

Sardana: Progress in the collaboration

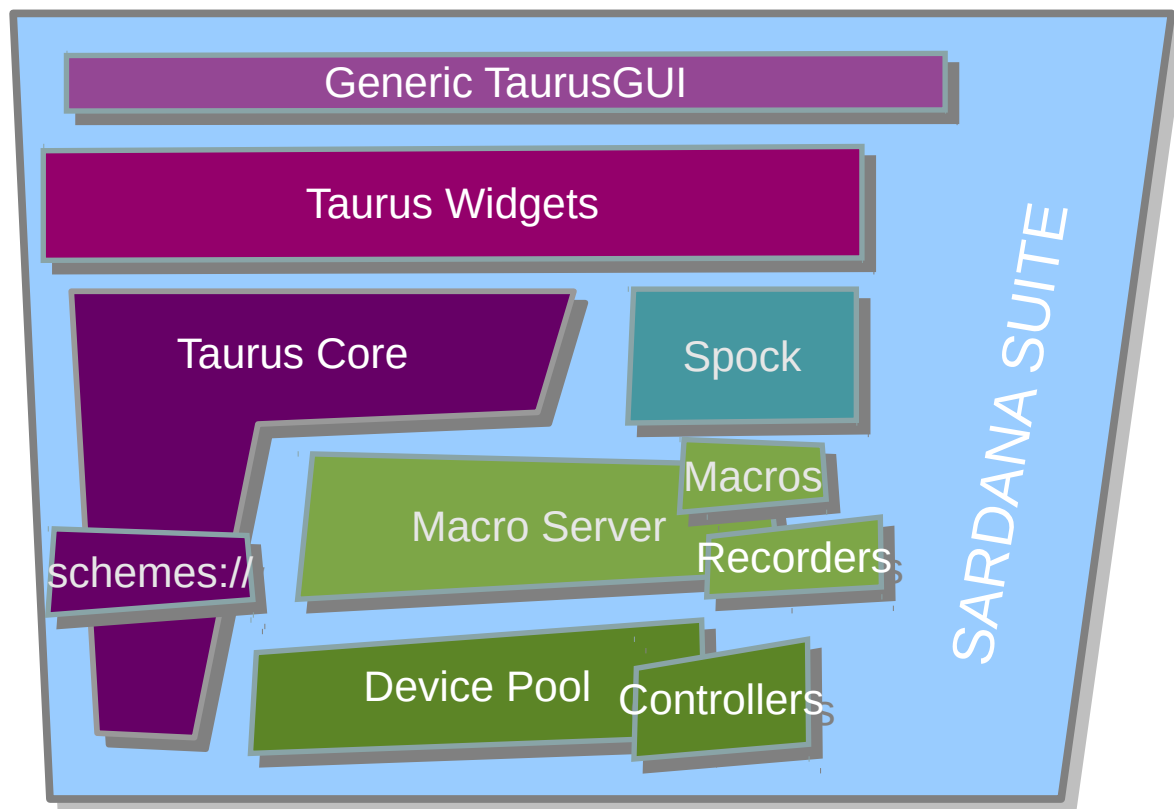


www.sardana-controls.org

Tango Meeting 2016 at ONERA - Toulouse



Zbigniew Reszela (Alba Synchrotron, Spain), on behalf of the Sardana Community



- Client-Server architecture
- Highly modular & customizable
- Python based
- Uses Tango
- Community-driven



