Taurus Status and Update

by

Carlos Pascual-Izarra

(On behalf of the Taurus community)

Introduction

What is Taurus

Taurus Structure

Taurus4 development (timeline)

Changes in taurus.core

Simplified, agnostic API

New model naming (validators and fragments)

Standardized values and units support

Backwards-compatibility

Changes in taurus.qt

New-style signals

Avoid icon resource files

Replacing Qwt dependency

Improving Community & Infrastructure

Transition to setuptools

Improving contribution workflow

Future priorities

Introduction

What is Taurus

Taurus Structure

Taurus4 development (timeline)

Changes in taurus.core

Simplified, agnostic API

New model naming (validators and fragments)

Standardized values and units support

Backwards-compatibility

Changes in taurus.qt

New-style signals

Avoid icon resource files

Replacing Qwt dependency

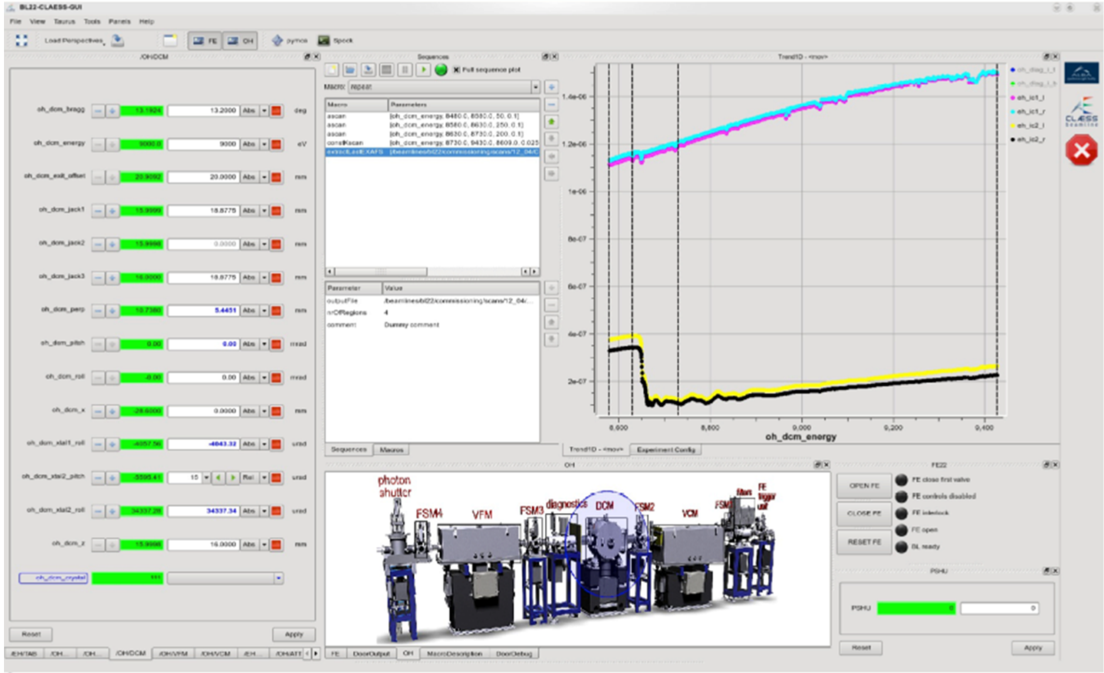
Improving Community & Infrastructure

Transition to setuptools

Improving contribution workflow

Future priorities

Taurus is...



*"**Taurus** is a **python** framework for control and data acquisition **CLIs** and **GUIs** in scientific/industrial environments. It supports multiple control systems or data sources: **Tango**, **EPICS**, **Spec**... New control system libraries can be integrated through plugins."*

- Widely used
- Production-ready
- Well supported
- Actively developed
- Free/Open Source
- Community-driven
- Modular
- Multi-platform
- Based on Python and Qt
- Easy to install

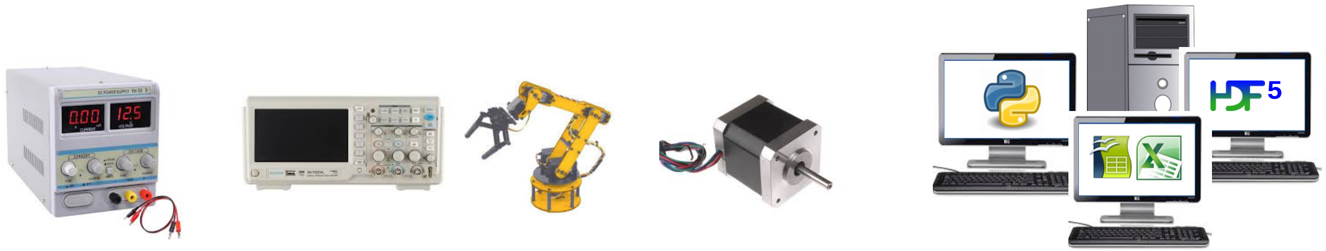
ALBA ESRF DESY MAX IV SOLARIS NATIONAL SYNCHROTRON RADIATION CENTRE
KIT Karlsruhe Institute of Technology eli extreme light infrastructure INSA RENNES CMAM cea
NEXEYA COSYLAB ONERA THE FRENCH AEROSPACE LAB OBSERVATORY SCIENCES PicoQuant TATA TATA CONSULTANCY SERVICES 3controls

