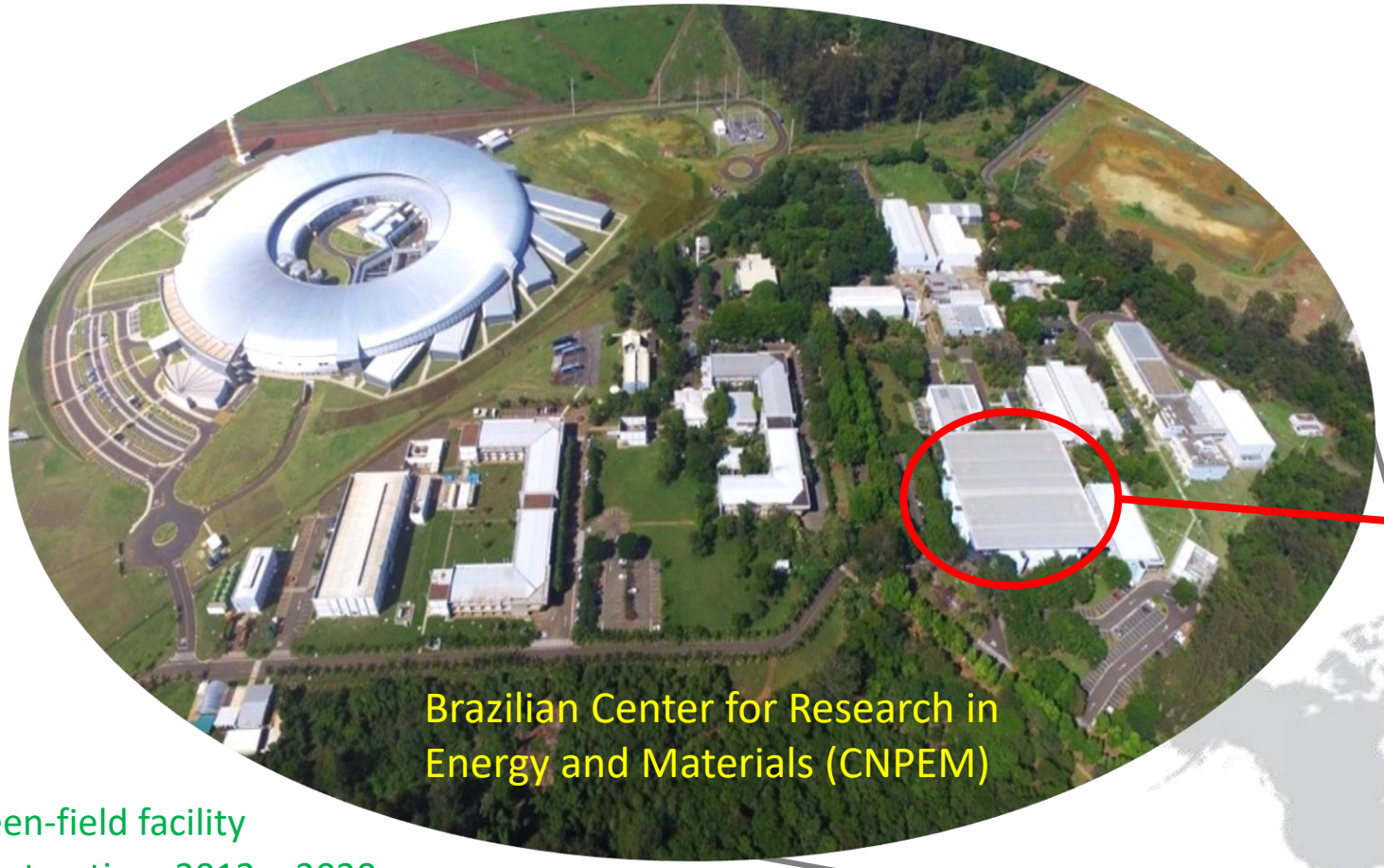
Python control tools at SIRIUS

Fernando Henrique de Sá,
on behalf of the SIRIUS Team

fernando.sa@lnls.br

Accelerator Middle Layer Workshop
DESY Hamburg (DE), June 19-21, 2024

SIRIUS – 4GSR in Operation



SIRIUS design parameters

Energy	3.0	GeV
Circumference	518.4	m
Emittance	250	pm.rad
Current (top-up)	350	mA

UVX, 2nd generation light source (1997-2019)

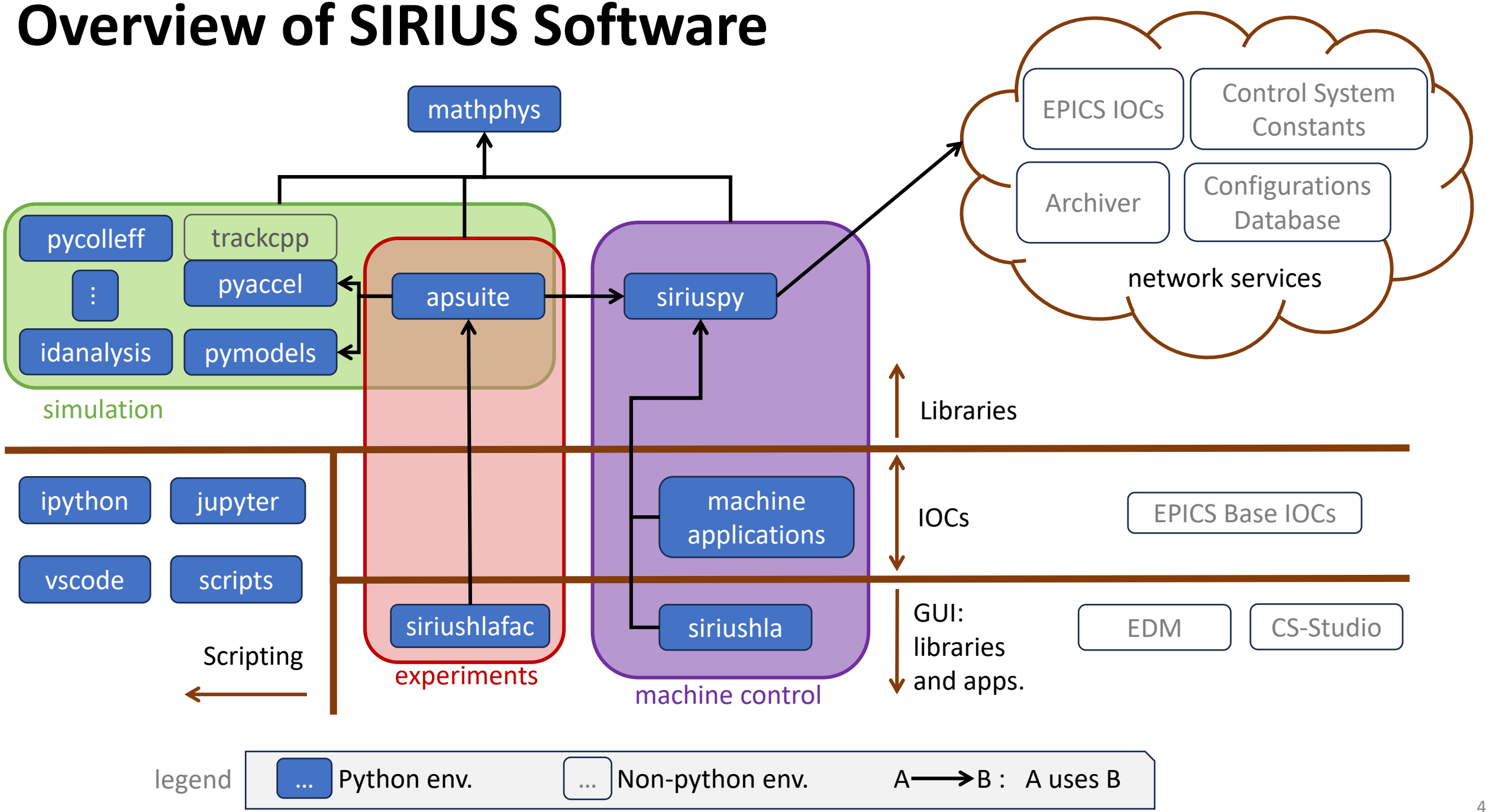
- Green-field facility
- Construction: 2012 – 2020
- Cost: US$ 500M (~85% spent in Brazil)
- 1st regular users call: Nov. 2022
- 10 beamlines in operation
- 100 mA in top-mode mode, uniform fill
- Phase-1 (end of 2024): 14 beamlines

Short history of python usage for SIRIUS

- In UVX → AT and MML (no python);
- 2011-2012: SIRIUS project → large simulations (DA and MA) → AT + *Tracy3*;
- 2013: *trackcpp* + (*pyaccel* and *pymodels*) → first SIRIUS models in python;
- 2014:
  - ✓ expansion of *pyaccel* → python for automated update of wiki SIRIUS parameters;
  - ✓ *trackcpp* substitutes *Tracy3* → incentive for python usage → DA and MA analysis migrated to python;
  - ✓ *pyjobs*: python server to distribute and manage SIRIUS simulations on PCs of CNPEM campus;
- 2015: virtual accelerator (VACA) → first contact with *pcaspy* and *pyepics* → *siriuspy* is created;
- 2016: good experience with *pcaspy* → all soft IOCs + PS IOCs in python → *machine-applications* creation;
- 2017: CS-Studio very disappointing → discovery of *PyDM* from SLAC → all control GUIs in python;