Recent releases &
events

New features &
bug fixes

Ongoing
developments

Roadmap &
Summary

Recent releases &
events

New features &
bug fixes

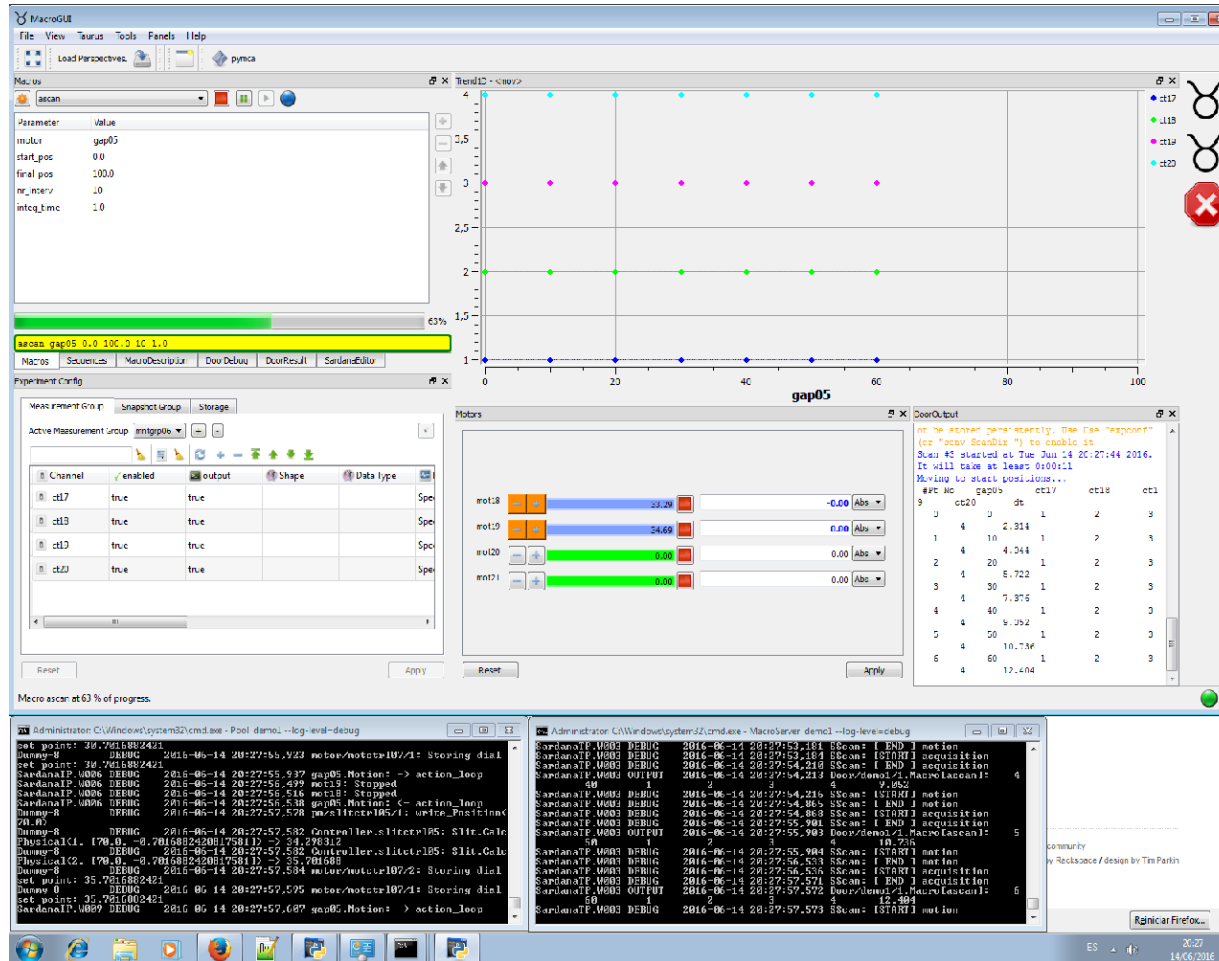
Ongoing
developments

Roadmap &
Summary

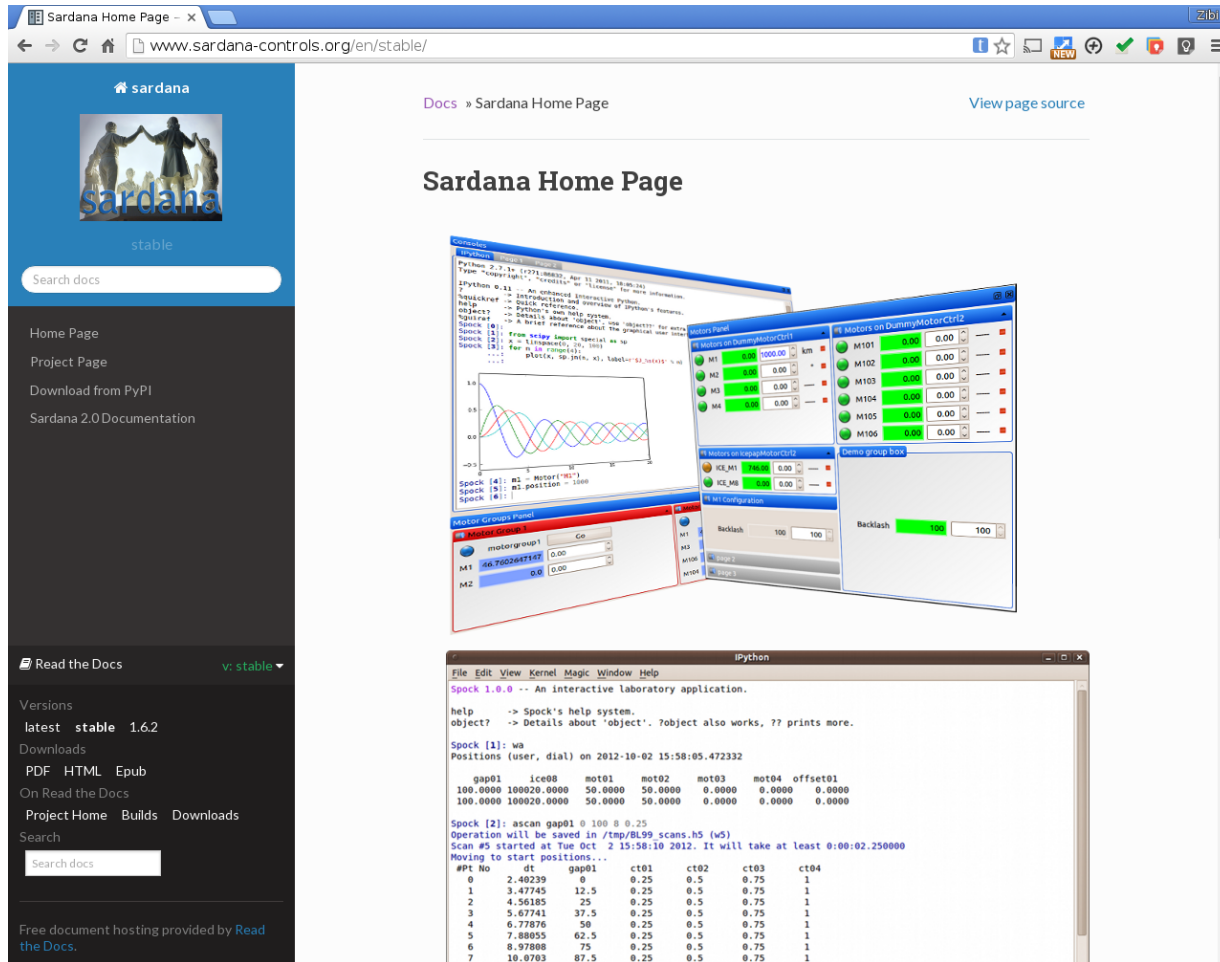
- **Allow Sardana execution on Windows (#228)**
- Allow reading of motor position when the motor is out of SW limits (#238)
- Improve speed of wa macro(#287)
- New macros dmesh and dmeshc (#283)
- Document DriftCorrection feature
- Fix meshc scan
- Solve bug in ascanc when using a pseudomotor (#353)
- **Sardana docs available in RTD (#5, #358)**
- Add option to display controller and axis number, in the output from wm/wa (#239)
- Allow undefine many elements at the same time, using udefelem (#127)
- Solve bugs related to loading macros/modules (#121 ,#256)
- Solve bug with PoolMotorTV showing AttributeError (#368 ,#369, #371)
- Solve bugs and features related with test framework (#249, #328, #357)

Releases – 1.6 (Jul15)

- Allow Sardana execution on Windows (#228)



- Sardana docs available in RTD (#5, #358)



The screenshot shows the Sardana Home Page in a web browser. The page has a blue header with the Sardana logo and the word 'stable'. Below the header is a search bar. The main content area is titled 'Sardana Home Page' and includes a 'Docs' link and a 'View page source' link. The page displays several panels: a 'Motor Groups Panel' showing a graph of motor positions, a 'Motor Groups Panel' showing a table of motor positions, and a 'Python' console window showing the output of a Spock command.

