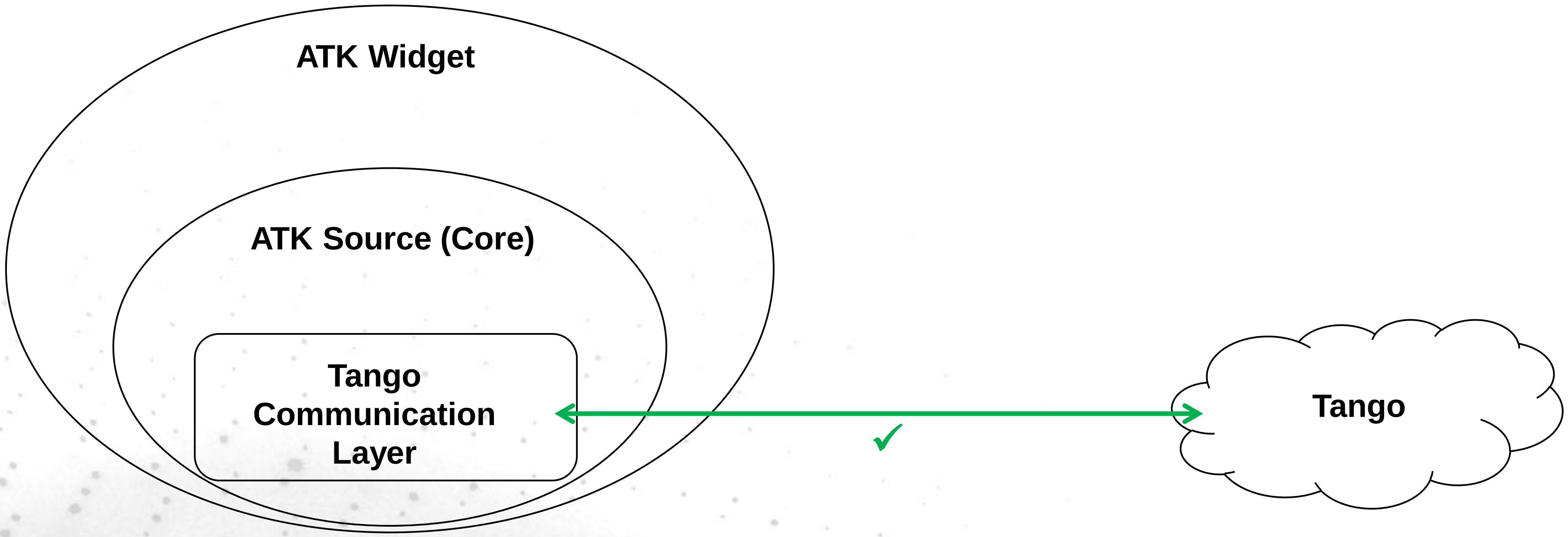
COMETE FRAMEWORK : G.U.I. CONNECTED TO MULTIPLE DATA SOURCES

Raphaël GIRARDOT : Synchrotron SOLEIL - FRANCE
On behalf of SOLEIL ICA team

Motivations of the project

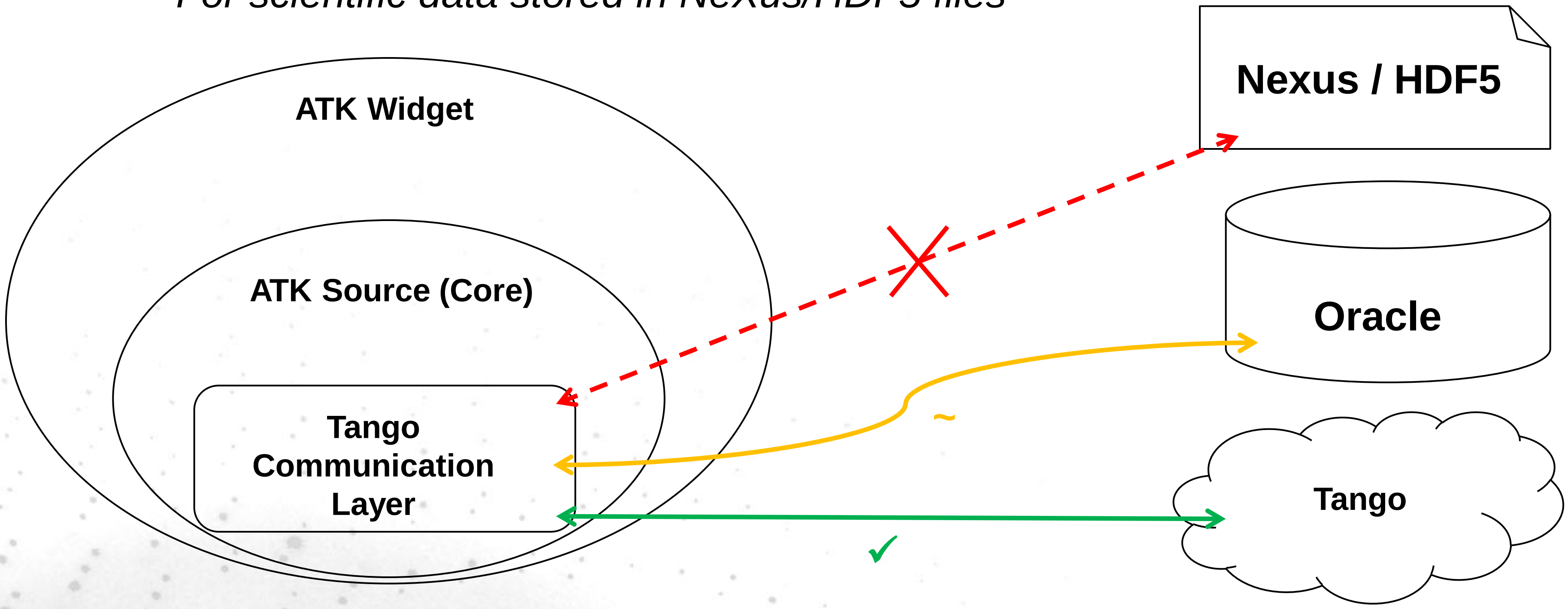
Motivations of the COMETE framework

- During the first years of SOLEIL construction, ICA team was focused on developing GUI applications on top of TANGO devices for the Control systems
- For this purpose, it was decided to use an existing framework: ATK



Motivations of the COMETE framework

