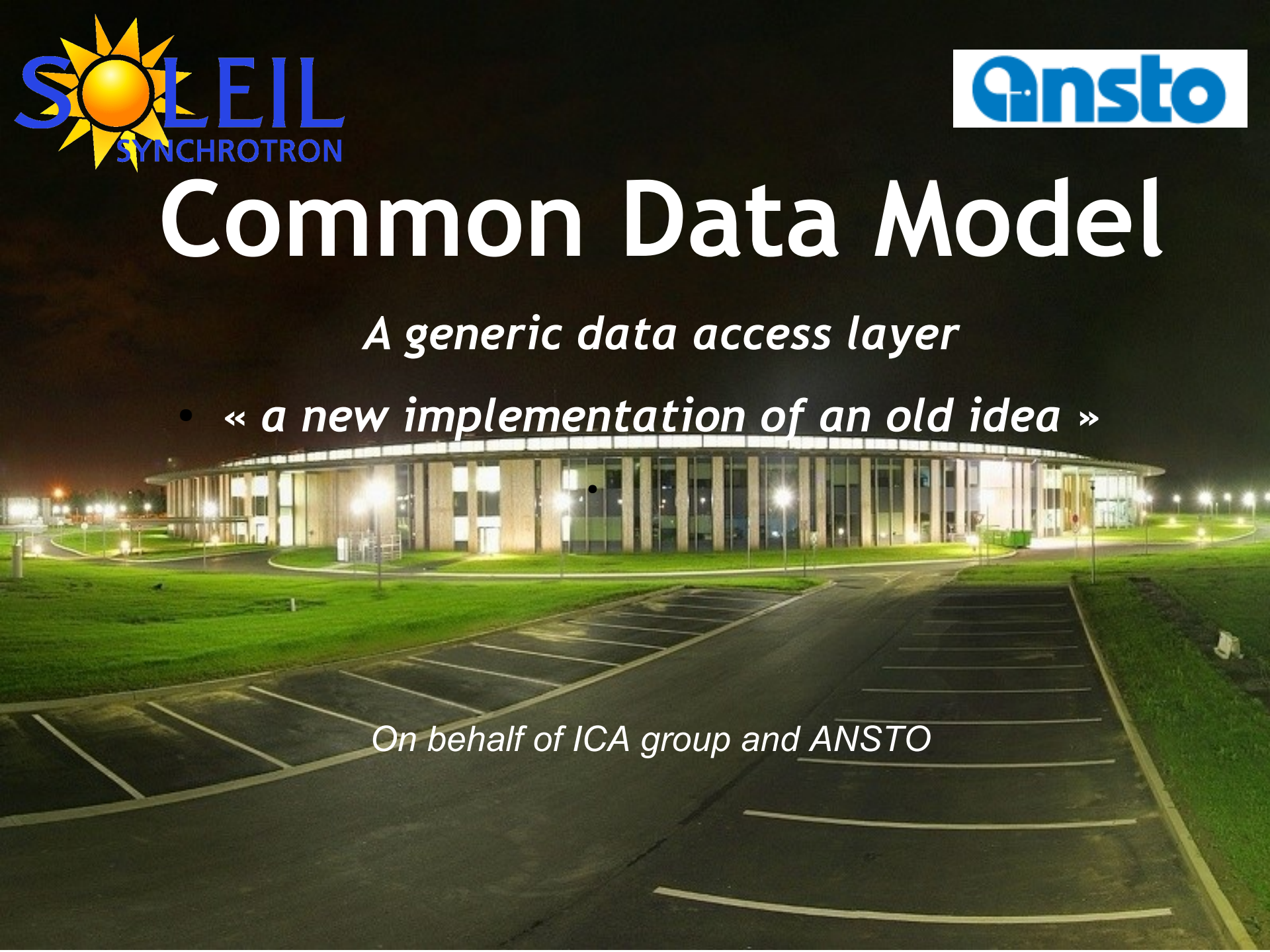


Common Data Model

A generic data access layer

- *« a new implementation of an old idea »*

On behalf of ICA group and ANSTO

SOLEIL SYNCHROTRON

Diversity

Is important

Agenda

- Which problems do we need to solve
  - ➔ *A quick feedback of SOLEIL after years of NeXus usage on a large scale basis*
  - ➔ *Which solutions are foreseen ?*
- The Common Data Model project
- The Common Data Model concepts
- Next steps of the project

