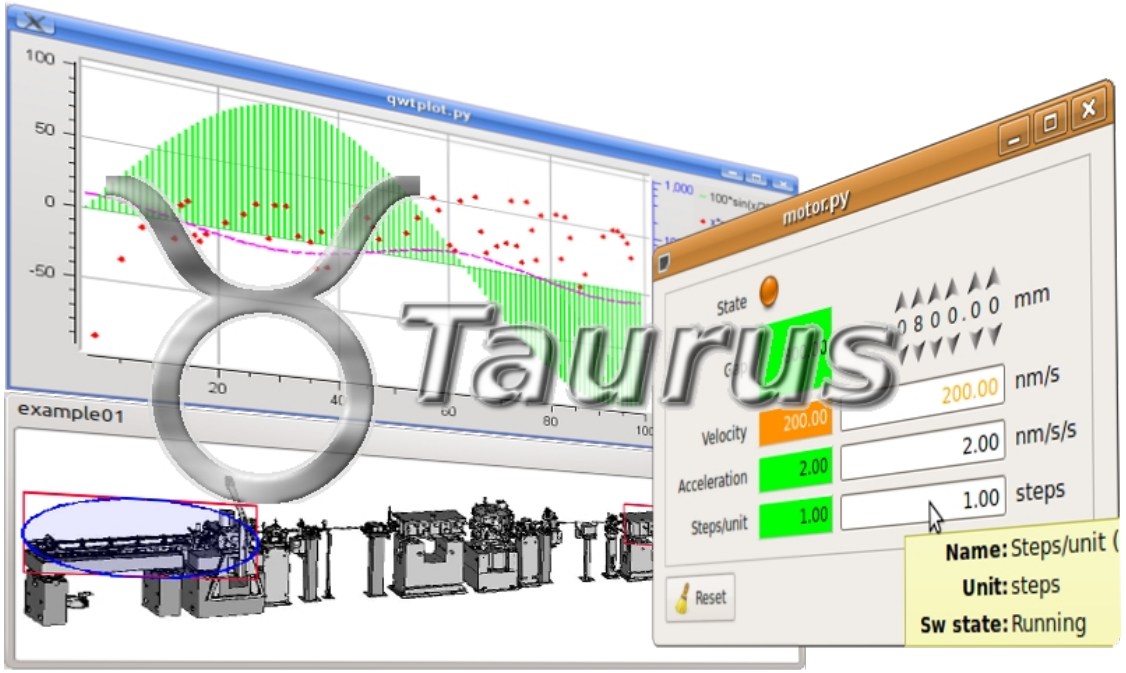
Taurus Status

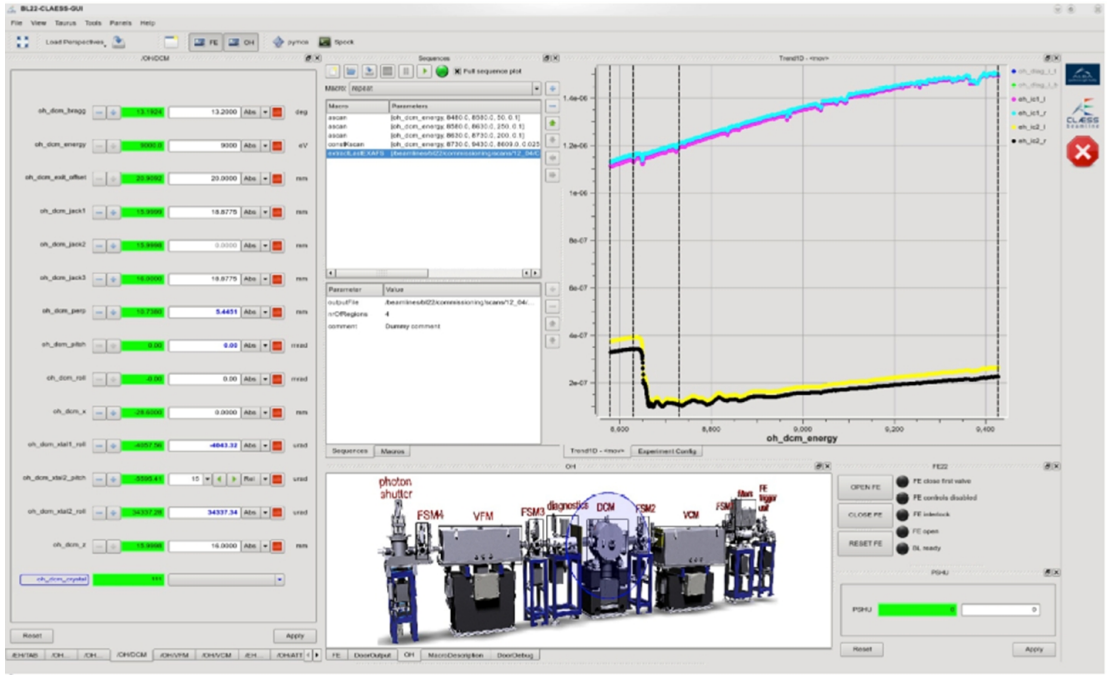
by

Carlos Pascual-Izarra
(on behalf of Alba's Controls group)

29th Tango Meeting,
May 2015



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
- Multi-platform
- Based on Python and Qt
- Easy to install

