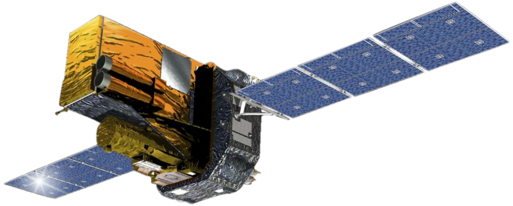
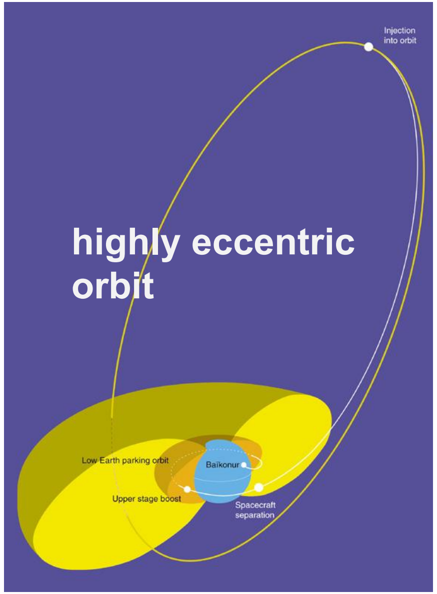
INTEGRAL Data Analysis

Volodymyr Savchenko



Distributed Computing in Astrophysics, Paris, APC, 2015

INTEGRAL: 2002-2029



2002 - Launched

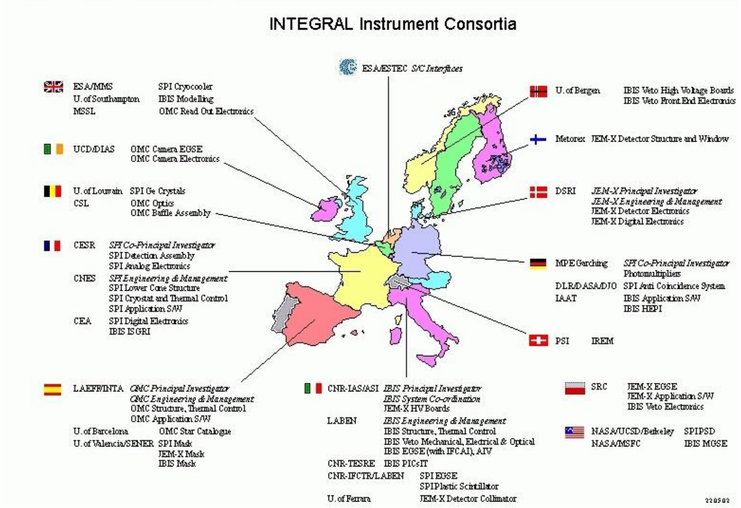
2006 - minimal scientific goals reached

2014 - maneuver to prepare for return

2015 - so far instruments suffered only small degradation, all operates well.

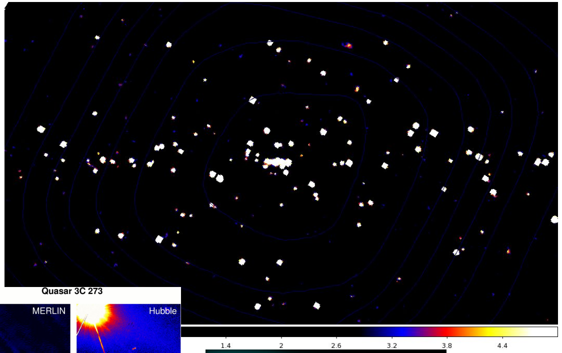
2025 - science return becomes difficult

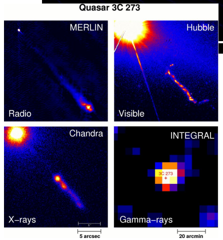
2029 - returns to Earth

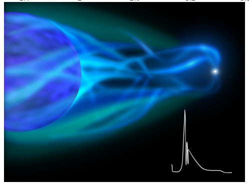


INTernational Gamma-Ray Laboratory

X-rays (3 keV - 500 keV) let us peek to the source of the most energetic phenomena in the universe