Structure of Taurus

TaurusGUIs

TaurusGUIs

External Hardware and data sources

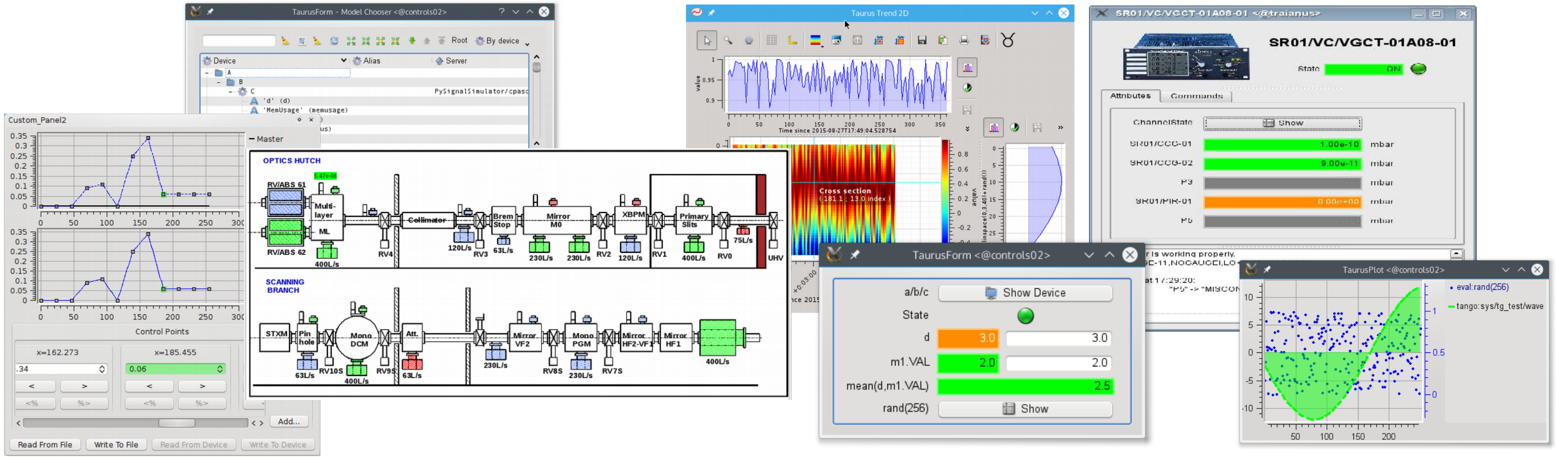


Structure of Taurus

TaurusGUIs

TaurusGUIs

Taurus Qt Widgets



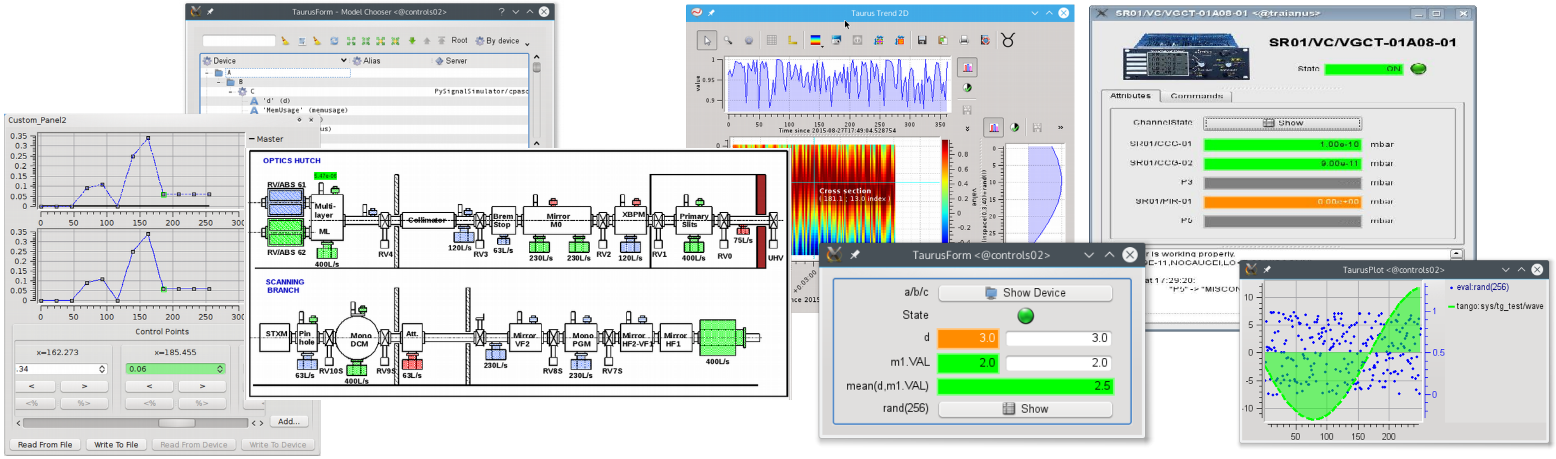
External Hardware and data sources

Structure of Taurus

TaurusGUIs

TaurusGUIs

Taurus Qt Widgets



Taurus Core

Taurus Core

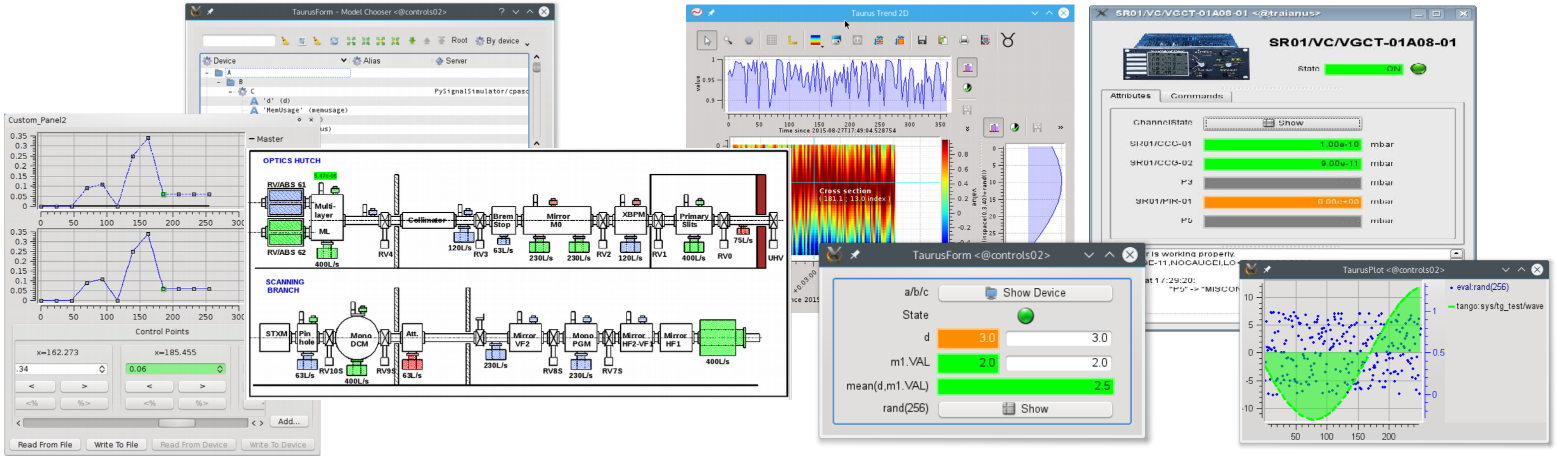
External Hardware and data sources

Structure of Taurus

TaurusGUIs

TaurusGUIs

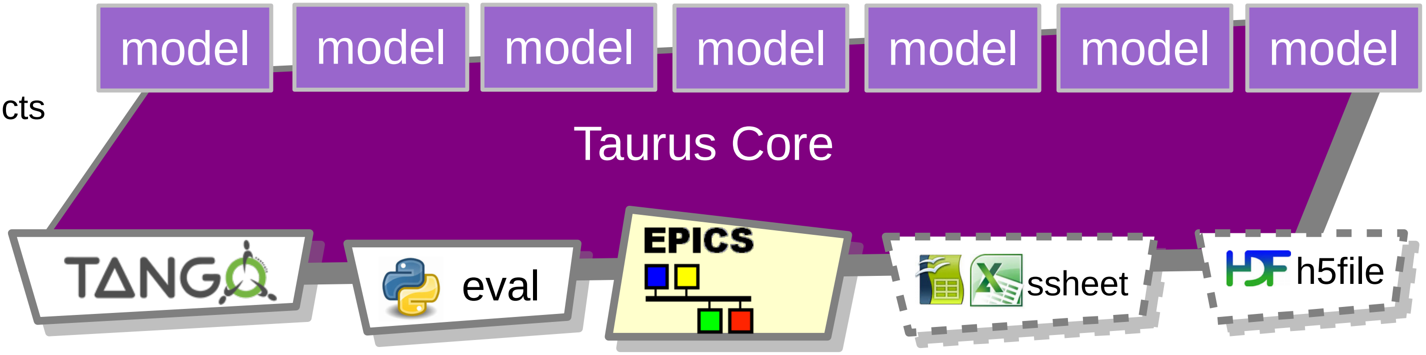
Taurus Qt Widgets



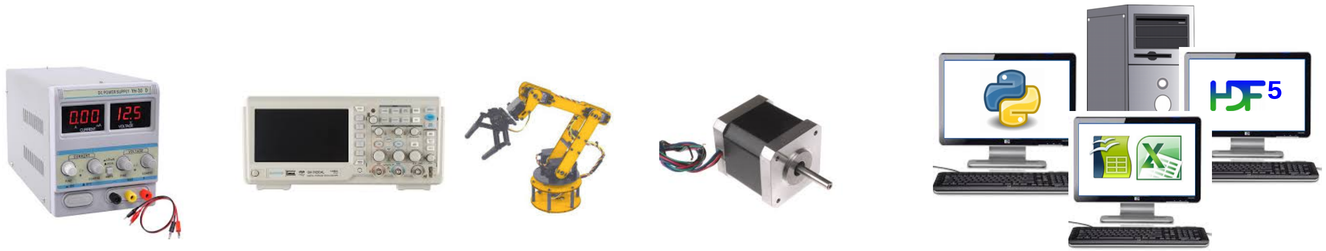
Taurus Core

Model Objects

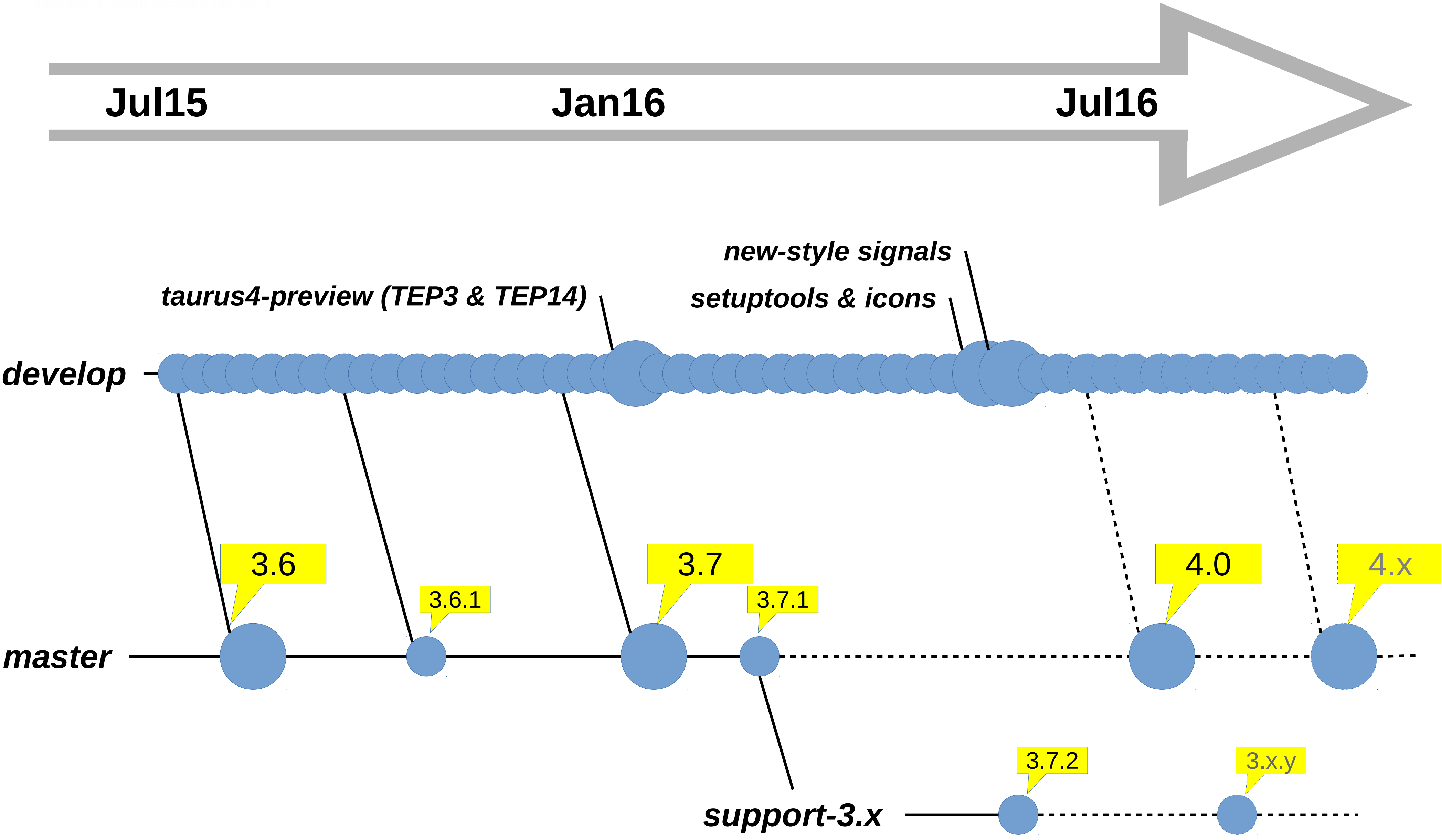
