HDF5 as a standard file format for synchrotron experiments

Eugen Wintersberger
GridKa 2016, 01.09.2016

HELMHOLTZ ASSOCIATION

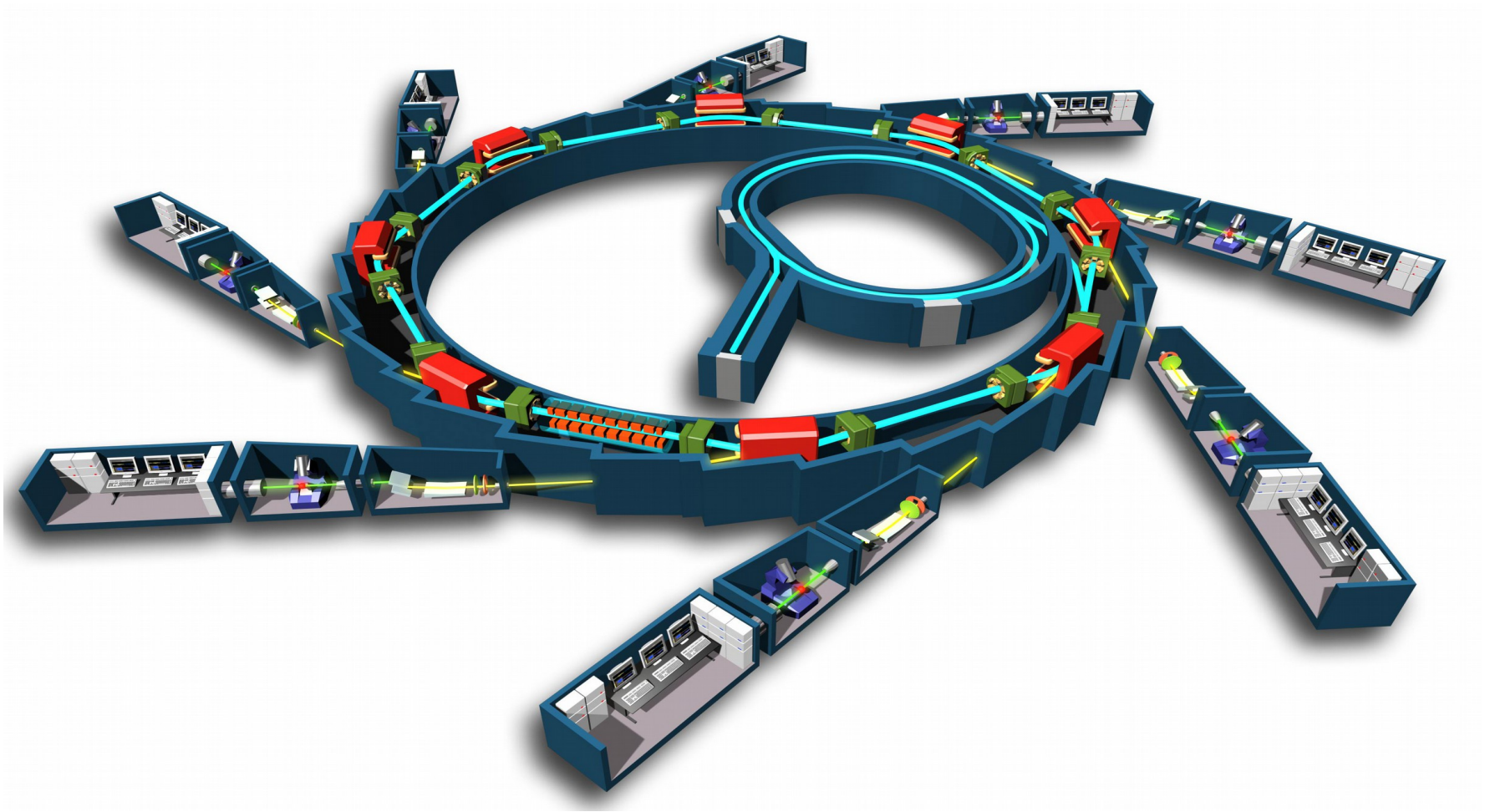
DESY

Overview

- What is a synchrotron and how to use it?
- Current status of file-formats and future demands
- HDF5 for synchrotron radiation experiments
- NeXus = HDF5 + semantics
- New developments in HDF5

What are synchrotrons?

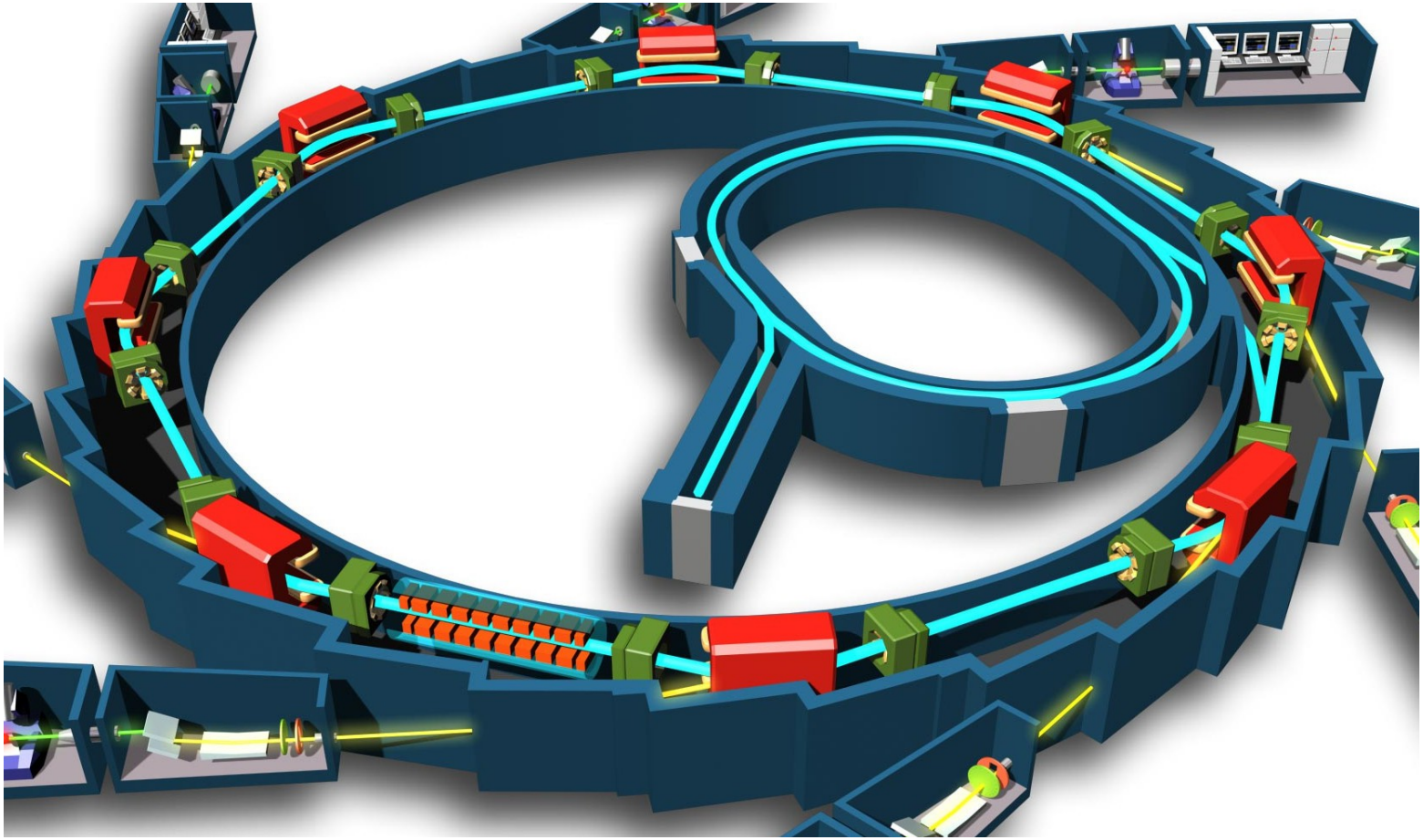
How does it work?



Copyright © EPSIM 3D/JF Santarelli

Produces x-rays for material science experiments!