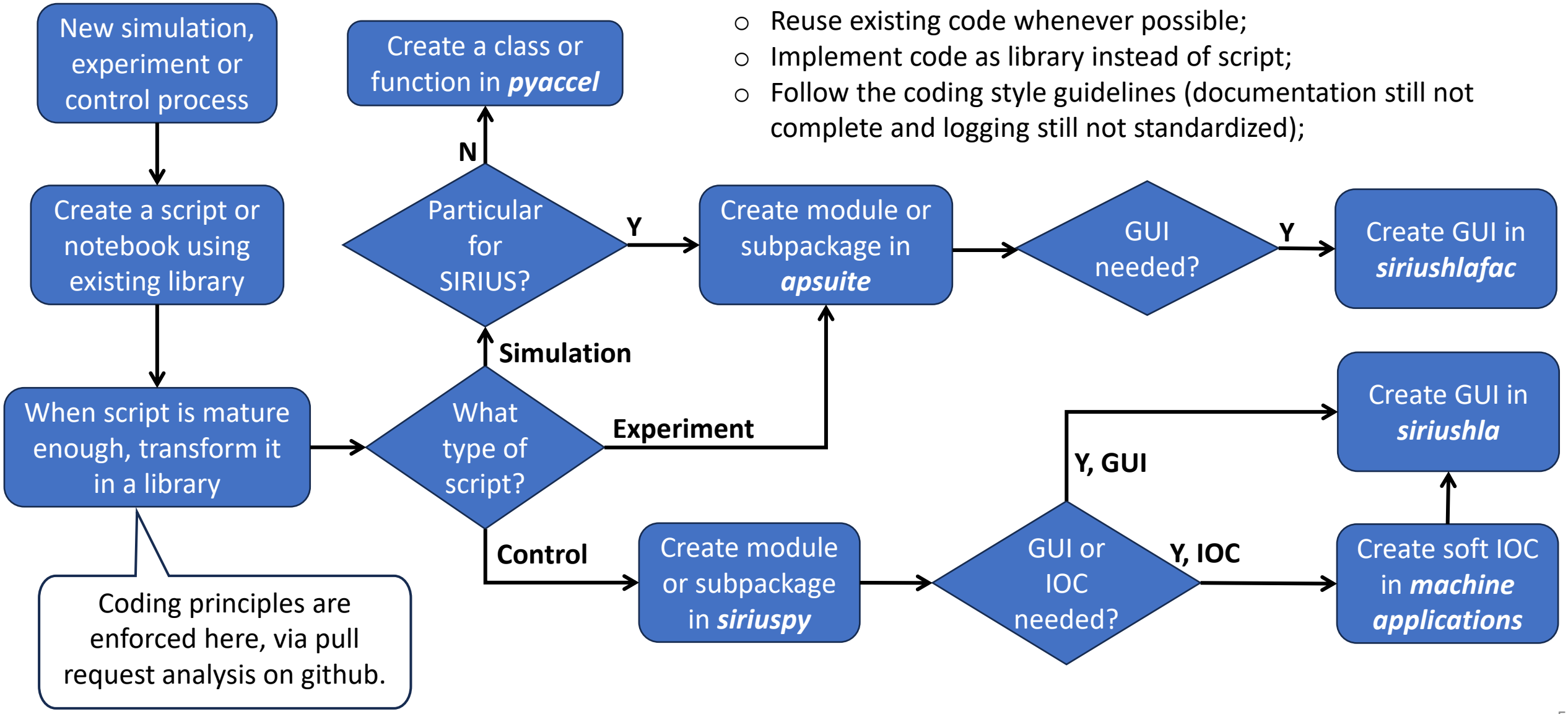
https://github.com/paulscherrerinstitute/pcaspy https://github.com/slaclab/pydm

Overview of SIRIUS Software



Python code development strategy

- Principles:
  - Reuse existing code whenever possible;
  - Implement code as library instead of script;
  - Follow the coding style guidelines (documentation still not complete and logging still not standardized);



Packages statistics

- All repositories are available at Github;
- Most of the code was developed over the last 10 years by 4 or 5 people, on average;
- Recently (last two years), developers' community inside LNLS has grown with the creation of a software group;
- Periodic deploy (~ 1 per month) in control room PCs with ansible;
- IOCs run in docker containers in dedicated workstations. Deploy whenever needed;

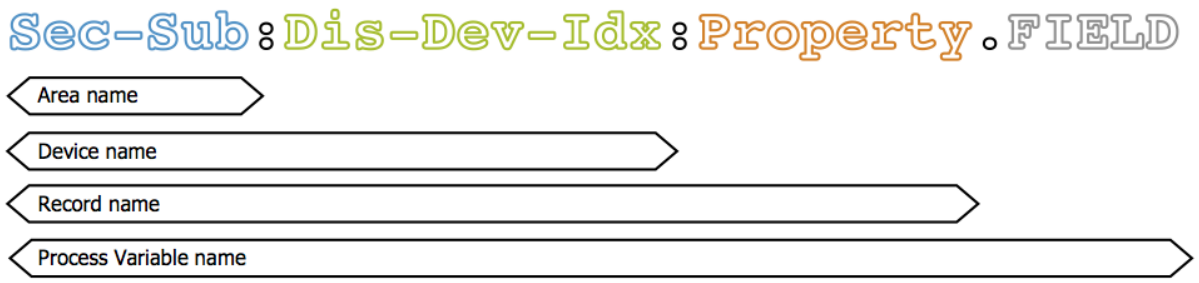
Package	Github page	Version	Pull Requests	Commits	Files	Lines
siriuspy	https://github.com/lnls-sirius/dev-packages	2.89.0	1062	9027	816	166934
siriushla	https://github.com/lnls-sirius/hla	1.1.0	674	4770	535	351650
machine-applications	https://github.com/lnls-sirius/machine-applications	3.48.0	285	2089	339	34831
apsuite	https://github.com/lnls-fac/apsuite	2.51.0	271	1872	96	29853
pyaccel	https://github.com/lnls-fac/pyaccel	3.18.1	93	613	43	14258
pymodels	https://github.com/lnls-fac/pymodels	1.18.1	84	661	43	8879
trackcpp	https://github.com/lnls-fac/trackcpp	4.10.4	48	323	67	39593
mathphys	https://github.com/lnls-fac/mathphys	2.9.0	26	233	25	7775
siriushlafac	https://github.com/lnls-fac/hlafac	0.10.1	22	336	17	1713

High-level control

Structure of *siriuspy*

- siriuspy
- clientconfigdb
- clientarch
- clientweb
- epics
- namesys
- devices
- ⋮
- Libs for specific control routines
- ⋮

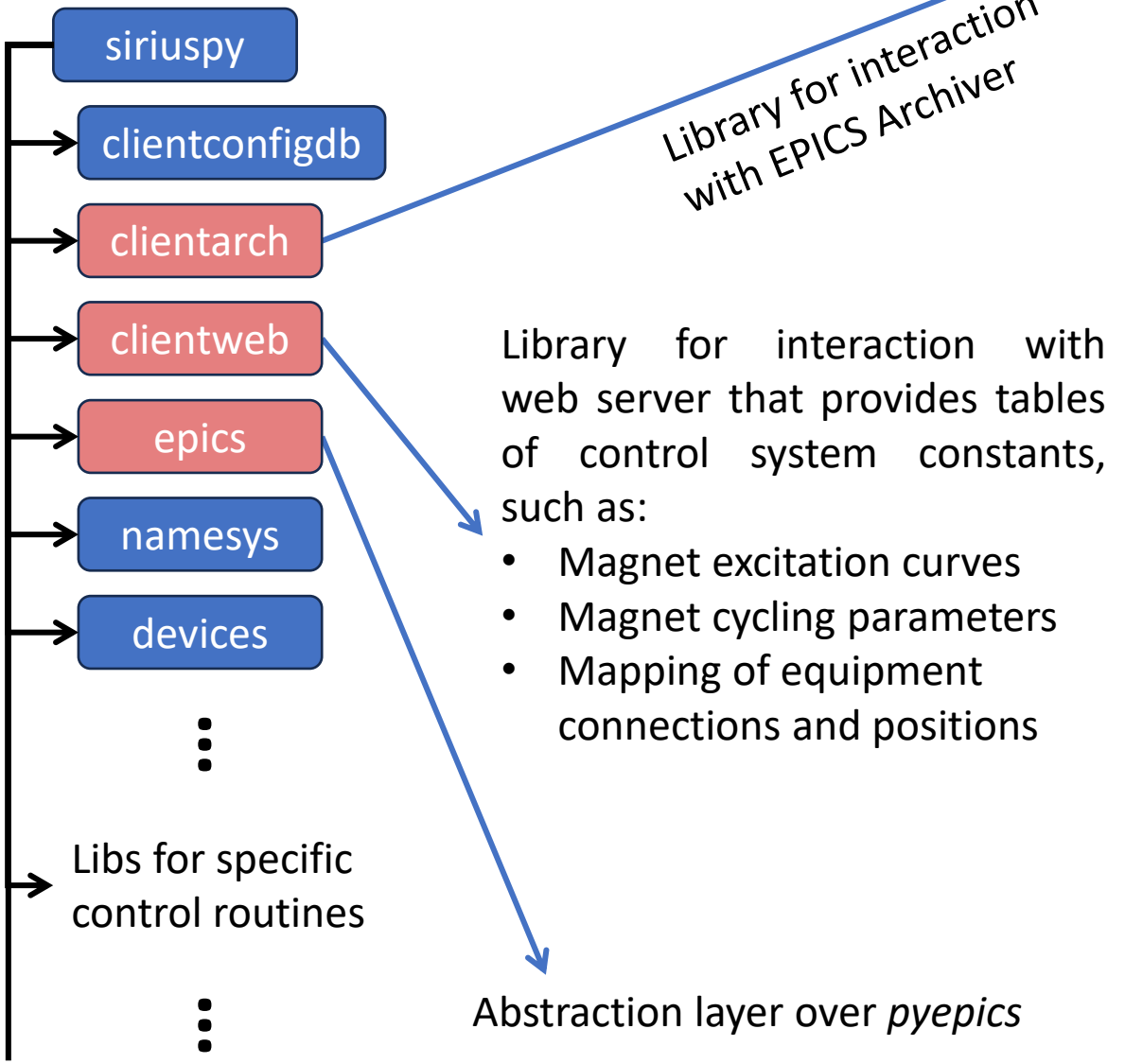
Library that transforms EPICS PV names into objects with properties that allow identification of the meaning of that PV, according to the naming system defined for SIRIUS.



