COMETE: A MULTI DATA SOURCE ORIENTED GRAPHICAL FRAMEWORK

G. Viguier, K. Saintin, Y. Huriez, M. Ounsy
SOLEIL Synchrotron, Gif-sur-Yvette, France.
R. Girardot, EXTIA, Sèvres, France

Abstract

Modern beamlines at SOLEIL need to browse a large amount of scientific data through multiple sources that can be scientific measurement data files, databases or TANGO control systems.

We created the COMETE[1] framework because we thought it was necessary for the end users to use the same collection of widgets for all the different data sources to be accessed. Conversely, for GUI application developers, the complexity of data source handling had to be hidden. As these two requirements are now fulfilled, our development team is able to build high quality, modular and reusable scientific oriented GUI software with consistent look and feel for end users.

CONTEXT

The SOLEIL ICA team is in charge of the control system and data reduction software on the beamlines. In the early years of development at SOLEIL, ICA has focused on the control system; today, however, the NeXus storage system is fully integrated into the control system through a set of dedicated devices. It is therefore possible to record any data coming from Tango devices into a NeXus file; SOLEIL beamlines produce daily experimental data files with sizes ranging from a few MB up to 100 GB. This is why there is a huge demand for data reduction applications.

The ATK[2] toolkit helped us to quickly develop applications dedicated to the control system and its supervision. The problem is that ATK is intimately linked to TANGO.

Data reduction software on the other hand is based on the experimental data NeXus file format and doesn't have a dedicated graphical toolkit. It is in this context that SOLEIL launched the COMETE project to propose a multi data source toolkit to help engineers to develop applications independently of the data source.

THE COMETE SOLUTION

Architecture

COMETE is a framework composed of three parts (Fig.1):

1. A set of graphical components (widgets) that are completely dissociated from a data source or even a data type.
2. A data source compatible with the graphical component, each corresponding to a data type.
3. Between the widgets and the data source, a communication layer called DataConnection Management.

Each component of this architecture will be described in the following sections.

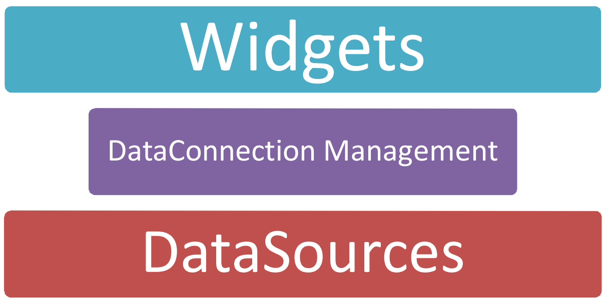


Figure 1: Comete Project Architecture.

Widgets

Comete Widgets are available in three implementations (Swing, SWT & AWT) and therefore integrate well into any existing JAVA software (Fig. 2). Anyone can use this set of widgets because they are based on the three major toolkits. In fact, a Comete Widget is a standard component with some code that makes it connectable to any COMETE data source.

COMETE components are dedicated to scientific data but anyone can easily add new widgets or use the existing components in another context.

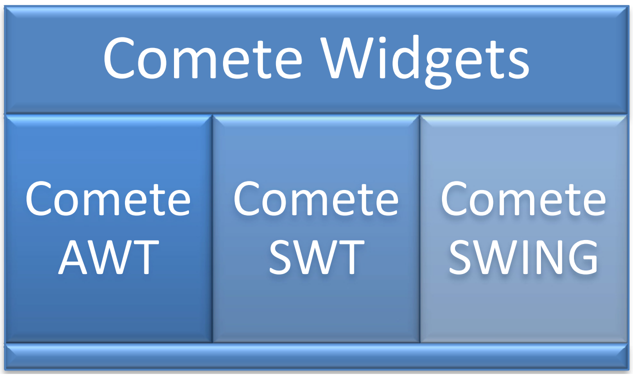


Figure 2: Comete Widget implementations.

The current widget can display:

- scalar data (textfield, spinner, wheelswitch, slider, etc.)
- spectrum data (chart viewer)
- images

Our image viewer is based on ImageJ [3] which is a popular image processing application. ImageJ is already used by many scientists, so it is very easy for them to use our viewer and they can run their old macros through our Java applications. Since our viewer encapsulates ImageJ, it has the same amount of image processing features as illustrated in Fig. 3.

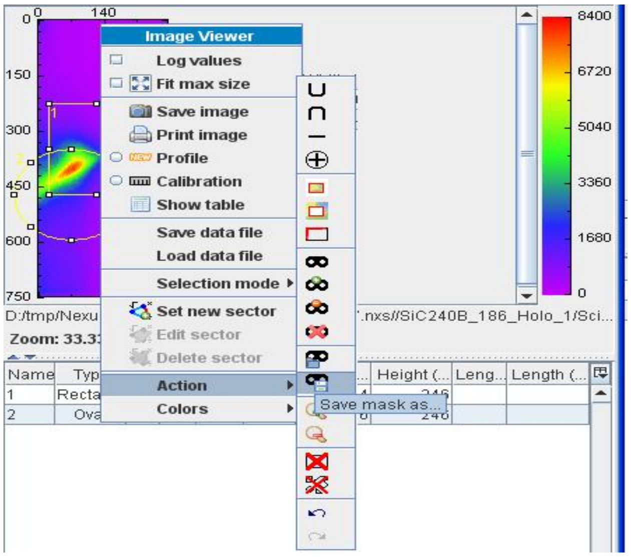


Figure 3: Comete Image Viewer features.