- Then the focus was set on providing data storage and management applications for technical and scientific data.
  - *For the technical data the Tango Archiving service storing Tango attributes in MYSQL or Oracle database*
  - *For scientific data stored in NeXus/HDF5 files*



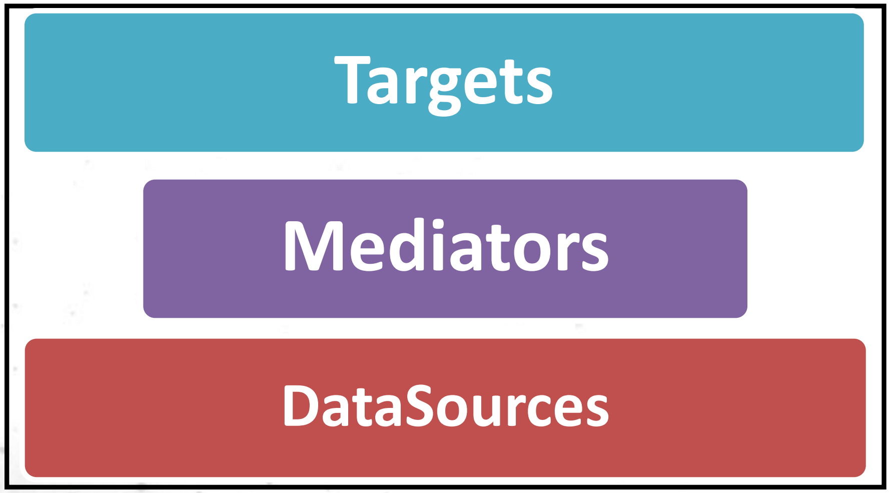
COMETE project started in 2009

Comete Architecture

COMETE Architecture : DataConnection Management

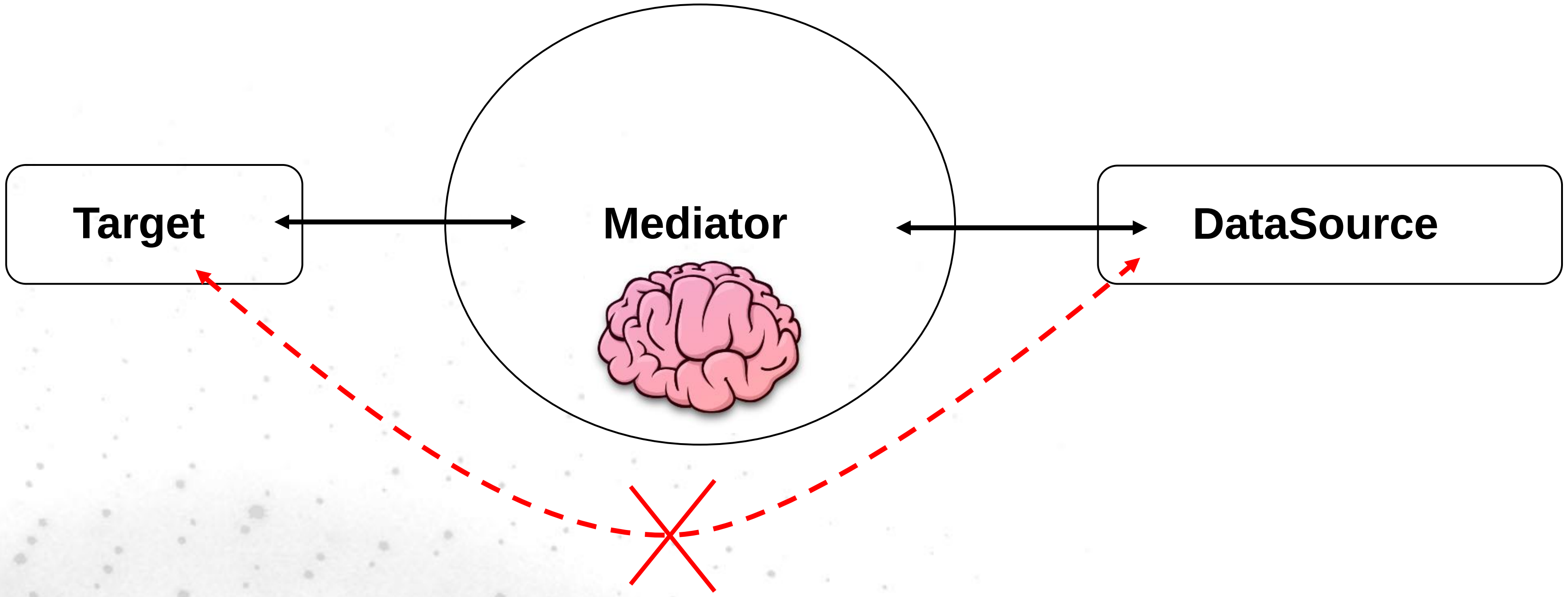
- The DataConnectionManagement module is a layer that allows connection between two abstract entities, called “Targets” and “Data Sources”