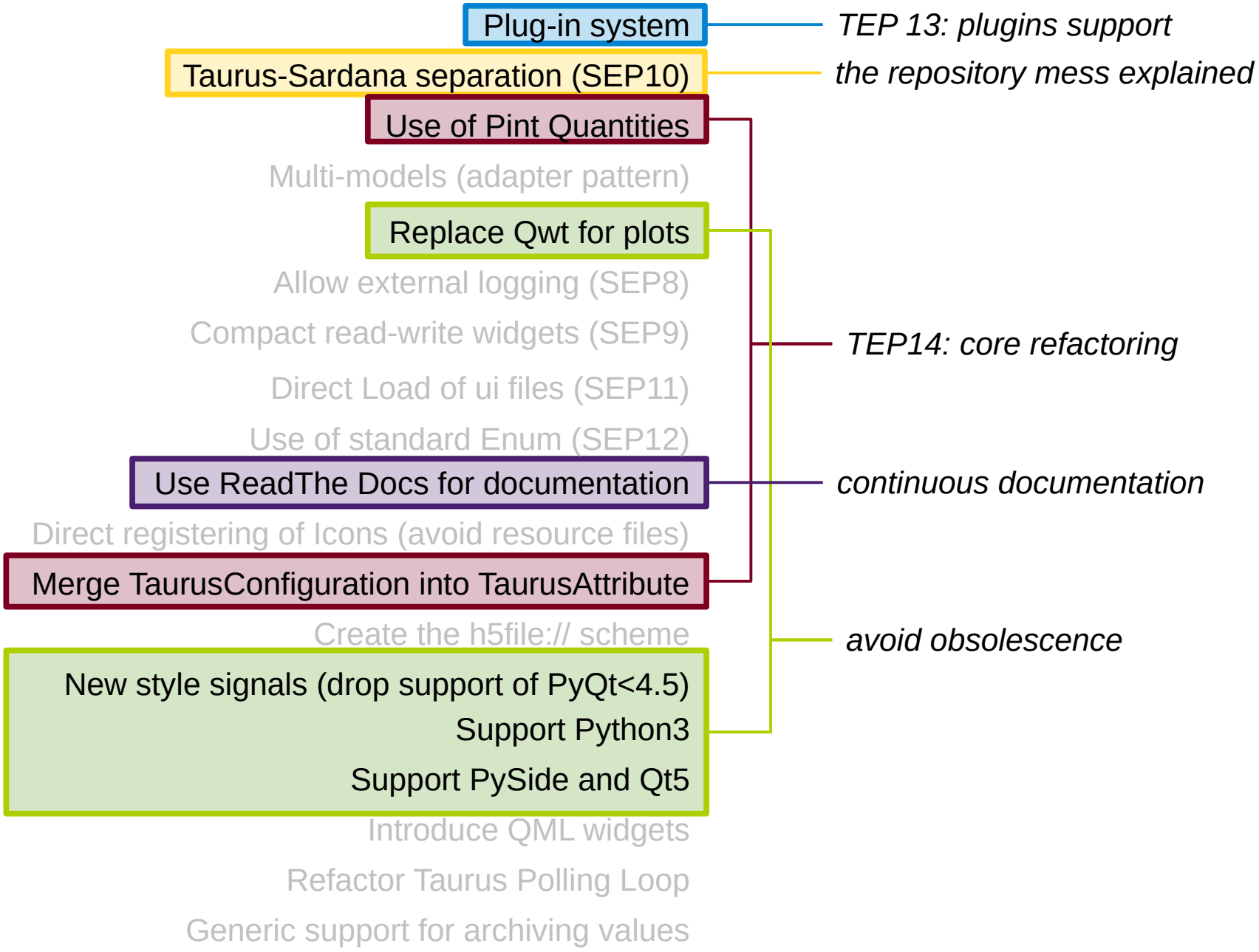
TO-DOs (one year ago)

- Plug-in system
- Taurus-Sardana separation (SEP10)
- Use of Pint Quantities
- Multi-models (adapter pattern)
- Replace Qwt for plots
- Allow external logging (SEP8)
- Compact read-write widgets (SEP9)
- Direct Load of ui files (SEP11)
- Use of standard Enum (SEP12)
- Use ReadThe Docs for documentation
- Direct registering of Icons (avoid resource files)
- Merge TaurusConfiguration into TaurusAttribute
- Create the h5file:// scheme
- New style signals (drop support of PyQt<4.5)
- Support Python3
- Support PySide and Qt5
- Introduce QML widgets
- Refactor Taurus Polling Loop
- Generic support for archiving values

TO-DOs (now)

Plug-in system	*Work in progress*
Taurus-Sardana separation (SEP10)	✔ ***Finished***
Use of Pint Quantities	*Prototype ready*
Multi-models (adapter pattern)	*No progress*
Replace Qwt for plots	*Resources committed*
Allow external logging (SEP8)	*No progress*
Compact read-write widgets (SEP9)	✔ ***Finished***
Direct Load of ui files (SEP11)	✔ ***Finished***
Use of standard Enum (SEP12)	*Little progress*
Use ReadThe Docs for documentation	✔ ***Finished***
Direct registering of Icons (avoid resource files)	*No progress*
Merge TaurusConfiguration into TaurusAttribute	*Work in progress*
Create the h5file:// scheme	*No progress*
New style signals (drop support of PyQt<4.5)	*Preliminary discussions*
Support Python3	*No progress*
Support PySide and Qt5	*No progress*
Introduce QML widgets	*No progress*
Refactor Taurus Polling Loop	✔ ***Finished***
Generic support for archiving values	*Preliminary discussions*

Topics


Plug-in system
Taurus-Sardana separation (SEP10)
Use of Pint Quantities
Multi-models (adapter pattern)
Replace Qwt for plots
Allow external logging (SEP8)
Compact read-write widgets (SEP9)
Direct Load of ui files (SEP11)
Use of standard Enum (SEP12)
Use ReadThe Docs for documentation
Direct registering of Icons (avoid resource files)
Merge TaurusConfiguration into TaurusAttribute
Create the h5file:// scheme
New style signals (drop support of PyQt<4.5)
Support Python3
Support PySide and Qt5
Introduce QML widgets
Refactor Taurus Polling Loop
Generic support for archiving values
TEP 13: plugins support
the repository mess explained
TEP14: core refactoring
continuous documentation
avoid obsolescence

Plug-in system — *TEP 13: plugins support*

Taurus-Sardana separation (SEP10)

Use of Pint Quantities

Multi-models (adapter pattern)

Replace Qwt for plots

Allow external logging (SEP8)

Compact read-write widgets (SEP9)

Direct Load of ui files (SEP11)

Use of standard Enum (SEP12)

Use ReadThe Docs for documentation

Direct registering of Icons (avoid resource files)

Merge TaurusConfiguration into TaurusAttribute

Create the h5file:// scheme

New style signals (drop support of PyQt<4.5)

