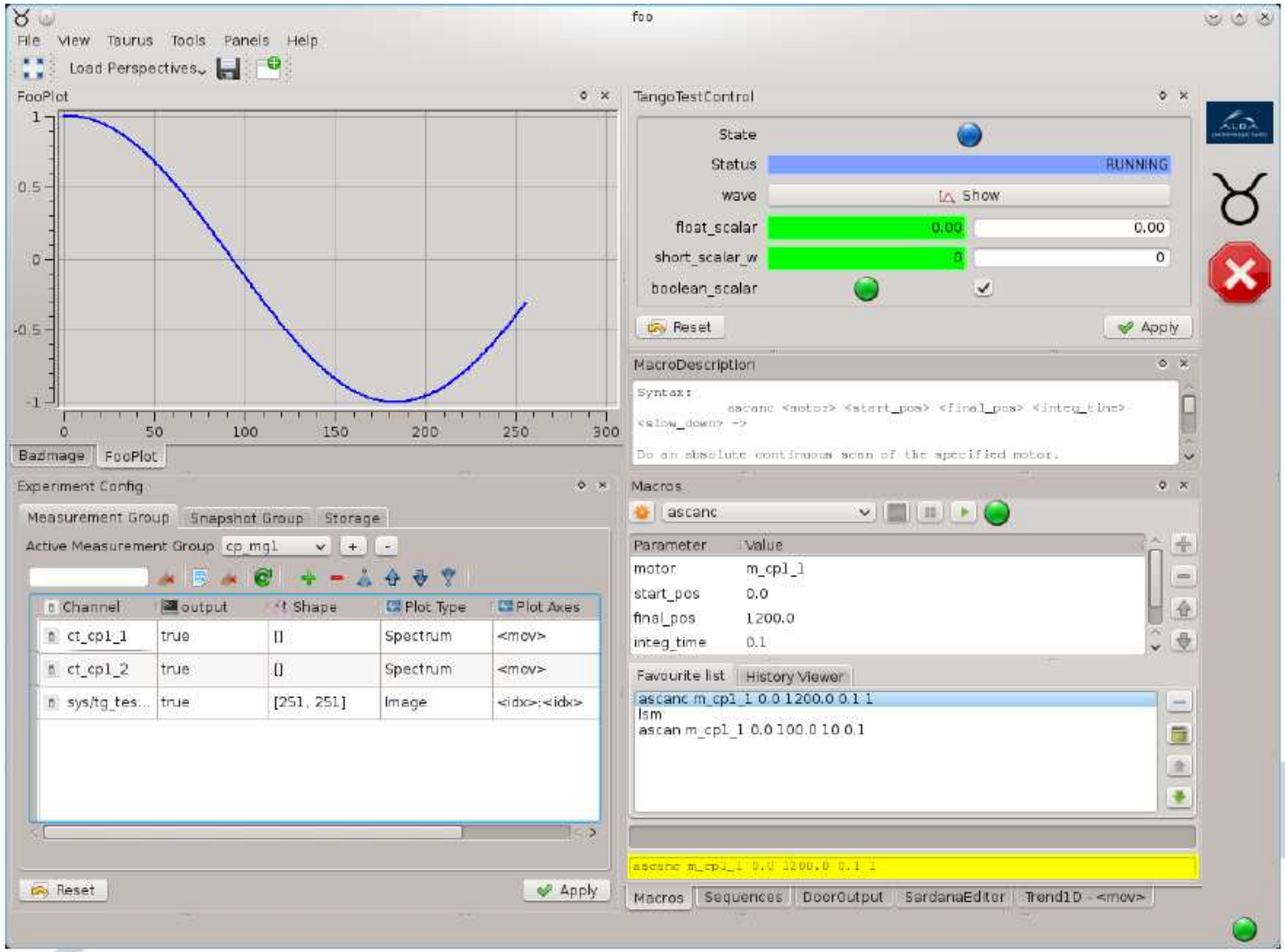


Live macro edition

Taurus Generic GUI Framework (i)