- DataConnectionManagement implements a Mediator pattern , as well as various other patterns such as Strategy, Observer



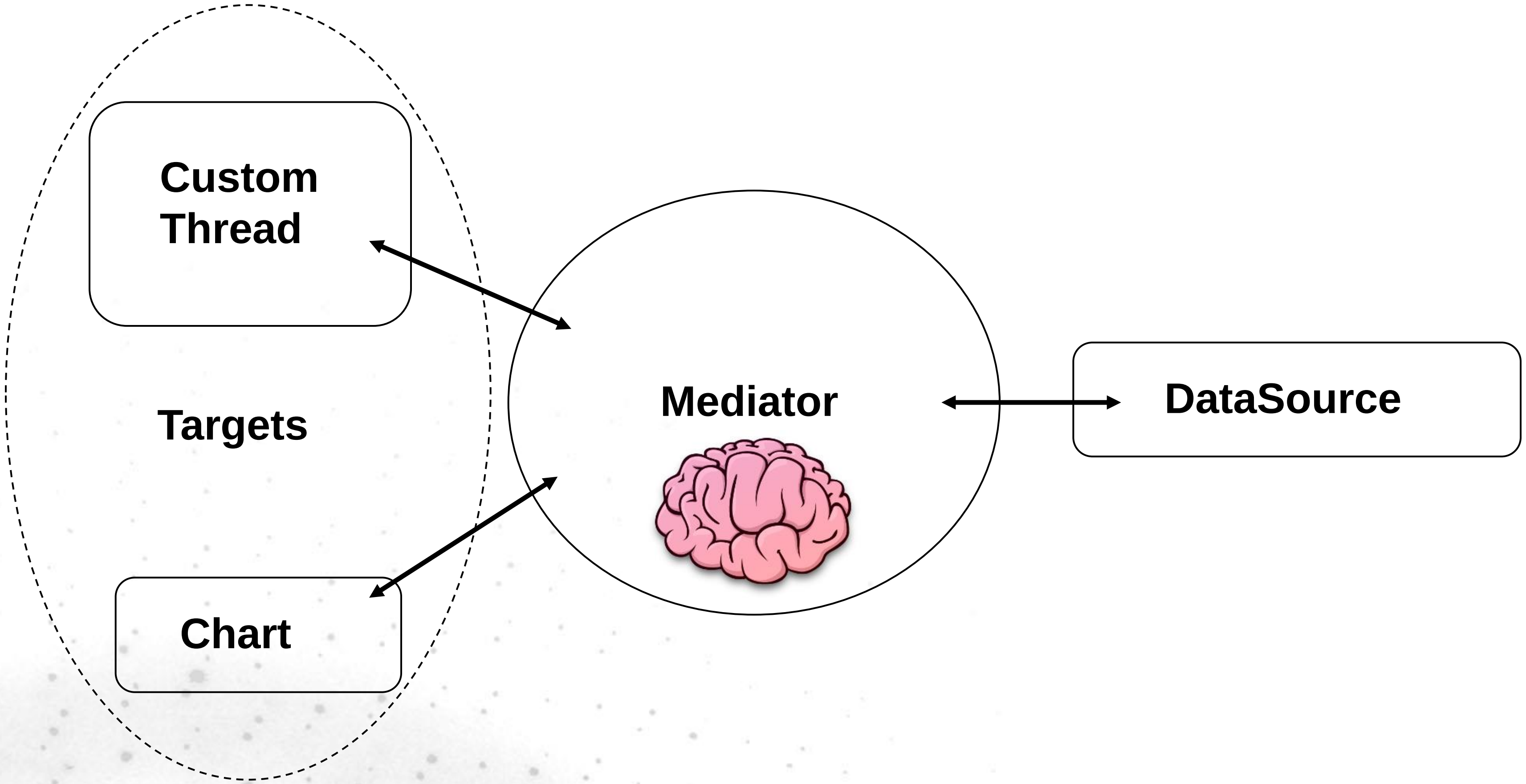
COMETE Architecture : DataConnection Management