DataConnection Management

The DataConnectionManagement module is a layer that allows connection between two entities, called "target" and "data container".

This system implements a Mediator pattern, as well as various other patterns such as Strategy, Observer etc. Mediator was chosen instead of MVC pattern because our two entities had to be completely independent from each other and from the system evolution.

The focus for this project is to provide COMETE with a solution to connect a widget with a data source, like Tango or NeXus, in a generic way. That means this system is an abstraction of these entities, and provides useful tools to make them communicate without knowing their type.

For example, thanks to the Comete architecture, anyone can superimpose a spectrum coming from a NeXus file with the same spectrum from the control system (see Figure 4).

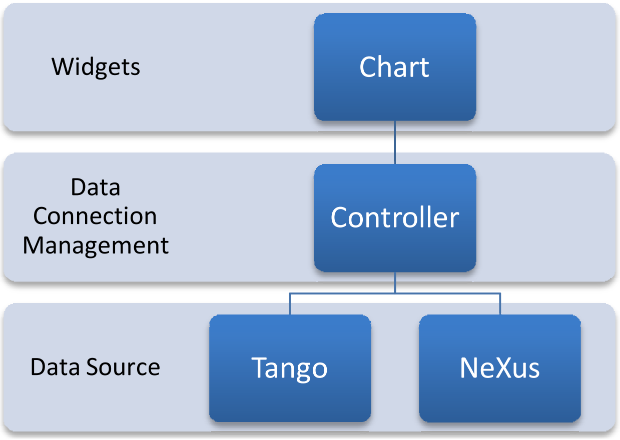


Figure 4: A widget connected to two sources.

DataSources

When someone wants to add a new data source to Comete, the only thing to do is to implement a class called AbstractCometeDataSource. This will make sure that the COMETE controller will be able to send data to the widget and vice-versa.

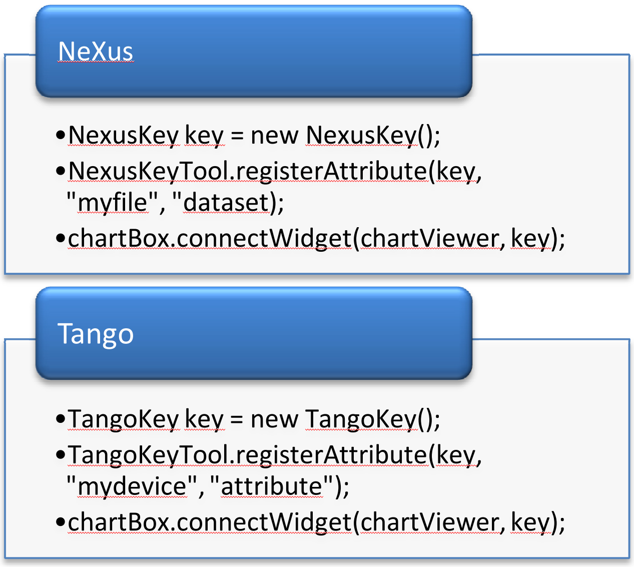
We made a TANGO implementation for control system software and we are working on a NeXus implementation for data reduction software. We plan to implement a data source for databases because we have an archiving system[4] built on ORACLE systems. Finally, we will have a data source for our sequencer software (Passerelle[5]) and thus enable our users to modify their sequences through our Java software.

CometeBox

The CometeBox module aims to simplify the use of COMETE. The fact that COMETE is a group of six different modules and the complexity of the architecture, especially the widget / source connection, makes it in fact difficult to use.

CometeBox operates on two levels. The first is for connection management: ordinarily, we need about ten to fifteen lines of code to make connections properly between entities, calling numerous functions coming from interconnected projects. With CometeBox, this is reduced to one line: one call to the correct function (Fig. 5).

The second goal of CometeBox is to centralize data handling: the connection process needs a wide range of data, some of them holding the same information. With CometeBox, this data is gathered and only information that is really necessary is stored.



```
•NexusKey key = new NexusKey();
•NexusKeyTool.registerAttribute(key,
 "myfile", "dataset");
•chartBox.connectWidget(chartViewer, key);
```

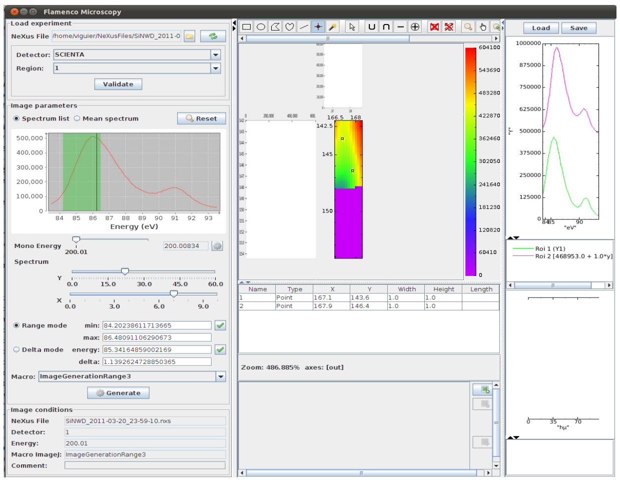
```
•TangoKey key = new TangoKey();
•TangoKeyTool.registerAttribute(key,
 "mydevice", "attribute");
•chartBox.connectWidget(chartViewer, key);
```

Figure 5: Connect a widget to a data source in 3 lines.

