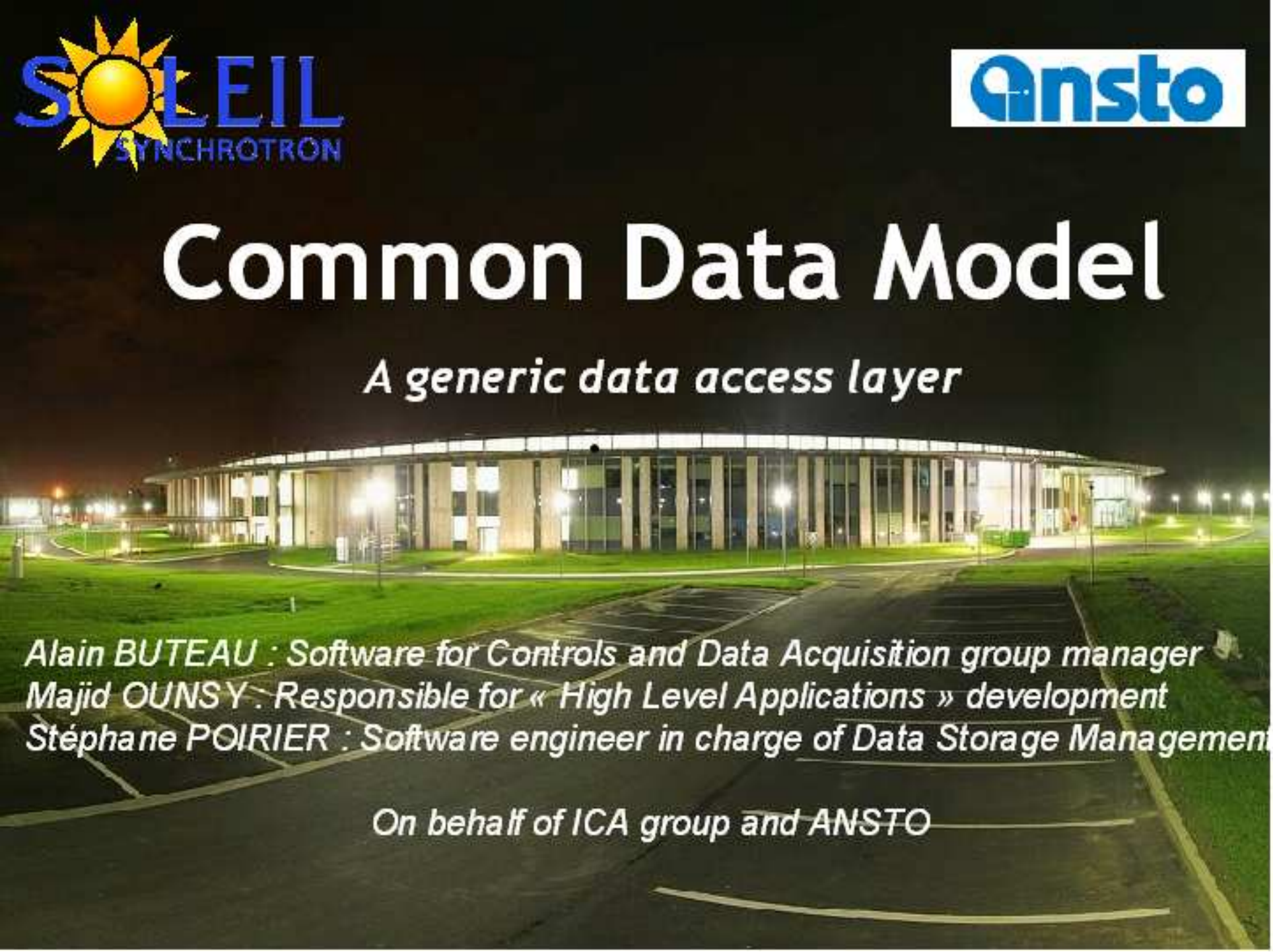

SOLEIL
SYNCHROTRON
Ansto
Common Data Model
A generic data access layer
Alain BUTEAU : Software for Controls and Data Acquisition group manager
Majid OUNSY : Responsible for « High Level Applications » development
Stéphane POIRIER : Software engineer in charge of Data Storage Management
On behalf of ICA group and ANSTO

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
  - *A live demo on a SAXS application*
- Next steps of the project