Support Python3

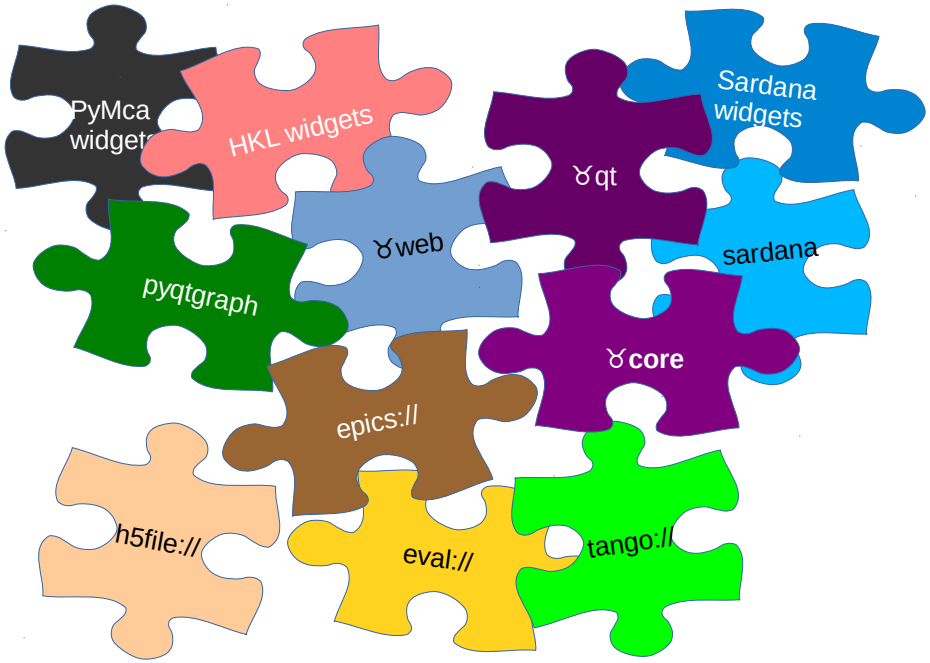
Support PySide and Qt5

Introduce QML widgets

Refactor Taurus Polling Loop

Generic support for archiving values

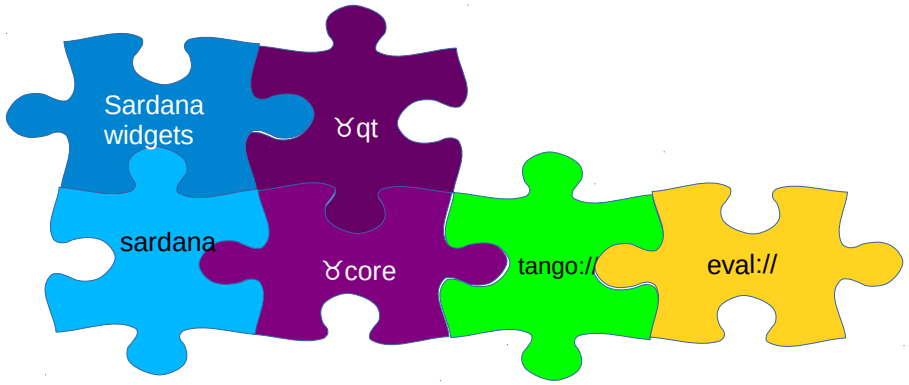
TEP13: Plugins support



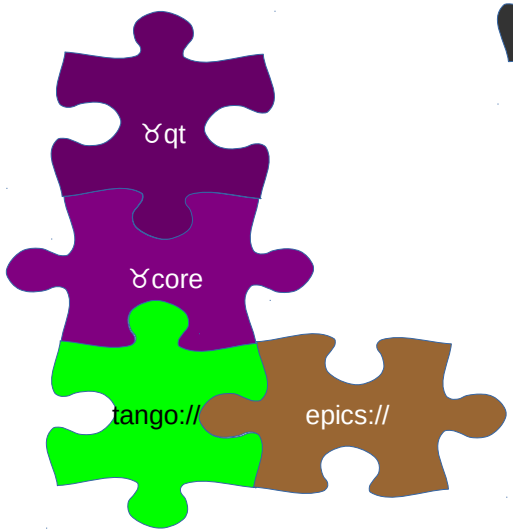
http://sf.net/p/tauruslib/wiki/TEP13

Plugins will make Taurus...

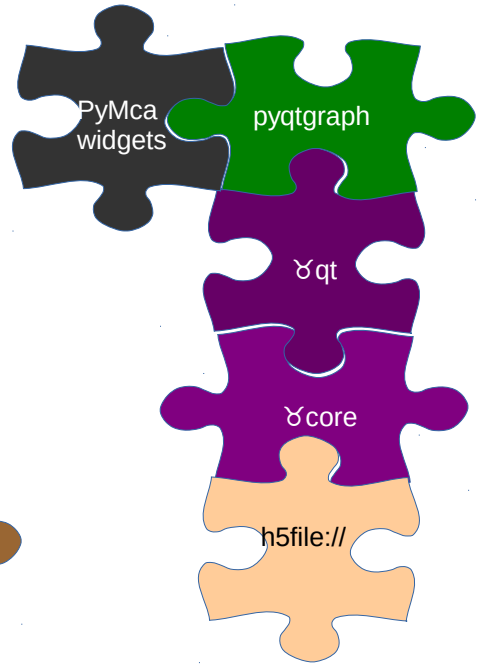
- Light: most dependencies optional
- Extendable for user specific need
- Taurus usable as a library for data analysis GUIs (not connected to control system at all)