- Mediator was chosen instead of MVC pattern because our two entities had to be completely independent from each other, to allow easily adding new widgets and sources



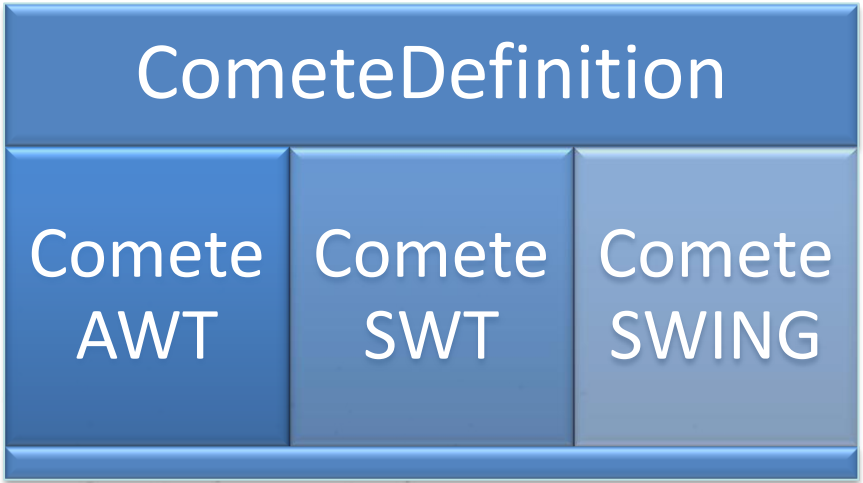
COMETE Architecture : DataConnection Management

- We decided to use Targets instead of Widgets, because we also needed to connect sources to non graphical targets (example: to custom processes)



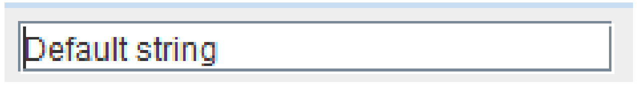
COMETE Architecture : Widgets

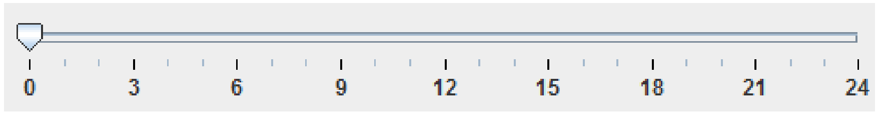
- COMETE **Widgets** are “**Targets**” specialization
- They musts comply with the interfaces described in CometeDefinition that makes them connectable to any data source
- Comete Widgets are available in three implementations (Swing, SWT & AWT)

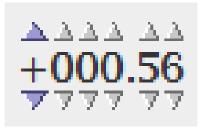


COMETE Architecture : Widgets

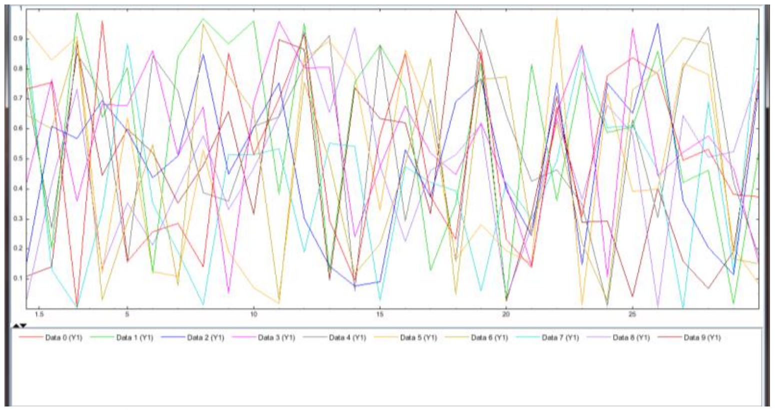
- Of course the COMETE framework allows adding easily new widgets.
- The **current library of widgets** can display:
  - ➢ *Scalar data (textfield, spinner, wheelswitch, slider, etc.)*