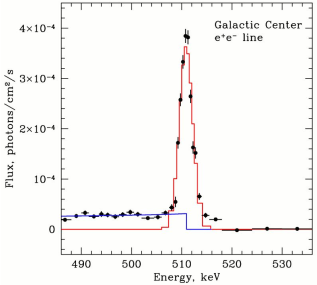


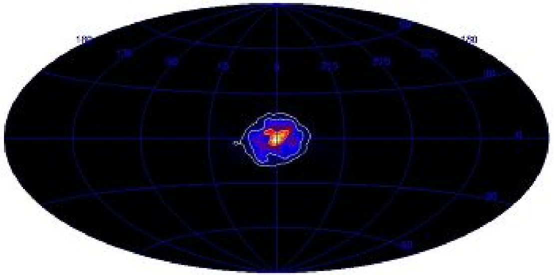
Both galactic and stellar scale **black holes and neutron stars**

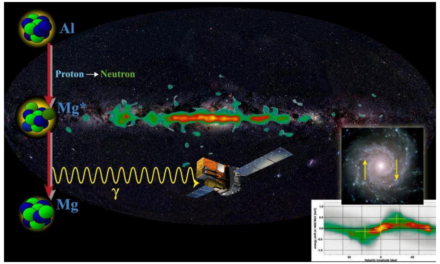
Sources persistent and **extremely variable**

Nuclear imaging of the galaxy (50 keV - 5 MeV)

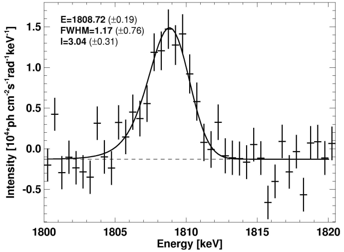
Positron annihilation in the galactic center