SOLEIL
SYNCHROTRON

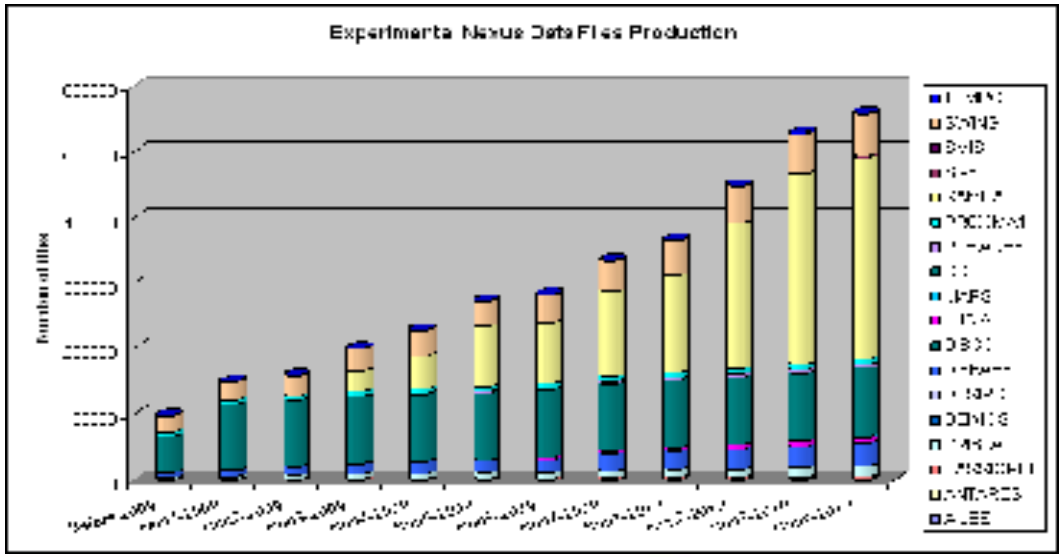
Which problems do we need to solve ?

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 :

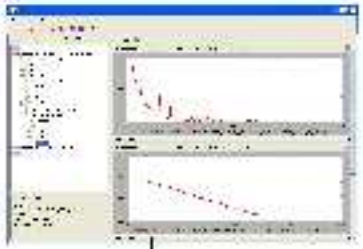
- ✓Choose NeXus/HDF5 data format as the “SOLEIL standard” on all our beamlines
- ✓Define a standard internal data file structure for experimental data storage
- ✓**That 's the way we followed at SOLEIL during the last 7 years**



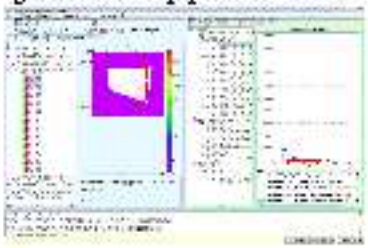
NeXus Files choice : Are we happy ?

File retrieval

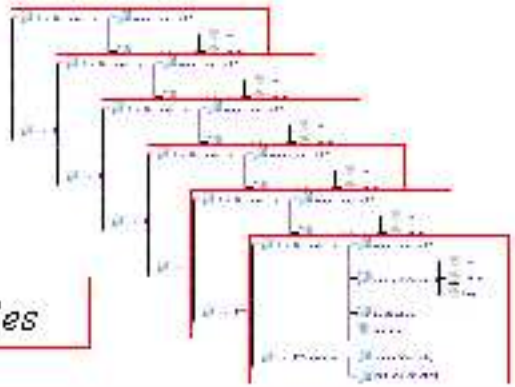
File browsing



SAXS Data Analysis foxtrot Application



NeXus Application Interface



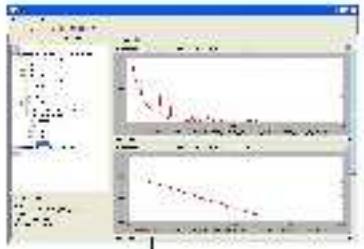
SOLEIL NeXus Files

- **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*

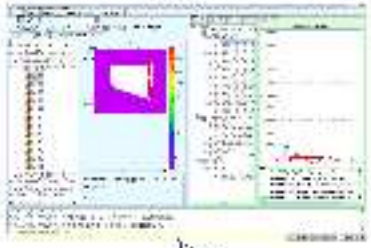
NeXus Files choice : Are we happy ?