***Example**: Taurus+Sardana as we use it now in ALBA*

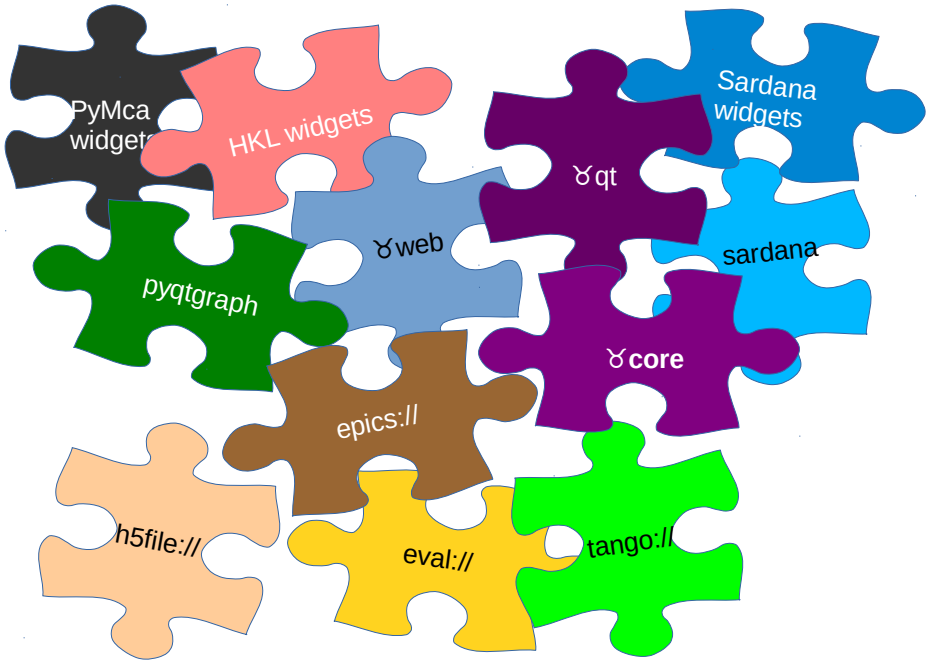


***Example**: Controlling a mixed Tango+EPICS environment*



***Example**:Taurus for Data Analysis (no control system)*

TEP13: Plugins support

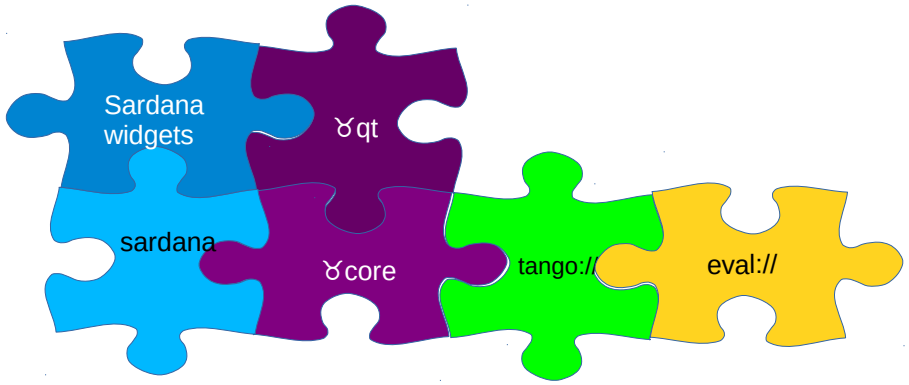


http://sf.net/p/tauruslib/wiki/TEP13

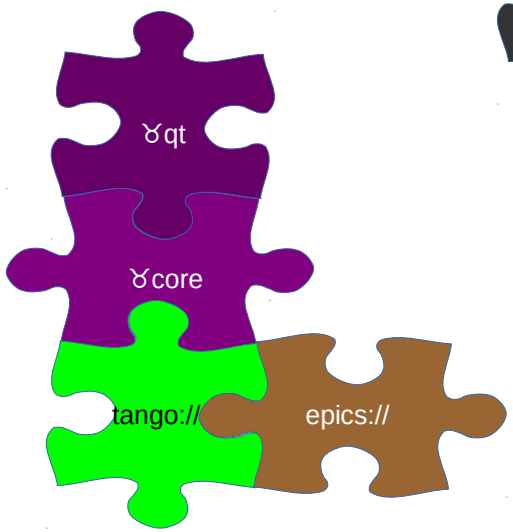
stevedore is the main candidate:

- based on setuptools entry points
- manages dependencies (requires/provides)
- systematic approach
- well documented

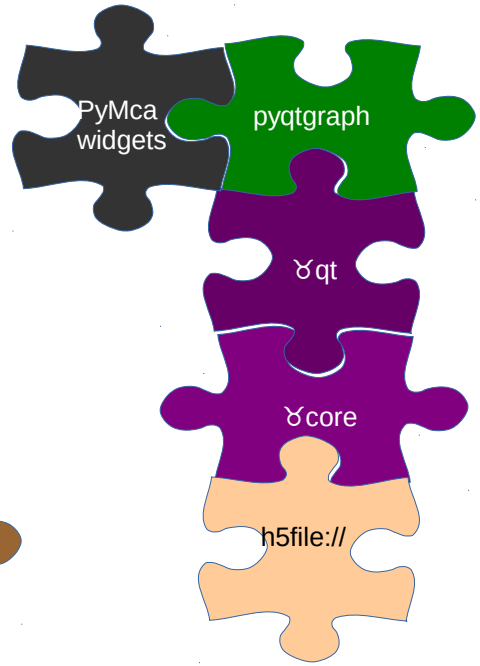
https://pypi.python.org/pypi/stevedore



***Example**: Taurus+Sardana as we use it now in ALBA*



***Example**: Controlling a mixed Tango+EPICS environment*



***Example**:Taurus for Data Analysis (no control system)*

Plug-in system

Taurus-Sardana separation (SEP10)

Use of Pint Quantities

Multi-models (adapter pattern)

Replace Qwt for plots

Allow external logging (SEP8)

Compact read-write widgets (SEP9)

Direct Load of ui files (SEP11)

Use of standard Enum (SEP12)

Use ReadThe Docs for documentation

Direct registering of Icons (avoid resource files)

Merge TaurusConfiguration into TaurusAttribute

Create the h5file:// scheme

New style signals (drop support of PyQt<4.5)