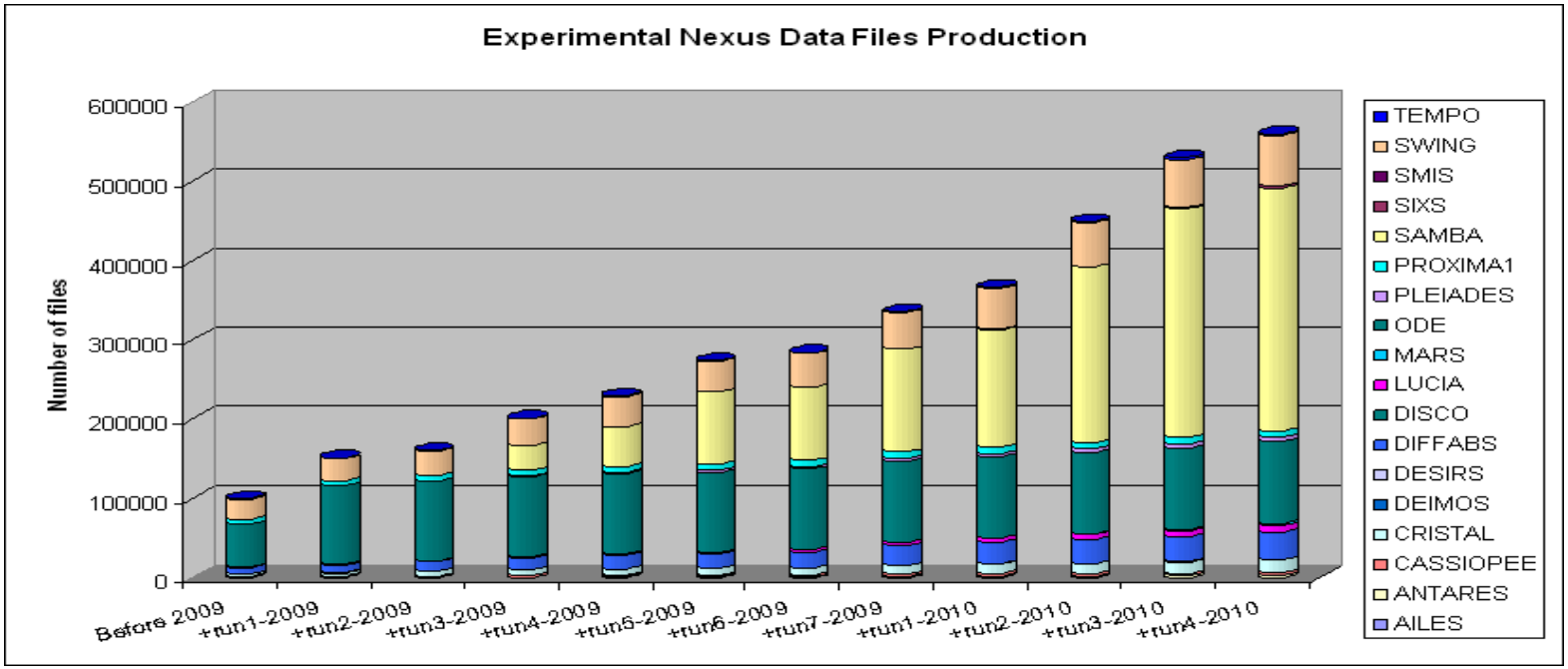
Which problems do we need to solve ?

A solution for which problem ?

- Find solutions to data format issues from the **data analysis point of view** ?
- Put in **common** different **algorithms** for analyzing data ?
- Find the most suitable ways to **exchange data** ?

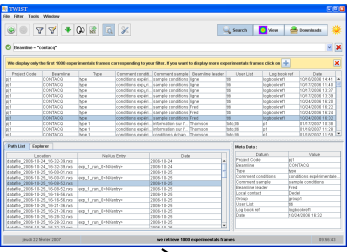
The foreseen solutions are :

- ✓Choose NeXus/HDF5 data format as the “SOLEIL standard” on all our beamlines
- ✓Define a standard internal data file structure for experimental data storage
- ✓**That 's the way we followed at SOLEIL during the last 7 years**

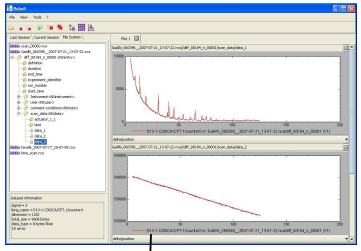


NeXus Files choice : Are we happy ?