Schemes



External Hardware and data sources



Development timeline



Introduction

What is Taurus

Taurus Structure

Taurus4 development (timeline)

Changes in taurus.core

Simplified, agnostic API

New model naming (validators and fragments)

Standardized values and units support

Backwards-compatibility

Changes in taurus.qt

New-style signals

Avoid icon resource files

Replacing Qwt dependency

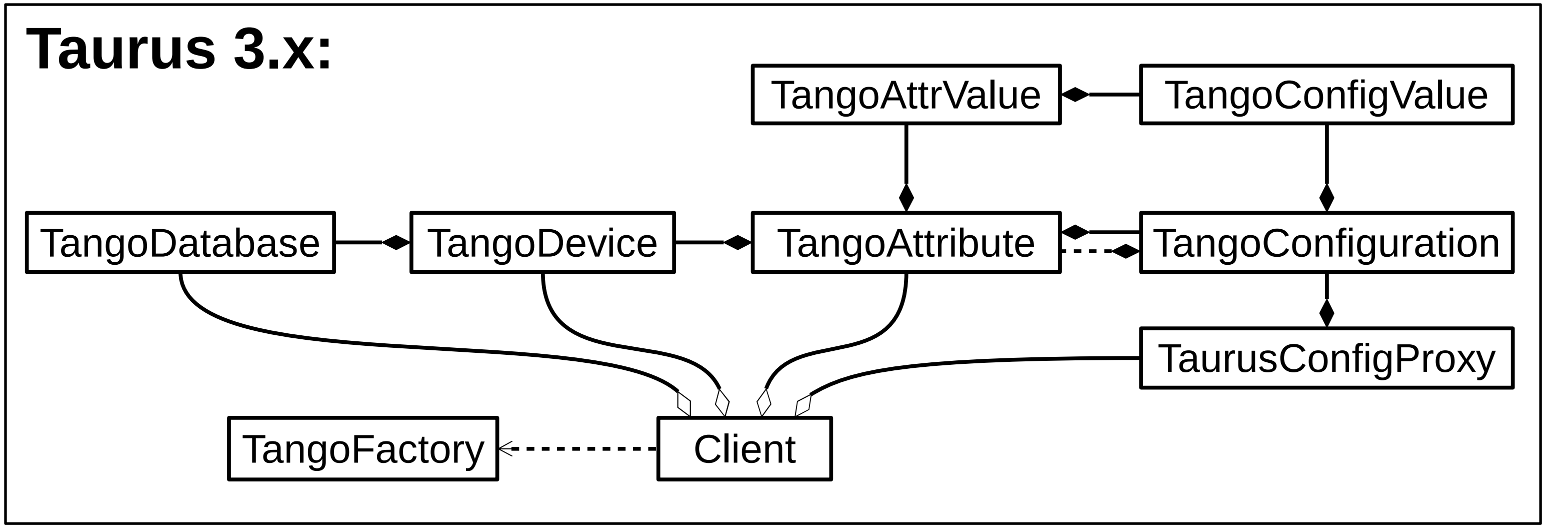
Improving Community & Infrastructure

Transition to setuptools

Improving contribution workflow

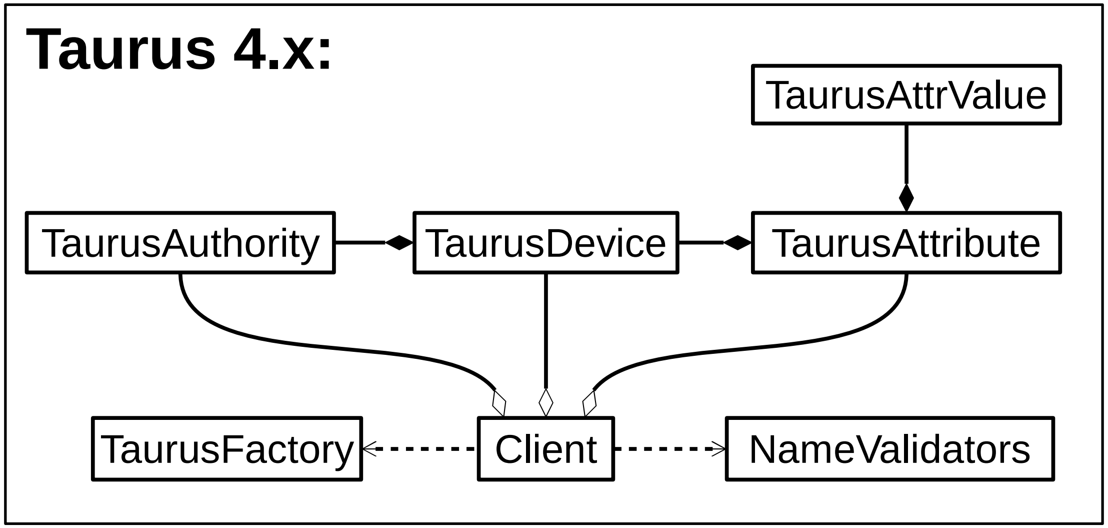
Future priorities

Taurus 4 core: simplified API



New in Taurus 4 core:

- Merged Attribute+Config
- Agnostic base classes
- Improved validators
- Model fragments support
- Agnostic helpers
- Backwards-compat. API



Examples of model names

scheme:authority/path?query#fragment

#	Model name (URI)	Scheme	Model type	Represented source of data/control object
1	*tango://foo:1234*	TANGO	Authority	**Tango** database listening to port *1234* of host *foo*
2	*tango://foo:1234/a/b/c*	TANGO	Device	**Tango** Device *a/b/c* registered in database *foo*
3	*tango:a/b/c/state*	TANGO	Attribute	**Tango** attribute *state* of device #2
4	*tango:a/b/c/d#units*	TANGO	Attribute	**Tango** attribute *d* of device #2 (*units* fragment)
5	*ca:XXX:m1.VAL*	EPICS	Attribute	**EPICS** process variable *XXX:m1.VAL*
6	*eval:({tango:a/b/c/d}+{epics:XXX:m1.VAL})*0.5*	eval	Attribute	**Calculated** average of the values of #4 and #5
7	*eval:rand(256)*	eval	Attribute	Random **generated** array of 256 values
8	*msenv://foo:1234/macroserver/bar/1/ScanDir*	[MSenv]	Attribute	*ScanDir* variable from **Sardana's environment**
9	*h5file:/mydir/myfile.hdf5*	HDF5	Device	File in **HDF5** format saved at */mydir/myfile*
10	*h5file:/mydir/myfile.hdf5:data/energy*	HDF5	Attribute	**HDF5** dataset *energy* of group *data* from file #9
11	*ssheet:myfile.ods:Sheet1.A1*		Attribute	Contents of cell A1 of Sheet1 of *myfile.ods* **spreadsheet**

Other suggested schemes:

Spec, LIMA, Madoca2, Archiving, SQL, Icat, Pasarelle, ASCII tables

Taurus 4 core: name validators

```
>>> val = taurus.Factory('tango').getAttributeNameValidator()

>>> val.isValid('tango:a/b/c/d')

True

>>> val.getNames('tango:a/b/c/d')

('tango://foo:1234/a/b/c/d', 'a/b/c/d', 'd')

>>> val.getUriGroups('tango://foo:1234/a/b/c/d')
{'__STRICT__': True,      '_devalias': None,        '_devslashname': 'a/b/c',
 '_shortattrname': 'd',   'attrname': '/a/b/c/d',   'authority': '//foo:1234',
 'devname': 'a/b/c',      'fragment': None,         'host': 'foo',
 'path': '/a/b/c/d',      'port': '1234',           'scheme': 'tango'}
```

Named groups in validators

	tango	eval	epics
All	**scheme, authority, path, query, fragment**		
authority	**host**, **port**		
device	**devname**, *_devalias, _devslashname, host, port*	**devname**, *_evalname, _evalclass*	**devname**
attribute	**attrname**, **_shortattrname, devname,** *_devalias, _devslashname, host, port*	**attrnname, _expr, _evalrefs,** *_subst, devname, _evalname, _evalclass,*	**attrname** *_field*

Taurus 4 core: name validators

scheme:authority/path?query#fragment

```
class TangoAuthorityNameValidator(TaurusAuthorityNameValidator):
    scheme = 'tango'
    authority = '//(?P<host>([\w\-_]+\.)*[\w\-_]+):(?P<port>\d{1,5})'
    path = '(?!)'
    query = '(?!)'
    fragment = '(?!)'
```

```
class EpicsAttributeNameValidator(TaurusAttributeNameValidator):
    scheme = '(epics|ca)'
    authority = '//'
    path = r'(?P<attrname>[a-zA-Z0-9_\-:;\<\>]+?(\.(?P<_field>[A-Z]+))?)'
    query = '(?!)'
    fragment = '[^# ]*'

    def getNames(self, fullname, **kwargs):
        groups = self.getUriGroups(fullname)
        if groups is None:
            return None
        complete = 'epics:%s' % groups['attrname']
        normal = groups['attrname']
        short = normal
        return complete, normal, short
```

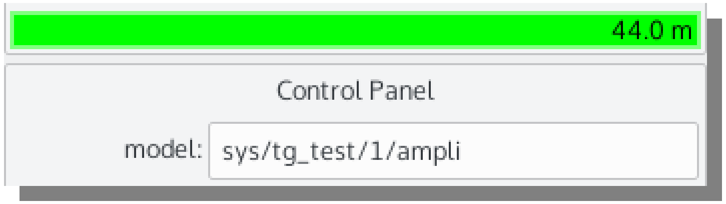
Taurus 4 core: fragments

$\underbrace{\texttt{scheme:authority/path?query}}\texttt{\#}\underbrace{\texttt{fragment}}$

model name
~ model object

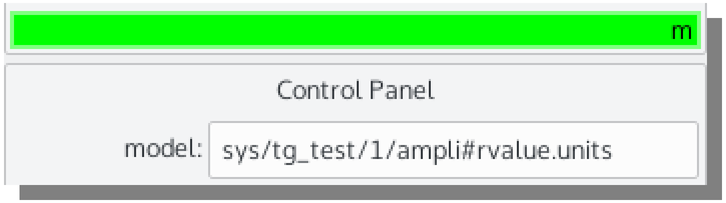
fragment name
~member of model object

Note: The fragment name is only known by the view/controller (not by the model object)


44.0 m

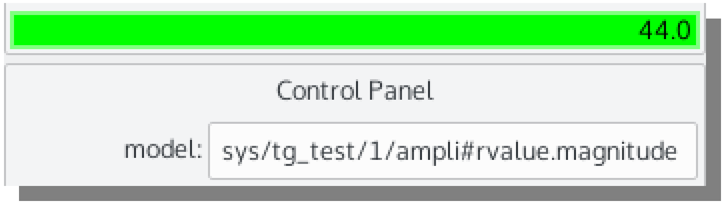
Control Panel

model: sys/tg_test/1/ampli


m

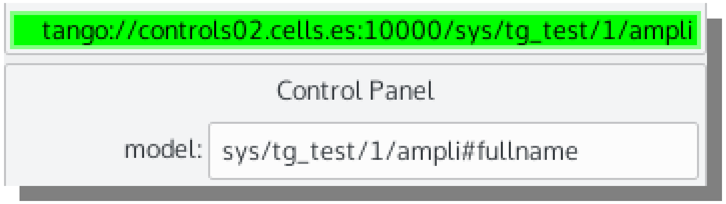
Control Panel

model: sys/tg_test/1/ampli#rvalue.units


44.0

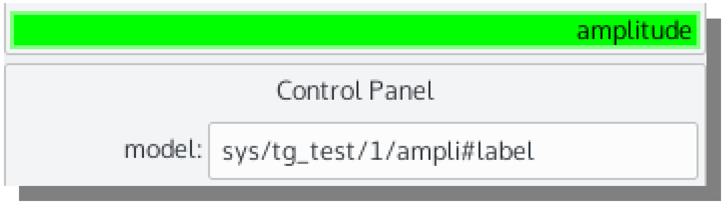
Control Panel

model: sys/tg_test/1/ampli#rvalue.magnitude


tango://controls02.cells.es:10000/sys/tg_test/1/ampli

Control Panel

model: sys/tg_test/1/ampli#fullname


amplitude

Control Panel

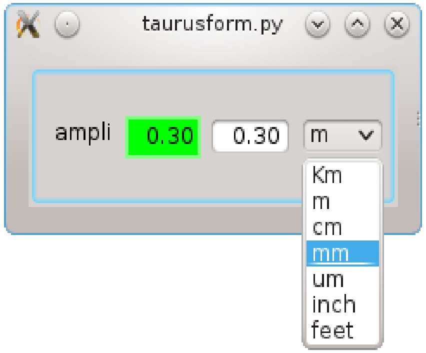
model: sys/tg_test/1/ampli#label

http://sf.net/p/tauruslib/wiki/TEP14

Taurus4 core: units

Allowed types for values, limits, alarms, etc.