File retrieval

File browsing



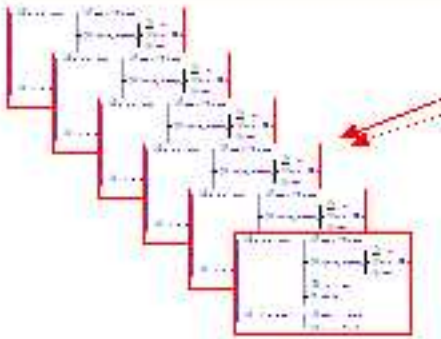
SAXS Data Analysis
foxtrot Application



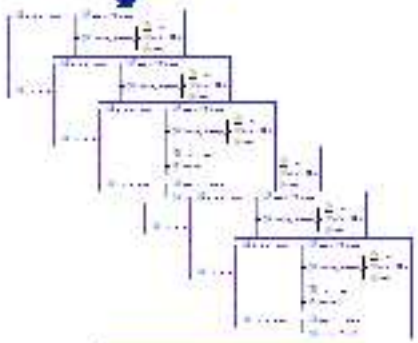
Data Analysis
Application B

Data Analysis
Application C

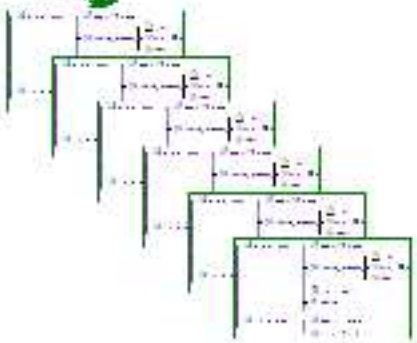
NeXus Application Interface



SOLEIL NeXus Files

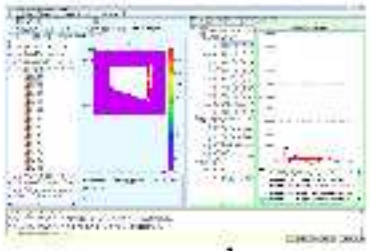


ESRF Files



DESY Files

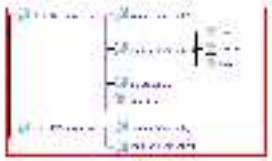
In reality we were blocked even before exchanging data with other institutes !!



SAXS Data Analysis foxtrot Application

- 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

NeXus Application Interface



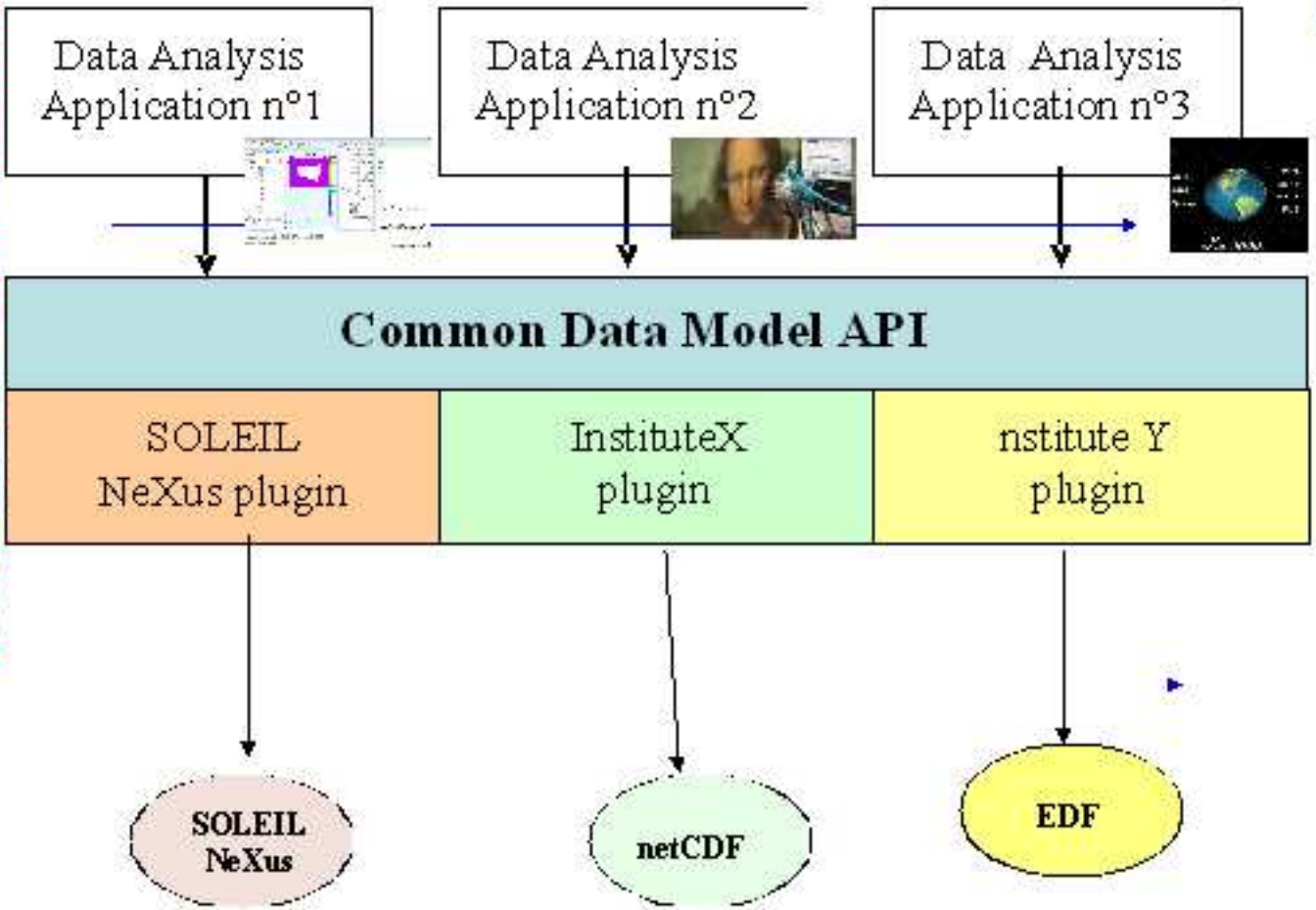
SOLEIL NeXus File created on Beamline SWINGs