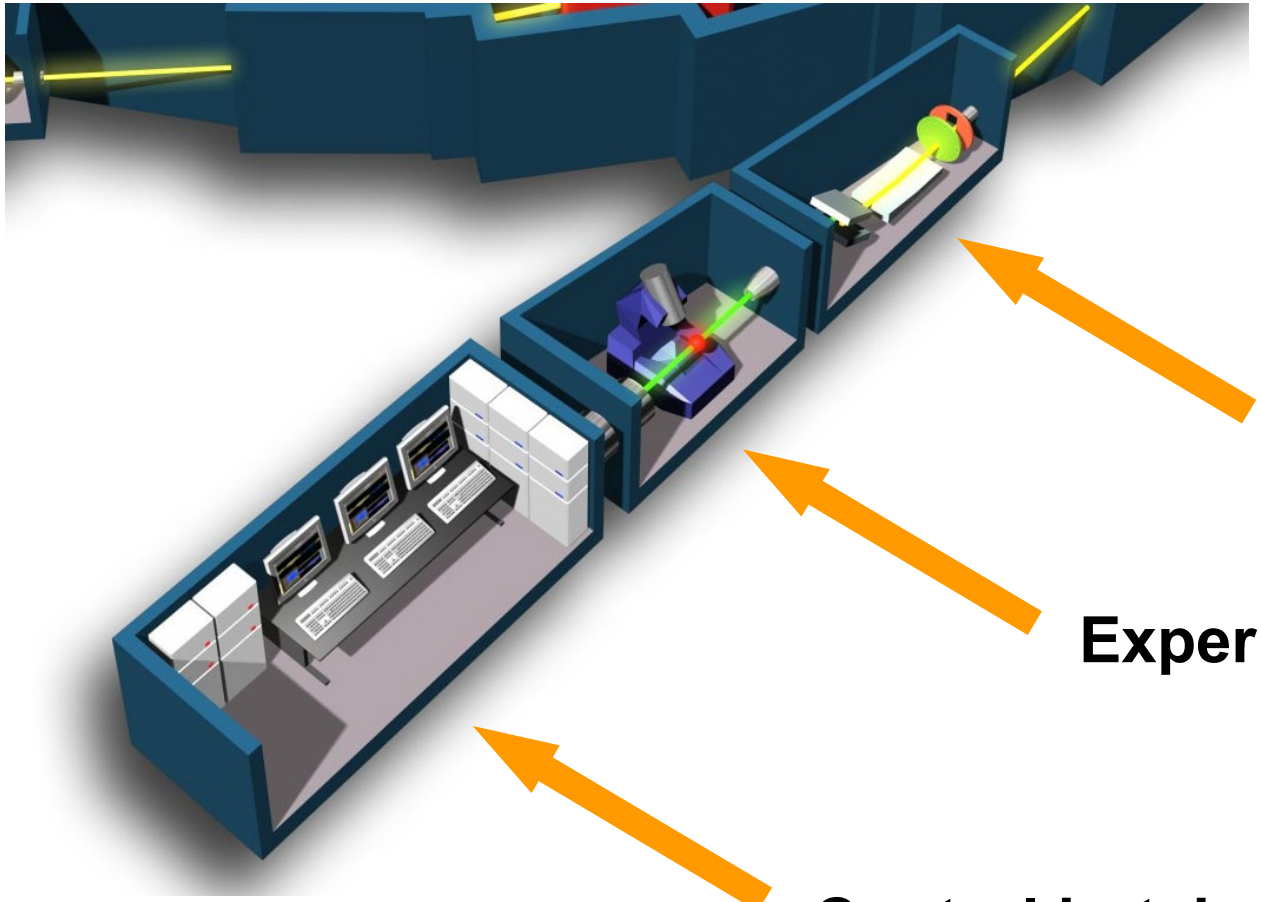
Storage ring and insertion devices



Copyright © EPSIM 3D/JF Santarelli

Experiment stations: the *beamline*

Copyright © EPSIM 3D/JF Santarelli

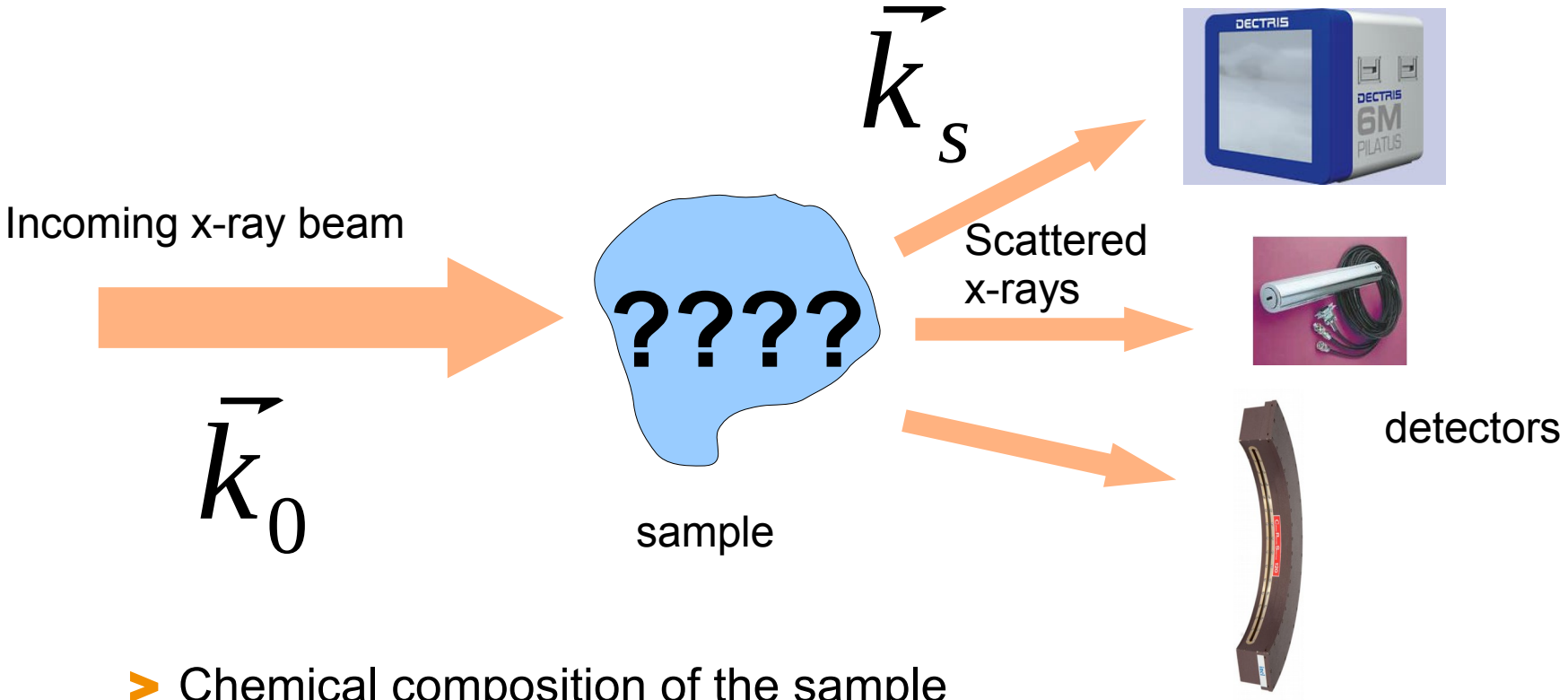


Optics hutch

Experimental hutch

Control hutch

What can we do with it?

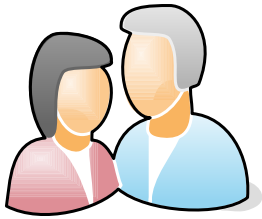


- Chemical composition of the sample
- Structural properties
- Electronic structure of the sample material
- Structure of proteins and viruses!

Who is using a synchrotron and how?

Scientists from external institutions

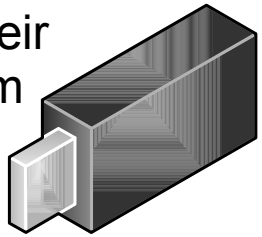
In-house researcher



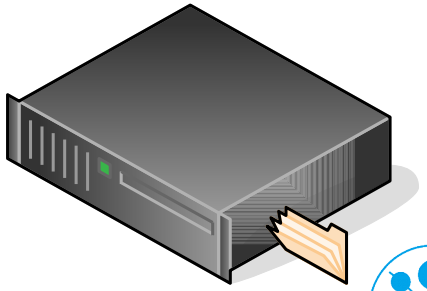
Beamtime: a period of time users are granted access to a particular beamline

Users collect data from their samples at a particular beamline. Data analysis is done either after the beamtime or even during the experiment.