Element	Description	Characters
Sec	Section	1 – 6
Sub	Subsection	
Dis	Discipline is the branch of knowledge indicating the context in which a device is used	1 – 6
Dev	Generic Device type. Two devices of the same generic device type provide the same function	1 – 12
Idx	The instance index is used to distinguish instances of devices of the same type in the same subsection and discipline.	0 – 6
Property	Signal Property, e.g. Current, time or temperature.	1 – 15
FIELD	The field is predefined for the EPICS record type. Example: value (VAL), units (EGU), alarmlimits (HIGH, LOW etc).	

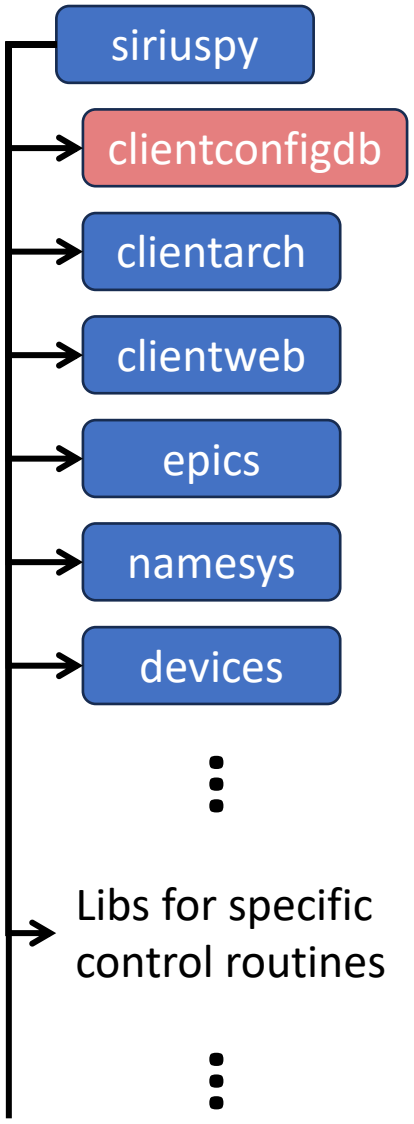
```
In [174]: from siriuspy.namesys import SiriusPVName

In [175]: pvn = SiriusPVName(
     ...:     'SI-01C1:PS-CH-1:Current-SP'
     ...: )

In [176]: isinstance(pvn, str)
Out[176]: True

In [177]: print_dir(pvn, tsiz=50)
is_cte_pv       dev             is_rb_pv
field           dis             substitute
idx             propty_name     propty_suffix
from_rb2sp      from_sp2rb      get_nickname
sub             device_name     is_sp_pv
area_name       prefix          sec
channel_type    is_cmd_pv       device_propty
propty          is_standard     is_write_pv

In [178]: pvn.sub
Out[178]: '01C1'

In [179]: pvn.device_name
Out[179]: 'SI-01C1:PS-CH-1'

In [180]: pvn.substitute(propty_suffix='RB')
Out[180]: 'SI-01C1:PS-CH-1:Current-RB'

In [181]: pvn.is_standard()
Out[181]: True

In [182]: pv2 = SiriusPVName(
     ...:     'SI-RF-DLLRF-01:PL:REF'
     ...: )

In [183]: pv2.is_standard()
Out[183]: False
```

Structure of *siriuspy*




```
In [115]: from siriuspy.clientarch import PVData, Time, PVDataSet, ClientArchiver

In [116]: clt = ClientArchiver()

In [117]: print_dir(clt)
DEFAULT_TIMEOUT          ENDPOINT                 SERVER_URL
connected                deletePVs                getAllPVs
getData                  getPVDetails             getPVsInfo
getPausedPVsReport       getRecentlyModifiedPVs   login
logout                   pausePVs                 renamePV
resumePVs                server_url               session
timeout

In [118]: pv = PVData('SI-Glob:AP-CurrInfo:Current-Mon')

In [119]: pv.time_start = Time(2024, 5, 6, 0, 10, 0)

In [120]: pv.time_stop = pv.time_start + 60 * 60 * 5  # 5 hours period

In [121]: pv.update()

In [122]: pv.value.shape, pv.timestamp.shape
Out[122]: ((149915,), (149915,))

In [123]: pvset = PVDataSet([
     ...:     'SI-Fam:PS-B1B2-1:Current-Mon',
     ...:     'SI-Fam:PS-B1B2-2:Current-Mon'
     ...: ])

In [124]: pvset.time_start = Time(2024, 5, 6, 0, 10, 0)

In [125]: pvset.time_stop = pvset.time_start + 60 * 60 * 5  # 5 hours period

In [126]: pvset.update()

In [127]: pvset[0].value.shape, pvset[1].value.shape
Out[127]: ((87799,), (87268,))
```

Structure of *siriuspy*

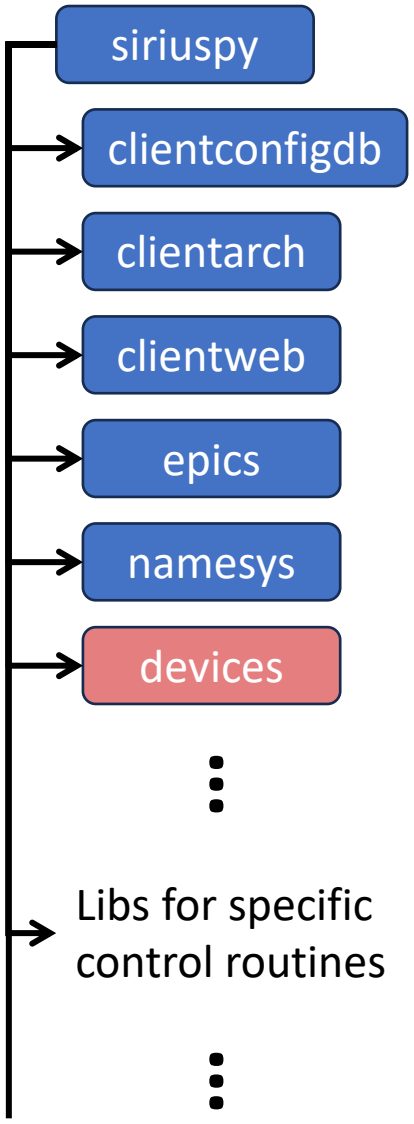


Library for interaction with web server that provides access and control of Mongo database where machine configuration files are saved.

Examples of configuration types:
- PVs defining machine state
- orbit response matrices
- orbits and trajectories of interest
- ID feedforward tables
- booster ramp parameters

```
In [2]: from siriuspy.clientconfigdb import ConfigDBClient

In [3]: clt = ConfigDBClient()

In [4]: print_dir(clt)
check_valid_configname          check_valid_value
compare_configs                 config_type
connected                       conv_timestamp_flt_2_txt
conv_timestamp_txt_2_flt        delete_config
find_configs                    get_config_info
get_config_types                get_config_types_from_templates
get_config_value                get_dbsize
get_nrconfigs                   get_value_from_template
insert_config                   rename_config
retrieve_config                 url


In [5]: mat = clt.get_config_value('ref_respmat',config_type='si_orbcorr_respm')

In [6]: np.array(mat).shape
Out[6]: (320, 281)

In [7]: reforb = clt.get_config_value('ref_orb', config_type='si_orbit')

In [8]: reforb.keys()
Out[8]: dict_keys(['x', 'y'])

In [9]: ref_conf = clt.get_config_value('ref_config', config_type='global_config')

In [10]: len(ref_conf['pvs'])
Out[10]: 2374
```

Structure of *siriuspy*



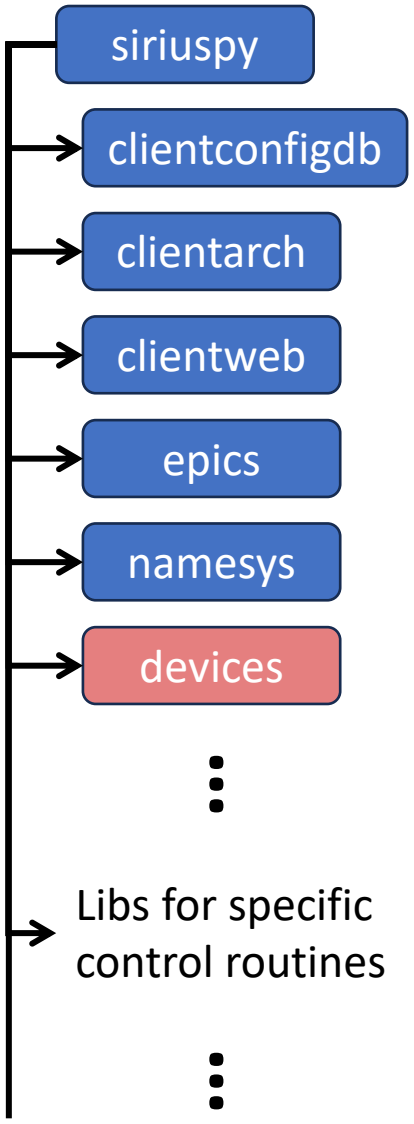
Interacts with a set of EPICS PVs from the same device or set of devices and group them in a single python object (similar to *ophyd* and class Device from *pyepics*).

```
In [248]: from siriuspy.devices import Device, Trigger, PowerSupply, SOFB

In [249]: print_dir(Device)
hosts                pv_ctrlvars          PROPERTIES_DEFAULT
pvnames              properties_in_use    pv_object
values               properties_all       disconnected_pvnames
set_auto_monitor     properties_added     CONNECTION_TIMEOUT
connected            devname              wait_for_connection
GET_TIMEOUT          pv_attribute_values  simulators
auto_monitor_status  PROPERTY_SEP         update
```

```
In [282]: from siriuspy.devices import DeviceSet, FamBPMs

In [283]: fambpm = FamBPMs(FamBPMs.DEVICES.SI, props2init='acq')

In [284]: isinstance(fambpm, DeviceSet)
Out[284]: True

In [285]: print_dir(fambpm)
RFFEATT_MAX                    PROPERTIES_ACQ
TIMEOUT                        wait_update_mturn_signals
get_switching_frequency        reset_mturn_flags
get_sampling_frequency         mturn_signals2acq
wait_update_mturn_flags        wait_acquisition_start
bpm_names                      csbpm
set_attenuation                calc_positions_from_amplitudes
DEVICES                        ALL_MTURN_SIGNALS2ACQ
get_slow_orbit                 set_switching_mode
devices                        wait_acquisition_finish
wait_update_mturn              get_mturn_timestamps
config_mturn_acquisition       update_mturn_initial_timestamps
cmd_abort_mturn_acquisition    bpms
reset_mturn_initial_state      get_mturn_signals
update_mturn_initial_signals   cmd_start_mturn_acquisition
wait_update_mturn_timestamps

In [286]: fambpm[0]
Out[286]: <siriuspy.devices.bpm.BPM at 0x7fc356489790>
```

```
In [267]: trig = Trigger('SI-01SA:TI-InjNLKckr')

In [268]: isinstance(trig, Device)
Out[268]: True

In [269]: print_dir(trig)
STATES                 status_str             cmd_lock_low_level
status                 controlled_channels    lock_low_level
cmd_enable             status_labels          cmd_unlock_low_level
source_options         total_delay_raw        duration
source                 nr_pulses              polarity_str
polarity               delay                  delay_raw
state_str              POLARITIES             lock_low_level_str
width_raw              LOCKLL                 total_delay
is_in_inj_table        delta_delay            state
low_level_triggers     cmd_disable            source_str
delta_delay_raw

In [270]: trig.delay  # [us]
Out[270]: 294619.6935363579

In [271]: trig['Delay-RB']  # create new PV if not connected yet.
Out[271]: 294619.6935363579

In [272]: trig.delay_raw  # hardware units
Out[272]: 36802975.0
```