Motor Groups Panel (Table):

Motor	Position	Offset
M1	0.00	0.00
M2	0.00	0.00
M3	0.00	0.00
M4	0.00	0.00
M5	0.00	0.00
M6	0.00	0.00

Python Console:

```
Spock 1.0.0 -- An interactive laboratory application.
help -> Spock's help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

Spock [1]: wa
Positions (user, dial) on 2012-10-02 15:58:05.472332
gap01 ice08 mot01 mot02 mot03 mot04 offset01
100.0000 100020.0000 50.0000 50.0000 0.0000 0.0000 0.0000
100.0000 100020.0000 50.0000 50.0000 0.0000 0.0000 0.0000

Spock [2]: ascan gap01 @ 100 0 0.25
Operation will be saved in /tmp/BL99 scans.h5 (w5)
Scan #5 started at Tue Oct 2 15:58:10 2012. It will take at least 0:00:02.250000
Moving to start positions...
#Pt No dt gap01 ct01 ct02 ct03 ct04
0 2.40239 0 0.25 0.5 0.75 1
1 3.47745 12.5 0.25 0.5 0.75 1
2 4.56185 25 0.25 0.5 0.75 1
3 5.67741 37.5 0.25 0.5 0.75 1
4 6.77876 50 0.25 0.5 0.75 1
5 7.88055 62.5 0.25 0.5 0.75 1
6 8.97808 75 0.25 0.5 0.75 1
7 10.0703 87.5 0.25 0.5 0.75 1
```



SOLARIS
NATIONAL SYNCHROTRON
RADIATION CENTRE



- Many thanks to the organizers – DESY!!!
- Two days workshop: October 6-7, 2015
- Highlights:
 - Feedback from the scientists – thanks!
 - Taurus4 (core refactoring, graphics widgets, etc.)
 - Continuous scans
 - Data management
 - Other technical topics: general hooks, limits protection, macro environment inheritance, Nexus recorder, experiment configuration GUI, Taurus access to MacroServer environment, configuration tools
 - Sardana collaborative work

- **Add HKL support (SEP4)**
- **Improve support of repeat macro parameters (#3, #466) -**
- **Add support to external recorders (#380, #409, #417) -**
- **Improve the Door status behaviour (#120, #427)**
- **Correct PoolPath precedence - now it respects the order (#6)**
- **Add macro tw to standard macros (#437)**
- **Add possibility to rename pool elements (#430)**

Recent releases &
events

New features &
bug fixes

Ongoing
developments

Roadmap &
Summary

- Proposed and implemented by T. Nuñez from DESY and F. Picca from SOLEIL: SEP4*
- Depends on hkl library authored by F. Picca
- Standard macros (try to follow SPEC syntax)
- Implemented as base PseudoMotor controller and specific to the diffractometer geometry subclasses e. g. **E6C**, **E4C**
- GUI: **hklscan**, **ubmatrix**, **diffractometeralignment**
- Give it a try with: **sar_demo_hkl**

diffractometer related macros

• addreflection	• computeub	• latticecal	• orswap	• setor0
• affine	• freeze	• loadcrystal	• pa	• setor1
• br	• getmode	• lscan	• savecrystal	• setorn
• ca	• hklscan	• newcrystal	• setaz	• th2th
• caa	• hscan	• or0	• setlat	• ubr
• ci	• kscan	• or1	• setmode	• wh

The image shows three overlapping windows from the diffractometer control software:

- diffractometeralignment**: A window for setting angles (mu, omega, chi, phi, gamma, delta) and hkl (H, K, L). It includes fields for 'Position', 'Move to', 'Select mode', 'Engine', and 'Mode'. There are also buttons for 'Stop', 'Store Ref.', and 'Mac'.
- hklscan**: A window for setting positions/limits (H, K) and angles (mu, omega, chi, phi, gamma, delta). It includes fields for 'Position', 'Move to', 'Start', 'Stop', 'Nb of inter', and 'Parameters' (Nb points, Sample time). There are buttons for 'Start', 'Stop', 'Display Angles', and 'Macroserver Connection'.
- ubmatrix**: A window for setting the UB Matrix (UB11, UB12, UB13, UB21, UB22, UB23, UB31, UB32, UB33) and U Vector (Ux, Uy, Uz). It also includes fields for 'Lattice' (a, b, c, alpha, beta, gamma) and 'Wavelength'. There are buttons for 'Update', 'Affine', 'ComputeUB', 'Reflections List', and 'Edit Reflections'.

* More details in: <http://sourceforge.net/p/sardana/wiki/SEP4/>

Macro Repeat Parameters

Previously: macro could define **only one** repeat parameter, it had to be **the last one**, **no nesting** was allowed.

```
@macro([[ 'mot', Moveable, None, 'motor to move'],
        ['positions', [[ 'pos', Float, None, 'position']],
        None, 'positions to move']])
def multimove(self, mot, *positions):
    for p in positions: mot.move(p)
```

Now: **any** number of repeat parameters is possible, located at **arbitrary** positions, **nesting** is allowed.

```
@macro([[ 'mot', Moveable, None, 'motor to move'],
        ['positions', [[ 'pos', Float, None, 'position']],
        None, 'positions to move'],
        ['integ_time', Float, 1, 'integration time']])
def multimovect(self, mot, positions, integ_time):
    for p in positions:
        mot.move(p)
        self.getMeasurementGroup('mntgrp01').count(integ_time)
```

Backwards incompatibility issues with some* macros developed for Sardana < 2 **that we have to rethink!**

- Macro function arguments (or **prepare & run** method arguments if the macro was developed as a class)
- Arguments of some macro API methods e. g. **execMacro**, **prepareMacro**, **createMacro**

* Only macros that were defining repeat parameters or internally calling other macros with repeat parameters

Macro Repeat Parameters

Previously: macro could define **only one** repeat parameter, it had to be **the last one**, **no nesting** was allowed.

Spock is not compatible with these macros yet!

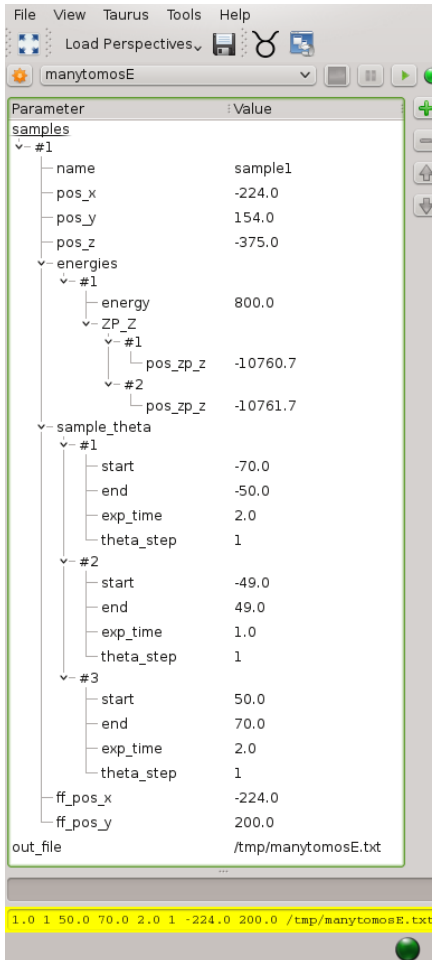
```
Door_1[1]: multimovect mot01 (0, 1, 2, 3, 4) 1
```

To be decided: how to extend the “spock syntax” with repeat parameter grouping?