	0D	1D	ND
bool	bool or numpy.bool	ndarray(dtype=bool)	ndarray(dtype=bool)
int & float	pint.Quantity	pint.Quantity	pint.Quantity
str	str	seq<str>	seq<seq<...<str>>>
bytes	bytes		
enums	enum.Enum (or taurus Enumeration)		



Advantages of Quatities:

- unified unit support for all schemes
- simple conversion
- transparent operation
- dimensionality is checked
- dimensionless quantities supported

http://pint.readthedocs.org

```
>>> a = taurus.Attribute('tango:motor1/position')
>>> p = a.read().rvalue
>>> p
<Quantity(0.03, 'meter')>
>>> print(p)
0.03 m
>>> print(p.magnitude)
0.03
>>> print(p.to('cm'))
3.0 cm
>>> a.alarms
[<Quantity(-100, 'millimeter')>, <Quantity(100, 'millimeter')>]
>>> print(a.alarms[1] - p)
70.0 mm
```

taurus 4 core: bck-compat

```
Model name "tango://sys/tg_test/1" is supported but not strictly valid. It is STRONGLY
recommended that you change it to strictly follow tango scheme syntax

DeprecationWarning: getDisplayUnit is deprecated (from tep14). Use .rvalue.units instead

DeprecationWarning: getMinAlarm is deprecated (from tep14). Use .alarms[0] instead
```

Taurus 3.x

```
taurus.Database(...)
taurus.Configuration(...)

'tango://a/b/c'
'tango://a/b/c/d?configuration=units'

'eval://a*x?a={tango://a/b/c/d};x=2'
'eval://dev=foo;rand()'
'eval://rand()?configuration=label'

attribute.read().value
attribute.read().w_value

dev.getSWState()
dev.getState()

PyTango.AttrDataFormat.IMAGE
...
```

Taurus 4.x

```
taurus.Authority(...)
taurus.Attribute(...)

'tango:a/b/c'
'tango:a/b/c/d#units'

'eval:a={tango:a/b/c/d};x=2;a*x'
'eval:@foo/rand()'
'eval:rand()#label'

attribute.read().rvalue  (.magnitude)
attribute.read().wvalue  (.magnitude)

dev.state
dev.stateObj.read().rvalue

taurus.DataFormat._2D
...
```

https://sourceforge.net/p/tauruslib/wiki/Taurus4-API_changes/

Introduction

What is Taurus

Taurus Structure

Taurus4 development (timeline)

Changes in taurus.core

Simplified, agnostic API

New model naming (validators and fragments)

Standardized values and units support

Backwards-compatibility

Changes in taurus.qt

New-style signals

Avoid icon resource files

Replacing Qwt dependency

Improving Community & Infrastructure

Transition to setuptools

Improving contribution workflow

Future priorities

New-style signals

Taurus 3.x: old-style signals, PyQt4 (> 4.4)

👎

```
class MyWidget(Qt.QWidget):

    def foo(self):
        self.connect(self, Qt.SIGNAL('mySignal(int)', self.bar)
        self.emit(Qt.SIGNAL('mySignal(int)', 123)
```

Taurus 4.x: new-style signals, PyQt4 (> 4.8) , PyQt5, PySide

👍

```
class MyWidget(Qt.QWidget):

    mySignal = Qt.pyqtSignal(int)

    def foo(self):
        self.mySignal.connect(self.bar)
        self.mySignal.emit(123)
```

For an automatic translation helper, see: https://github.com/cpascual/fixsignals

Avoid icon resource files

Taurus 3.x:

- needs to build / distribute resource files
- buggy workaround for supporting theme icons

```
from taurus.qt.qtgui.resource import getIcon, getThemeIcon
icon1 = getIcon(':/actions/edit-cut.svg')
icon2 = getThemeIcon('computer')
```

Taurus 4.x:

- Does not use resource files. It registers icon paths instead.
- Proper theme icons support in all OS (see taurus.qt.qtgui.icon)

```
import taurus.qt.qtgui  # this registers taurus icon paths
icon1 = Qt.QIcon('actions:edit-cut.svg')
icon2 = Qt.QIcon.fromTheme('computer')
```

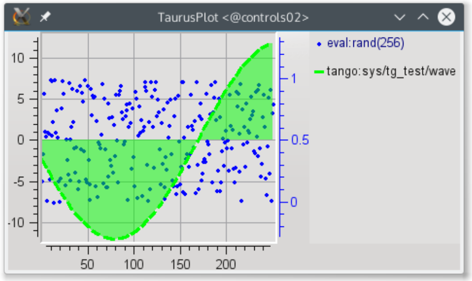
- Provides backwards-compatibility layer

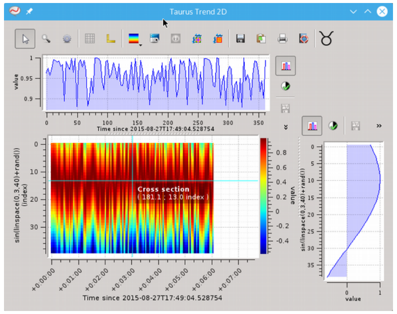
```
DeprecationWarning: taurus.qt.qtgui.resource is deprecated (from 4.0).
 Use taurus.qt.qtgui.icon instead

DeprecationWarning: getIcon(":/actions/edit-cut.svg") is deprecated (from 4.0).
 Use Qt.QIcon("actions:edit-cut.svg") instead

DeprecationWarning: getThemeIcon is deprecated (from 4.0).
 Use QIcon.fromTheme instead
```

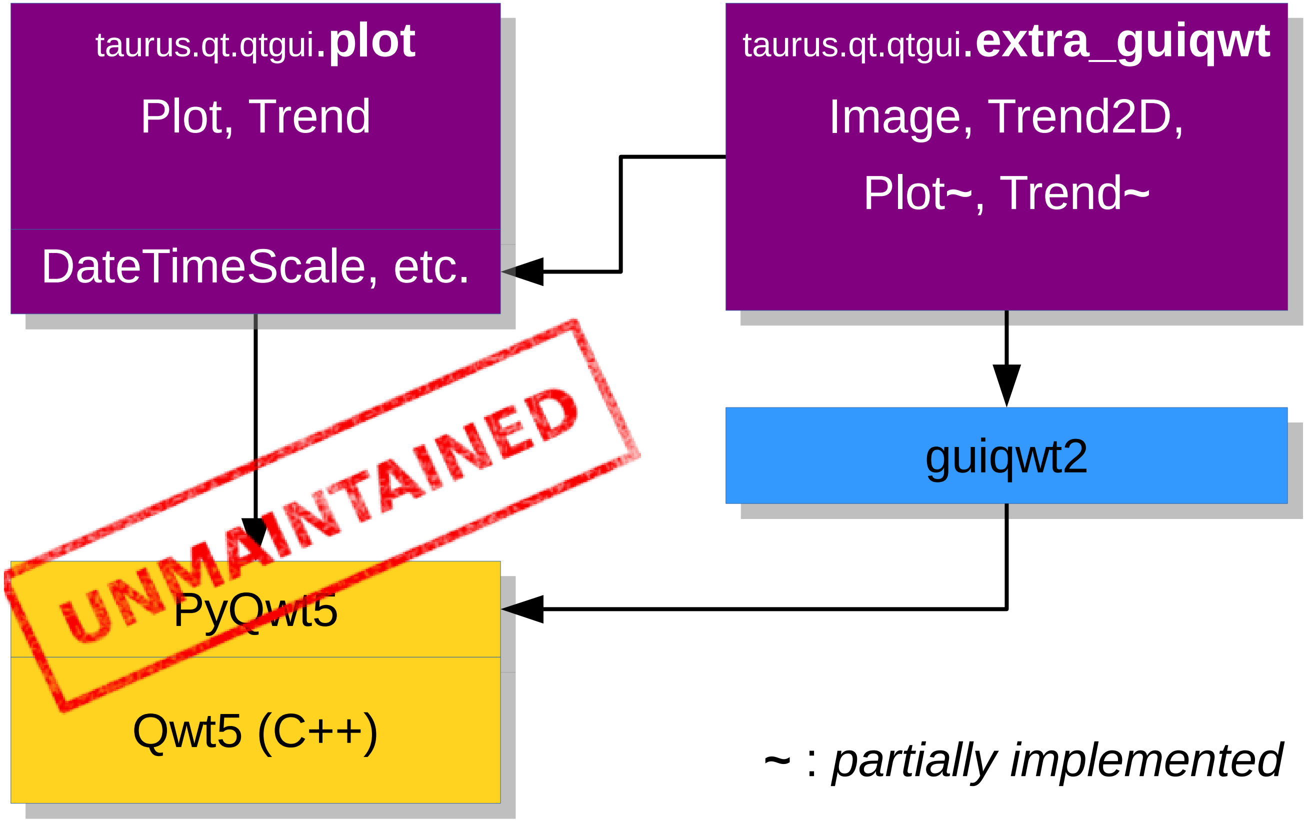
Replacing Qwt5





Current situation:

- No bugfixes
- No support for python3
- No support for Qt5
- Plots & Trends: PyQwt5
- Images and Trend2D: guiqwt2
- extra_guiqwt tools: PyQwt5



taurus.qt.qtgui.**plot**

Plot, Trend

DateTimeScale, etc.

taurus.qt.qtgui.**extra_guiqwt**

Image, Trend2D,

Plot~, Trend~

guiqwt2

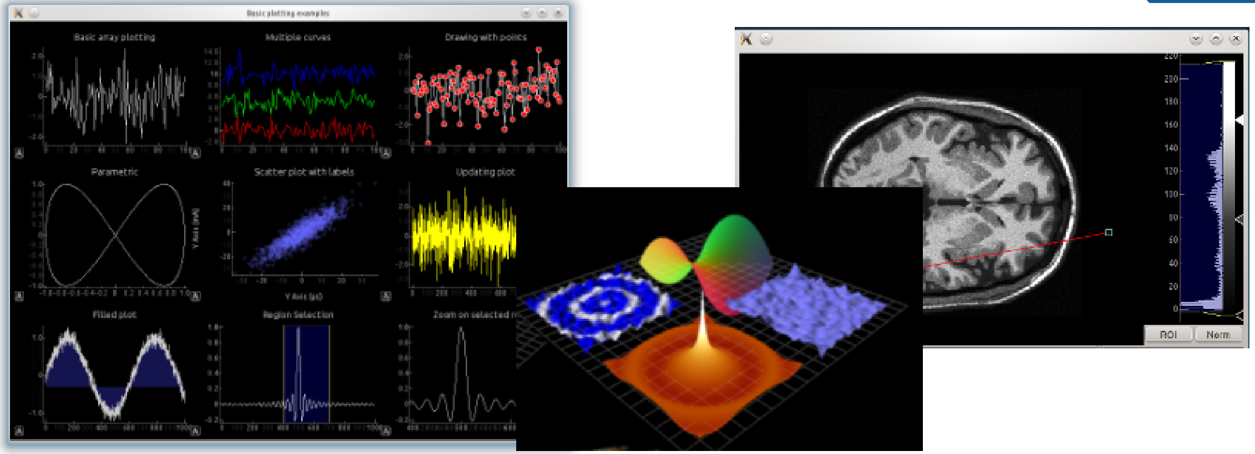
PyQwt5

Qwt5 (C++)

UNMAINTAINED

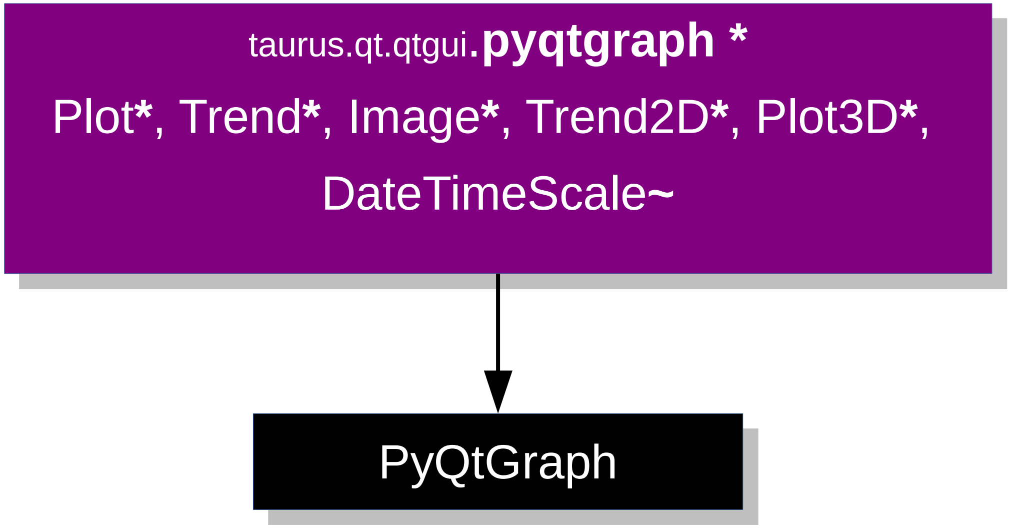
~ : *partially implemented*

Replacing Qwt5



Alternative - PyQtGraph:

- 👍 Fast
- 👍 Supports 3D (with OpenGL)
- 👍 Nice, simple API
- 👍 Lively community and good forum
- 👎 Single maintainer
- 👎👎 Need to implement **everything**