Al 26

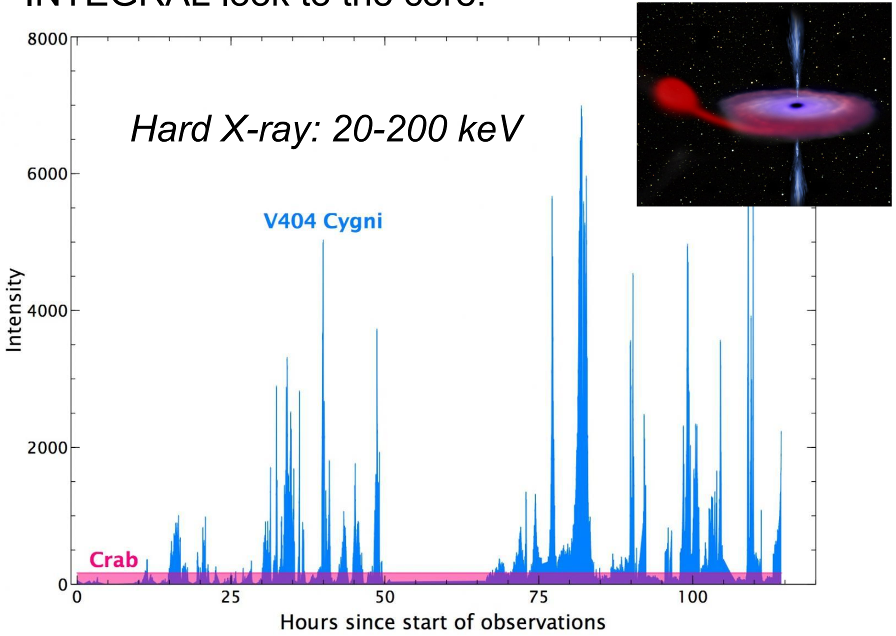


INTEGRAL continues to provide results

“Monster back hole wakes up“

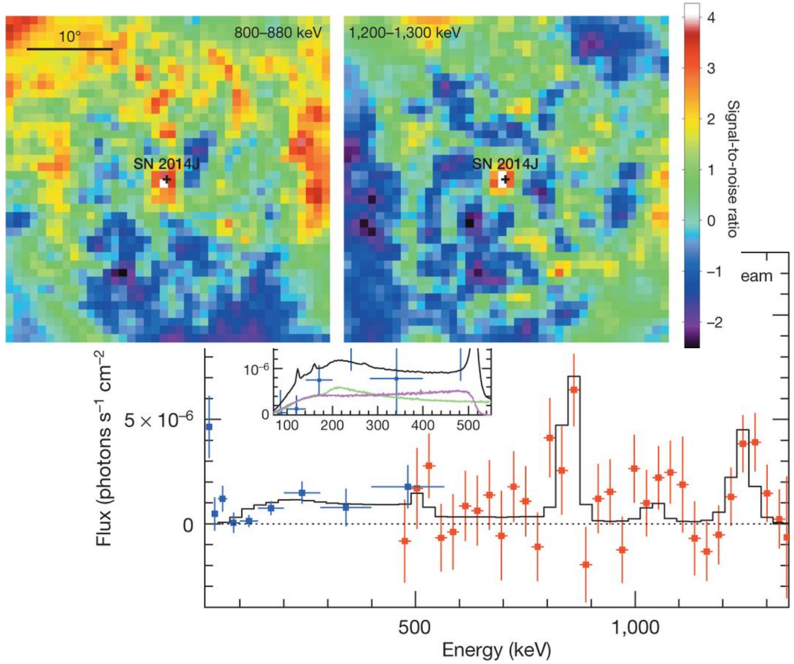
(ESA Press release, 2015)

INTEGRAL look to the core!



Gamma-ray lines from supernovae

INTEGRAL is the only one capable to look!