Structure of *siriuspy*



Interacts with a set of EPICS PVs from the same device or set of devices and group them in a single python object (similar to *ophyd* and class Device from *pyepics*).

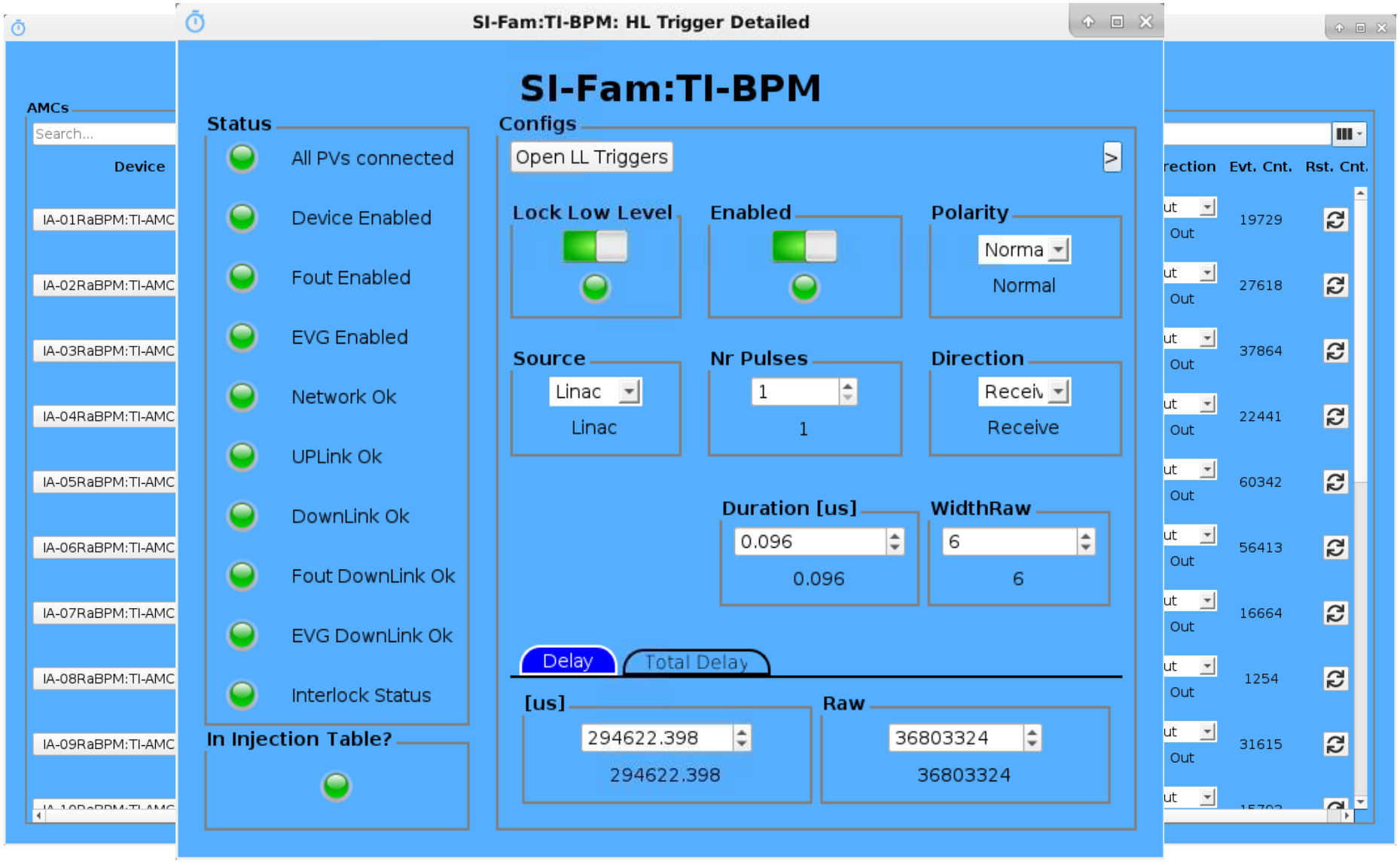
```
In [248]: from siriuspy.devices import Device, Trigger, PowerSupply, SOFB

In [249]: print_dir(Device)
hosts                pv_ctrlvars          PROPERTIES_DEFAULT
pvnames              properties_in_use    pv_object
values               properties_all       disconnected_pvnames
set_auto_monitor     properties_added     CONNECTION_TIMEOUT
connected            devname              wait_for_connection
GET_TIMEOUT          pv_attribute_values  simulators
auto_monitor_status  PROPERTY_SEP         update
```

```
In [287]: import siriuspy.devices as devices

In [288]: print_dir(devices, tsiz=120)
ID                       SIRFACAmp              BLInterlockCtrl        FamFOFBLamp            BunchbyBunch
FOFBCtrlSysId            TuneCorr               DevicesSync            FOFBPSLamp             BPM
WIG                      afc_acq_core           DeviceSet              CurrInfoSI             LIModltr
FOFBCtrlRef              PSCorrSOFB             StrengthConv           InjSysStandbyHandler   DevLILLRF
Energy                   FamFastCorrs           PSProperty             Tune                   MachShift
BORFRampStandbyHandler   ScraperV               EqualizeBPMs           EVG                    RFKillBeam
orbit_interlock          EGFilament             PowerSupplyPU          RFGen                  EGPulsePS
IDBase                   LinacStandbyHandler    DELTA                  SIRFDCAmp              FOFBCtrlLamp
EGun                     Trigger                Device                 AFCACQLogicalTrigger   APU
idff                     PowerSupplyFBP         EPU                    ASPPSCtrl              fofb
FamFOFBControllers       CurrInfoLinear         fofb_acq               bpm_eq                 Screen
LILLRF                   TuneFrac               DCCT                   HLTiming               FOFBPSSysId
FOFBCtrlDCC              DVFImgProc             BORFCavMonitor         PosAng                 EGTriggerPS
BaseOrbitIntlk           TranspEff              InjSysPUModeHandler    SILLRFPreAmp           SIRFCavMonitor
BLM                      BORF300VDCAmp          AFCPhysicalTrigger     InjCtrl                BOLLRFPreAmp
PowerSupplyFC            DVF                    OrbitInterlock         SOFB                   FamFOFBSysId
ICT                      ASLLRF                 PUMagsStandbyHandler   CurrInfoTranspEff      RFCav
FamBPMs                  PAPU                   PowerSupply            EGBias                 HLFOFB
EGHVPS                   PSApplySOFB            BORFDCAmp              TuneProc               ScraperH
BPMDCC                   BPMOrbitIntlk          ASMPSCtrl              BOPSRampStandbyHandler IDFF
bpm_fam                  CurrInfoAS             LIEnergy               Event                  CurrInfoBO
```