Users take their data with them via mobile storage.



Data is stored on facility provided storage and made accessible via file servers or web interfaces.



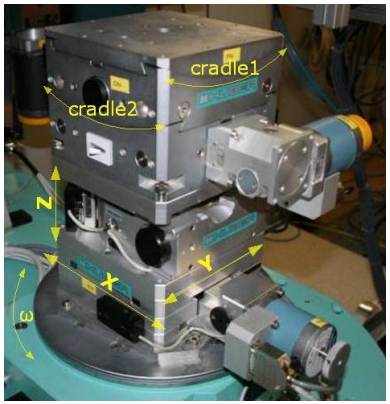
A synchrotron at a glance is ...

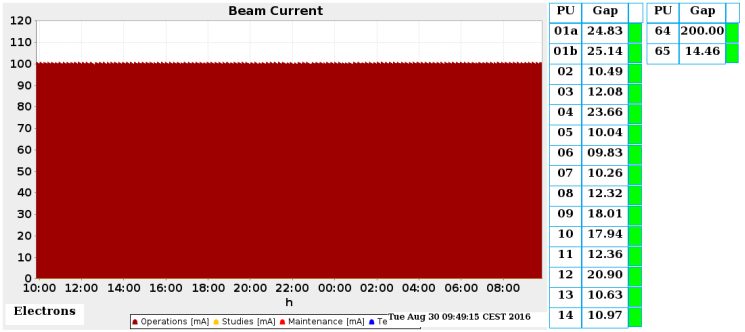
- A brilliant light source for x-rays
- Is used for all kinds of experiments in material science
- Can operate several experiment stations in parallel

Current status of file-formats and future demands

What data is currently stored?

- Detector data (quite obvious)
  - Recorded intensities
  - Detector position
- Sample parameters
  - Temperature
  - Position
  - Pressure
  - Voltages and currents
  - Stress
- Other beamline parameters
  - Ring current
  - Monitor readings





PU	Gap
01a	24.83
01b	25.14
02	10.49
03	12.08
04	23.66
05	10.04
06	09.83
07	10.26
08	12.32
09	18.01
10	17.94
11	12.36
12	20.90
13	10.63
14	10.97

PU	Gap
64	200.00
65	14.46

Scalar and 1D data – ASCII files

- Point and 1D detectors
- Sample parameters
- Motor positions
- …..

Typically stored in ASCII files as

> Tables

> Single column of data (1D detectors)

> Key-Value pairs

There are many ASCII formats around!

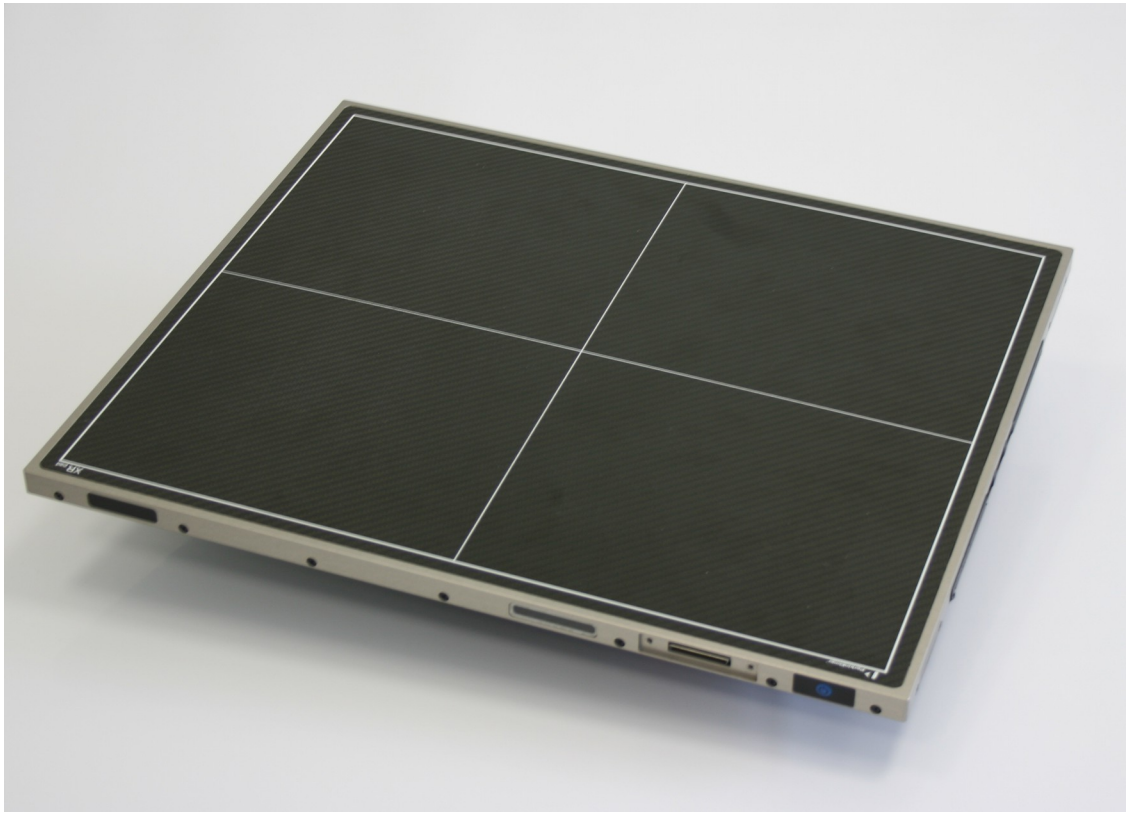
STATE	FIPS	LAT_DMS	LONG_DMS		WATERUSE		WELLDPTH		SAMPDATE		SAMPTIME
2s	5n	11s	11s	11s	11s	11s	11s	11s	11s		
88273001	AL	01023	315553	882730	U	1780	26JAN1981		1200		1
86133501	AL	01041	313657	861335			06SEP1989		1200	<	1
86133502	AL	01041	313657	861335			06SEP1989		1640	<	1
0	AL	01055	340630	861540			19SEP1995		800	<	1
87295901	AL	01057	332524	872959		207	24MAY1982		1600		1
87291301	AL	01057	333659	872913		54	07JUL1982		1030	<	1
87310301	AL	01057	333701	873103			28MAY1980		1200		1
87310202	AL	01057	333709	873102		18	24JUN1982		815		1
87284901	AL	01057	333717	872849		84	07JUL1982		1235		1
87302801	AL	01057	333724	873028			17JUN1982		815		1
87290301	AL	01057	333733	872903			13JUL1982		1230		1
87290302	AL	01057	333733	872903		30	13JUL1982		1300	<	1

DESY

Image data – binary formats, one image per file!

Binary image formats

- TIFF
- CBF
- MAR
- CIF
- SPE
- EDF



Important: every single image is stored in a separate file!

DESY

Challenges of today's experiments

- Large number of images (~200 000) → high data volumes
- Increasing importance of meta-data
- Easily access data stored in a file from many applications
- Current HR-policies of facilities make it difficult to hire staff on permanent positions

Can we do this with current technology?

No, we cannot!

- One image per file is bad
- One measurement → many files → we have to keep things together
- Current formats do not provide an easy means of integrating meta-data
- Maintain IO code for each of the formats is expensive

DESY

Requirements for a new file format for our experiments

- Reduce the number of files
- Provide an easy means of adding meta-data
- Large amounts of data should be written and read to and from the file at high throughput
- Keep maintenance efforts low → use existing technology if possible
- The technology used should be well established and provided with reasonable funding to ensure long-term maintenance
- Easy access to the data

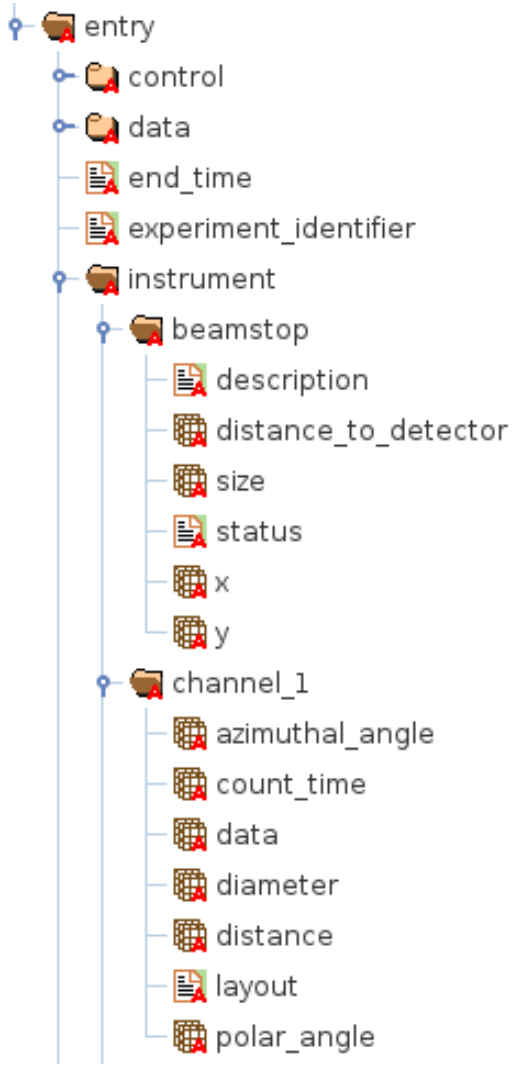
DESY

HDF5

What is HDF5?