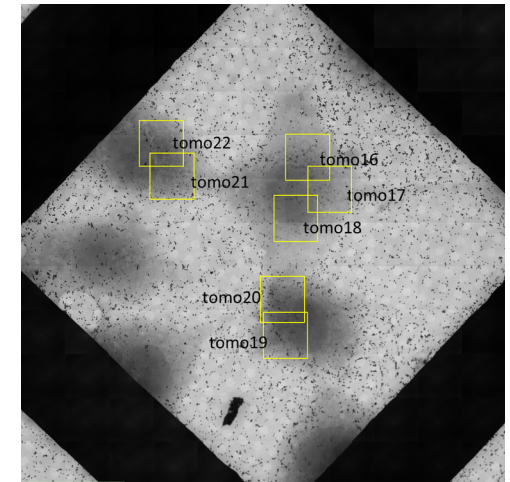
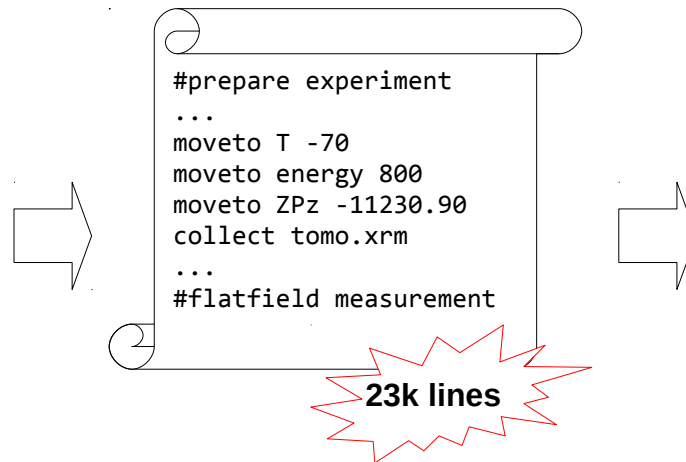
Backwards incompatibility issues with some* macros developed for Sardana < 2!

- Macro function arguments (or **prepare** & **run** method arguments if the macro was developed as a class)
- Arguments of some macro API methods e. g. **execMacro**, **prepareMacro**, **createMacro**

* Only macros that were defining repeat parameters or internally calling other macros with repeat parameters

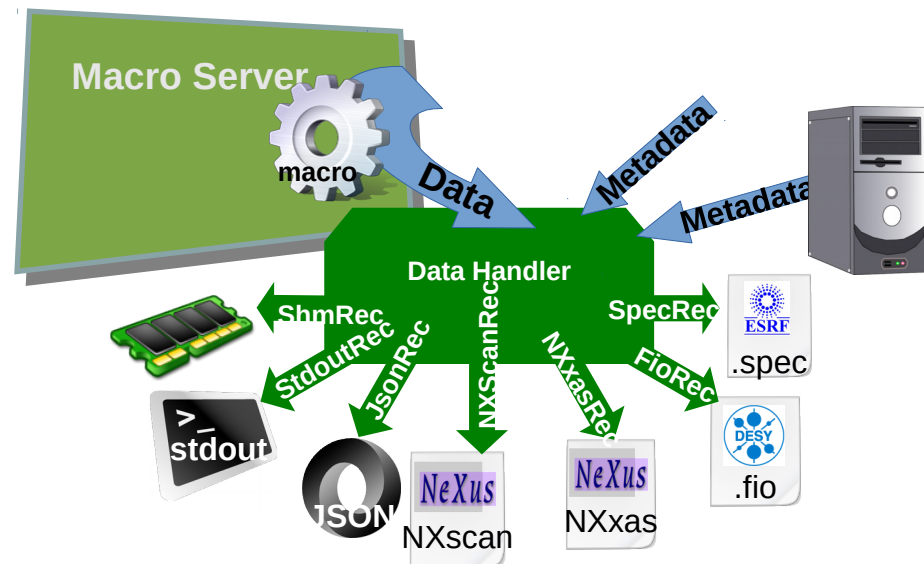


- BL09-MISTRAL (Tomography) – experiment control is out of Sardana
- MacroExecutor and macros resulted to be great tools for mocking up scripts and GUI



- TODO: MacroExecutor's copy&pasting repetitions e.g. duplicating samples

- Data recorders are used by scan macros and encapsulate specificity of delivering data to its destination.
- Now Sardana **allows to plug-in** recorders and override the standard ones.
- Storage recorders are configurable by **data file extension** or explicit selection:
ScanRecorder env. var.
- Developed by J. Kotanski from DESY – thanks!



BL04 – MSPD (Material Science and Powder Diffraction) – continuous scan.

```
#S 29 madscan 60.0 60.1 1.0 0.03
#U sicilia
#D 1466092212.0
#C Acquisition started at Thu Jun 16 17:50:12 2016
#N 23
#L Pt_No  pd_oc  oc  i1  i2  i3  i4  i5  i6  i7  i8  i9  i10  i11  i12  i13  i14  i15  temp  it  oct  temp2  dt
0 nan 59.99307375 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 nan 0.03 0.03 -0.0719291534424 nan
```

SPEC file recorder has a predefined order of columns, and stores all of them.

```
from sardana.macroserver.recorders.storage import SPEC_FileRecorder

class MAD_FileRecorder(SPEC_FileRecorder):
    def __init__(self, filename=None, macro=None, **pars):
        pars['labels'] = ['Pt_No', 'oc', 'i1', 'i2', 'i3', 'i4', 'i5',
                          'i6', 'i7', 'i8', 'i9', 'i10', 'i11', 'i12',
                          'i13', 'i14', 'i15', 'temp', 'temp2']
        SPEC_FileRecorder.__init__(self, filename=filename,
                                   macro=macro, **pars)
```

Data processing program expects different order of columns.

```
#S 3 madscan 60.0 60.1 1.0 0.03
#U sicilia
#D 1466070995.0
#C Acquisition started at Thu Jun 16 11:56:35 2016
#N 19
#L Pt_No  oc  i1  i2  i3  i4  i5  i6  i7  i8  i9  i10  i11  i12  i13  i14  i15  temp  temp2
0 59.993014375 0 0 0 0 0 0 0 0 0 0 0 0 0 0 nan -0.0790229644775
```

Recent releases &
events

New features &
bug fixes

Ongoing
developments

Roadmap &
Summary

- Now: hooks can be attached to macros e. g. scans but requires either defining a new macro or using sequencer.
- Motivation: customize macros without defining new ones.
- Apply to all macros defining a given hook place
- Set/Unset in MacroServer environment e. g.

```
Door_1[1]: senv HOOKS "{ 'post-step': ['mymacro'] }"  
Door_1[2]: senv ascan.HOOKS "{ 'post-step': ['mymacro'] }"  
Door_1[3]: senv MyDoor.HOOKS "{ 'post-step': ['mymacro'] }"
```