Support Python3

Support PySide and Qt5

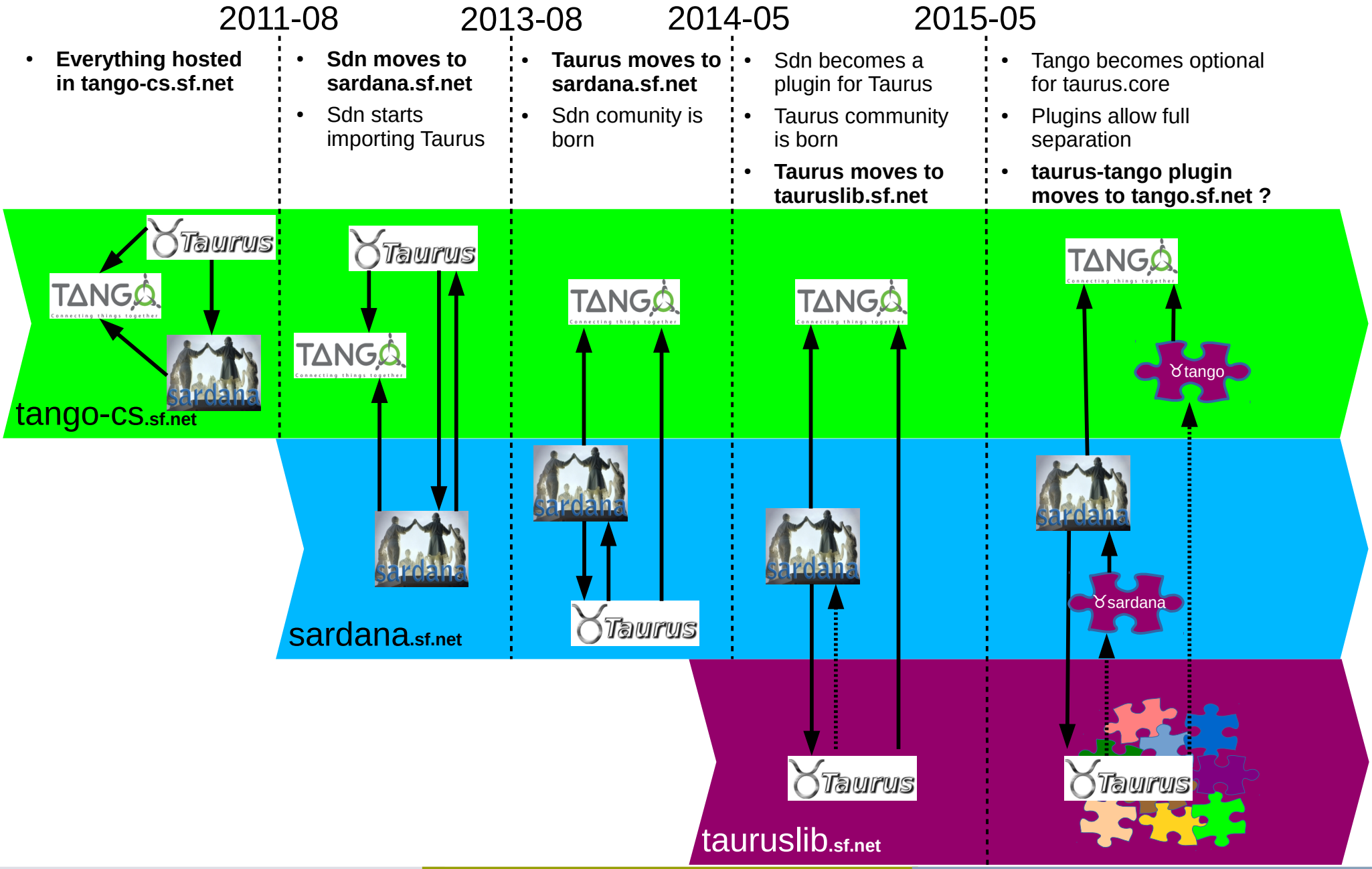
Introduce QML widgets

Refactor Taurus Polling Loop

Generic support for archiving values

TEP 13: plugins support the repository mess explained

The repository mess explained



Plug-in system

Taurus-Sardana separation (SEP10)

Use of Pint Quantities

Multi-models (adapter pattern)

Replace Qwt for plots

Allow external logging (SEP8)

Compact read-write widgets (SEP9)

Direct Load of ui files (SEP11)

Use of standard Enum (SEP12)

Use ReadThe Docs for documentation

Direct registering of Icons (avoid resource files)

Merge TaurusConfiguration into TaurusAttribute

Create the h5file:// scheme

New style signals (drop support of PyQt<4.5)

Support Python3

Support PySide and Qt5

Introduce QML widgets

Refactor Taurus Polling Loop

Generic support for archiving values

TEP 13: plugins support

the repository mess explained

TEP14: core refactoring

TEP14: core refactoring

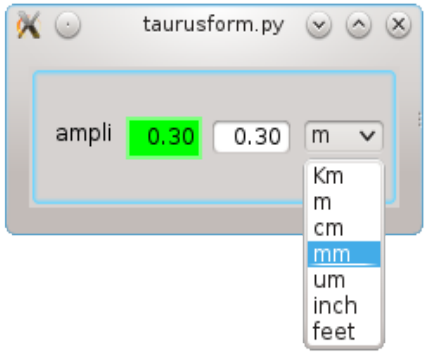
Use Quantities for **int** and **float** types:

	0D	1D	ND
bytes	bytes		
str	str	seq<str>	seq<seq<...<str>>>
bool	bool numpy.bool	ndarray (dtype=bool)	ndarray (dtype=bool)
int / float	pint.Quantity	pint.Quantity	pint.Quantity