- Binary file format
- Stores data in trees of groups and datasets
- Datasets are multidimensional arrays of arbitrary type
- Compress individual datasets
- Meta-data to groups and datasets via attributes
- Objects can be accessed via their path

```
/entry/instrument/channel_1/count_time
```



How to access data stored in HDF5 files?

- via a C-library with bindings for
  - C++
  - Python
  - Java
  - Fortran
  - C#
  - Perl
  - Erlang
  - R

```c
#include <hdf5>

int main(int,char**)
{
    double *data;
    hid_t file_id = H5Fopen("detector_data.h5",H5F_ACC_RDONLY,H5P_DEFAULT);
    hid_t dset    = H5Dopen(file_id,"scan_1/detector/data",H5P_DEFAULT);
    hid_t dspace  = H5Dget_space(dset);

    hssize_t n = H5Sget_simple_extent_npoints(dspace);
    hid_t dtype = H5Tcopy(H5T_NATIVE_DOUBLE);
    data = (double*)malloc(n*sizeof(double));
    H5Dread(dset,dtype,H5S_ALL,H5S_ALL,(void*)data);

    return 0;
}
```

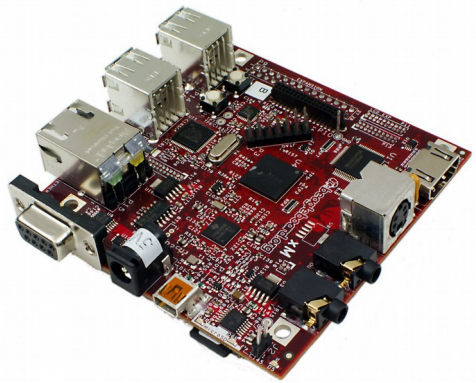
- Commercial products
  - Matlab
  - IDL
  - Mathematica
  - Igor Pro
  - LabView

```python
import h5py
import numpy
from matplotlib import pyplot

f = h5py.File("detector_data.h5")
images = f["/scan_1/detector/data"][...]

pyplot.imshow(images[0,:,:])
pyplot.show()
```


DESY

Available for many platforms and architectures

- **Linux**
  - X86_64
  - ArmHF
  - Power64



- **Windows**

- **OSX**

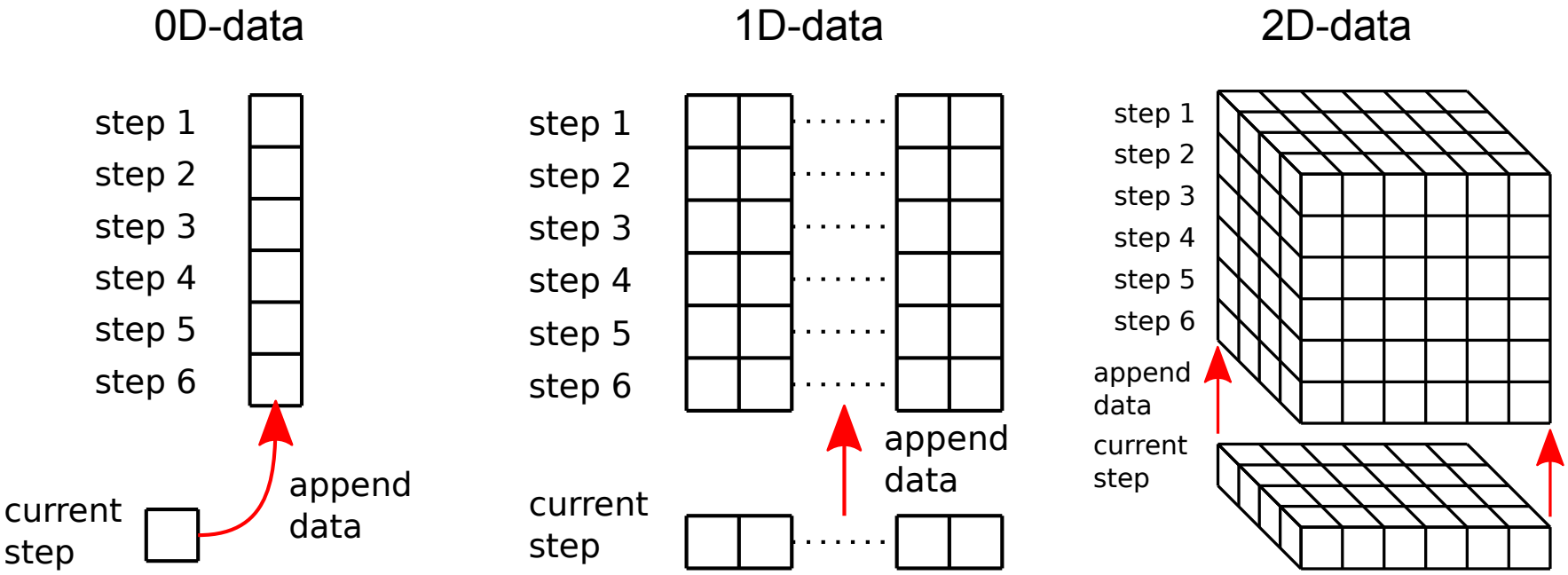
- **AIX**

- **Solaris**

DESY

Storing data in HDF5 files

Datasets in HDF5 can grow!



> 0D-, 1D-, and 2D-data can easily be stored in the same file

> Could use groups to organize the data

> Meta-data can easily be added as scalar dataset or attached to an object as an attribute

Is HDF5 the solution for all our problems?