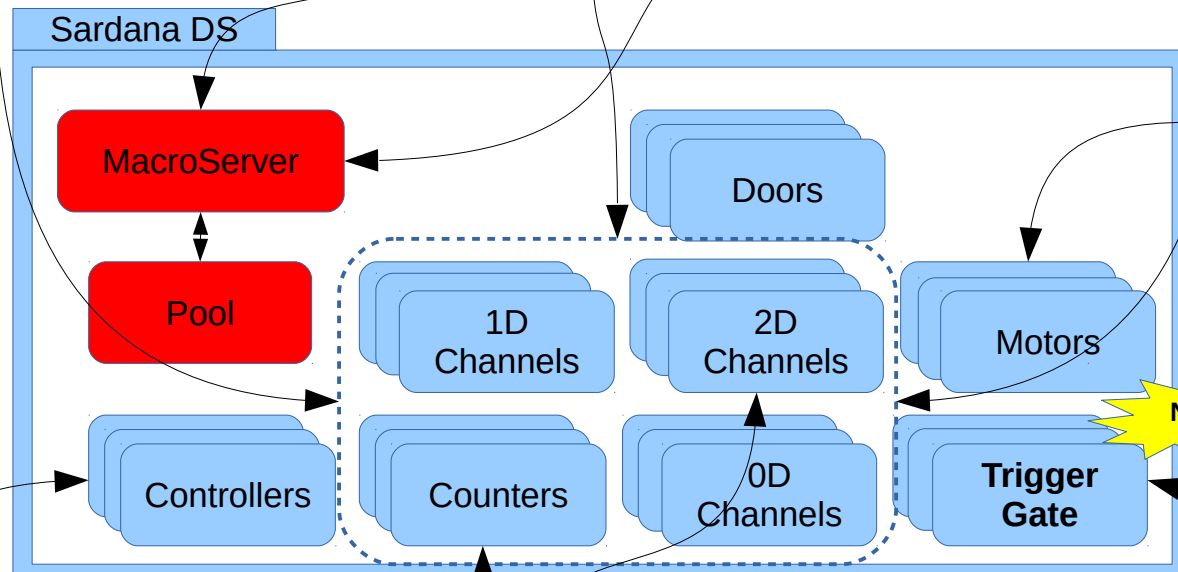
- Status:
 - Implemented and widely used at Petra experiments (DESY) since more than one year.
 - Integration into Sardana core is in progress.

Continuous Scans

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer



Motors –
linear & non-linear
trajectories

Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

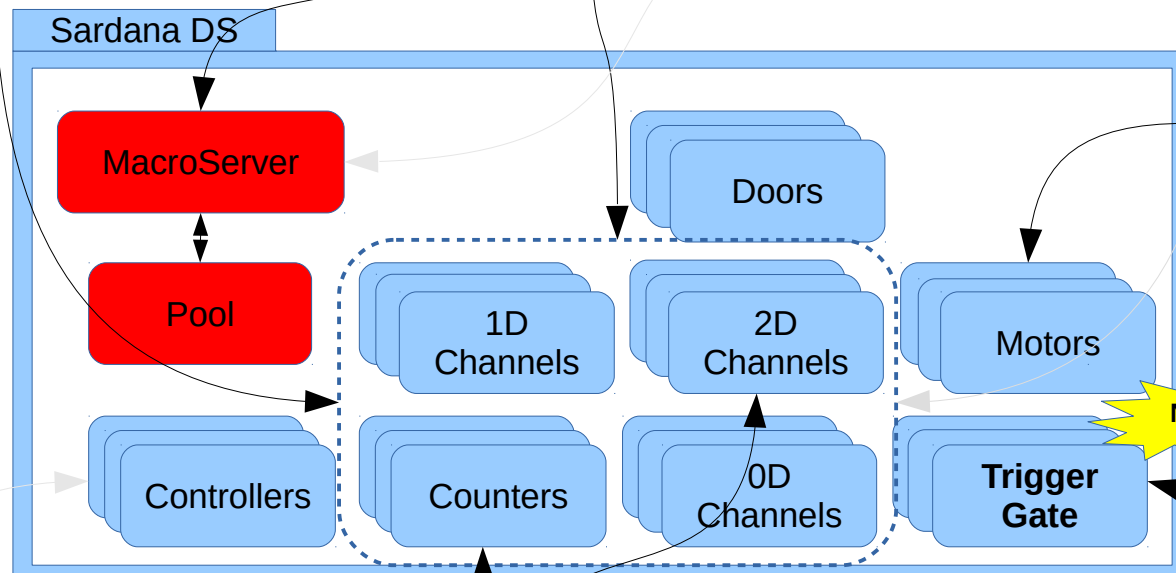
TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories



Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

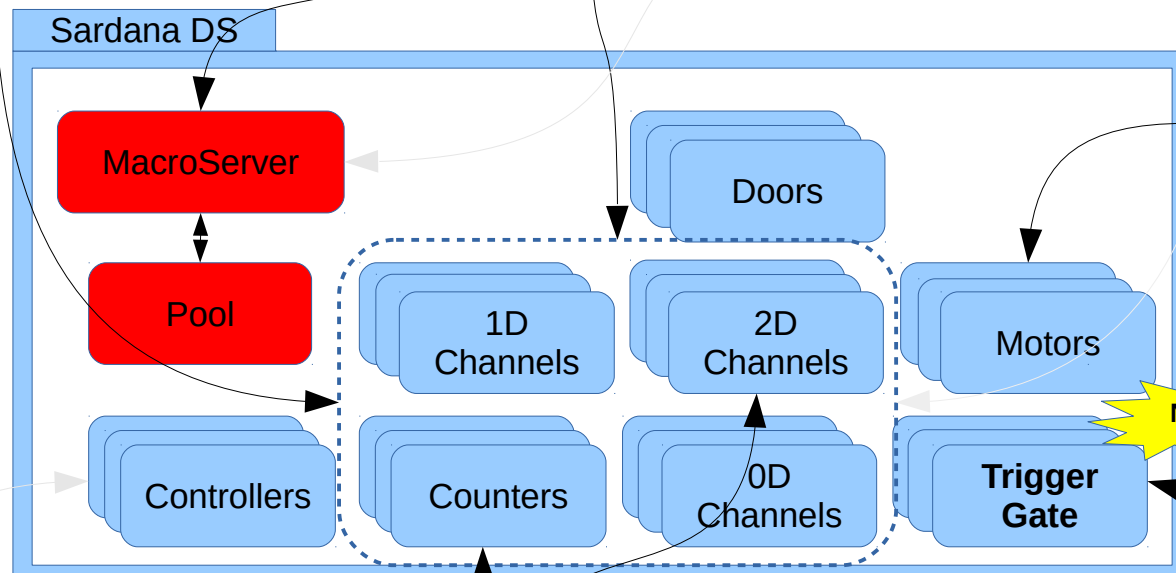
TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories

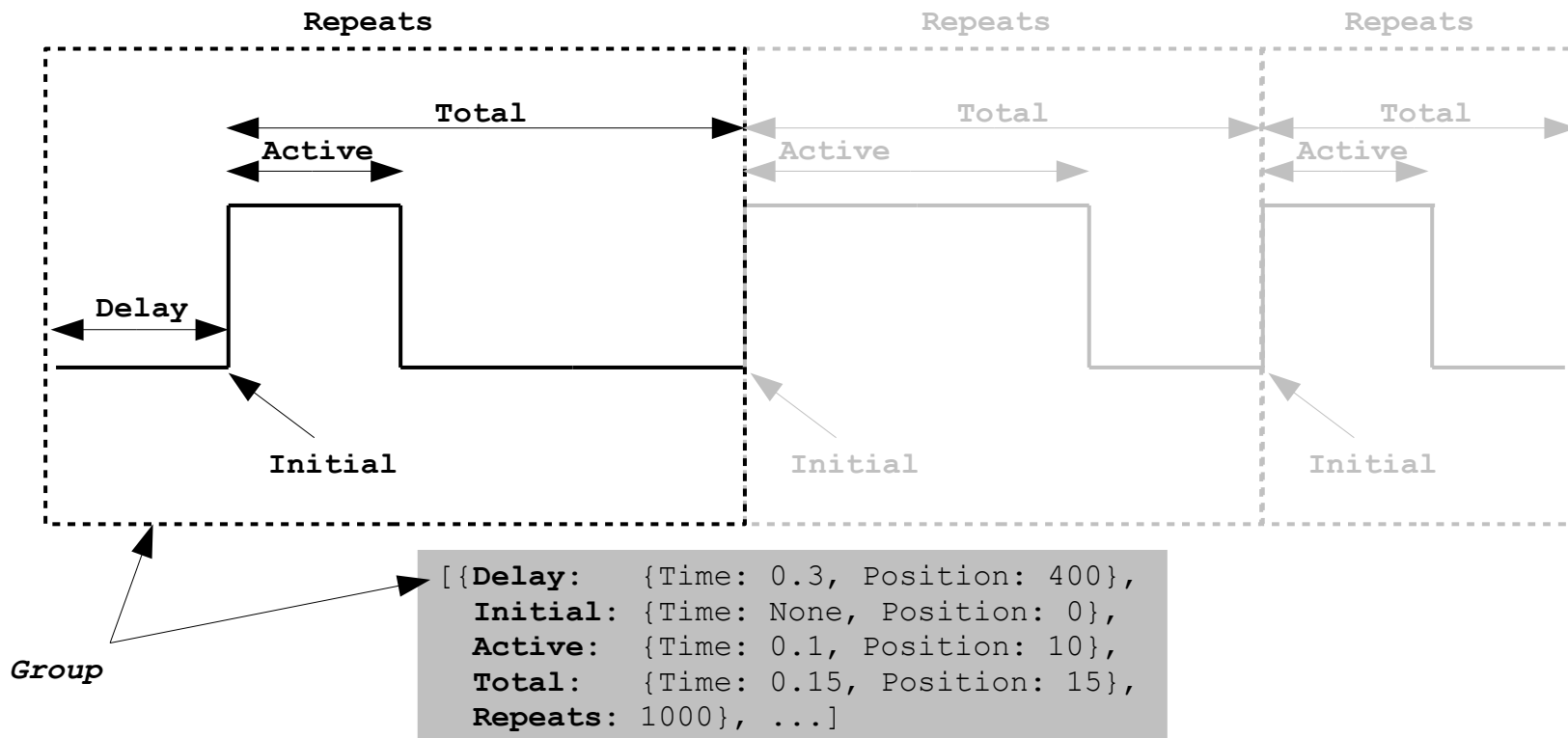


Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

TriggerGate – new element type:
time & position domain;
equidistant and arbitrary

- TriggerGate controller allows to **plug-in** hardware with the synchronization role.
- The API based on the **group** concept allows to define non-equidistant acquisitions.
- Redundant information, e. g. in time and position domains, allows sophisticated synchronizations: **initial position & active time**.
- **Initial** parameter allows to schedule acquisitions in the future.

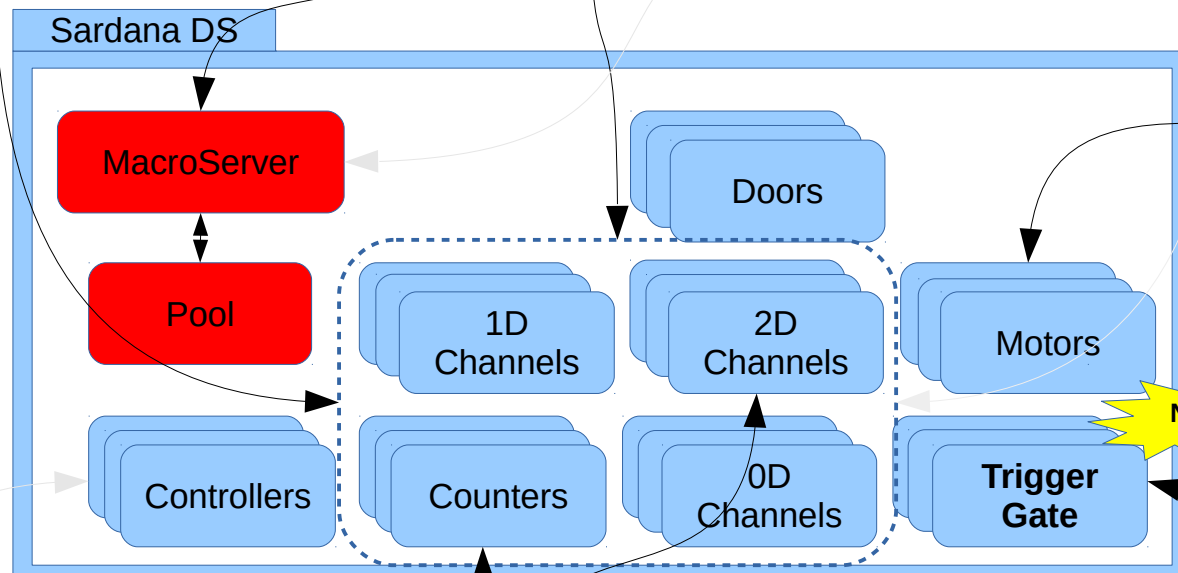


Measurement Group – relation channel – trigger:
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories

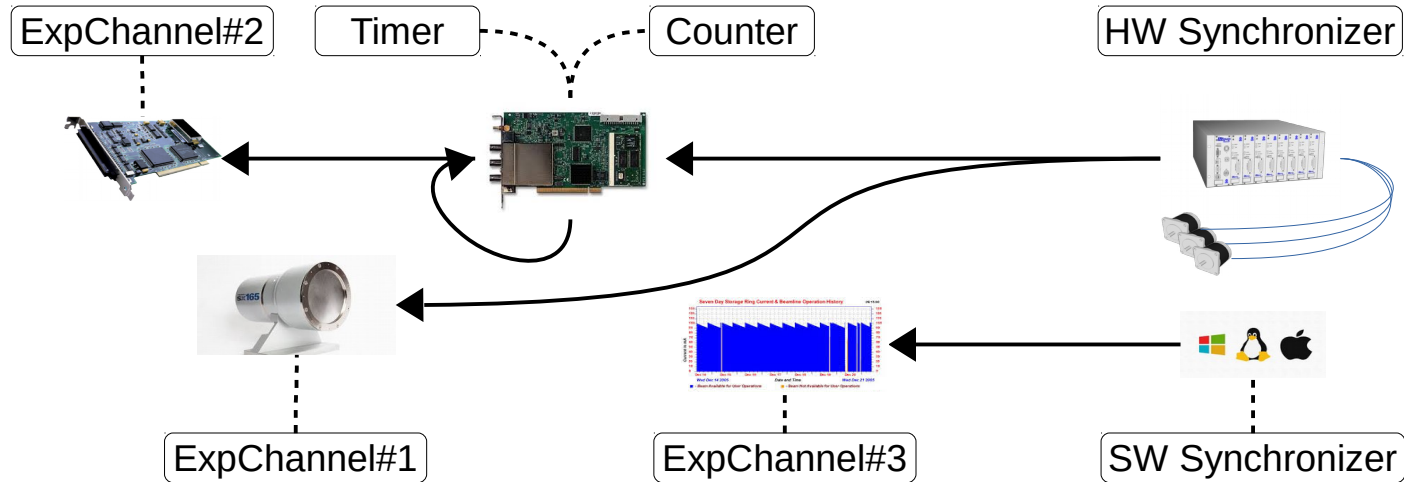


Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

Meas. Group Configuration



Exemplary setup involved in a continuous scan comprising mixture of hardware and software synchronization

Channel	Control	Synchronizer
Timer	Trigger	HW Synchronizer
ExpChannel#1	Trigger	HW Synchronizer
Counter	Gate	Timer
ExpChannel#2	Gate	Timer
ExpChannel#3	Trigger	SW Synchronizer

Direction of the
synchronization
control

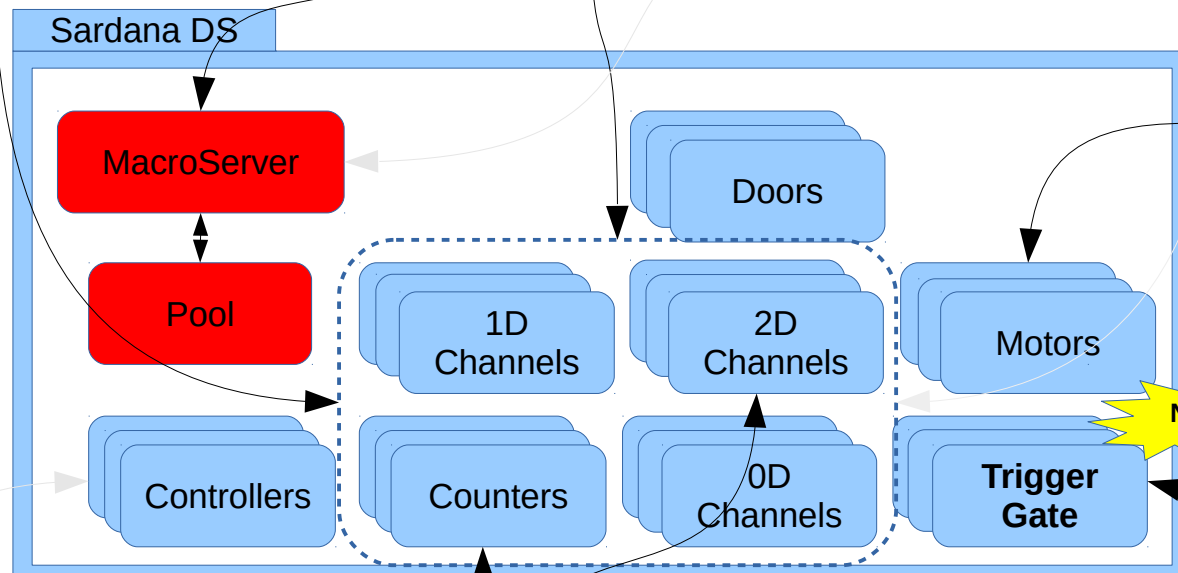
Measurement Group configuration expressed by 1-to-1 relation between the Synchronizer and the Experimental Channel

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories



Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

Old controllers will still be compatible with step scans!

AcqSynch:

- SoftwareTrigger
- HardwareTrigger
- SoftwareGate
- HardwareGate

LoadOne accepts number of repetitions.

Conditional programming.

ReadOne returns:

- single value
- chunk with values

```
class MyCounterTimerController(CounterTimerController):

    def __init__(self, *args, **kwargs):
        CounterTimerController.__init__(self, *args, **kwargs)

    def SetCtrlPar(self, par, value):
        # set controller parameter e.g.
        # timer, monitor, acquisition_mode, ...
        if par == 'trigger_type':
            self.acq_synch = value

    def LoadOne(self, ind, integ_time, repetitions):
        # load integration time and repetitions

    def StartOne(self, ind, value=None):
        # start your channel

    def ReadOne(self, ind):
        # read acquired data
        if self.acq_synch == AcqSynch.HardwareTrigger:
            ret = [...]
        elif self.acq_synch == AcqSynch.SoftwareTrigger:
            ret = ...
        # return float or seq<float>
        return ret

    def StateOne(self, ind):
        # read state

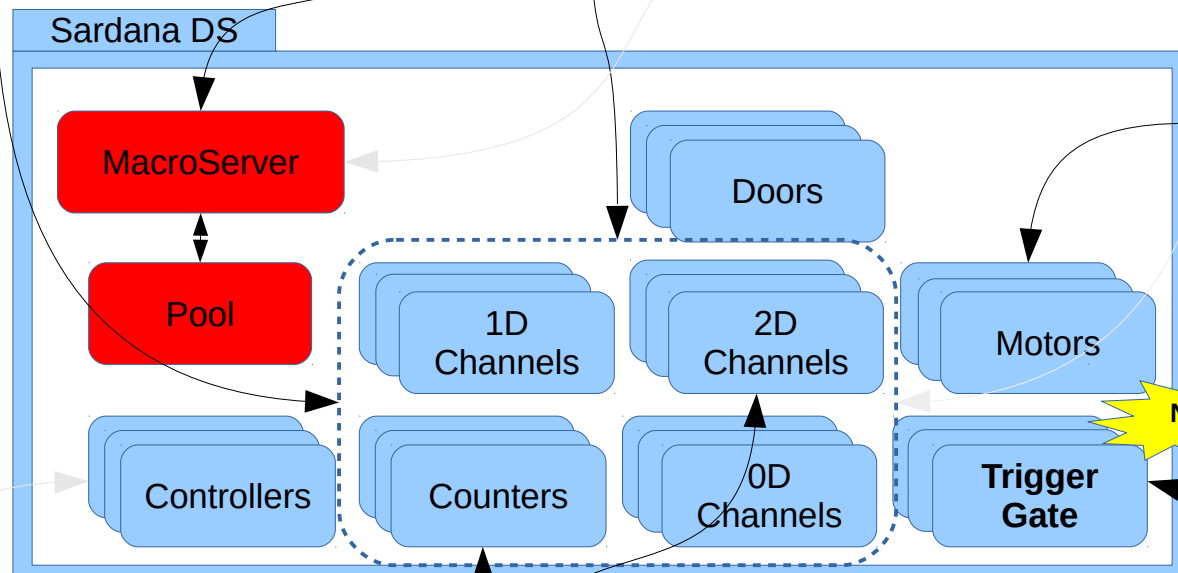
    def AbortOne(self, ind):
        # abort your channel
```

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories



Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

Data buffering & interpolation

- Every acquired value is stamped with the absolute time and the acquisition **index**.
- GSF **receives data in chunks** and fills the records based on the indexes.
- Software synchronized channels do not guarantee to provide data for each record.
- **Zero order hold** (constant interpolation) is applied in case of skipped acquisitions in order to fill the gaps.
- Interpolated data must be easily distinguishable from the raw data.
- Things get complicated with the PseudoCounters...

Measurement Group – relation channel – trigger;
co-existence of software and hardware synchronization

Data collection, merging and storage –
timestamp, raw + interpolated data

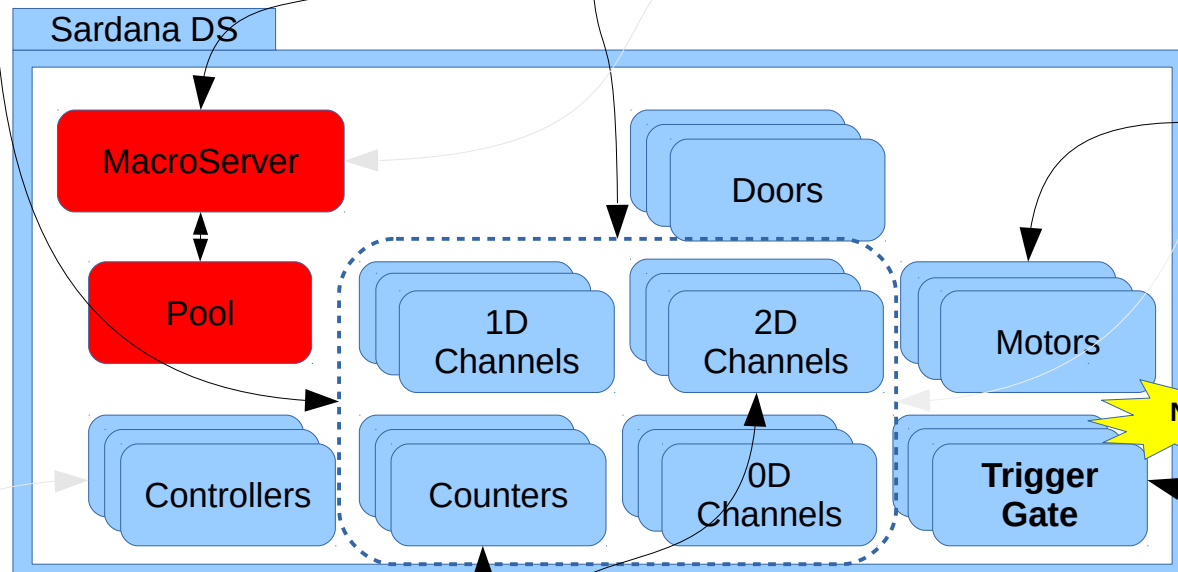
75%

60%

Different distribution schemes
require different data transfer

Motors –
linear & non-linear
trajectories

90%



Controller - handle multiple
hardware functionalities

Experimental Channels - aware of multiple
acquisition & report multiple data

90%

TriggerGate – new element type;
time & position domain;
equidistant and arbitrary

80%

Recent releases &
events

New features &
bug fixes

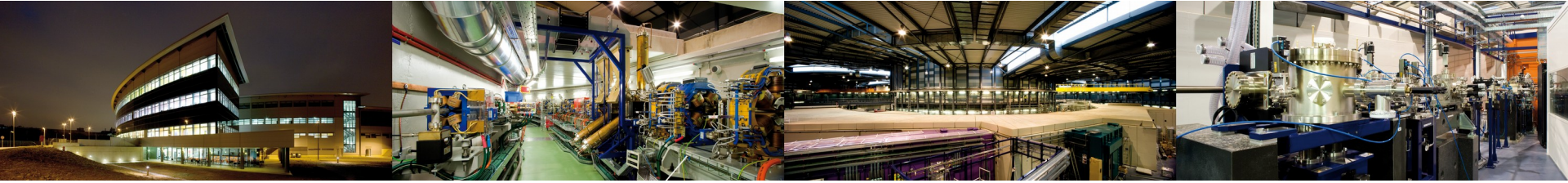
Ongoing
developments

Roadmap &
Summary

- Sardana must be adapted to Taurus4 (core + widgets)
- Next release (Jul16) will be mainly bug fixes release.
- Sardana similarly to Tango will migrate its development platform to Github.
- Sardana Workshop 2016 (satellite to NOBUGS).
- Hopefully Jan17 release will bring standard, flexible and transparent continuous scans to Sardana.

- Satellite to the NOBUGS conference:
- Thursday, 20 October (08:00 – 17:00)
- Morning session focused on the Experience Feedback from beamline scientists and new developers.
- The afternoon session will be an open space discussion around the technical hot topics: Continuous/Fly scan, Nexus integration, Taurus migration and new scheme system ...
- Subscribe to it!
- Propose discussion topics!
- <https://nobugs.esss.se>





Thank you!