SOLEIL NeXus File created on Beamline CRISTAL

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 , two processes are going on in parallel
  - The NeXus committee with scientists defines “application definitions”
    - Which information are mandatory to analyse data for each experimental technique (SAXS, EXAFS, ..)
  - 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

These processes are complementary

- When an Application Definition is adopted by NIAC
  - it will be natively included in all the plugins
- During the Application Definition Process
  - CommonData Model plugins adapt themselves to different data formats
  - while data analysis applications still use the same CDM functions to access data
- When Multiple Experiment techniques are mixed
  - An analysis application can have its own logical organization of the combined Application Definitions with CDM

The Common Data Model project

Genesis: the GDM project

- From the mid 2000's ANSTO started to develop a generic data analysis framework. It provide:
  - a GUI (GumTree Workbench)
  - a middleware system (GumTree Server)
  - a source code framework (GumTree Framework)
  - see http://gumtree.codehaus.org/ for mode information
- 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

First steps toward collaboration

- At the same time SOLEIL has developed a Data production process based upon the NeXus format
- Since 2009, we started the COMETE project, a Java framework that aims to ease data visualization and data reduction applications
- During ICALEPS 2009, a meeting between T. LAM (ANSTO), M. Ounsy & A. Buteau (Soleil) and A. Götz (ESRF) showed that our institutes are enough mature on the topic to be able to start concrete collaboration.

A statement from ANSTO GumTree project

More than a primitive data type

The CDM Array class is more than a primitive Java or C++ array. It is a **scientific data object with file IO**, allowing you to slice and dice arrays and to do math. These methods are the same as found in the default interface implementation provided by netcdf

Yet an other statement from the GumTree project

The CDM API has an **error** object, that provides propagation of count uncertainties based on Poisson statistics with every math operation. This is extensible to other uncertainty calculations

Note that an uncertainties property has been added to NXdata at the 2010 NIAC, hence providing a standard way to store uncertainties to file

Collaboration status

- Collaboration between SOLEIL & ANSTO
  - ➡ started at the end of 2009
  - ➡when ANSTO sent to SOLEIL its current implementation of the GumTree Data Model.
- This collaboration is based on a 'Gentlemen Agreement' between our institutes

Source code sharing

- Since 02/2010, SOLEIL has a write access to the SVN repository of the CommonDataModel project
  - The CDM is a sub-project of GumTree.
  - The SVN repository is localized here: https://svn.codehaus.org/gumtree/datamodel/
  - Last week the first official version (1.0) of CDM was released by ANSTO
  - This version includes the dictionary mechanism proposed by SOLEIL

The CommonDataModel concepts

The Common Data Model

- The CDM is a API dedicated to data access, that should be the data access layer of new data reduction applications
- Using a plug-ins mechanism, it allows transparently accessing data from several data formats:
  - It's the responsability of institutes to develop the plug-in is accordance with theirs data formats
  - Currently available formats: NetCDF, NeXus, EDF
- It offers two ways to access data:
  - The conventional one, which need a strong knowledge of data-files structures,
  - A new, innovative way, which no longer need any knowledge about the data file structure, through a ***dictionary*** mechanism.