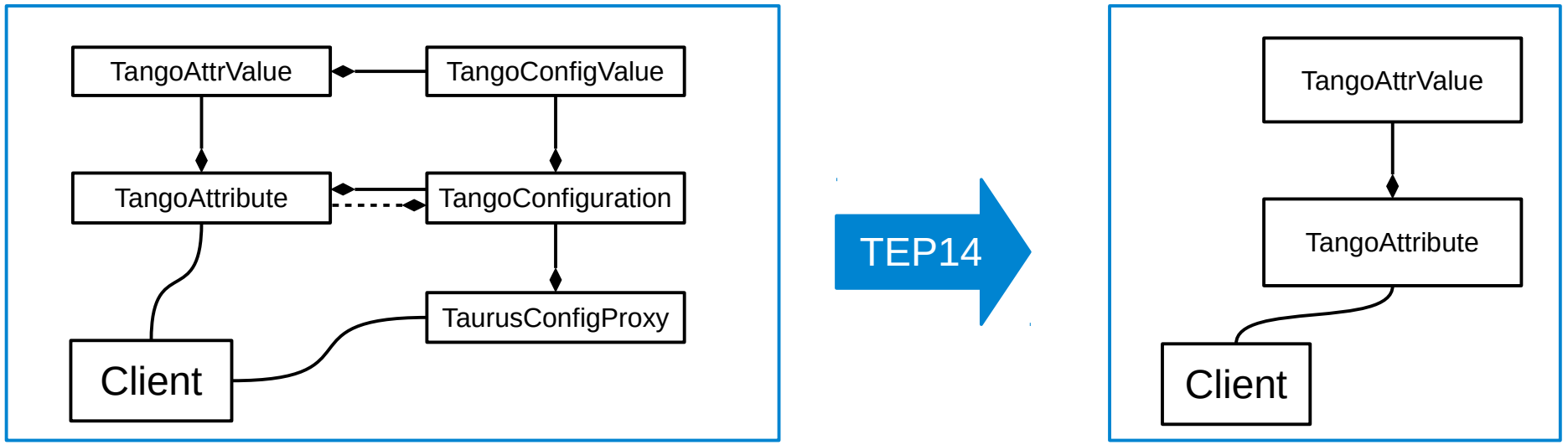
http://pint.readthedocs.org

1st implementation ready in taurus.core of tep14 branch

```
>>> a = taurus.Device('sys/tg_test/1').ampli
>>> a
<Quantity(0.3, 'meter')>
>>> print(a)
0.3 meter
>>> print(a.to('mm'))
300.0 millimeter
```



Merge attribute and configuration objects:



Plug-in system

Taurus-Sardana separation (SEP10)

Use of Pint Quantities

Multi-models (adapter pattern)

Replace Qwt for plots

Allow external logging (SEP8)

Compact read-write widgets (SEP9)

Direct Load of ui files (SEP11)

Use of standard Enum (SEP12)

Use ReadThe Docs for documentation

Direct registering of Icons (avoid resource files)

Merge TaurusConfiguration into TaurusAttribute

Create the h5file:// scheme

New style signals (drop support of PyQt<4.5)

Support Python3

Support PySide and Qt5

Introduce QML widgets

Refactor Taurus Polling Loop

Generic support for archiving values

TEP 13: plugins support

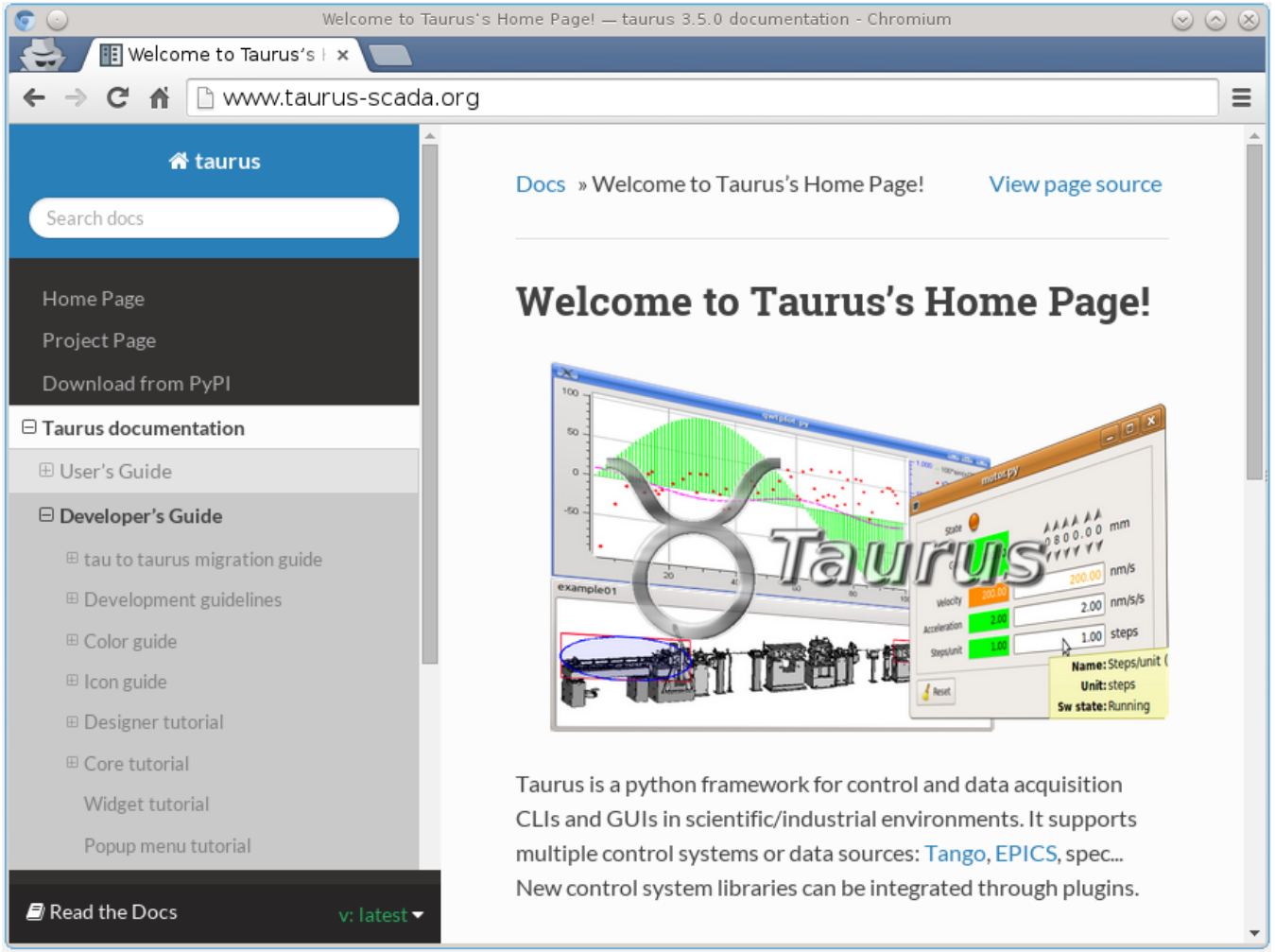
the repository mess explained

TEP14: core refactoring

continuous documentation

Continuous documentation

http://www.taurus-scada.org
http://www.sardana-controls.org } Hosted in http://readthedocs.org