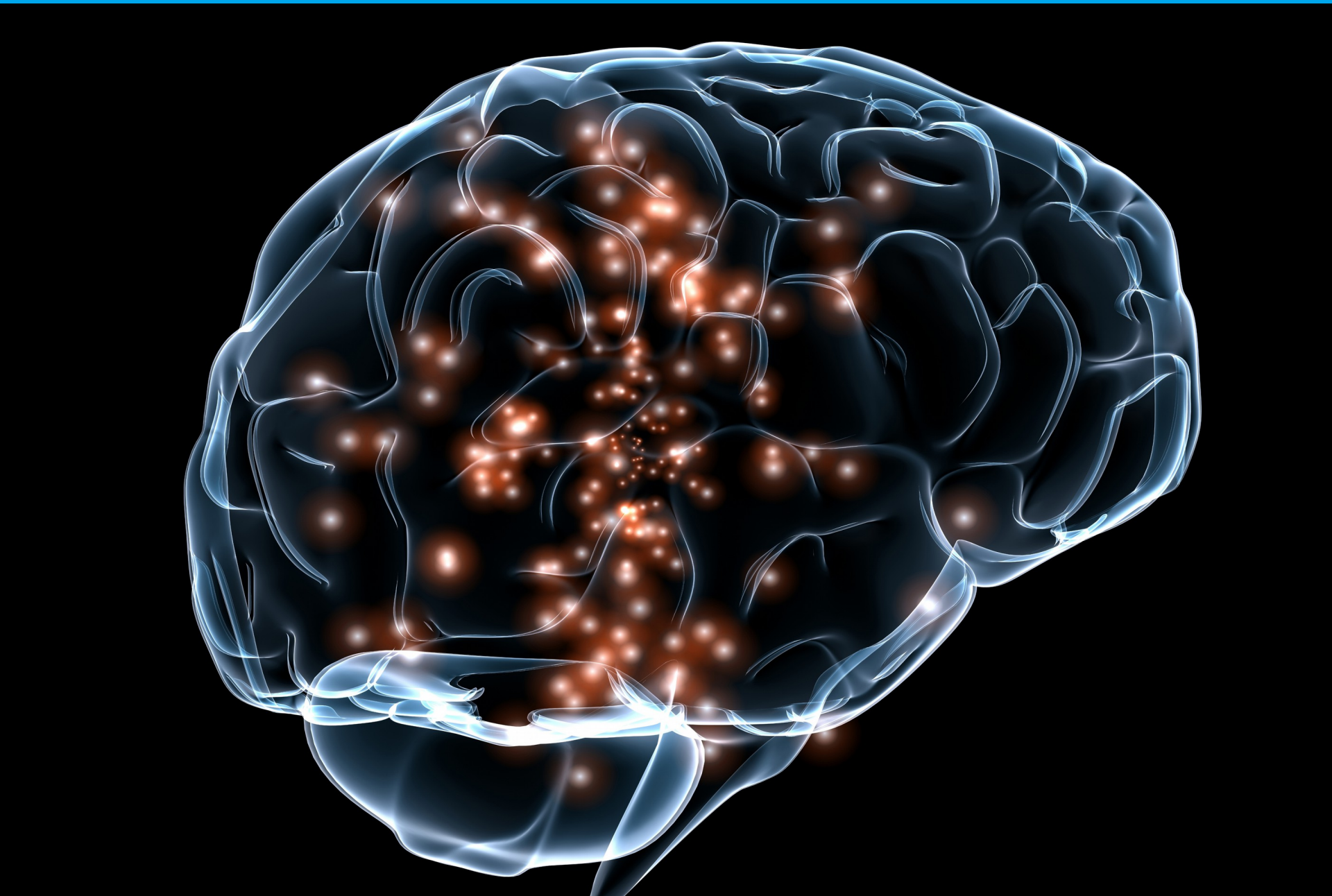
All data in a single file?	OK
High performance?	OK
Maintenance and funding?	OK (yes, we could do better)
Meta-data	Easy to add, but ...

Problems

- HDF5 knows nothing about synchrotrons and beamlines.
- Can access objects only by there name
- Would have to standardize virtually all names in the file

DESY

NeXus – adding semantics to HDF5



A little history ...

In the mid 1990s the Neutron scattering community faced a similar problem.

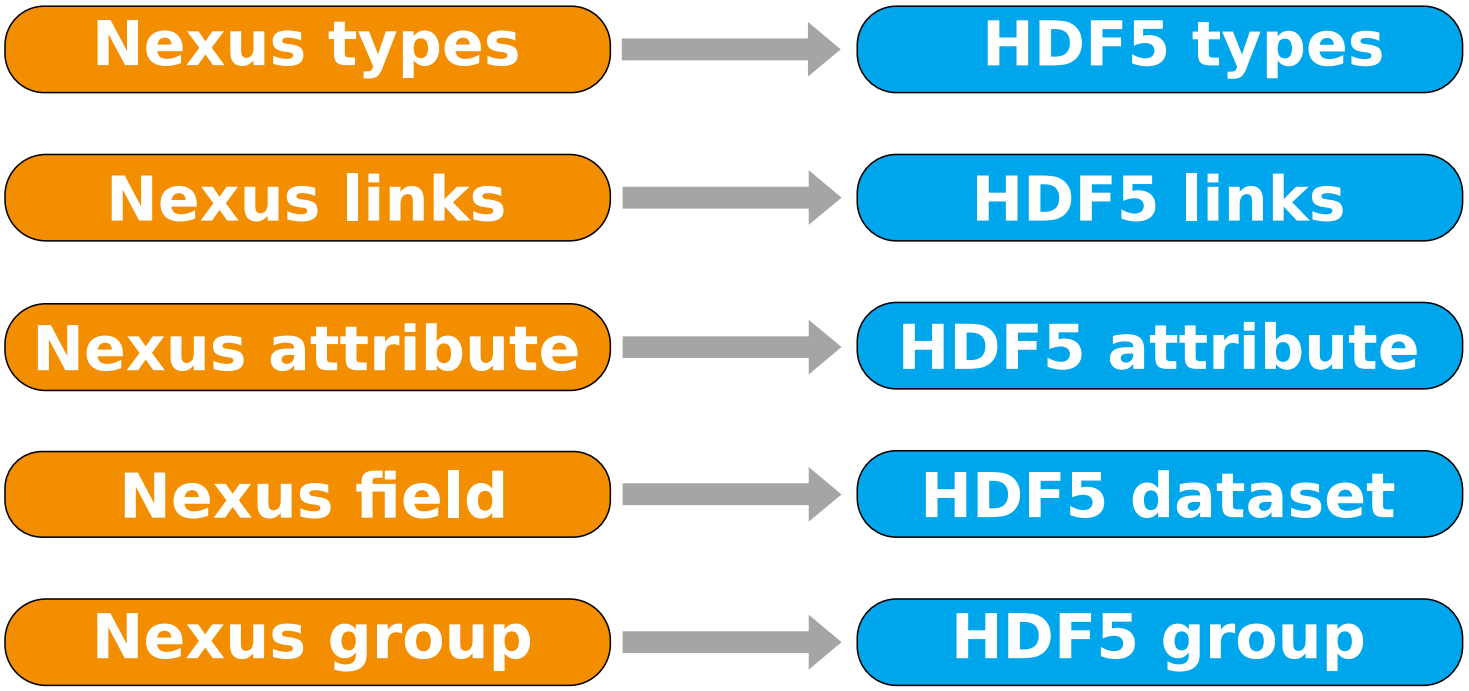
- Easily share data between users and facilities
- Develop a common set of analysis tools
- Store meta-data about the experiment

They invented the NeXus file format

- Originally based on XML, latter HDF4
- Data is organized in a hierarchal tree structure
- Based on a small set of objects: *groups*, *fields*, *attributes* and *links*
- Access to the data was provided via a simple C-API with bindings to C++, Fortran, Python, etc.

DESY

Today: NeXus uses HDF5 as its physical file format



From its basic building blocks NeXus and HDF5 are virtually eaqual, thus it is easy to use HDF5 as the physical file format for NeXus.

DESY

NeXus: a layered view

Layer 3: application definitions

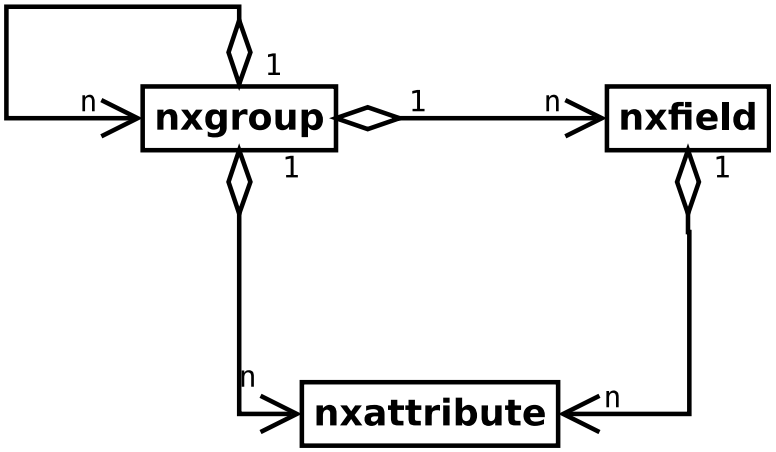
Describing standard data for a particular experimental technique.

Layer 2: base classes

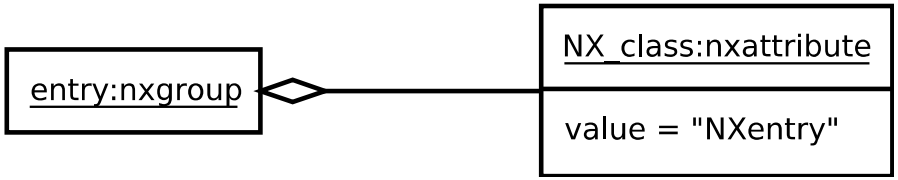
Describing standard objects for objects at a beamline

Layer 1: nxgroup, nxfield, nxattribute, nxlink, types

The basic building blocks of a NeXus tree.