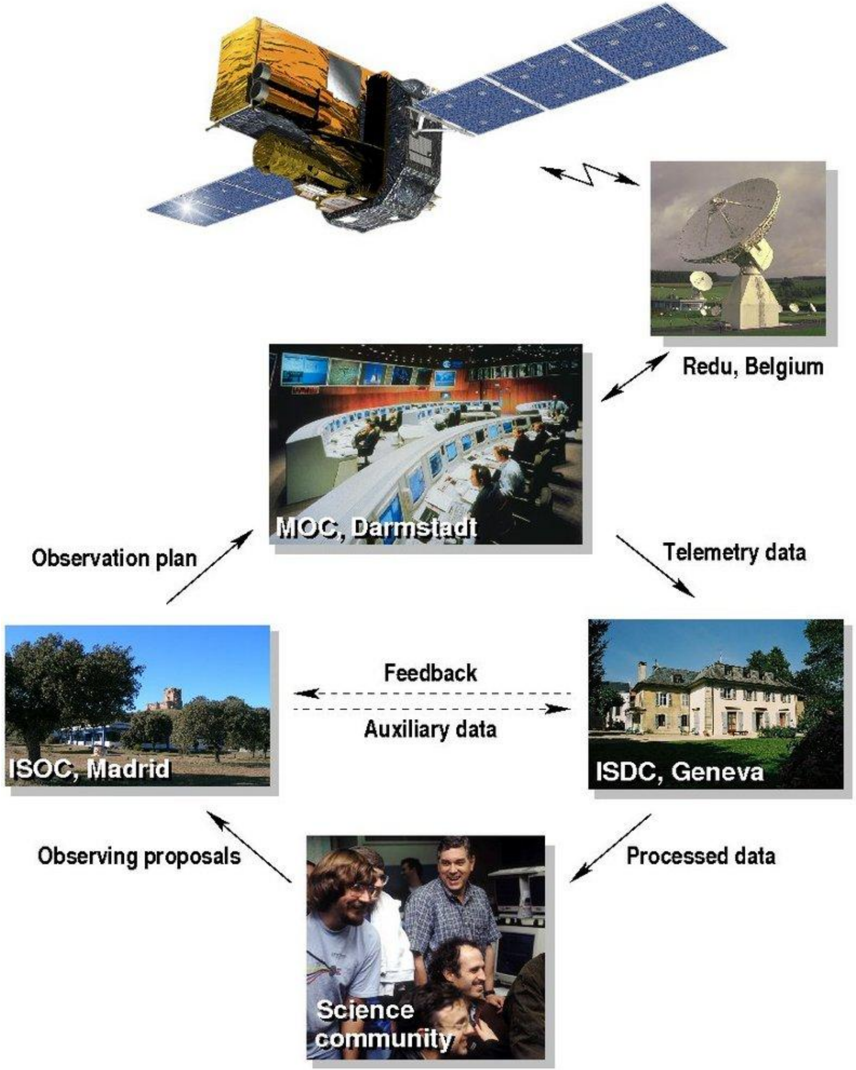


Gamma-ray sky is variable, something new happens all the time!

INTEGRAL is an observatory

Observation Data is provided **freely** to the community

Software and calibration is distributed and has to meet quality requirements: portability, reliability



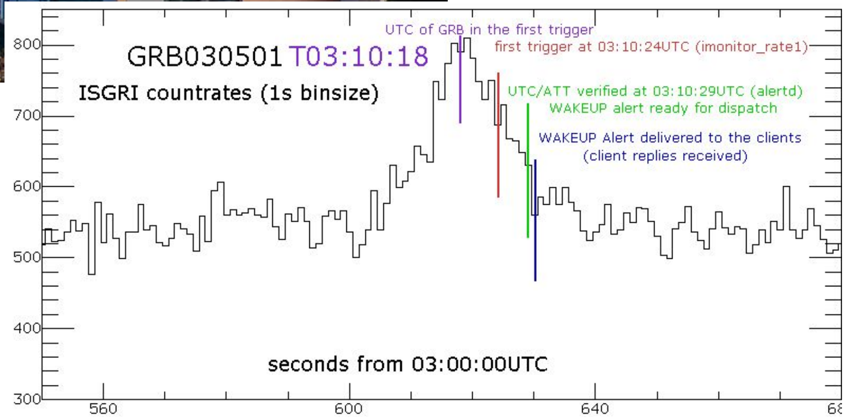
INTEGRAL operations

Quick Look Analysis and Burst Alert System

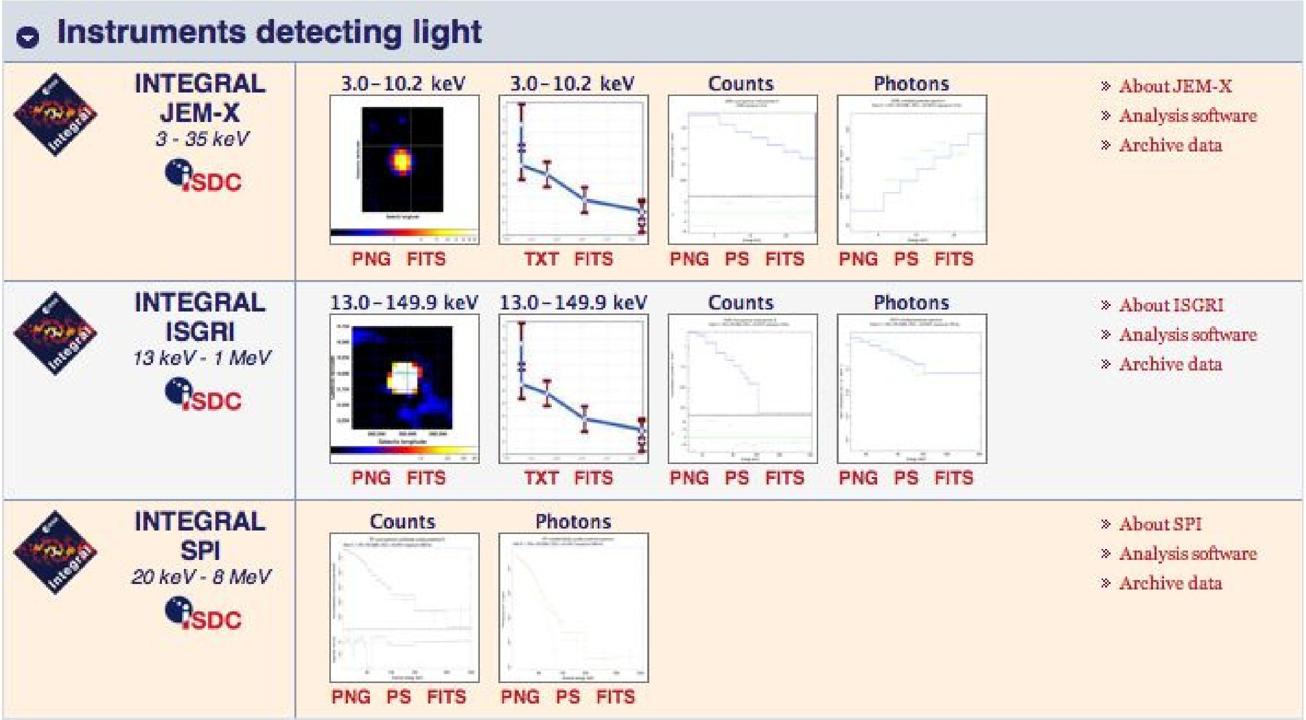
Telemetry is searched in **real time for bursts,** on-ground, the alerts are distributed.