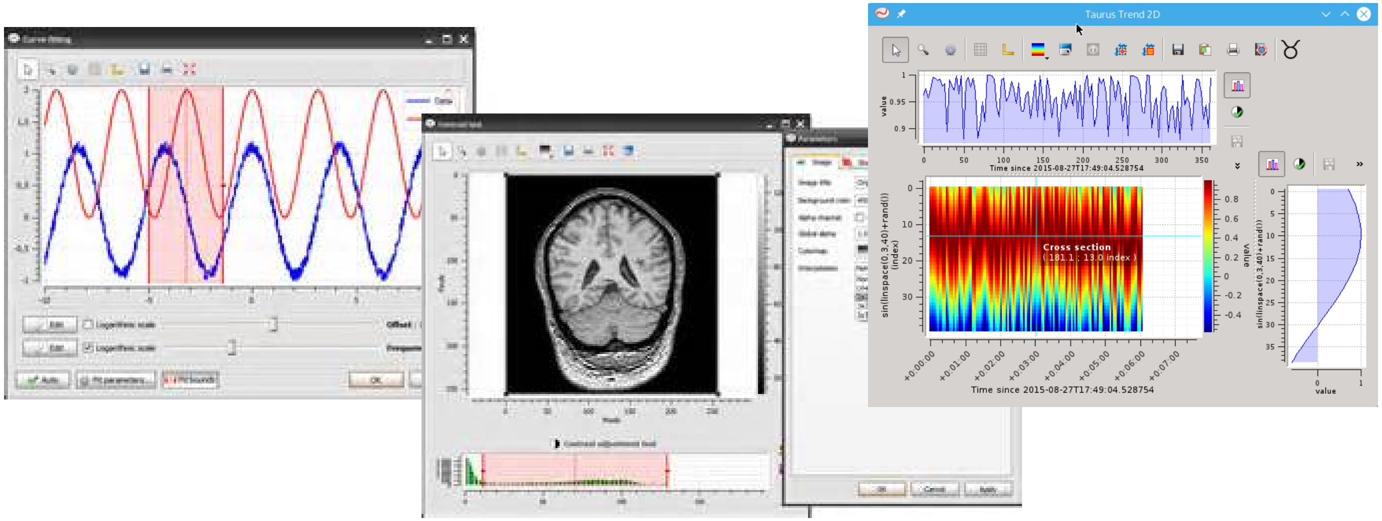
taurus.qt.qtgui.**pyqtgraph ***

Plot*, Trend*, Image*, Trend2D*, Plot3D*, DateTimeScale~

PyQtGraph

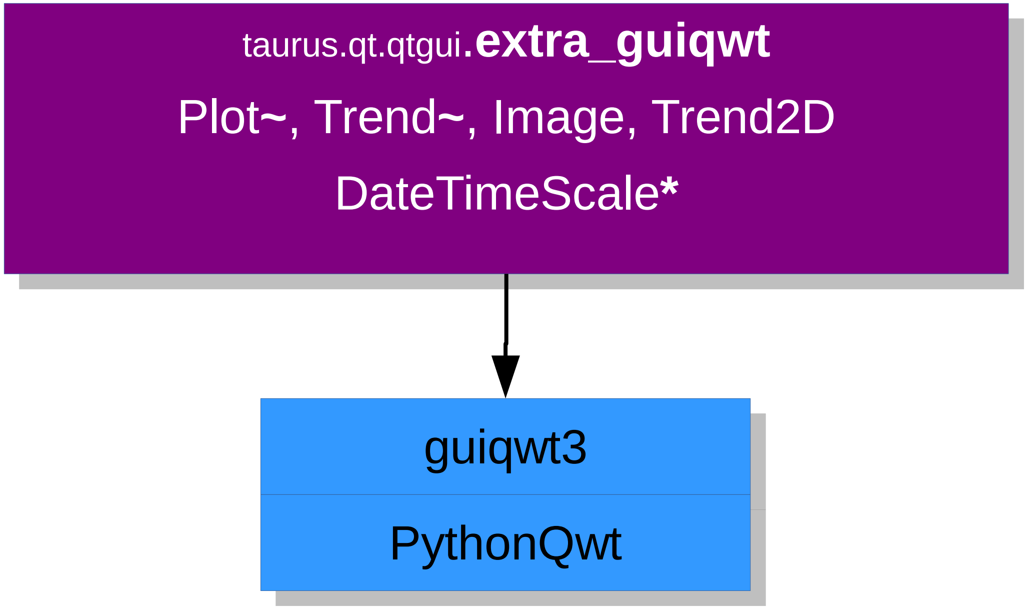
~ : partially implemented
** : not implemented*

Replacing Qwt5



Alternative - guiqwt3:

- We can reuse most of extra_guiqwt
- Lots of great tools
- No 3D support
- Awkward, badly documented API
- Single, busy, maintainer
- Bad support, quasi-dead mailing list



taurus.qt.qtgui.**extra_guiqwt**

Plot~, Trend~, Image, Trend2D

DateTimeScale*

guiqwt3

PythonQwt

~ : partially implemented
** : not implemented*

Introduction

What is Taurus

Taurus Structure

Taurus4 development (timeline)

Changes in taurus.core

Simplified, agnostic API

New model naming (validators and fragments)

Standardized values and units support

Backwards-compatibility

Changes in taurus.qt

New-style signals

Avoid icon resource files

Replacing Qwt dependency

Improving Community & Infrastructure

Transition to setuptools

Improving contribution workflow

Future priorities

Move to setuptools

Taurus 3.x:

- uses **distutils**
  - pip requires “--egg” parameter to work 👎
  - module and package_data lists must be maintained manually 👎
- heavily customized *setup.py* (~1000 lines) 👎
  - difficult to maintain 👎
  - non-standard installation commands 👎

Taurus 4.x:

- uses **setuptools**
  - enables plugin support via “`entry_points`” 👍👍
  - automated launcher script creation (multi-platform) 👍
  - nice extra commands: “*`develop`*” , ”*`test`*”, “*`build_sphinx`*”, ... 👍
- New, simpler *setup.py* created from scratch (~100 lines) 👍

Improving contribution workflow

Current situation

sourceforge

- either use tickets and merge-request (bad interface) or emails (complex) 👎
- Poor integration between mailing list and tickets 👎
- Saturation of mailing list with administrative emails 👎
- No out-of-the-box solution for **public** Continuous Integration 👎👎

gerrit+Jenkins

- Need to configure & administer for **public** usage 👎👎
- Used internally by other members of Tango 👍
- Open source 👍

Tested proposal

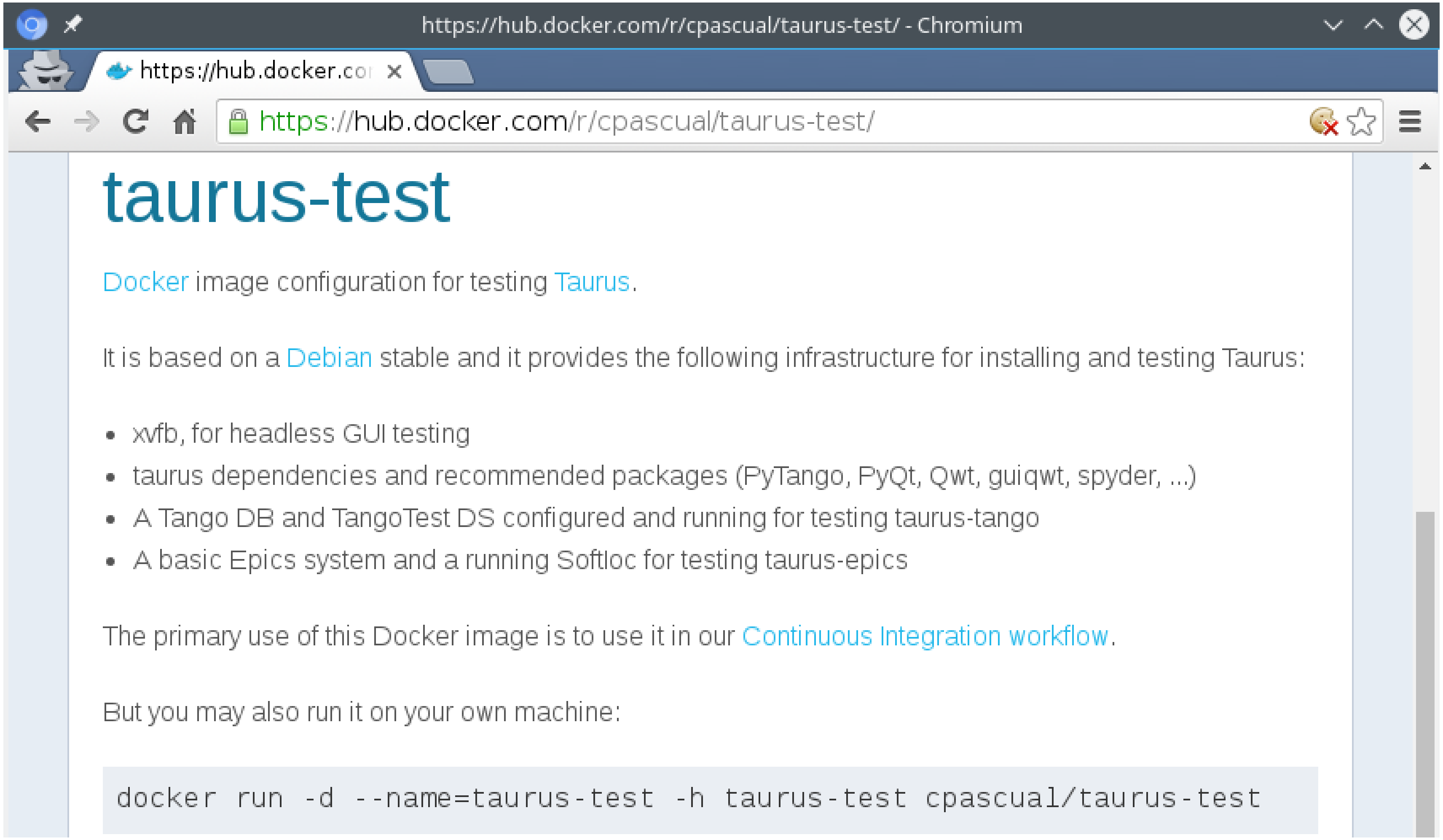
GitHub

- best interface 👍
- Public Continuous Integration (via Travis) 👍
- Integrated with with ReadTheDocs 👍
- Tango's choice 👍👍

GitLab

- good interface 👍
- Public Continuous Integration allowing own workers (e.g. for windows) 👍👍
- Used internally by some members of Tango 👍
- Open Source 👍

Improving contribution workflow


https://hub.docker.com/r/cpascual/taurus-test/ - Chromium
https://hub.docker.co
https://hub.docker.com/r/cpascual/taurus-test/

taurus-test

Docker image configuration for testing Taurus.

It is based on a Debian stable and it provides the following infrastructure for installing and testing Taurus:

- xvfb, for headless GUI testing
- taurus dependencies and recommended packages (PyTango, PyQt, Qwt, guiqwt, spyder, ...)
- A Tango DB and TangoTest DS configured and running for testing taurus-tango
- A basic Epics system and a running SoftIoc for testing taurus-epics

The primary use of this Docker image is to use it in our Continuous Integration workflow.