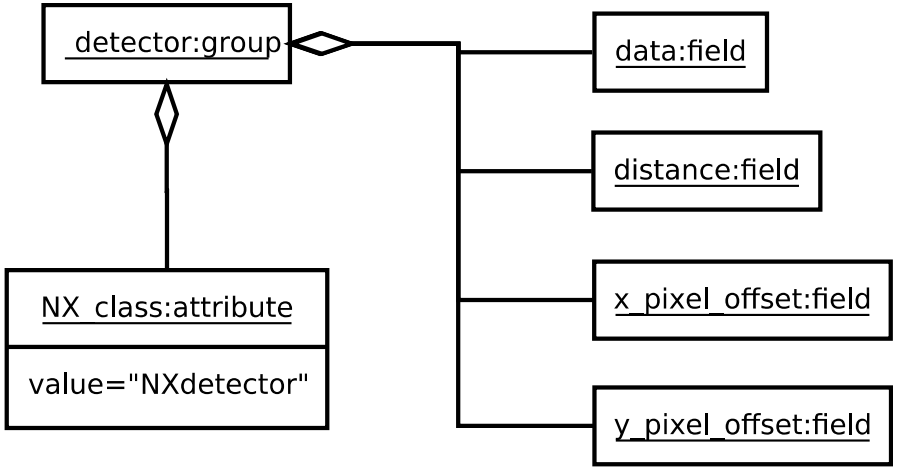


Layer 2: Types and base classes



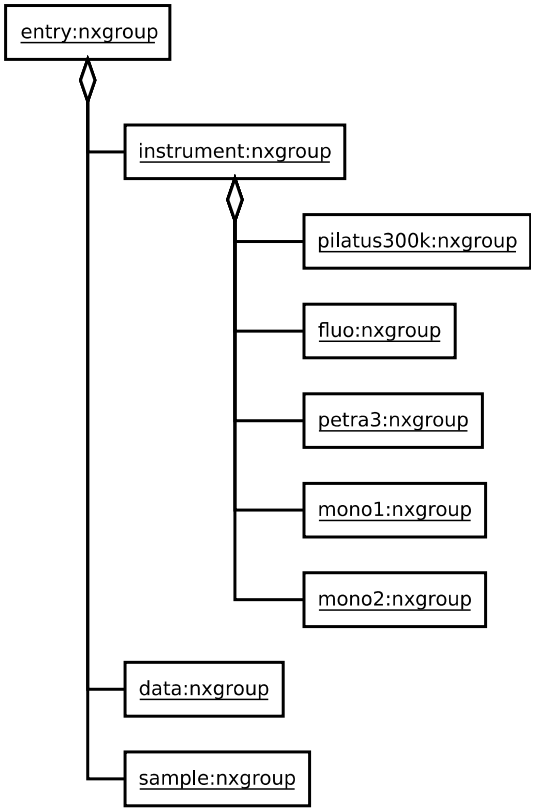
Every group has an attribute *NX_class* which determines its type (base class).

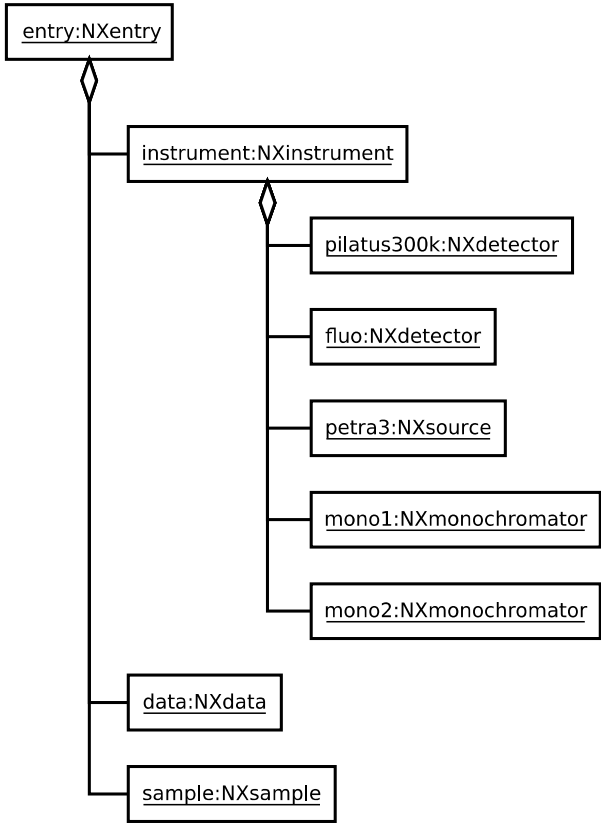
The type (base class) determines which fields on can expect to be present within a group.



Types and their related fields can be found in the official NeXus documentation!

The NeXus default tree – why do we need types





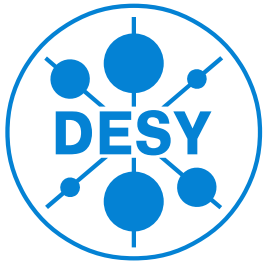
1) Types tell us to what kind of object the data in a group belongs

2) We can search for objects of a particular type without having to know their precise names.

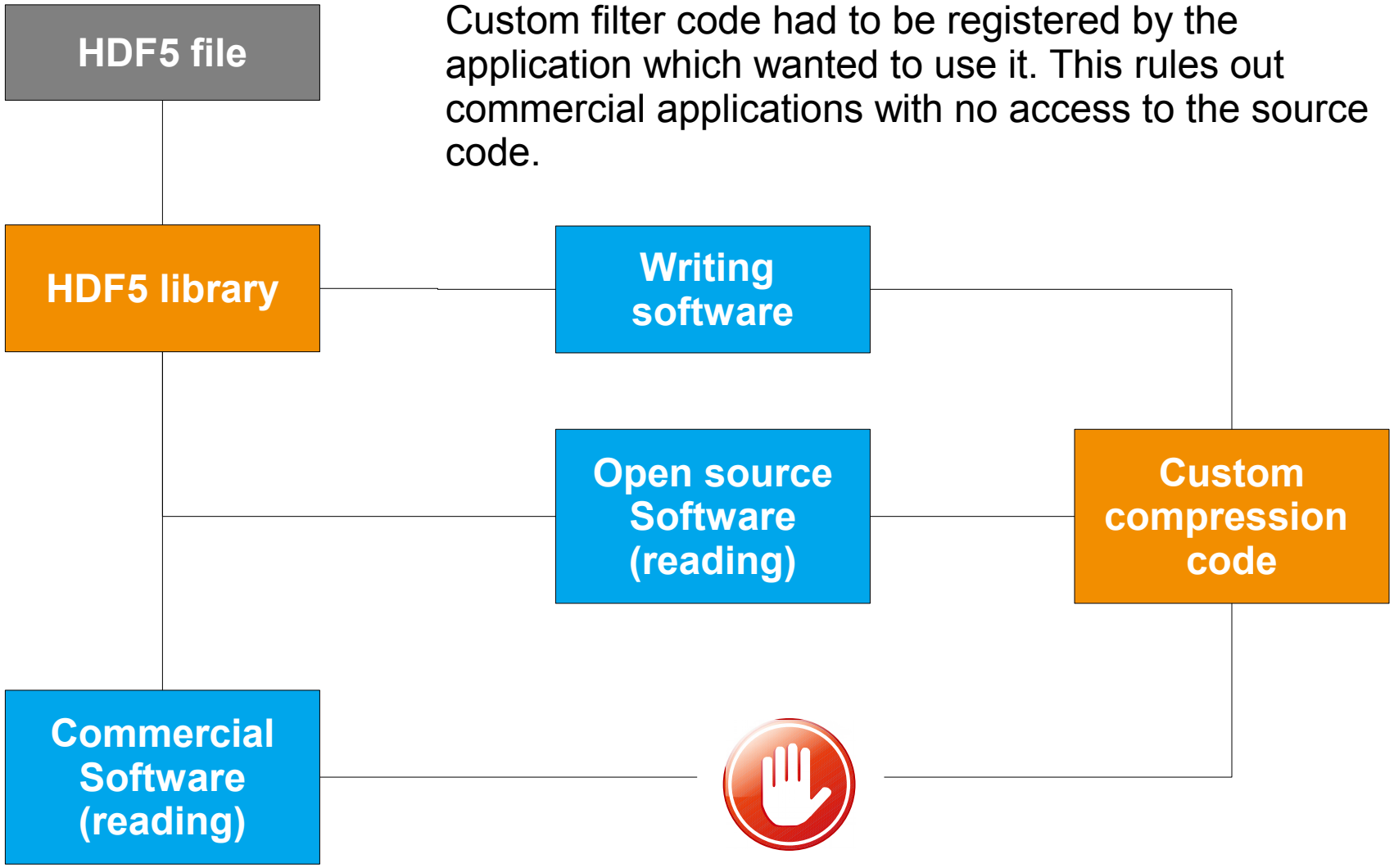
Extending HDF5

HDF5 features developed with the synchrotron community

- Filter chain bypass
- External filter interface
- Single Writer / Multiple Reader → SWMR
- Virtual Datasetse (VDS)