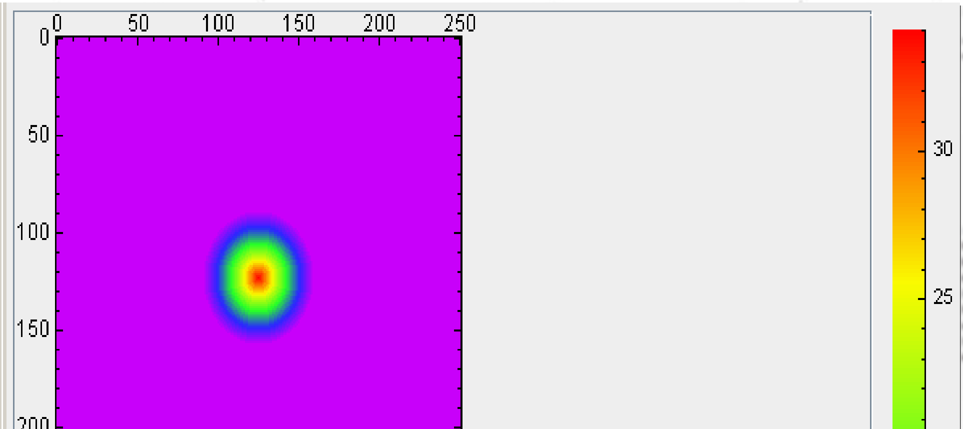


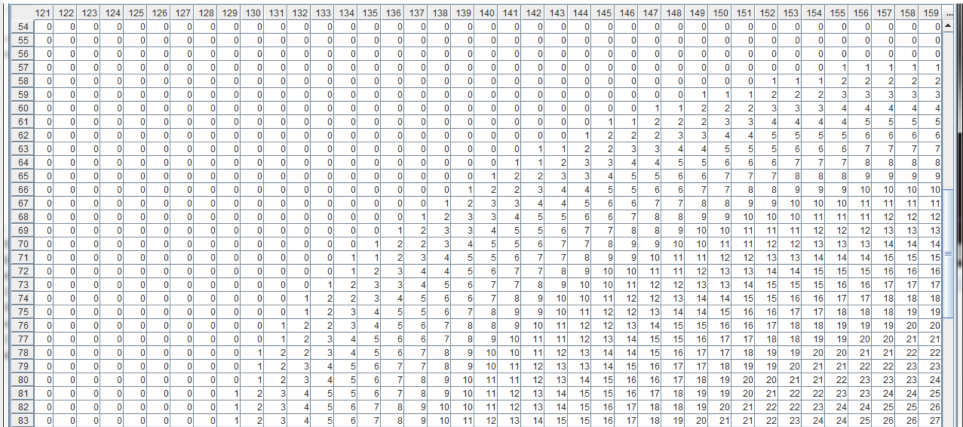


  - ➢ *Spectrum data (chart viewer)*



  - ➢ *Images (image viewer, tables)*

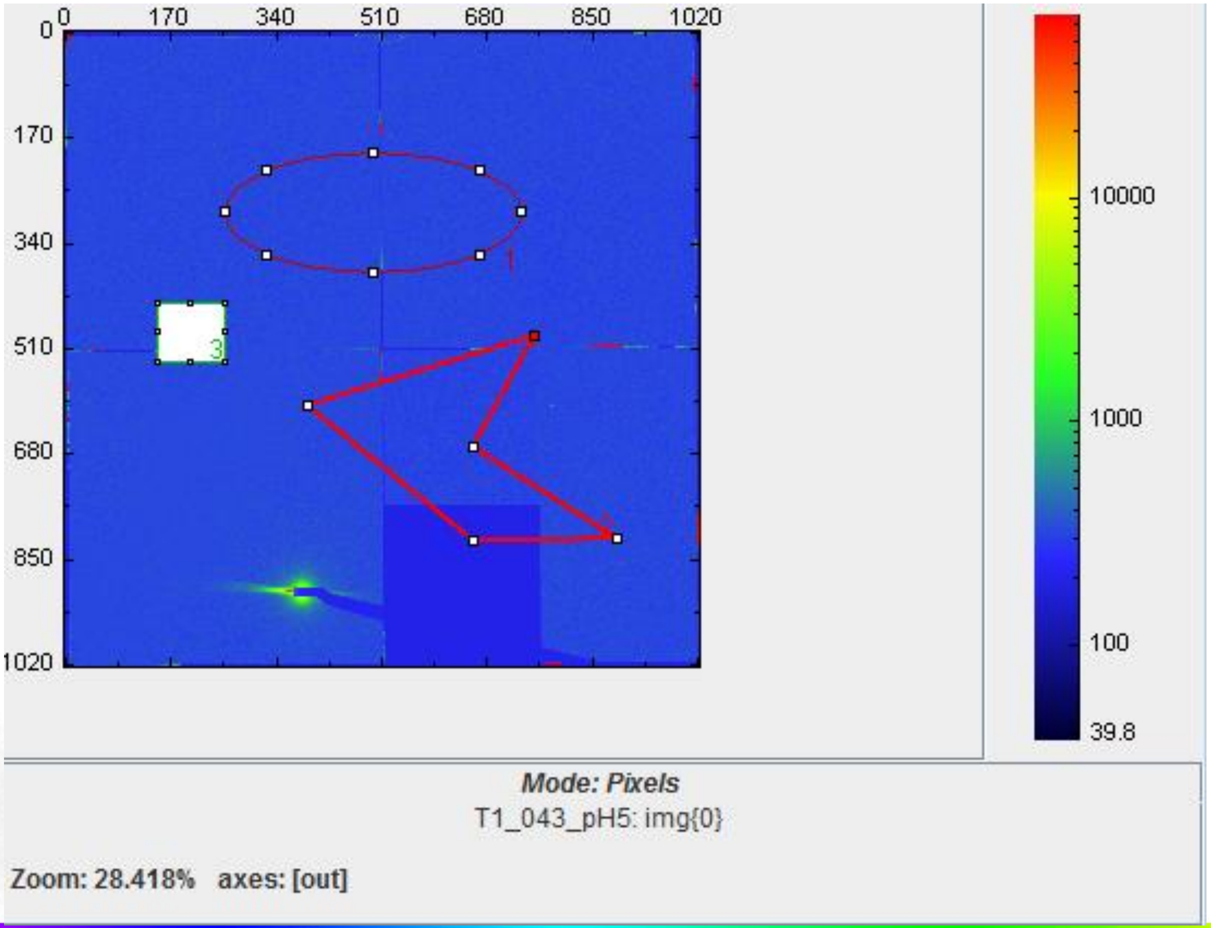




	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	2	2	2	2	2
59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	2	2	2	3	3	3	3	3
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	2	2	3	3	3	4	4	4	4	4
61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	2	2	3	3	4	4	4	4	5	5	5	5
62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	2	3	3	4	4	5	5	5	5	6	6	6	6
63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	2	3	3	4	4	5	5	5	6	6	6	7	7	7	7
64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	3	3	4	4	5	5	6	6	6	7	7	7	8	8	8	8
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	3	3	4	5	5	6	6	7	7	7	8	8	8	9	9	9	9
66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	3	4	4	5	5	6	6	7	7	8	8	9	9	9	10	10	10	10
67	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	3	4	4	5	6	6	7	7	8	8	9	9	10	10	10	11	11	11	11
68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	3	4	5	5	6	6	7	8	8	9	9	10	10	10	11	11	11	12	12	12
69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	3	4	5	5	6	7	7	8	8	9	10	10	11	11	11	12	12	12	13	13	13
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	3	4	5	5	6	7	7	8	9	9	10	10	11	11	12	12	13	13	13	14	14	14
71	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	3	4	5	5	6	7	7	8	9	9	10	11	11	12	12	13	13	14	14	14	15	15	15
72	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	4	5	6	7	7	8	9	10	10	11	11	12	13	13	14	14	15	15	15	16	16	16