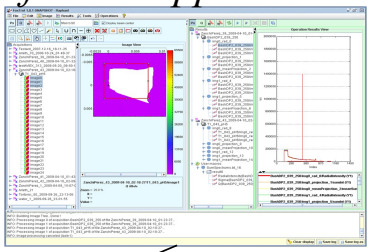
File retrieval



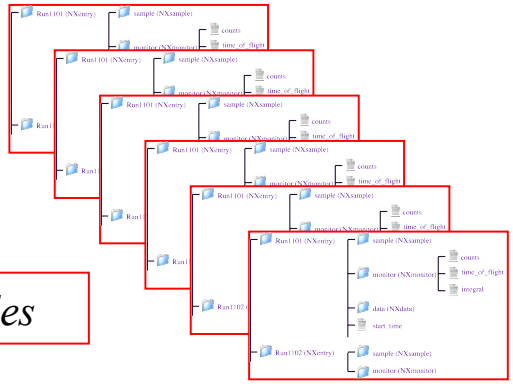
File browsing



SAXS Data Analysis foxtrot Application



NeXus Application Interface

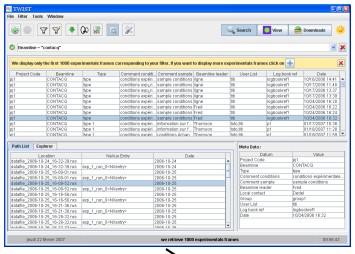


SOLEIL NeXus Files

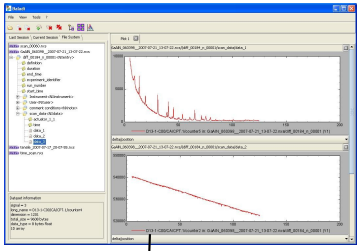
- NeXus is a good and efficient storage format
- Thanks to a unique API and a « SOLEIL standardized internal data organization », we could :
  - ✓*develop common software solutions*
  - ✓*Decouple the development of Acquisition softwares from Data Analysis software*

NeXus Files choice : Are we happy ?