But you may also run it on your own machine:

```
docker run -d --name=taurus-test -h taurus-test cpascual/taurus-test
```

TO-DOs (from last year)

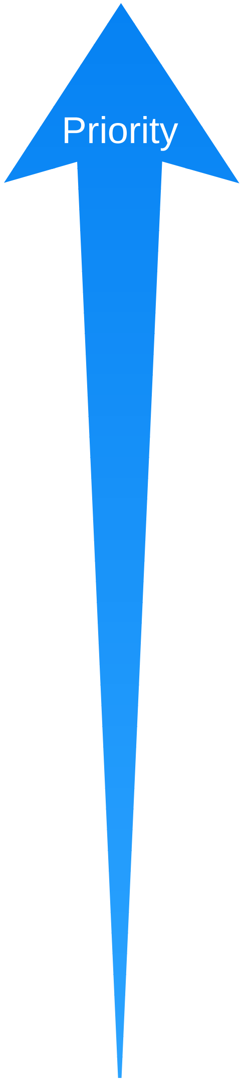
Tango isolation (TEP3)	💡	💬	🛠
Use of Pint Quantities (TEP14)	💡	💬	🛠
Merge TaurusConfiguration into TaurusAttribute (TEP14)	💡	💬	🛠
Multi-models	💡	💬	🛠
Allow external logging (SEP8)	💡	💬	🛠
Plug-in system (TEP13)	💡	💬	🛠
Direct registering of Icons (avoid resource files)	💡	💬	🛠
Use of standard Enum (SEP12)	💡	💬	🛠
Create the h5file:// scheme	💡		
Replace Qwt for plots	💡	💬	
New-style signals (ticket 187)	💡	💬	
Support Qt5 and PySide (ticket 245)	💡	💬	🛠
Introduce QML widgets	💡		
Support Python3 (ticket 266)	💡		
Generic support for archiving values	💡	💬	
Improve code contribution workflow	💡	💬	
Use Continuous Integration	💡	💬	

TO-DOs (now)

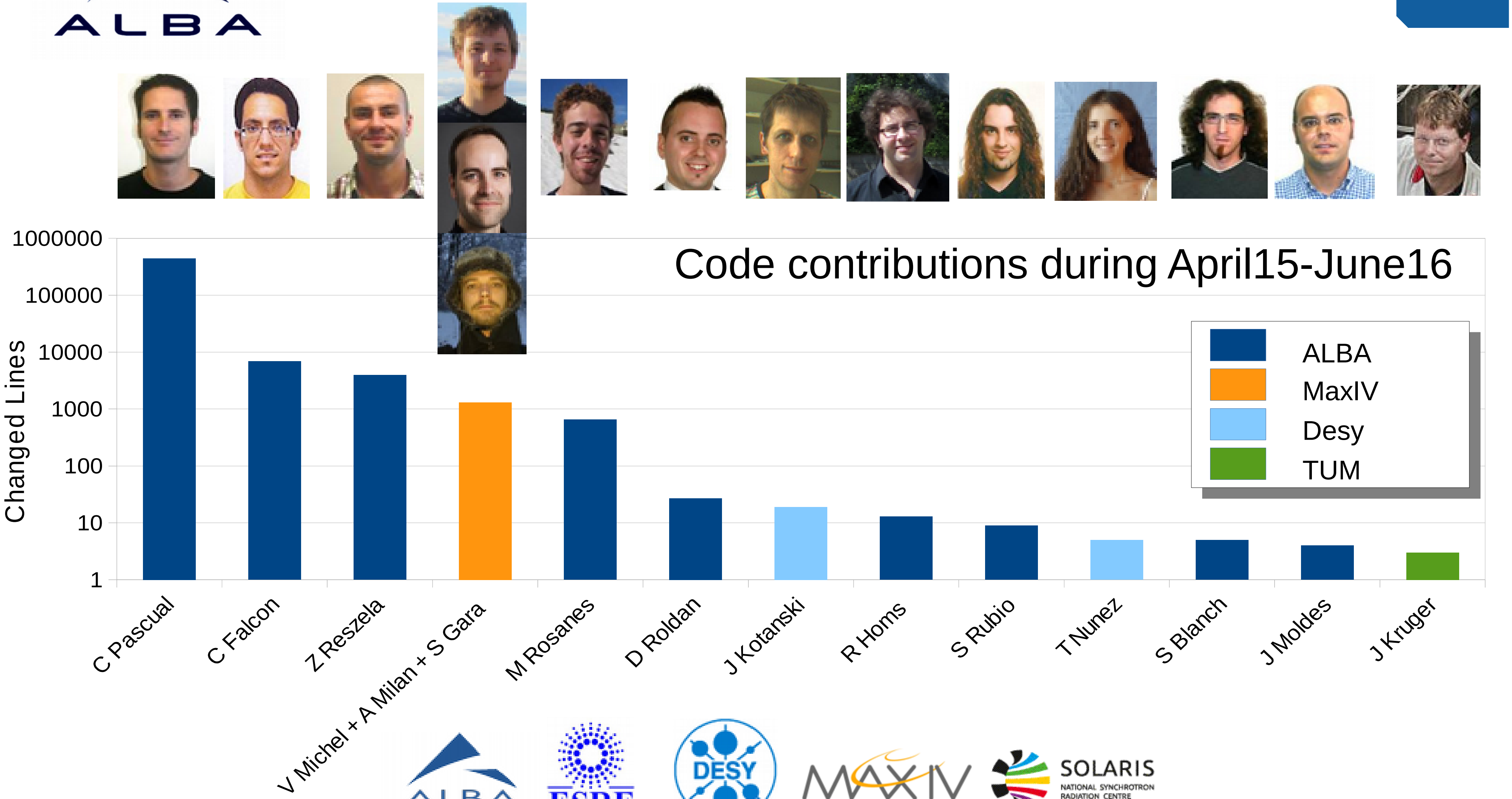
Tango isolation (TEP3)	💡	💬	🛠	✔	*Finished*
Use of Pint Quantities (TEP14)	💡	💬	🛠	✔	*Finished*
Merge TaurusConfiguration into TaurusAttribute (TEP14)	💡	💬	🛠	✔	*Finished*
Multi-models	💡	💬	🛠		*No progress*
Allow external logging (SEP8)	💡	💬	🛠		*No progress*
Plug-in system (TEP13)	💡	💬	🛠		*Work in progress*
Direct registering of Icons (avoid resource files)	💡	💬	🛠	✔	*Finished*
Use of standard Enum (SEP12)	💡	💬	🛠		*No progress*
Create the h5file:// scheme	💡				*No progress*
Replace Qwt for plots	💡	💬	🛠		*Work in progress*
New-style signals (ticket 187)	💡	💬	🛠	✔	*Finished*
Support Qt5 ~~and PySide~~ (ticket 245)	💡	💬	🛠		*Work in progress*
Introduce QML widgets	💡				*No progress*
Support Python3 (ticket 266)	💡	💬			*Little progress*
Generic support for archiving values	💡	💬			*No progress*
Improve code contribution workflow	💡	💬	🛠		*Work in progress*
Use Continuous Integration	💡	💬	🛠		*Work in progress*

TO-DOs (next 12 months)

- Make taurus4 deprecation-warning-free 💡 💬 🛠
- Replace Qwt for plots 💡 💬 🛠
- Support Qt5 (ticket 245) 💡 💬 🛠
- Improve code contribution workflow 💡 💬 🛠
- Plug-in system (TEP13) 💡 💬 🛠
- Create the h5file:// scheme 💡
- Taurus.qt tango isolation 💡 💬 🛠
- Use Continuous Integration 💡 💬 🛠
- Multi-models 💡 💬 🛠
- Use of standard Enum (SEP12) 💡 💬 🛠
- Support Python3 (ticket 266) 💡 💬
- Generic support for archiving values 💡 💬
- Allow external logging (SEP8) 💡 💬 🛠



The **community** made it possible



...and special thanks to F. Picca for Debian packaging!