Examples of high-level control tasks

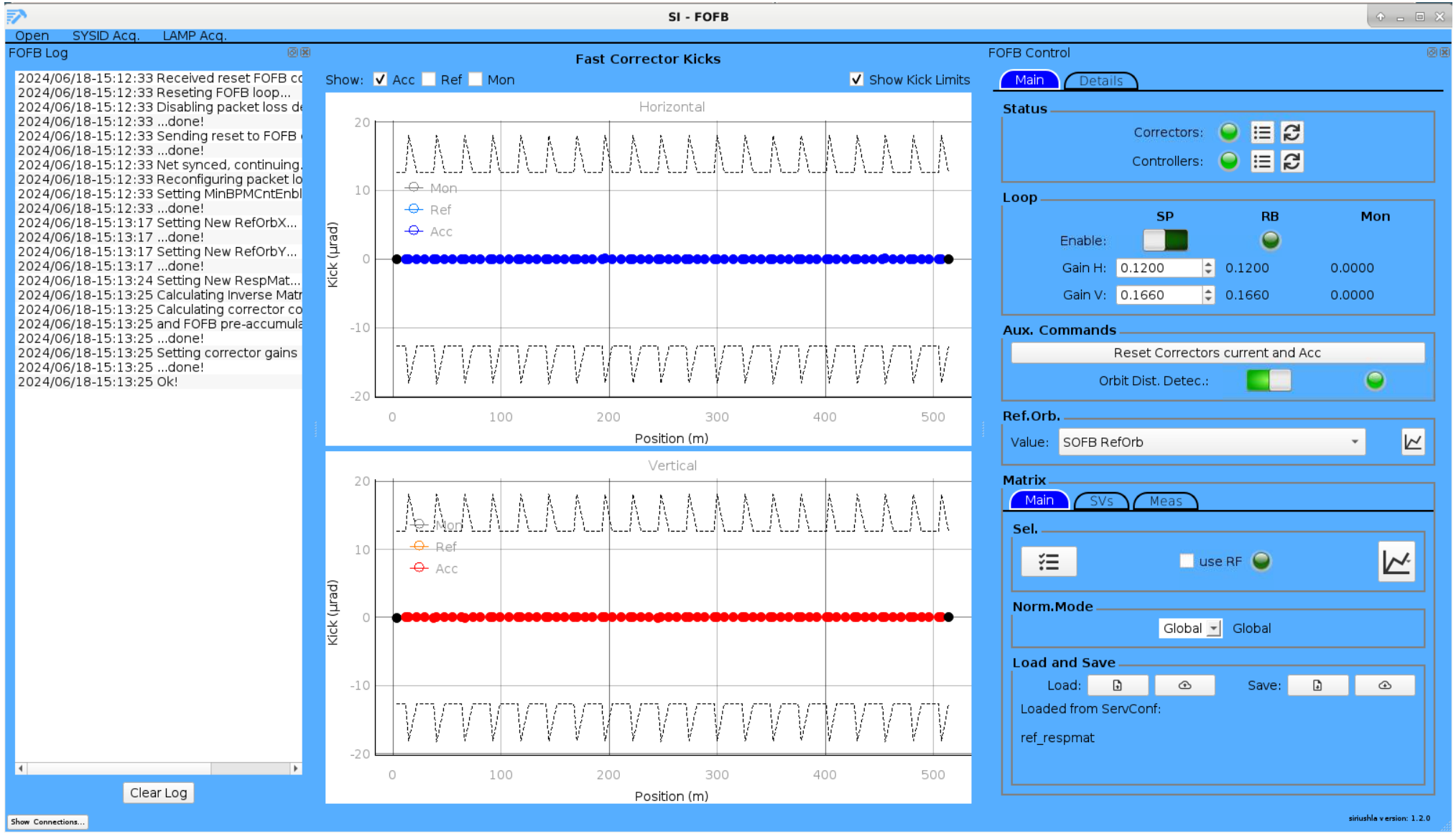
Common Tasks Performed by Soft IOCs

- Distributed systems architecture abstraction and orchestration: **timing**, FOFB, orbit interlock;



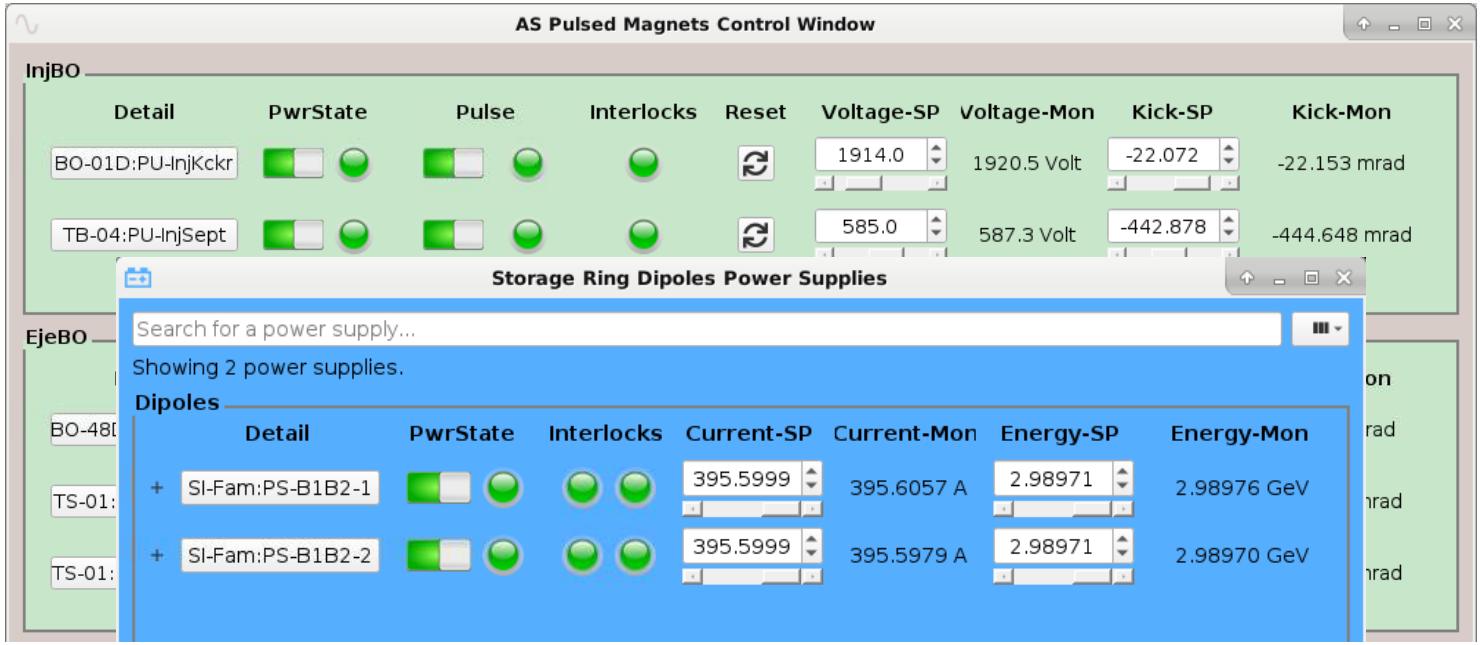
Common Tasks Performed by Soft IOCs

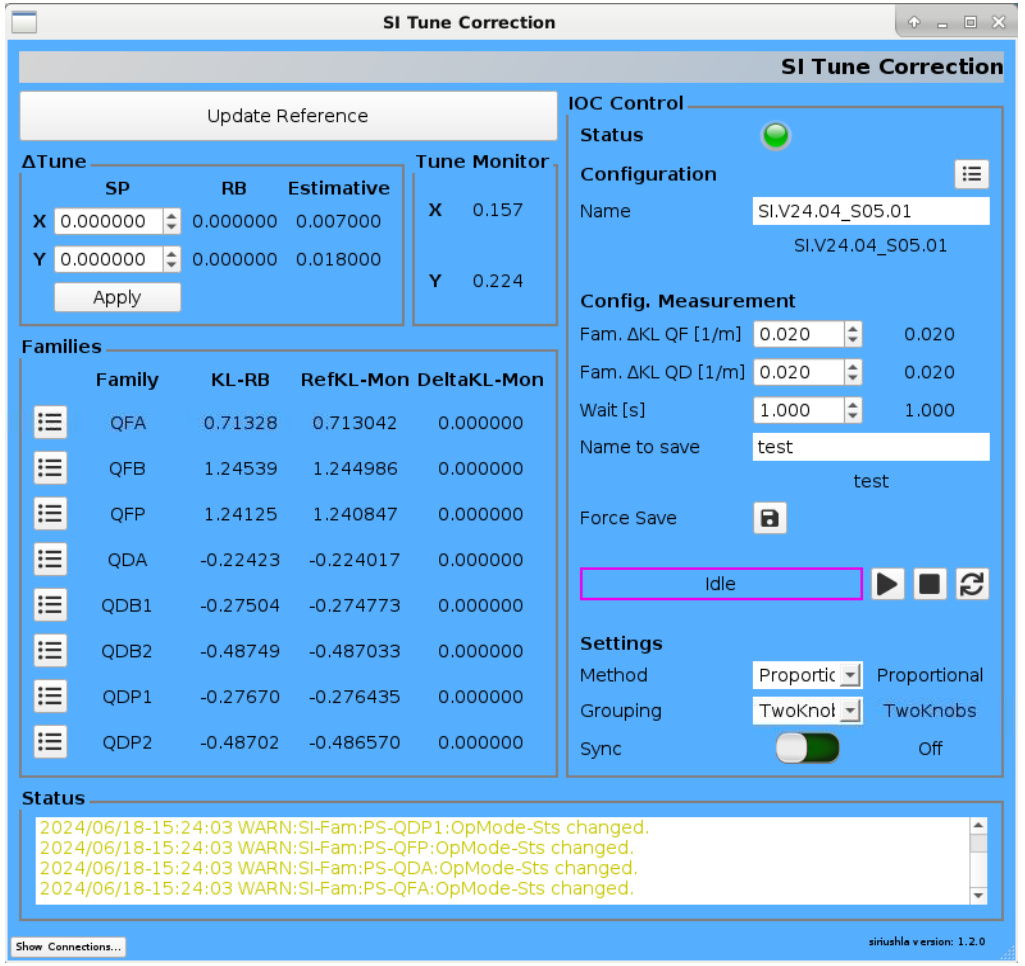
- Distributed systems architecture abstraction and orchestration: timing, **FOFB**, orbit interlock;