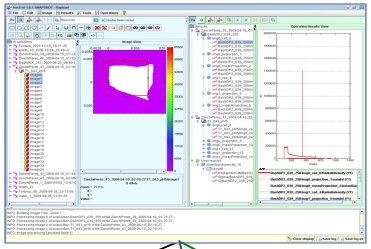
File retrieval



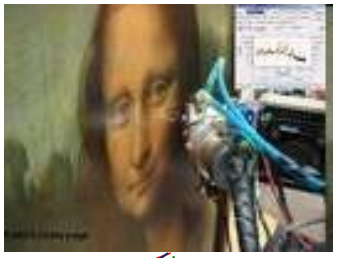
File browsing



SAXS Data Analysis foxtrot Application

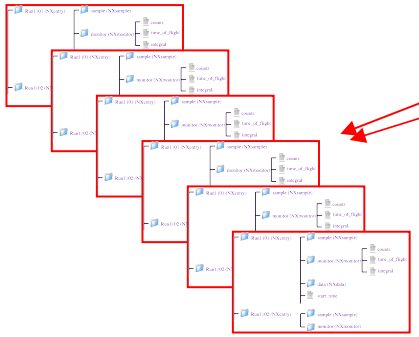


Data Analysis Application B

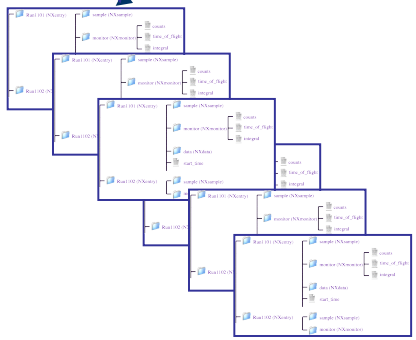


Data Analysis Application C

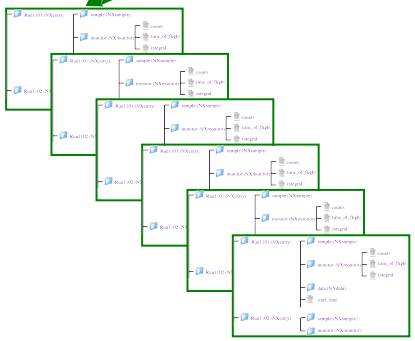
NeXus Application Interface



SOLEIL NeXus Files



ESRF Files

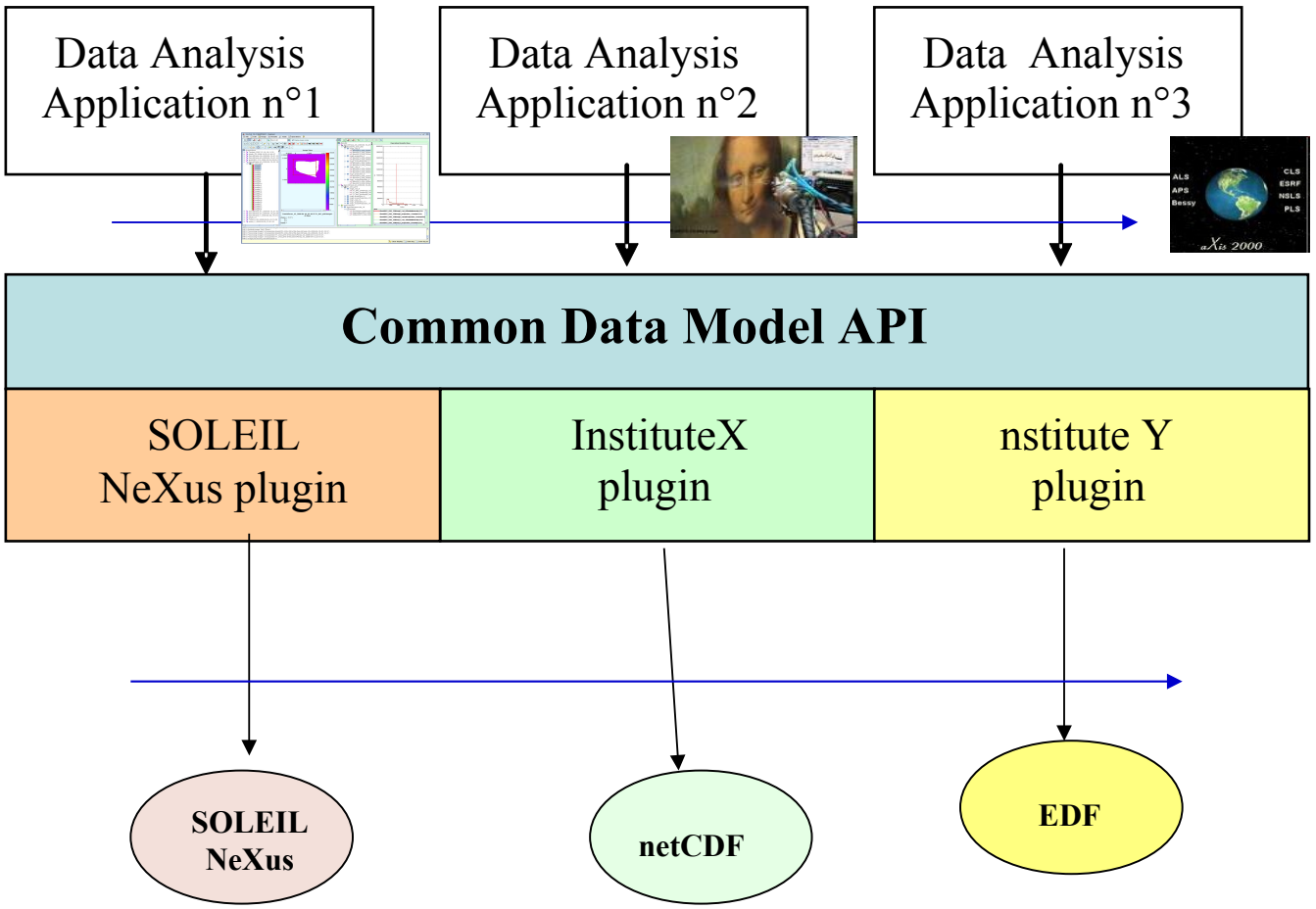


DESY Files

Our proposal in January 2010 at the Hyperspectral meeting at ESRF

Application developper adapt their application to the Common DataAccess API ONLY ONCE

Each institute implement plugin for each of its data formats



What's happening since then ?