Sample code

- Using the contentional way
  - Opening a file

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

  - Getting data

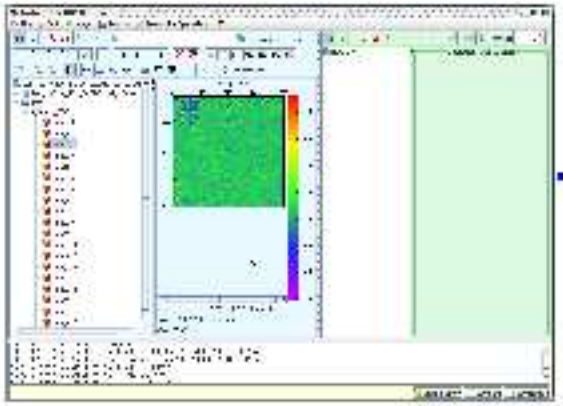
```
IGroup root = dataset.getRootGroup();
String groupName = "sub_group_name";
IGroup subGroup = root.getGroup(groupName);
String itemName = "data_item_name";
IDataItem item = root.getDataItem(itemName);
IGroup aGroup = subGroup.getGroupWithAttribute(“attribute name”, value);
IDataItem anItem = subGroup.getDataItemWithAttribute(“attribute name”, value);
```

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 don't have to care about data organization in the files.
- To achieve this the CDM API introduce the notion of *dictionary*
- A dictionary is
  - Some ***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,...)
  - Please see keywords as named concepts. If a name seems incorrect you can define a synonym of it and use it in data reduction application

The dictionary mechanism

- In the default implementation dictionary are XML files
- Each dictionary is defined by two files:
  - a file were some keywords are declared through a basic organization
    - This organization define groups where interrelated data items are declared
    - One can define only one group if one prefer a flat organization
  - a file were these keywords are linked with the data files organization.
    - It's a *map* where keywords are linked to data path
    - Keywords synonyms can be added in this files



keywords Declaration file

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

  <group><key>monochromator|crystal</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>lambda</key>
    <return>
       <path>path/to/user/name</path>
    </return>
  </entry>
  ...
</map-def>
```

0100110
1001110
0100110
11110...

keywords declaration file

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

  <group><key>user</key>                                   <!-- logical group where
    <entry id="u1"><key>name</key></entry>
    <entry id="u2"><key>e-mail</key></entry>
  </group>

  <group><key>source</key>
    <entry id="s1"><key>facility-name</key></entry>
    <entry id="s2"><key>type</key></entry>                 <!-- 'X-ray', 'Neutron' -->
  </group>

  <group><key>monochromator|crystal</key>          <!-- 'monochromator' or 'crystal' -->
    <entry id="m1"><key>wavelength</key></entry>
  </group>

  <group id="data">
    <entry id="d1"><key>raw-data</key></entry>
  </group>

</data-def>
```

- This first document (item 'data-def') is generic and may be distributed to application developers

Keywords mapping

```
<map-def name="Experiment name">            <!-- ex: EXAFS, SAXS,... -->
  <entry id="u1"><return><path>path/to/user/name</path></return></entry>

  <entry id="s1"><return><path>path/to/facility/name</path></return></entry>

  <entry id="m1">
    <key>lambda</key>                       <!-- a synonym of the keyword 'wavelength' -->
    <return><path>path/to/user/name</path></return>
  </entry>
  ...
</map-def>
```

- The second document is related to a particular file format/plug-in
- However the 'data-def' element content may be written from a existing application point of view in order to ease its adaptation of the dictionary mechanism

Using dictionary

- The CDM API offers two ways to accessing data:
  - Direct access to the data, this implies a deep knowledge of the datafiles structures
  - Indirect access through the dictionary mechanism.
- Code sample:

```
// 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

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;
}
```

The future of the project

Status of collaborations on the CDM project

- The java API version 1.0 is now officially released and available on codehaus
- The porting of the API to C++ and python is now the next development task
- ESRF is studying the current API and may join the collaboration in the next months
- First version of CDM documentation will be sent to Dr. Bernstein for review by the HDF group
- We are expecting from this meeting :
  - *New examples of application to adapt to the CDM*
  - *New collaboration on the implementation of CDM plugins*