Common Tasks Performed by Soft IOCs

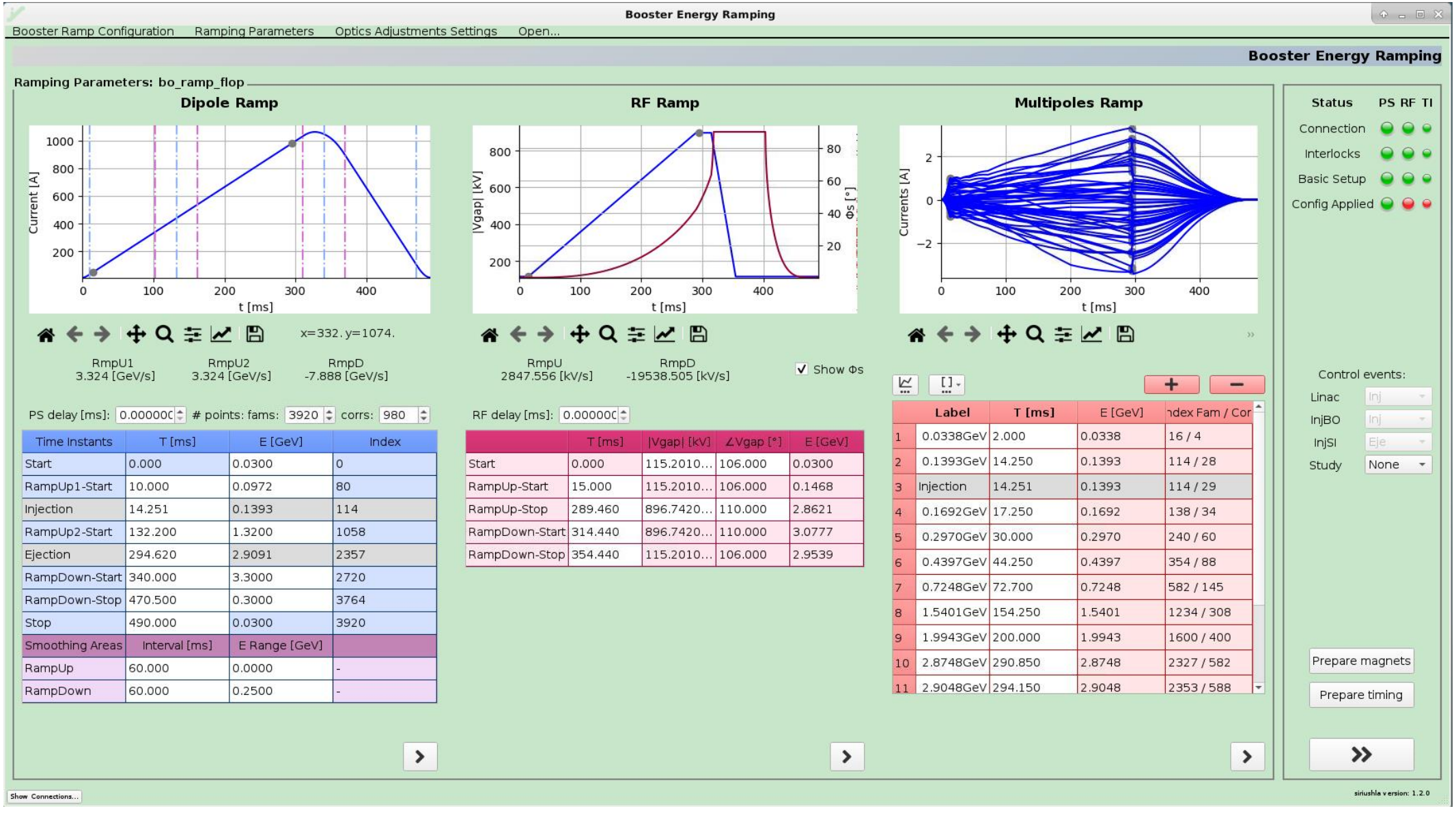
- Distributed systems architecture abstraction and orchestration: timing, FOFB, orbit interlock;
- **Conversion from hardware to physics units**: power supplies, **pulsed magnets**, etc;
- Subsystems diagnostics: power supplies, RF, pulsed power supplies, LINAC devices, etc.;
- Calculation of lifetime, integrated stored current, injected current, etc.;
- Slow orbit correction and orbit response matrix measurement;
- Injection trajectory (TbT) analysis and correction;
- Calculation of injection efficiency between accelerators;
- Correction of position and angle of injected beam;
- **Tune** and chromaticity **correction**;
- ...





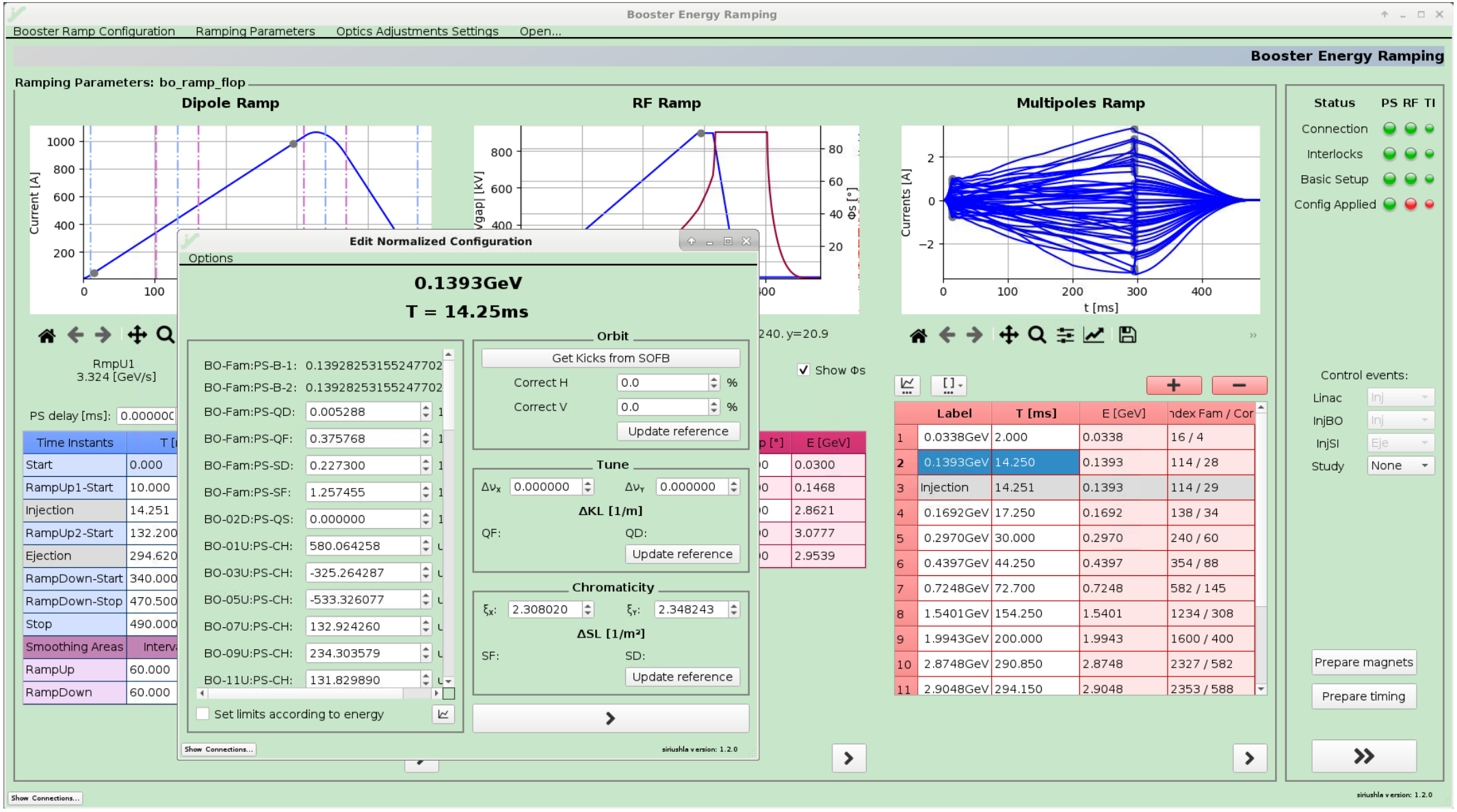
Common Tasks Performed by GUIs

- Booster ramp configurations management and tuning;