In **near-real time analysis pipeline** is run (single CPU) to extract standard results

Tools for **parallel analysis** are used for rapid tuned **extensive investigation**.



VO and HEAVENS

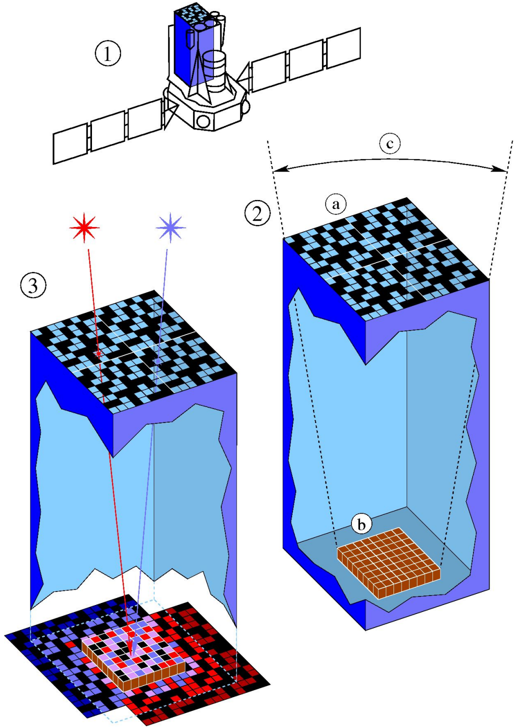


Processed results for some standard analysis are provided: **http://www.isdc.unige.ch/heavens/**