- Since January 2010 , 2 processes are going on in parallel
  - ➔The NeXus committee continues trying to standardise file organisation
  - ➔SOLEIL with ANSTO proposed to make a first Common Data Model implementation
    - And as a proof of concept , to adapt our SAXS application to the CDM in order to be able do read data coming from SOLEIL and ESRF

The Common Data Model project

Genesis: the GDM project

- From the mid 2000's ANSTO started to develop a generic data analysis framework called GumTree
- Its data access layer, called GumTree Data Model :
  - ➔ was originally derived from the NetCDF format
  - ➔and extended to support NeXus through a new dedicated plug-in
- At the same time SOLEIL has developed a Data production process based upon the NeXus format
- In 2009, we started the COMETE project, a Java framework that aims to ease data visualization and data reduction applications

The CommonDataModel concepts

The Common Data Model

- The CDM is a API dedicated to data access, that should be the data access layer for new data reduction applications
- Using a plug-ins mechanism, it allows transparently accessing data from several data formats:
  - ➔It's the responsibility of institutes to develop the plug-in is accordance with theirs data formats
  - ➔Currently available formats: NetCDF, NeXus, EDF

.

Sample code

- Using the conventional way
  - ➔Opening a file

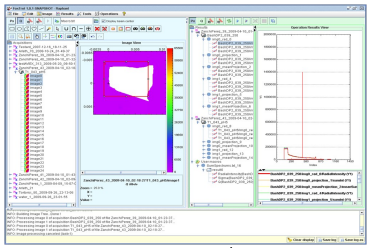
```
URI fileURI = new URI("file:/C:/example.hdf");
IDataset dataset = Factory.createDatasetInstance(fileURI);
dataset.open();
…
dataset.close();
```

  - ➔Getting data

```
IGroup root = dataset.getRootGroup();
String groupName = "sample";
IGroup subGroup = root.getGroup(groupName);
String itemName = "temperature";
IDataItem item = root.getDataItem(itemName);
```

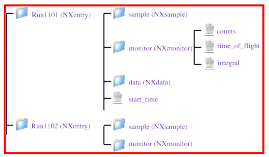
- **First benefit : You must no longer care about the kind of file format (EDF, HDF, CBF, etc...)**
- **But you need to know internal file organization**
  - ➔**In this example you need to know that temperature information is located in the sample group**

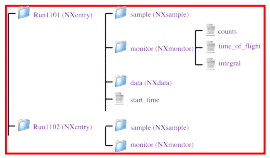
In reality we were blocked even before exchanging data with other institutes !!



SAXS Data Analysis foxtrot Application

NeXus Application Interface





- On SWING beamline, data organisation had been fixed by a « SAXS oriented » data acquisition sequence
- On CRISTAL beamline, data organisation inside the NeXus file had been fixed by a «Powder diffraction oriented » data acquisition sequence

SOLEIL NeXus File created on Beamline SWINGs

SOLEIL NeXus File created on Beamline CRISTAL

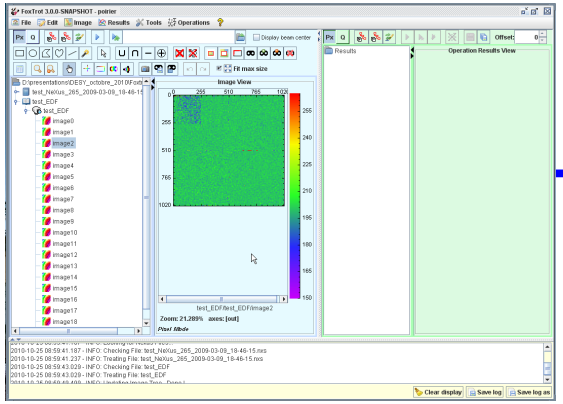
Introducing dictionary mechanism

- The main point of CDM is to allow a data analysis application that do not care about file format.
- We think it's not sufficient. Applications developers should'nt have to care about data organization in the files.
- To achieve this the CDM API introduce the notion of *dictionary*
- A dictionary is
  - ➔ A list of ***keywords*** (organized or not in separate groups, it doesn't matter)
  - ➔ A set of association between these ***keywords*** and ***data items*** in a specific data structure (NeXuS, NetCDF,...)