73	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	3	4	5	6	7	7	8	9	10	10	11	12	12	13	13	14	15	15	15	16	16	17	17	17
74	0	0	0	0	0	0	0	0	0	0	0	1	2	2	3	4	5	6	6	7	8	9	10	10	11	12	12	13	14	14	15	15	16	16	17	17	18	18	18
75	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	5	6	7	8	9	9	10	11	12	12	13	14	14	15	16	16	17	17	18	18	18	19	19
76	0	0	0	0	0	0	0	0	0	0	1	2	2	3	4	5	6	7	8	8	9	10	11	12	12	13	14	15	15	16	16	17	18	18	19	19	19	20	20
77	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	6	7	8	9	10	11	11	12	13	14	15	15	16	17	17	18	18	19	19	20	20	21	21
78	0	0	0	0	0	0	0	0	0	1	2	2	3	4	5	6	7	8	9	10	10	11	12	13	14	14	15	16	17	17	18	19	19	20	20	21	21	22	22
79	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	7	8	9	10	11	12	13	13	14	15	16	17	17	18	19	19	20	21	21	22	22	23	23
80	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	11	12	13	14	15	16	16	17	18	19	20	20	21	21	22	23	23	23	24
81	0	0	0	0	0	0	0	0	1	2	3	4	5	5	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	19	20	21	22	22	23	23	24	24	25
82	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	10	11	12	13	14	15	16	17	18	18	19	20	21	22	22	23	24	24	25	25	26
83	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	20	21	21	22	23	24	24	25	26	26	27

COMETE Architecture : Widgets

- CometeSwing image viewer is based on **ImageJ** which allows using ImageJ macros and ROI definitions within the component



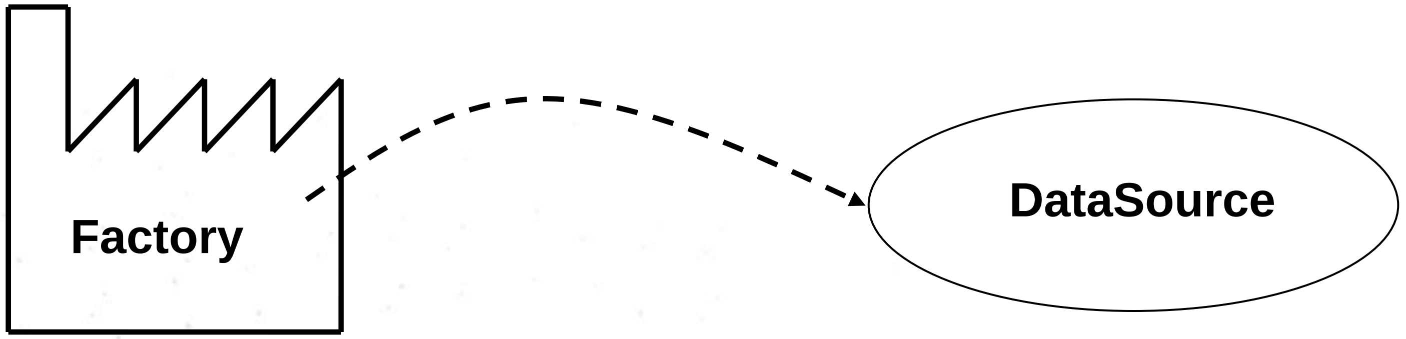
Mode: Pixels

T1_043_pH5: img{0}

Zoom: 28.418%   axes: [out]