INTEGRAL has many masks

resolving sources in hard X-ray - gamma-ray band is quite complicated

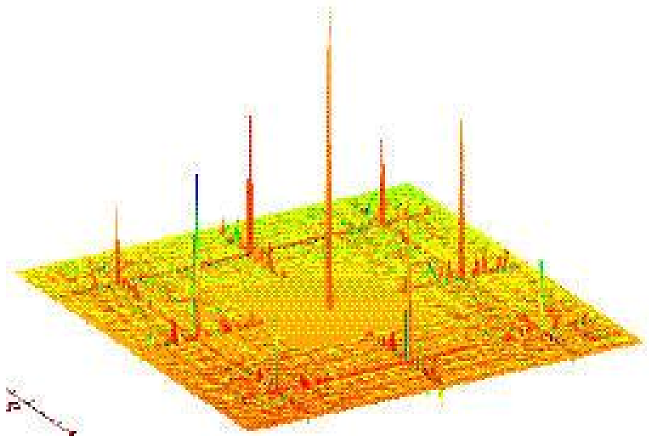
IBIS: finer elements, thinner mask

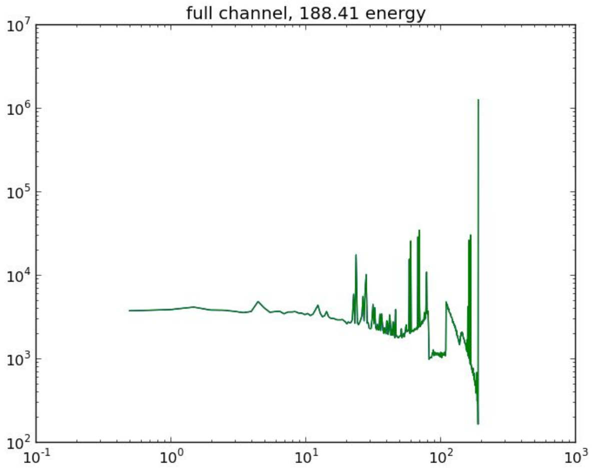


SPI: thicker and coarser mask

How is this analysed?..

Reconstruction **depends strongly on the source model and the response** (image and spectrum)





Point source image (somewhat transformed shadow) - **PSF**

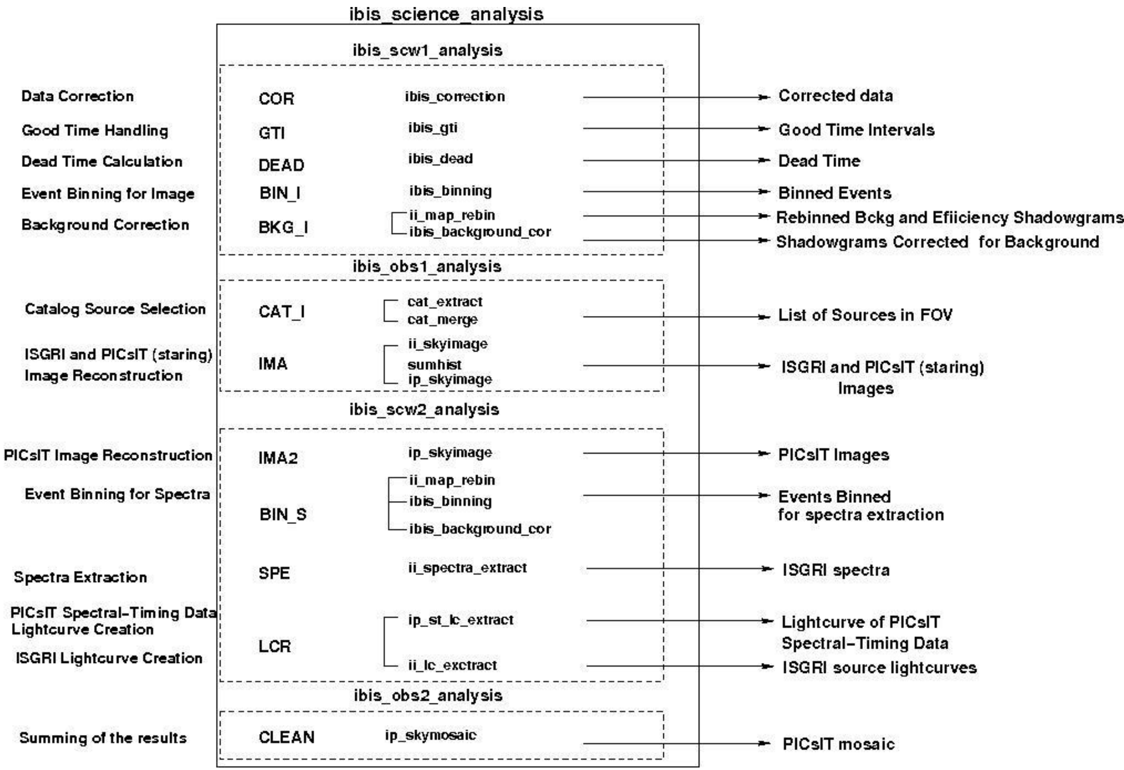
Monochromatic source spectrum - **RMF**

Sources are all mixed up in 30 deg field of view: global fit is used and an iterative procedure.