- Git commits trigger rebuilding of docs
- We can auto-generate docs for any branch (now: stable & develop)
- Pdf and epub versions downloadable
- Improved search & indexing
- Mobile phone friendly

Plug-in system

Taurus-Sardana separation (SEP10)

Use of Pint Quantities

Multi-models (adapter pattern)

Replace Qwt for plots

Allow external logging (SEP8)

Compact read-write widgets (SEP9)

Direct Load of ui files (SEP11)

Use of standard Enum (SEP12)

Use ReadThe Docs for documentation

Direct registering of Icons (avoid resource files)

Merge TaurusConfiguration into TaurusAttribute

Create the h5file:// scheme

New style signals (drop support of PyQt<4.5)

Support Python3

Support PySide and Qt5

Introduce QML widgets

Refactor Taurus Polling Loop

Generic support for archiving values

TEP 13: plugins support

the repository mess explained

TEP14: core refactoring

continuous documentation

avoid obsolescence

Avoid obsolescence

PyQwt

- Affects: plots, trends, images,...
- Unmaintained since 2013.
- Urgency: very high
- Migration effort: high (write much code from scratch)
- Migration path: PyQtGraph (eventually Vispy?)
- Ref: http://pyqtgraph.org

Old-style Signals

(and other deprecated APIs)

- Affects: widgets
- Deprecated in PyQt>4.4 ; unsupported in PyQt5 & PySide.
- Urgency: medium-high (no Qt4 in next Debian ~3years)
- Migration effort: medium
- Migration path: support PyQt>4.6 (or go for PyQt>=5 ?)
- Ref: http://pyqt.sf.net/Docs/PyQt5/pyqt4_differences.html

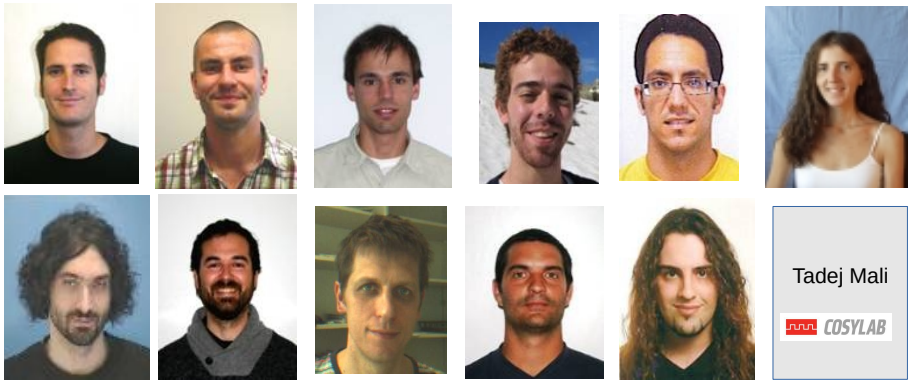
Python2

- Affects: all taurus
- Deprecated but supported till 2020
- Urgency: low
- Migration effort: medium (semiautomatic, mostly needs testing)
- Migration path: support both Python2.7 & Python>3.2
- Ref: https://docs.python.org/3/howto/pyporting.html

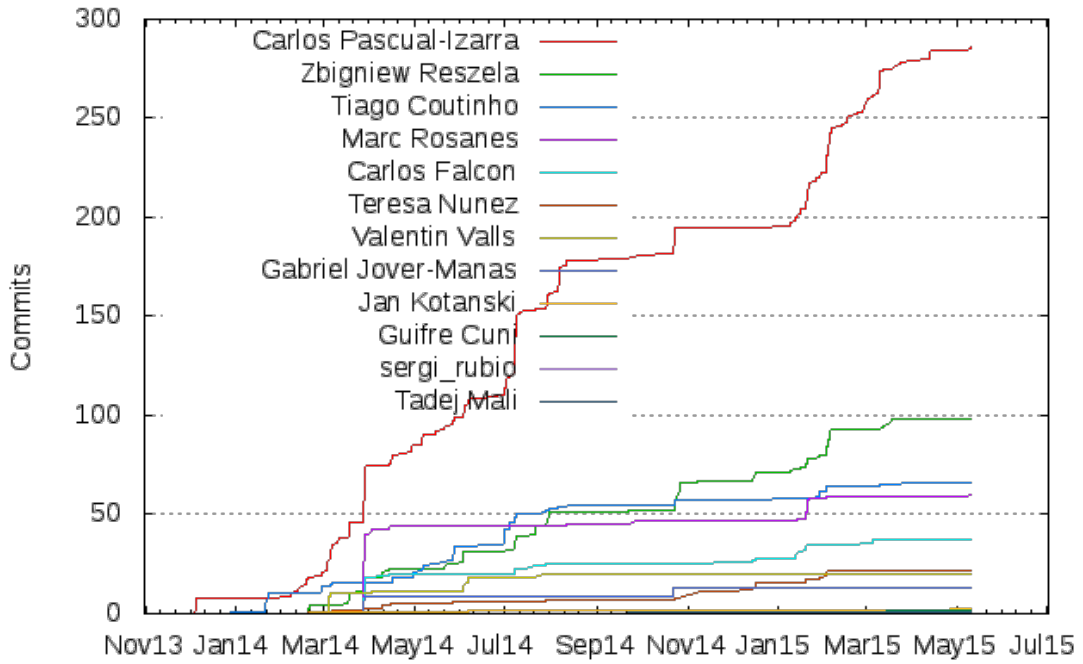
The **community** made it possible

Some figures since Jan14 release

- **3 releases (Jul14, Debian8, Jan15)**
- **9 TEPs being worked on** (4 accepted, 1 being voted, 2 being implemented, 2 being drafted)
- **345 commits to develop**
- **126 tickets** (62 solved)
- **~1800 emails** (~80 of them in users list)