COMETE Architecture : DataSources

- For better development separation, and in order to manage extra services around sources (like polling), DataSources creation follow the factory design pattern



COMETE Architecture : DataSources

- When someone wants to add a new data source to Comete, this person will implement a class that extends AbstractDataSource
- This guarantees that the COMETE mediator will be able to send data to the widget and vice-versa
- The developer will also make the factory (IDataSourceFactory) that handles sources production and data refreshment (if any)

- Today the **following data sources are available** within the COMETE framework:
    - *A TANGO implementation to access Control System data*
    - *A NeXus implementation to access scientific data from data reduction applications.*
    - *A first SQL implementation to access technical data produced by the Tango Archiving system*

COMETE Architecture : CometeBox

- The CometeBox module aims to simplify the use of COMETE for developers who intend to use the IDataSourceFactory
- When you want **to connect a target to some source** produced by an IDataSourceFactory, you have to do following steps:

  - *Instantiate your target*
  - *Instantiate your mediator*
  - *Instantiate your IDataSourceFactory*
  - *Instantiate a key*
  - *Ask your IDataSourceFactory to produce your source from this key*
  - *Ask your mediator to connect your source to your target*

- And of course, this allows accessing only 1 data.

COMETE Architecture : CometeBox

- CometeBox simplifies this access.
- It also offers the possibility to automatically connect your target to some meta-data around your source (for example, a state or quality concerning your source)

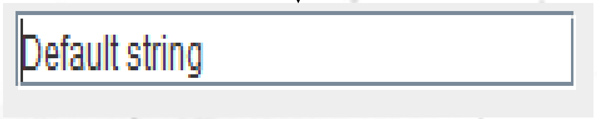
- To connect your target to a source and all its meta-data, you have to do following steps:
  - *Instantiate your target*
  - *Instantiate your CometeBox*
  - *Instantiate a key*
  - *Ask your CometeBox to connect your target to your key*

Use cases

Use cases : Connexion to a tango attribute

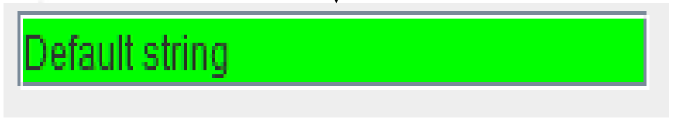
Case 1: Without CometeBox

```
TextField field = new TextField();
StringMediator mediator = new StringMediator();
TangoDataSourceFactory factory = new
TangoDataSourceFactory();
TangoKey key = new TangoKey();
TangoKeyTool.registerAttribute(«tango/tangotest/1
/string_scalar», key);
mediator.addLink(field,
factory.createDataSouce(key));
[...]
```


Default string

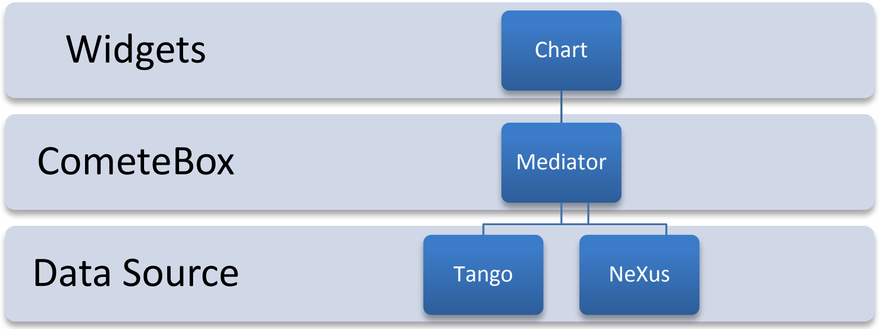
Case 2: With CometeBox

```
TextField field = new TextField();
StringScalarBox box = new StringScalarBox();
TangoKey key = new TangoKey();
TangoKeyTool.registerAttribute(«tango/tangot
est/1/string_scalar», key);
box.connectWidget(field,key);
[...]
```