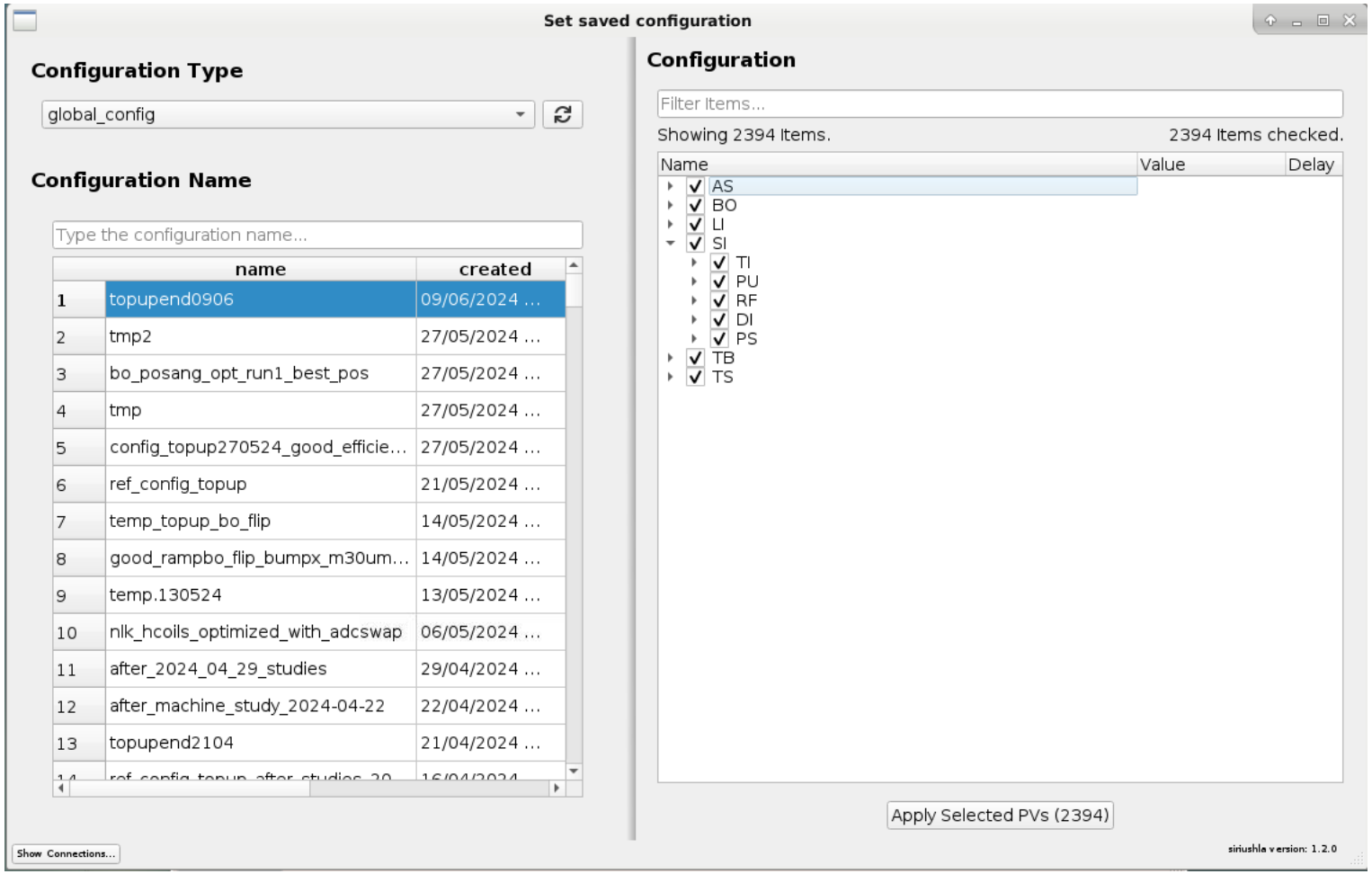
Common Tasks Performed by GUIs

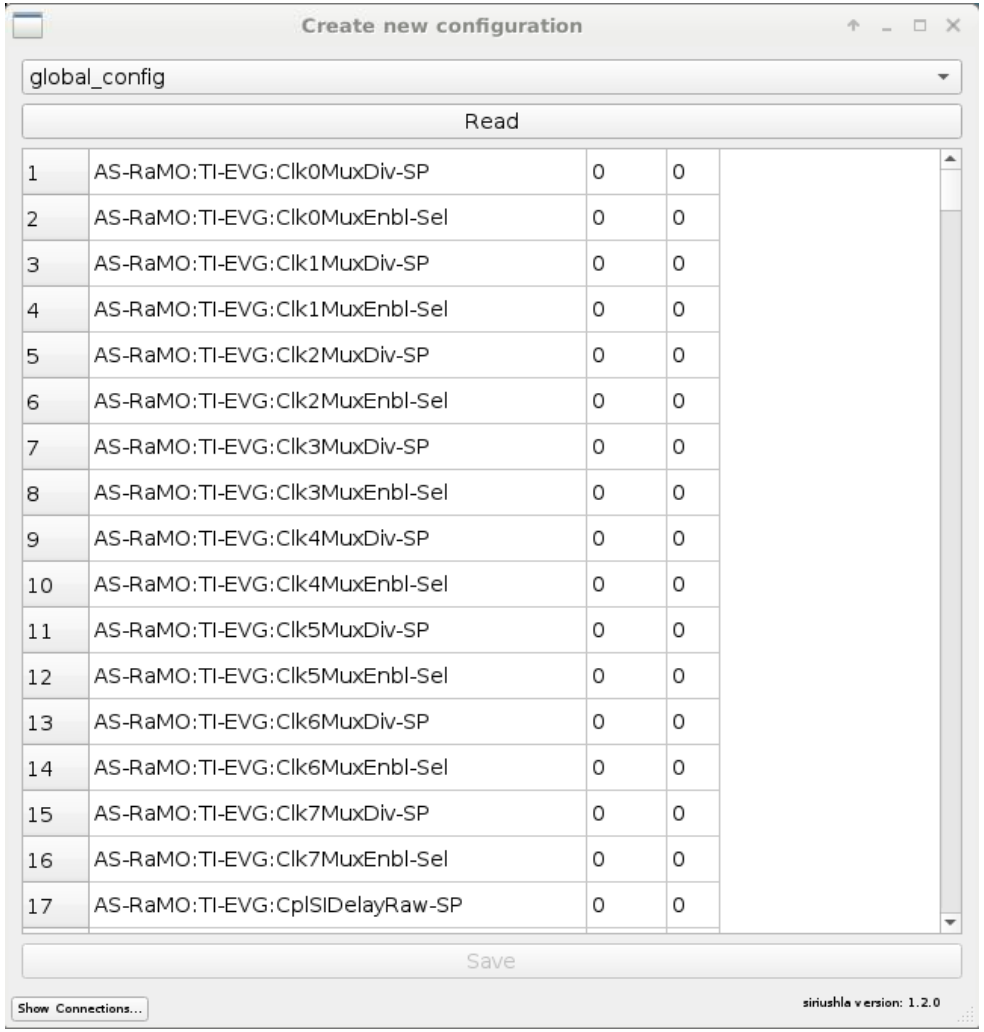
- Booster ramp configurations management and tuning;



Common Tasks Performed by GUIs

- Booster ramp configurations management and tuning;
- Power supplies testing, turn-off and turn-on routines;
- **Save & restore machine configurations;**
- Magnets cycling and standardization;





Experiments control

Structure of *apsuite*

- apsuite
  - optimization → Contains classes of generic optimization algorithms (e.g., RCDS, Simulated annealing, scanning, PSO)
  - commisslib → Contains classes to drive specific experiments
  - utils.py → Defines base classes for *optimization* and *commisslib*. Standardize how data and metadata is saved and loaded.
  - Simulation and LOCO packages

```
You, 6 months ago | 2 authors (You and one other)
class DataBaseClass:
    """."""

    def __init__(self, params=None):
        """."""
        self.data = dict()
        self.params = params

    def to_dict(self) -> dict: ...

    def from_dict(self, info: dict): ...

    def save_data(self, fname: str, overwrite=False, compress=False): ...

    def load_and_apply(self, fname: str): ...

    @staticmethod
    def load_data(fname: str): ...
```

```
You, 6 months ago | 2 authors (You and one other)
class ParamsBaseClass:
    """."""

    def to_dict(self): ...

    def from_dict(self, params_dict): ...
```

```
murilobalves, 3 years ago | 2 authors (You and one other)
class MeasBaseClass(DataBaseClass):
    """."""

    def __init__(self, params=None, isonline=True):
        """."""
        super().__init__(params=params)
        self.isonline = bool(isonline)
        self.devices = dict()
        self.analysis = dict()
        self.pvs = dict()

    @property
    def connected(self): ...

    def wait_for_connection(self, timeout=None): ...
```

```
murilobalves, 16 months ago | 3 authors (You and others)
class ThreadedMeasBaseClass(MeasBaseClass):
    """."""

    def __init__(self, params=None, target=None, isonline=True): ...

    @property
    def target(self): ...

    @target.setter
    def target(self, func): ...

    def start(self): ...

    def stop(self): ...

    @property
    def ismeasuring(self): ...

    def wait_measurement(self, timeout=None): ...

    def _run(self): ...
```

Data organization for experiments

- Shared folder among all PCs of the control room;
- Also mounted read-only in our PCs;
- Backup saved every day;
- All experiments should have its own folder:
  - With name starting with the date and a short description;
  - Should contain script that ran the experiment and data + figures + processing files + etc. created;
- Easy to find similar experiments done in the past;
- Reuse and adapt old scripts made in previous experiments;

```
lnls556-linux:data_by_day$ pwd
/home/fernando/shared/screens-iocs/data_by_day
lnls556-linux:data_by_day$ ls
2023-01-16-SI_bba                                    2023-09-24-SI_mach_shutdown
2023-01-16-SI_orbit-stability-leakfield              2023-09-25-SI_BbB_timing_setup
2023-01-17-SI-optics_analysis_LOCO                   2023-09-25-SI_orbit_stability
2023-01-17-SI_orbit-stability-leakfield-orbitacquisition  2023-10-01-SI_mach_shutdown
2023-01-23-SI_epu-characterization                   2023-10-02-SI_FOFB_SYSID
2023-01-24-SI_FOFB_anthenas_gains                    2023-10-02_SI_llrf
2023-01-30_SI_FOFB_50kHz                             2023-10-03-SI_FOFB
2023-02-06-BO_propagated_dispersion_TBBO             2023-10-03-SI_vertical_dispersion_correction
2023-02-06-LI_energy_emittance_meas                  2023-10-09-SI-gap_voltage_calibration
2023-02-06-SI_orbit_stability                        2023-10-09-SI_injection_collimation_with_scrapers
```