Commits per author since Jan14 release

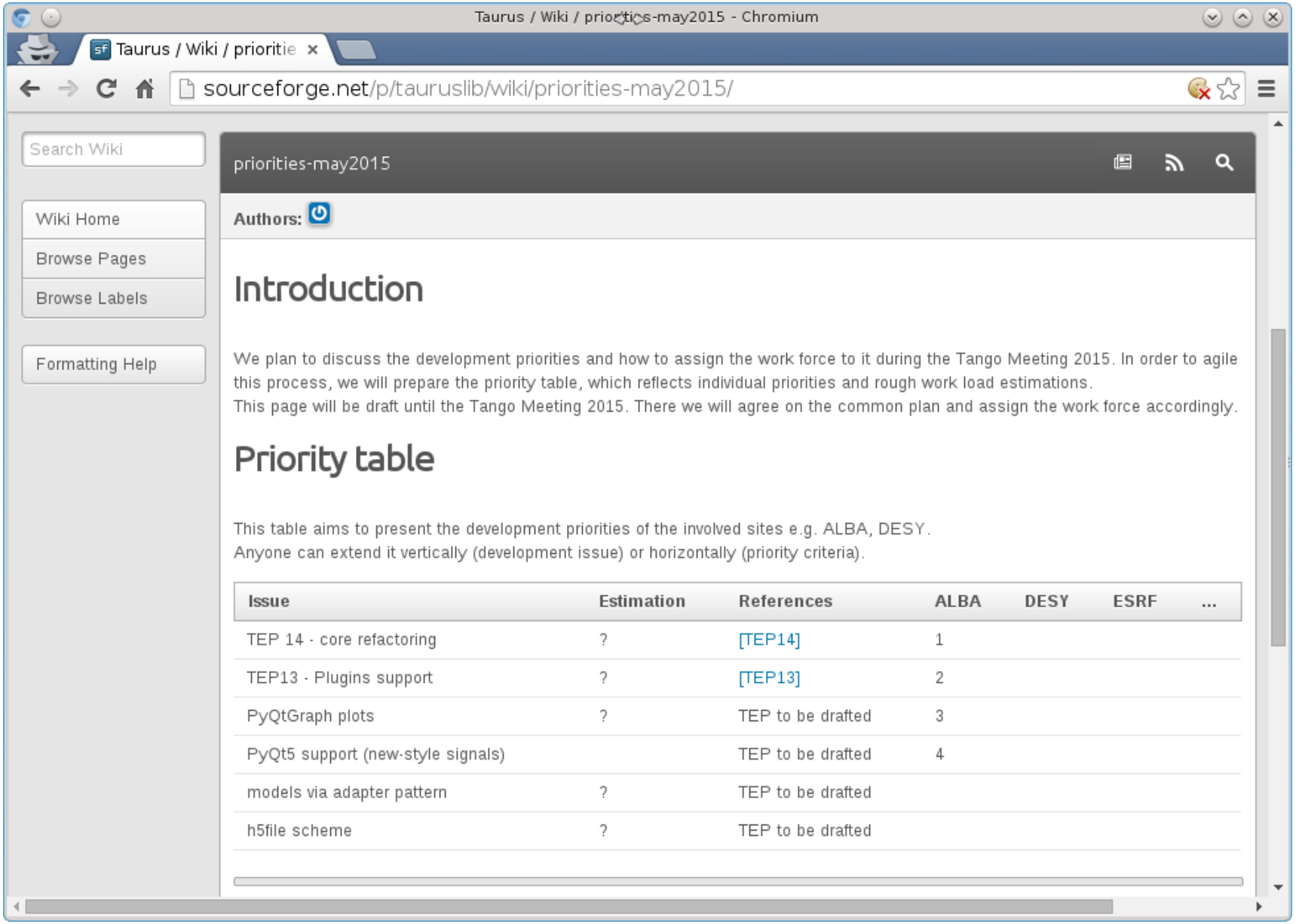


Room for improvement:

- **Integration process:**
  - **devel list is flooded with admin mails**
  - **unbalanced load on integrators**
  - **gerrit could be handy**
  - **review is open, but only integrators currently do it**
- **TEPs should be more agile**

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

Priorities for next releases



http://sf.net/p/tauruslib/wiki/priorities-may2015/

TEP3: new model syntax

TEP3 requires model names to be proper URIs (RFC3986-compliant)

```
Auth: <scheme>:<authority>[/<path>][?<query>][#<fragment>]
Dev:  <scheme>:[<authority>/]<path>[?<query>][#<fragment>]
Attr: <scheme>:[<authority>/]<path>[?<query>][#<fragment>]
Conf: <scheme>:[<authority>/]<path>[?<query>]#<fragment>
```

Old:	New:
tango://a/b/c/d?configuration=label	tango:a/b/c/d#label
tango://a/b/c/d	tango:a/b/c/d
tango://mydb:10000	tango://mydb:10000
eval://10*{tango://a/b/c/d}	eval:10*{tango:a/b/c/d}
eval://rand(a)?a=12	eval:a=123;rand(a)
eval://dev=Foo;bar()	eval:@Foo/bar()

http://sourceforge.net/p/tauruslib/wiki/TEP3/

multimodels in Taurus

- **Current Situation:**
  - widgets inherit from TaurusBaseComponent
  - base classes assume connection to **just 1 model**
  - Other cases require particular solutions

```
# standard widgets connect to 1 model
e = TaurusValueLineEdit()
e.model = 'a/b/c/d'

# but some parse the name and
# connect to other things!
m = PoolMotorTV()
m.model = 'a/b/c' # but it connects to:
                  #   'a/b/c/pos'
                  #   'a/b/c/state'

# and some widgets receive a list
# to manage it on their own!
f = TaurusForm()
f.model = ['a/b/c/d', 'e/f/g/h']
```

- **Coming (soon?)**: use the **adapter pattern**
  - widgets implemented as a pure Qt
  - model-management methods are inserted by **adapter decorators**
  - Multiple models supported with homogeneous API

```
@modelable('positionModel')
@modelable('stateModel')
class NewMotor(QWidget):
        ...

m = NewMotor()
m.stateModel = 'a/b/c/state'
m.positionModel = 'a/b/c/pos'
```

```
@list_modelable
class NewPlot(PyQtGraph.PlotWidget):
        ...

p = NewPlot()
m.model = ['a/b/c/d', 'e/f/g/h']
```