Performance

Five years ago, SOLEIL scientists were working with spectrum detectors or small image cameras. Performance was not a big deal because the data flow was not really impressive. Today this is not true, scientists are buying fast acquisition detectors with big CCDs and they store each frame into NeXus files.

COMETE needs to have a fast and efficient mechanism to transfer data from the source toward the widget. The first version of COMETE was really slow because it was only based on Java objects (*for example: each integer transferred was converted into an Integer Java object*). We thought that this would make COMETE a modern Framework but the memory footprint of every COMETE data reduction program was too expensive. Then we changed our philosophy from 'all object' to 'no object'. That means that today, each data item is transferred through COMETE as a primitive type. Today we are able to load stacks of images from our high data throughput multi detectors experiments [6] in a reasonable time.



NeXus File /home/viguier/NeXusFiles/SiNWD_2011-0
Detector: SCIENTA
Region: 1
Spectrum list / Mean spectrum
Mono Energy 200.01 200.00834
Range mode min: 84.20238611713665
max: 86.48091106290673
Delta mode energy: 85.34164859002169
delta: 1.1392624728850365
Macro: ImageGenerationRange3
Image conditions
NeXus File SiNWD_2011-03-20_23-59-10.nxs
Detector: 1
Energy: 200.01
Macro ImageJ: ImageGenerationRange3
Comment:

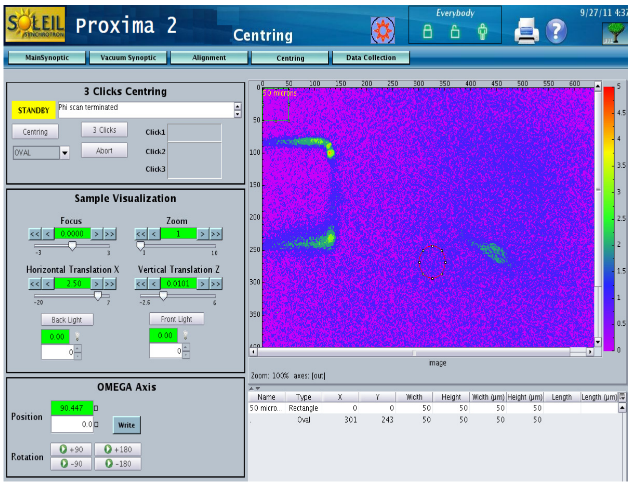
Name	Type	X	Y	Width	Height	Length
1	Point	167.1	143.6	1.0	1.0	
2	Point	167.9	146.4	1.0	1.0	

Zoom: 486.885% axes: [out]

Figure 6: Data reduction software dedicated to microscopy.

EXAMPLES

Fig. 6 and Fig. 7 are two examples of graphical applications based on COMETE and using the same component list with different data sources.



Zoom: 100% axes: [out]

Name	Type	X	Y	Width	Height	Width (µm)	Height (µm)	Length	Length (µm)
50 micro...	Rectangle	0	0	50	50	50	50		
.	Oval	301	243	50	50	50	50		

Figure 7: MXCube clone on PX2 beamline.

CONCLUSION

Today COMETE is a great tool for SOLEIL's developers. NeXus novices are able to build smart and effective data reduction software; anyone can develop a new supervision application without the help of TANGO gurus. In the future, we plan to optimize COMETE by implementing new refreshing systems. Data Reduction developers are looking forward to another feature: a mechanism of filters between the data source and the widget. Multiple filters can then be stacked and applied in a given order to an original Data Source to transform it to the final Data Source that will be really viewed by the COMETE widget.

REFERENCES

[1] G.Viguier, K.Saintin
http://sourceforge.net/apps/trac/comete/
[2] F. Poncet, J.L. Pons, "Tango Application Toolkit", ICALEPS'05, Geneva, Oct 2005.
[3] ImageJ : Image processing and analysis in Java
http://rsb.info.nih.gov/ij/
[4] J. Guyot, M. Ounsy, S. Pierre-Joseph Zephir "Status of the TANGO Archiving System", ICALEPS 2007
[5] A. Buteau, M. Ounsy, G. Abeille "A Graphical Sequencer for SOLEIL Beamline Acquisitions", ICALEPS'07, Knoxville, Tennessee - USA, Oct 2007.
[6] "Distributed Fast Acquisitions System for Multi Detector Experiments", ICALEPCS 2011, WEPKN003