The dictionary mechanism

Each dictionary is defined by two files:

- ➔ a file were some keywords are declared through a basic organization (this file is written by the Analysis Application developer)
  - This organization define groups where interrelated data items are declared
  - One can define only one group if one prefer a flat organization
  - This allows the developers to define his own View on the data
- ➔ a file were these keywords are linked with the data files organization (this file is written by the DataAcquisition application developer)
  - It's a *map* where keywords are linked to data path



keywords Declaration file

```
<data-def name="Experiment name">
<!-- ex: EXAFS, SAXS,... →

  <group><key>monochromator</key>
    <entry id="m1">
      <key>wavelength</key>
    </entry>
  </group>

</data-def>
```

CDM

Files format plugin

keywords Mapping file

```
<map-def name="Experiment name">
<!-- ex: EXAFS, SAXS,... -->

  <entry id="m1">
    <key>wavelength</key>
    <return>

<path>NXEntry/NXMonochromator/wavelength
</path>
    </return>
  </entry>
  ...
</map-def>
```

0100110
1001110
0100110
11110...

Accessing data using dictionary

```
// Getting a handle to the data structure
IDataset aDataset = Factory.instantiateDataset("{data file
location}");
…
// Getting a reference to the root logical group, indicating we use
the dictionary API
ILogicalGroup rootGroup = aDataset.getLogicalRootGroup();

// Getting the energy data from the monochromator logical group
IDataItem energyData = rootGroup.getDataItem("monochromator:energy");
…
```

Dictionary mechanism benefits

- For developers: focus only on data analysis and reduction
  - ➔Quickly develop applications without taking care of data formats
  - ➔Make new data reduction apps natively usable by other institutes without adaptation
- For users of our institutes: using applications able to:
  - ➔process data acquired from several beamlines/institutes
  - ➔Process old public data acquired by other scientists
  - ➔Larger use of data analysis applications

Status of sofware development

- Version 2.0 of java CDM API has been released by ANSTO last week
  - ➔Including first implementation of dictionary mechanism
- SOLEIL uses this CDM library in the foxtrot application
  - ➔Providing transparent access to SOLEIL Nexus files and ESRF EDF file
  - ➔No time for demo: just believe me

The future of the project

The next steps

- We are expecting from this meeting :
  - ➔*New examples of application to adapt to the CDM :*
    - *EDNA ?*
    - *Scientific workbench ?*
    -
  - ➔*New collaboration on the implementation of CDM plugins*
- Collaboration on the C++ implementation
- Use CDM as the abstraction layer to share data analysis applications

The foxtrot application case

Real case study: a SAXS application

- During the meeting "*HDF5 as a Hyperspectral Data Analysis format*" at ESRF in january, 2010 we (SOLEIL) had announced a scientific 'proof of concept' before the end of the year.
- The prototype is based on our SAXS beamline data reduction application, called *Foxtrot*.
  - ➔ Foxtrot is used as a production application on the SWING beamline. It read and process NeXus files since two years.
  - ➔ The prototype is a modified version of Foxtrot, that use CDM and dictionary, and able to process NeXus files as well as ESRF Data Format (EDF) files

How foxtrot get the distance sample - detector ?

```
public Double getDistance(String filePath, String acquisitionName)
{
  ILogicalGroup root = getLogicalRootGroup(filePath);
  IKey key = generateKey(filePath, "distance");
  IDataItem item = root.getDataItem(key);
  result = extractNumber(Double.class, item);
  return result;
}
```

How foxtrot get the data of the detector ?

```
ImageData getImageData(AcquisitionData acquisitionData, int imageIndex)
{
  String filePath = acquisitionData.getParent().getFilePath();
  ILogicalGroup root = getRootGroup(filePath);
  IKeyFilter filter = generateKeyFilter(filePath, FilterLabel.INDEX,
                                        imageIndex);
  IKey key = generateKey(filePath, "image");
  key.pushFilter(filter);
  IDataItem item = root.getDataItem(key);
  IArray data = null;
  if (item != null)
    data = item.getData();
  ...
  return imageData;
}
```