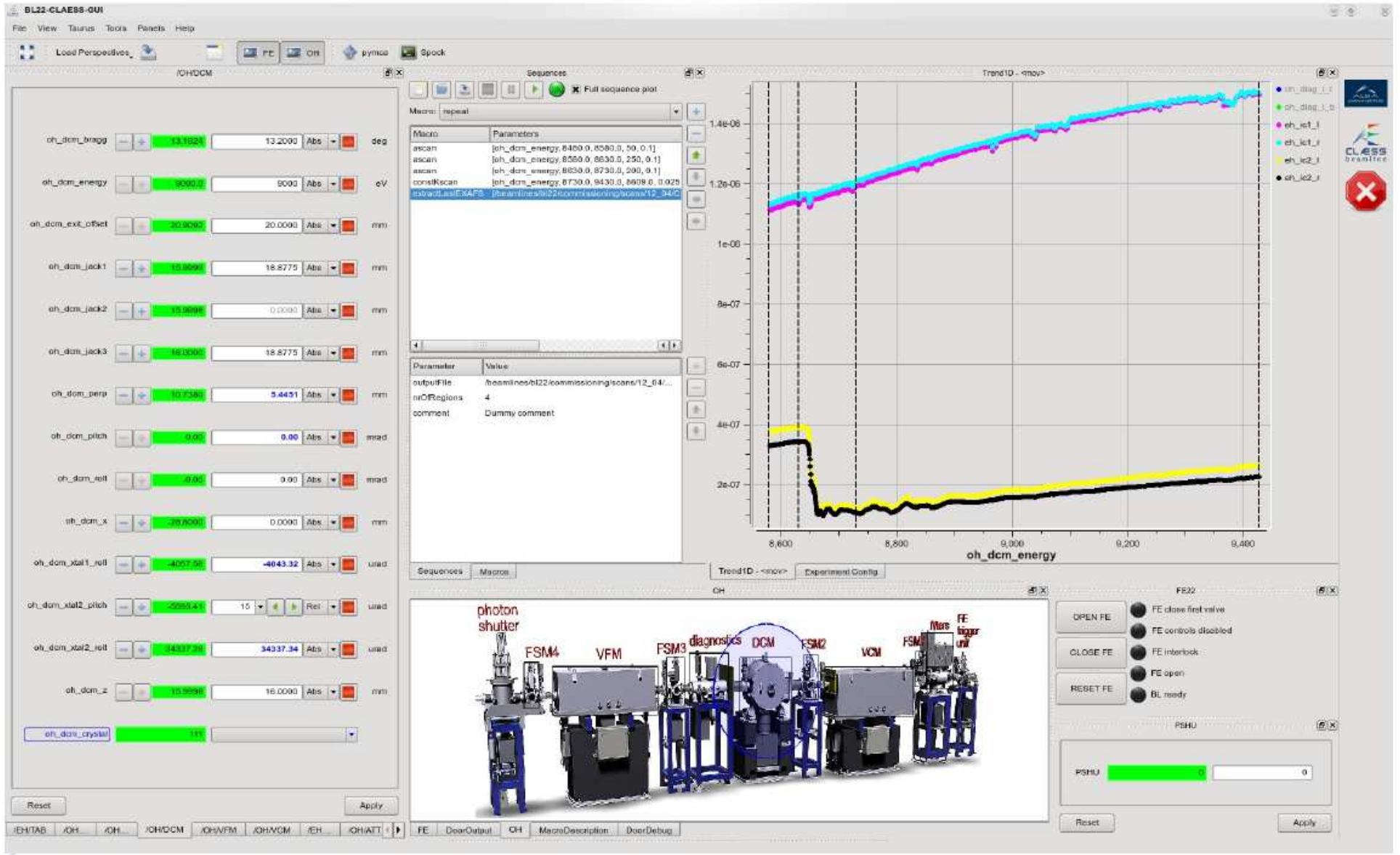


Taurus Generic GUI Framework (ii)

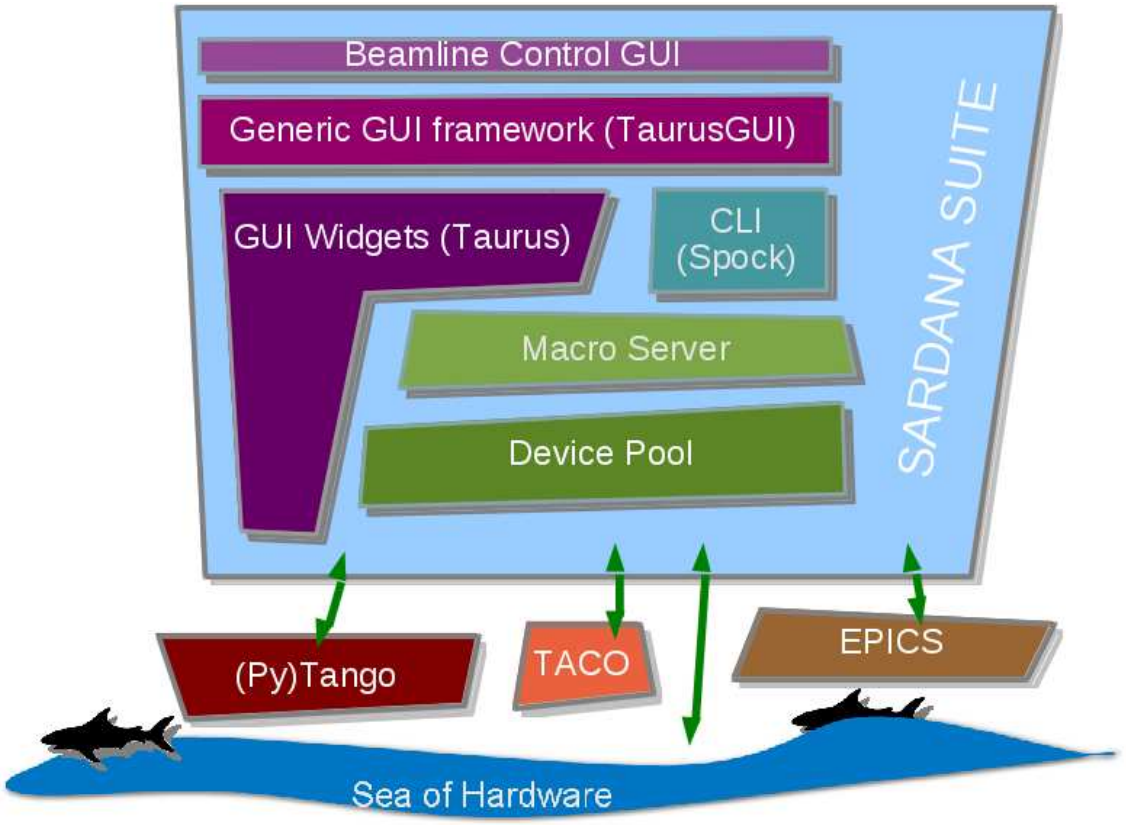


- ✔Dynamic
- ✔User-configurable
- ✔Common Look&Feel
- ✔Based on config file
- ✔Allows perspectives

Beamline Control GUI



Questions?


Beamline Control GUI
Generic GUI framework (TaurusGUI)
GUI Widgets (Taurus)
CLI (Spock)
Macro Server
Device Pool
SARDANA SUITE
(Py)Tango
TACO
EPICS
Sea of Hardware