Examples of experiments

Experiments performed by scripts/notebooks

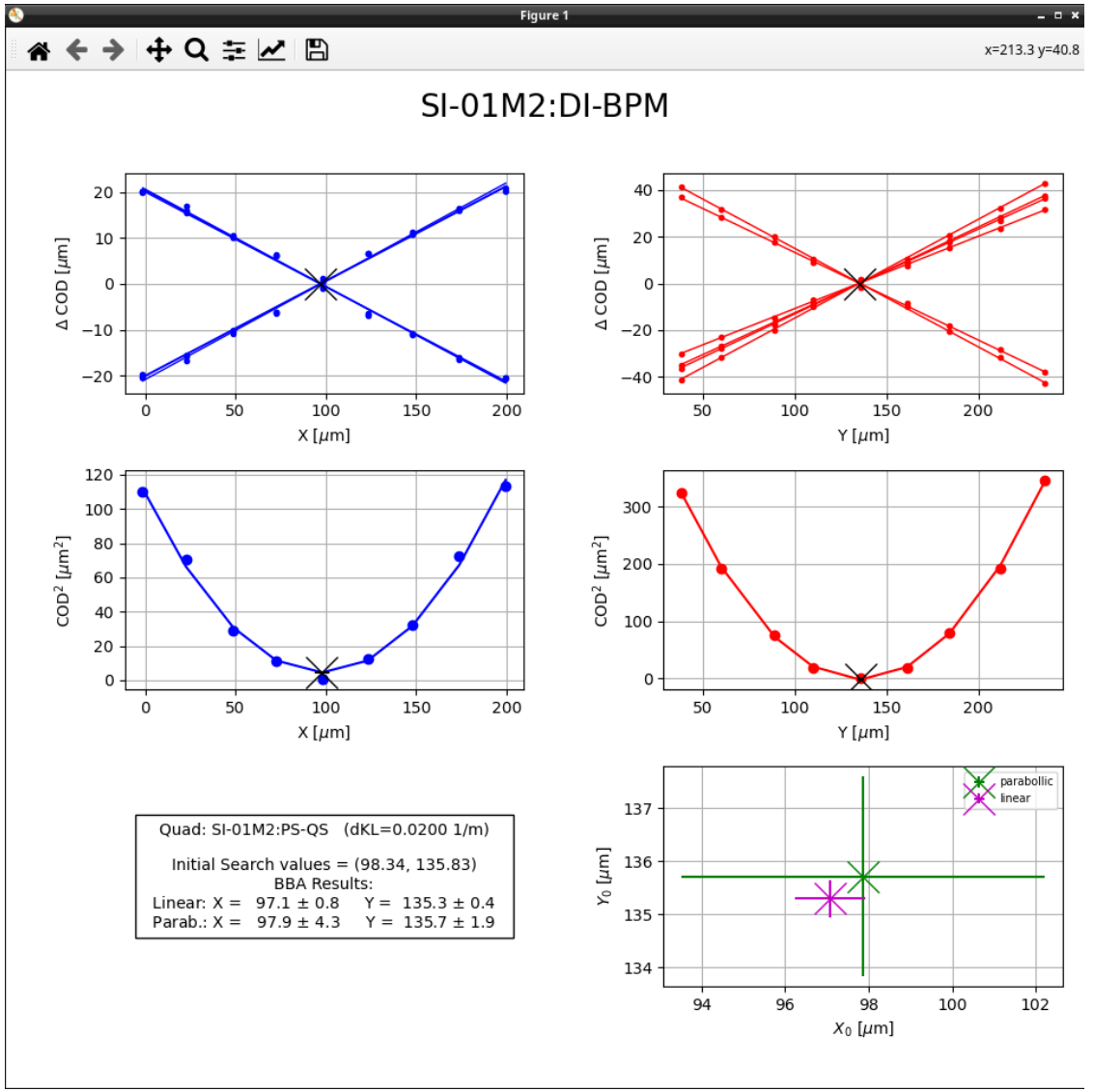
- Acquisition of synchronized fast orbit or trajectory (TbT) from all BPMs;
- Measurement of orbit response matrix using AC excitation of correctors;
- LOCO analysis;
- **BBA of storage ring BPMs**;

```python
#!/usr/bin/env python-sirius
""".""

import time

from apsuite.commisslib.measure_bba import DoBBA
from siriuspy.clientconfigdb import ConfigDBClient

fname = "bba_after_2024_06_17_shutdown"
# Use SOFB GUI to correct orbit and save it in
# servconf with name below.
ref_orb = f"ref_orb_{BBA_FNAME:s}"

if __name__ == "__main__":
    print(f"loading si_orbit: {ref_orb}")
    cltorb = ConfigDBClient(config_type="si_orbit")
    orb = cltorb.get_config_value(ref_orb)

    dobba = DoBBA()
    dobba.params.deltaorbx = 100
    dobba.params.deltaorby = 100
    dobba.params.wait_correctors = 0.3
    dobba.params.wait_quadrupole = 0.5
    dobba.params.quad_deltakl = 0.02
    dobba.params.sofb_nrpoints = 20
    dobba.params.sofb_maxorberr = 5
    dobba.data["scancenterx"] = orb["x"]
    dobba.data["scancentery"] = orb["y"]
    dobba.bpms2dobba = dobba.data['bpmnames']

    dobba.wait_for_connection()
    time.sleep(2)
    print(dobba)
    print(80 * "#")
    print("Starting BBA")
    dobba.start()
    while True:
        if dobba.wait_measurement(2 * 60):
            break
        dobba.save_data(fname, overwrite=True)
    dobba.save_data(fname, overwrite=True)
```

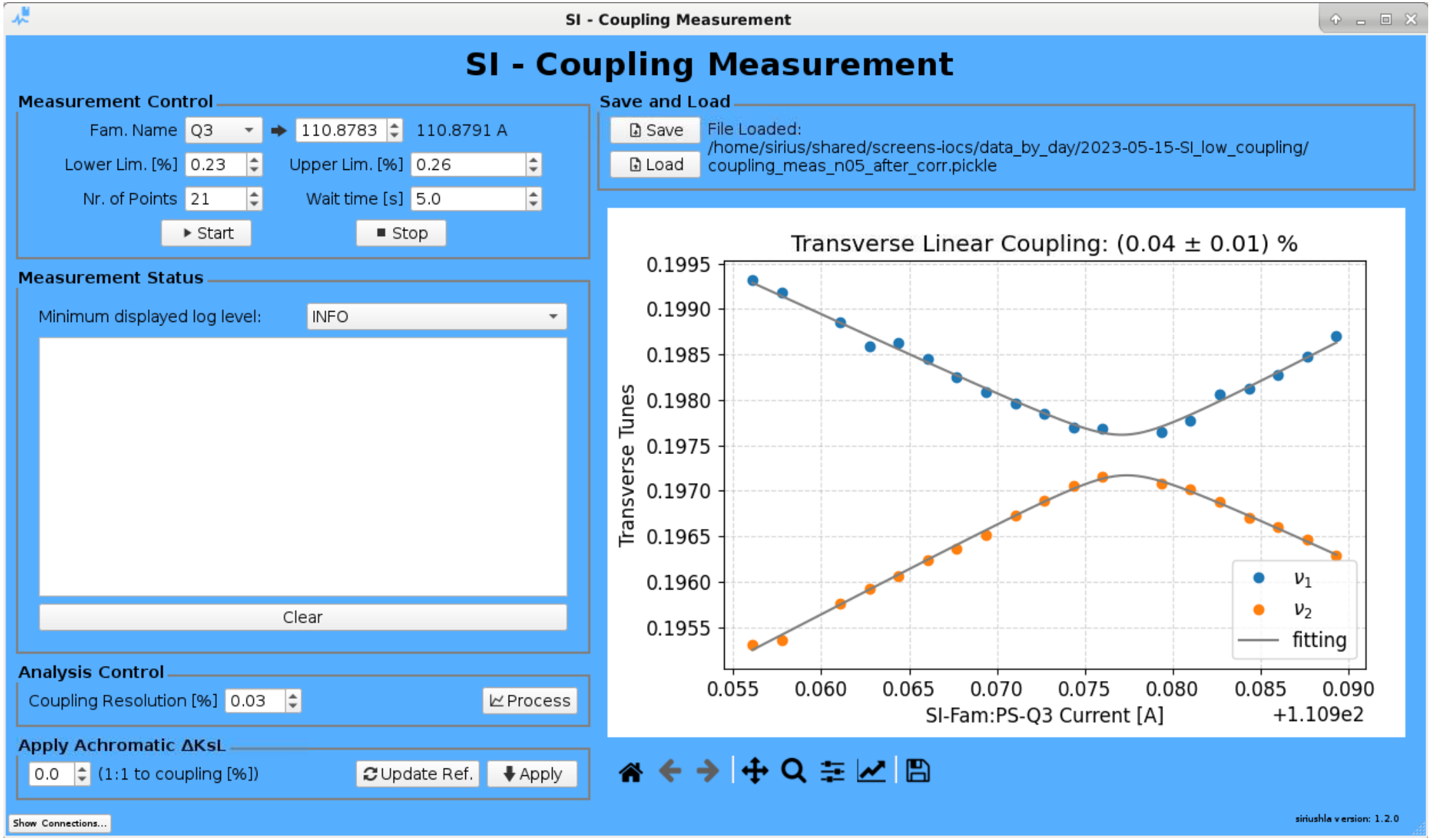
Experiments performed by scripts/notebooks

- Acquisition of synchronized fast orbit or trajectory (TbT) from all BPMs;
- Measurement of orbit response matrix using AC excitation of correctors;
- LOCO analysis;
- **BBA of storage ring BPMs**;

```
lnls556-linux:2024-06-17-SI_bba$ ipython-mamba-sirius
Python 3.9.2 | packaged by conda-forge | (default, Feb 21 2021, 05:02:46)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.18.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from mathphys.functions import load

In [2]: data = load('bba_after_2024_06_17_shutdown.pickle')

In [3]: data.keys()
Out[3]: dict_keys(['data', 'params'])

In [4]: data['data'].keys()
Out[4]: dict_keys(['bpmnames', 'quadnames', 'scancenterx', 'scancentery', 'measure'])

In [5]: data['params'].keys()
Out[5]: dict_keys(['deltaorbx', 'deltaorby', 'meas_nrsteps', 'quad_deltakl', 'quad_nrcycles',
'wait_correctors', 'wait_quadrupole', 'timeout_wait_orbit', 'sofb_nrpoints', 'sofb_maxcorriter
', 'sofb_maxorberr'])

In [6]: from apsuite.commisslib.measure_bba import DoBBA

In [7]: dobba = DoBBA(isonline=False)

In [8]: dobba.load_and_apply('bba_after_2024_06_17_shutdown.pickle')

In [9]: dobba.process_data(mode='symm')

In [10]: dobba.make_figure_bpm_summary(dobba.params.BPMNAMES[0])
```



Experiments performed by GUIs



Summary

- Python is the most used language for the SIRIUS high-level control system:
  - Span over simulation, experiments and control;
  - Provides a very versatile and flexible environment for developing new tools and perform complex tasks;

- The code development workflow from scripting to library or GUI is well-defined and encouraged:
  - Code duplication is very rare;
  - Creation of applications for non-expert users is possible;
  - Use of basic libraries in scripts still possible;

- Experiments control and data organization:
  - *apsuite* defines a standard way of saving data and metadata and creating experiment drivers;
  - Current methods does not handle partial loading of saved data. This feature would be useful;
  - Use of shared folder in control room PCs works well to separate and store experiment data, but does not provide security against accidental data losses;

Summary

- *siriuspy* works well as an architecture and control system abstraction layer:
  - *devices* and *epics* subpackages abstract EPICS;
  - *clientconfigdb* and *clientweb* abstract configurations and constants servers.
  - Several high-level control tasks already performed by soft IOCs or servers;

- Regarding virtual simulators:
  - Seamless exchange between real machine and virtual accelerator (VACA) via VACA_PREFIX environment variable existed in the past. However, VACA was discontinued a few years ago;
  - Use of virtual twin simulator is desired, but most of the measurements we develop nowadays involves acquisition of synchronized fast orbit, TbT, or bunch-by-bunch data. A virtual twin in a future middle-layer with such capabilities would be very useful.

- In my perspective, the decision we took 10 years ago to move towards python and away from MATLAB really paid off!

- I hope we can contribute with the community in the creation of a general python middle layer.

Thank you for your attention!

Thanks also to Ana Clara Oliveira for helping with the preparation of this presentation!