Default string

Use cases : Connexion to multiple sources

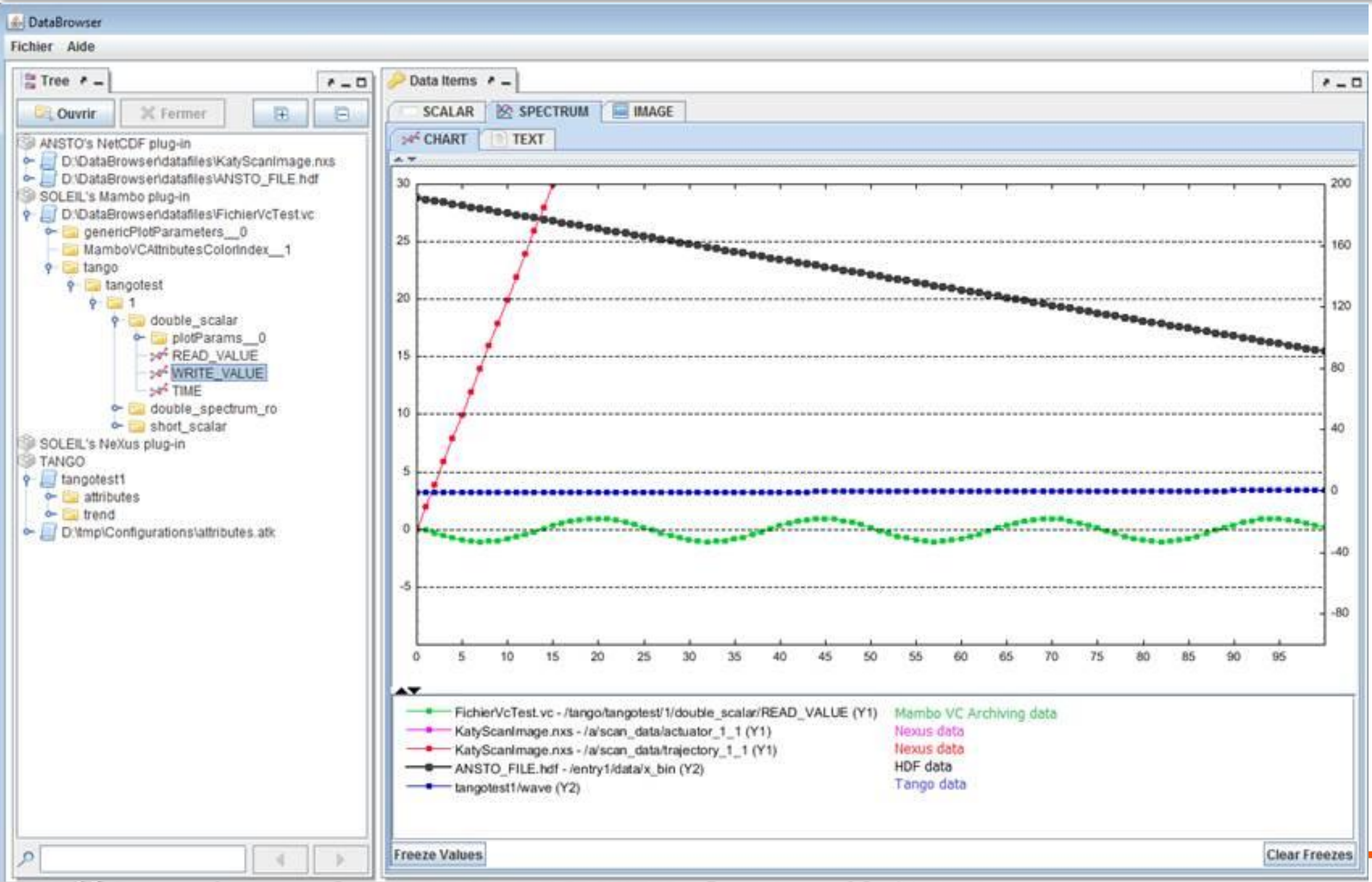
- A typical use case is to connect the same chart to both a tango and a NeXus source, in order to superpose spectrums from both sources



- This use case can be illustrated by following code:

```
Chart chart = new Chart();
ChartBox box = new ChartBox();
TangoKey tKey = new TangoKey();
TangoKeyTool.registerAttribute(«my/tango/device/myAttribute», tKey);
box.connectWidget(chart, tKey);
NexusKey nxKey = new NexusKey();
NexusKeyTool.registerDataSet(nxKey, filePath, datasetPath);
box.connectWidget(chart, nxKey);
[...]
```

Use cases : Chart connected to multiple sources



- FichierVcTest.vc - /tango/tangotest/1/double_scalar/READ_VALUE (Y1) — Mambo VC Archiving data
- KatyScanImage.nxs - /a/scan_data/actuator_1_1 (Y1) — Nexus data
- KatyScanImage.nxs - /a/scan_data/trajectory_1_1 (Y1) — Nexus data
- ANSTO_FILE.hdf - /entry1/data/x_bin (Y2) — HDF data
- tangotest1/wave (Y2) — Tango data

Conclusion

COMETE Framework : Summary

COMETE is the result of about 10 years of experience on GUI applications at SOLEIL.
It is now a **mature** and powerful framework **widely and daily used** by our developers

- The library of available **GUI** components is very **rich**
- Its architecture is **adaptable** in many contexts and other GUI frameworks

SOLEIL is open to collaborations on the project:

- With contributors (new widgets, new data sources, data services, etc…)
- With users, making feedback on their use and needs.

Questions ?

comete@synchrotron-soleil.fr