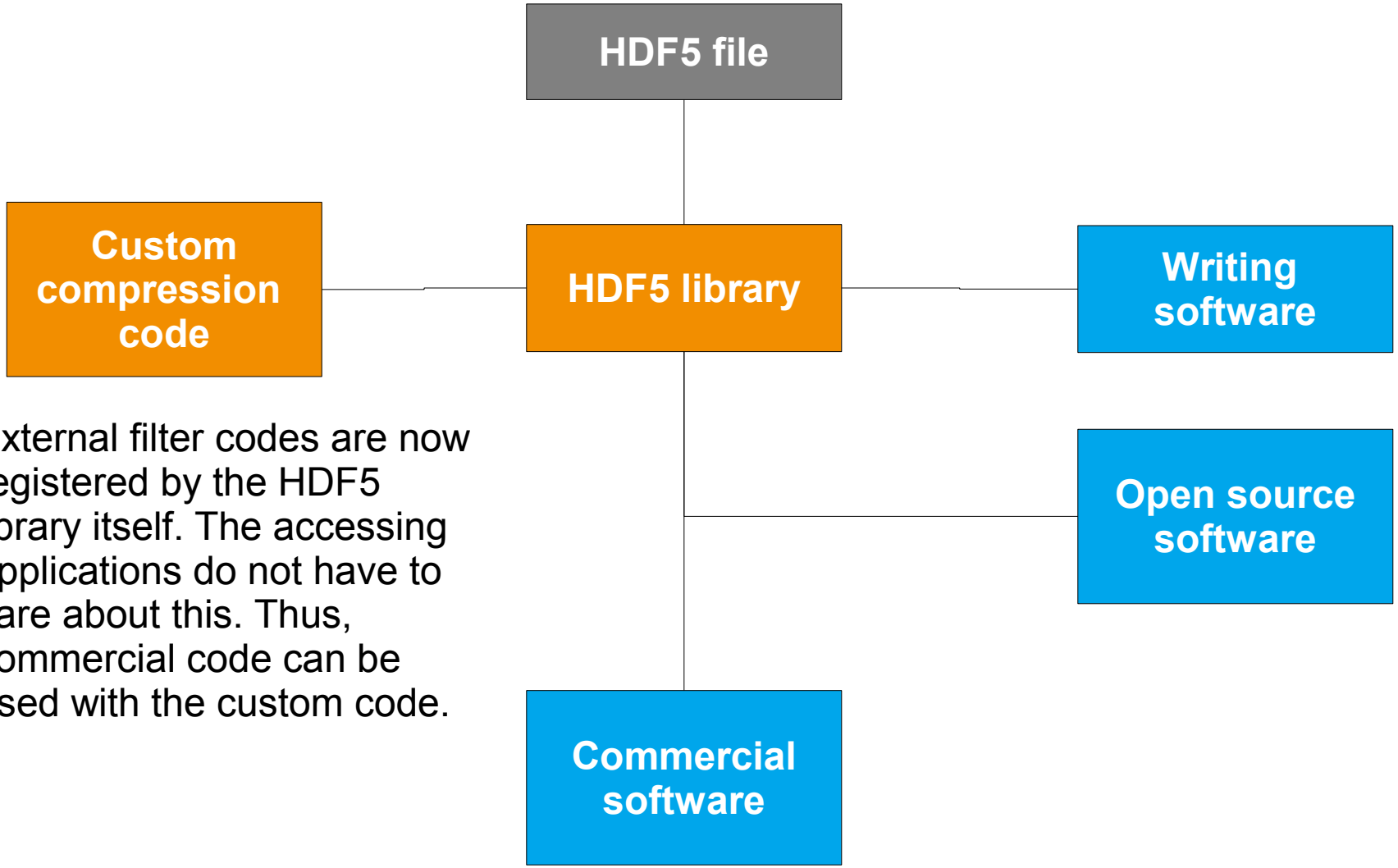


External filters – the original problem



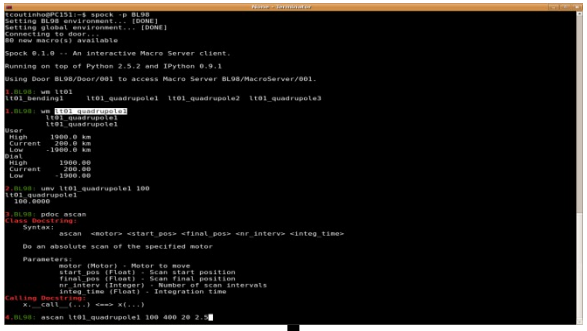
Custom filter code had to be registered by the application which wanted to use it. This rules out commercial applications with no access to the source code.

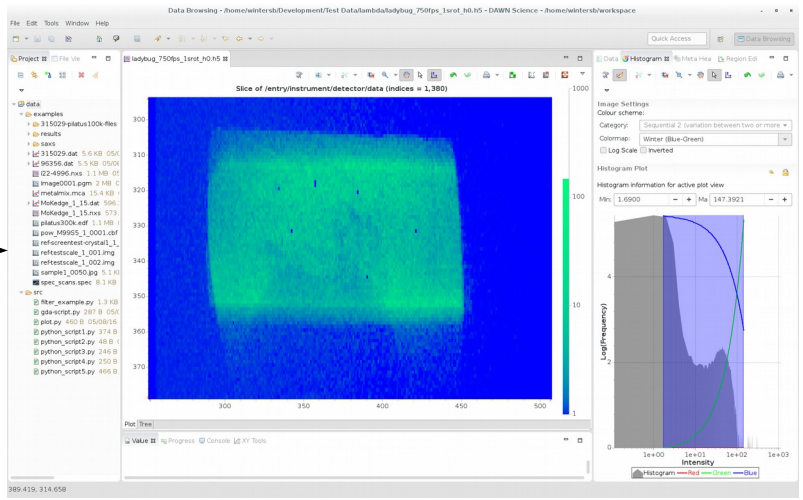
The new external filter interface

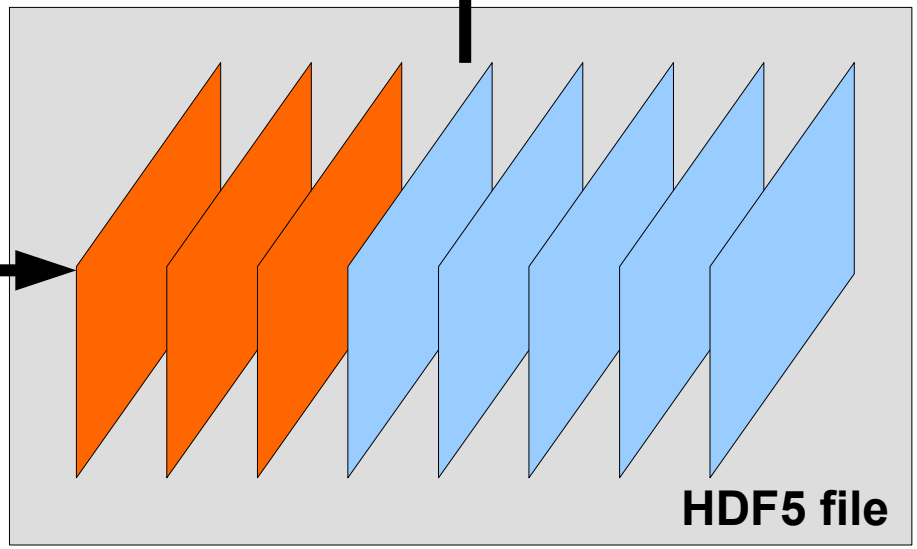


External filter codes are now registered by the HDF5 library itself. The accessing applications do not have to care about this. Thus, commercial code can be used with the custom code.

Single Writer / Multiple Reader - SWMR





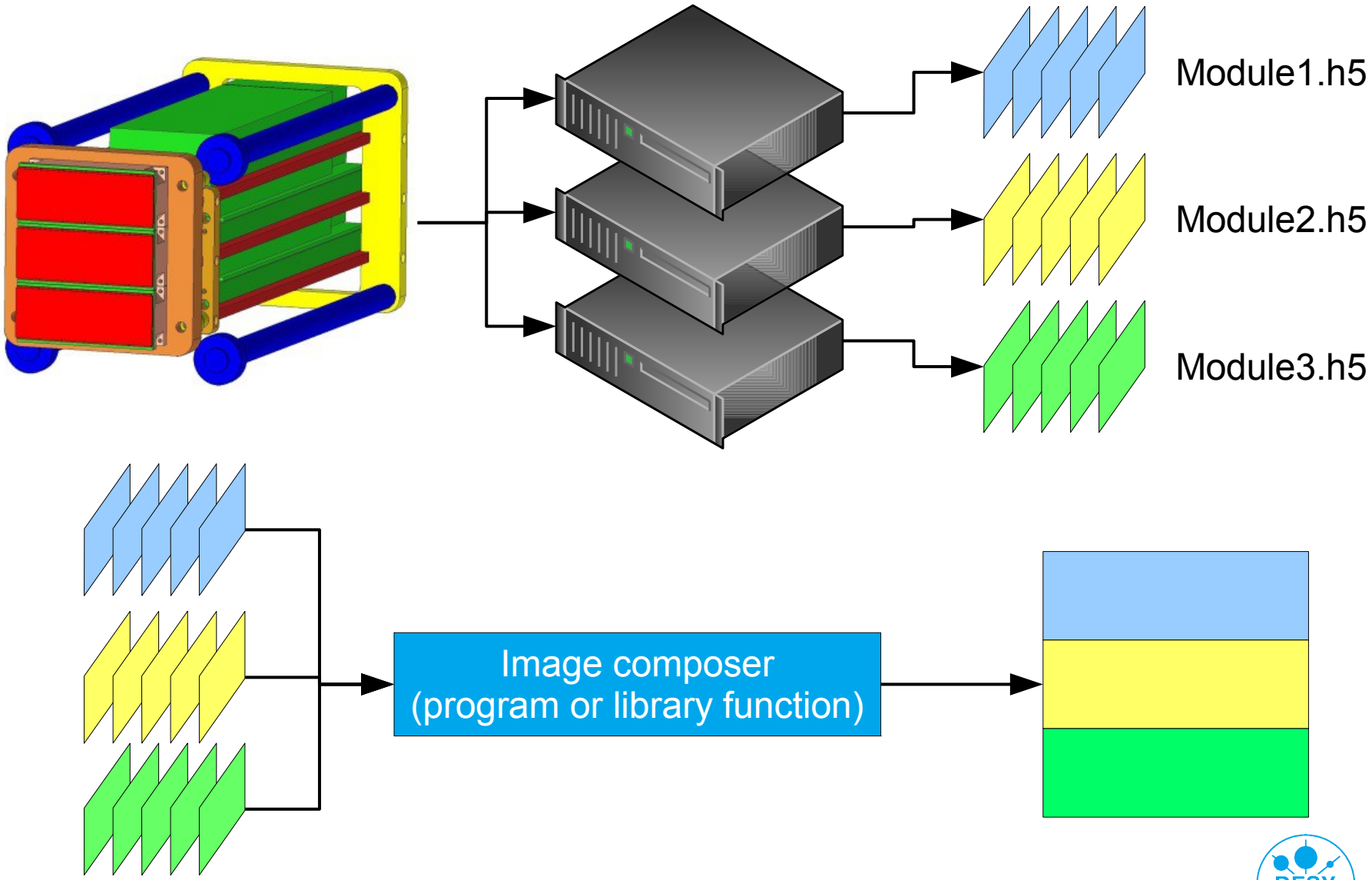


The viewer would never see the newly written images!

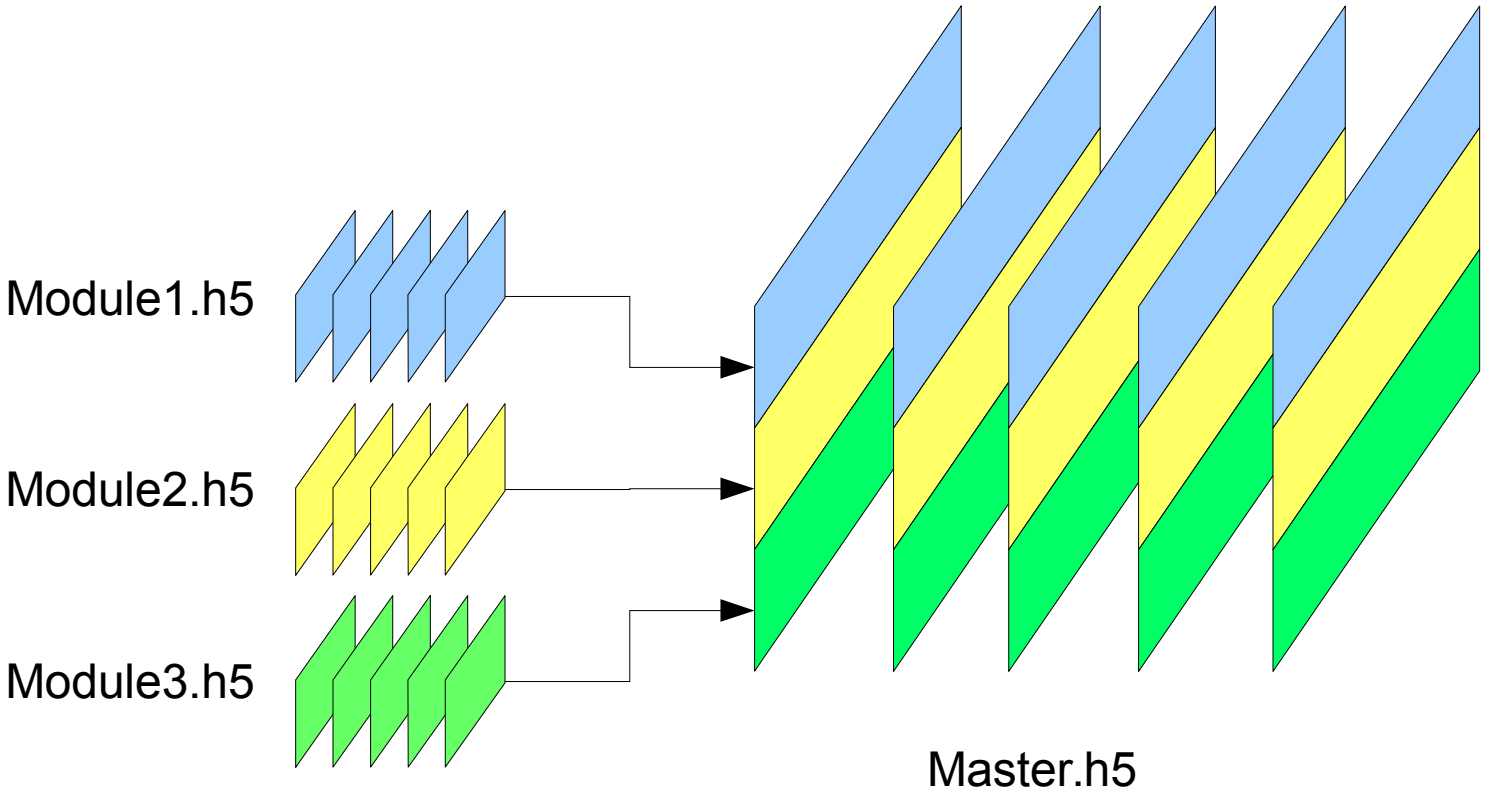
SWMR mode allows regular updates on existing objects in the file! (since 1.10)

DESY

Virtual Datasets – VDS: Use-Case



Virtual Datasets – VDS: master file



Data remains in the individual module files but can be accessed via a single dataset in the master file!

Conclusion so far

- **HDF5** satisfies the basic demands for actual synchrotron radiation experiments
- **NeXus** is a set of rules how to organize the objects within an HDF5 file!
- Testing the format currently at DESY – works quite well.

Problems

- Some technical problems in the HDF5 library had to be solved!
- Convincing software developers to support the new file format
- Convince users and beamline scientists to use the new format

DESY