Response response is non-local, and evolving, and not fully understood it has to be **verified as part of the analysis**.

Official Pipeline - OSA: linear and single-thread

- Correct the data with predefined instrument characteristics
- Iteratively search for sources and refine the sky model
- Fit the complete sky model to the data

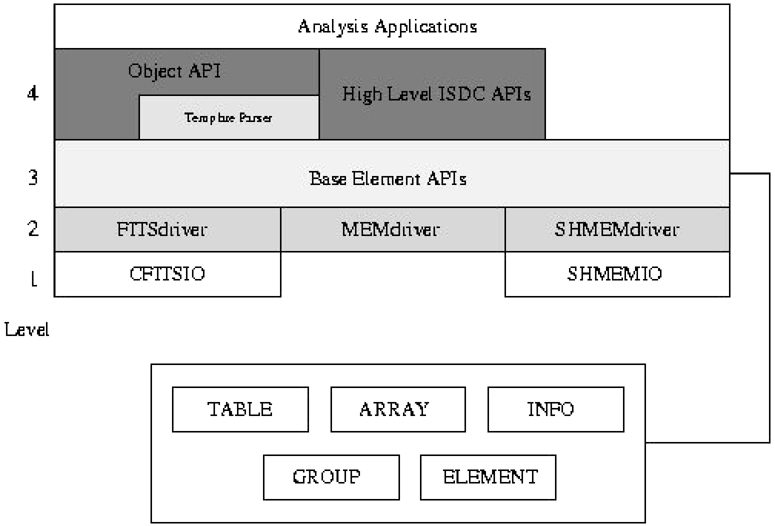


DAL: Data Access Layer

Jennings 1998

Isolates the analysis software from the specifics of the data formats while at the same time providing new **data abstraction** and access capabilities

Supports the creation and manipulation of **hierarchical data sets** which may span multiple files and, in theory, multiple computer systems



designed as common-interest framework, so why is it not widely known?

OSA and DAL

good:

data structures allow **data abstraction** independent of the storage

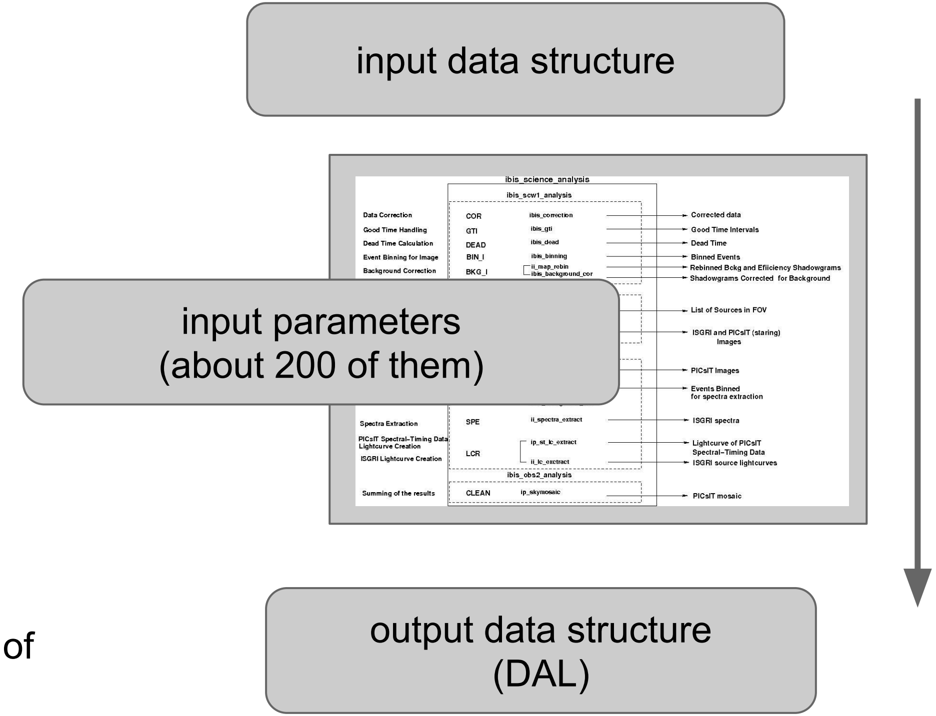
strictly controlled **interfaces** reduce uncertainties of data handling standards in large collaboration

bad:

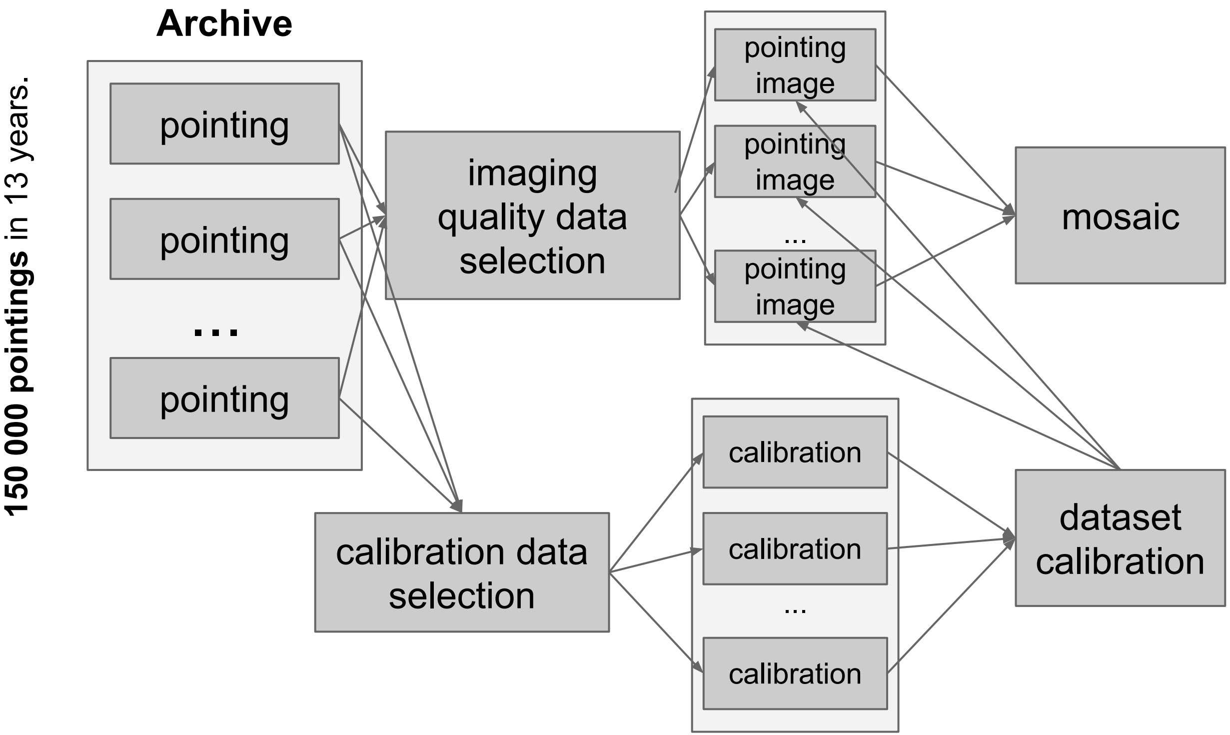
data structure make **simultaneous writing impossible**

consistency checks involve opening of a large structure, making **reading or writing very slow**

it is difficult to track down individual contributions of pipeline elements: somethings goes wrong, or any parameters slightly changed - **repeat all from scratch**



Data and analysis structure



Each pointing hundreds of different final products must be stored: images and spectra in different reconstruction settings.
Thousands of diverse intermediate results may be additionally stored to speed-up the analysis.

To run this kind of analysis?

- User approach - always run complete pipeline - compromise computer time and storage over human time

  not manageable in large scale analysis

- Determine reusable results and manipulate data structures:

  costs human time every time - not generic

- modify the pipeline so that it will be able to run in parallel, and reuse the results naturally from a variety of sources

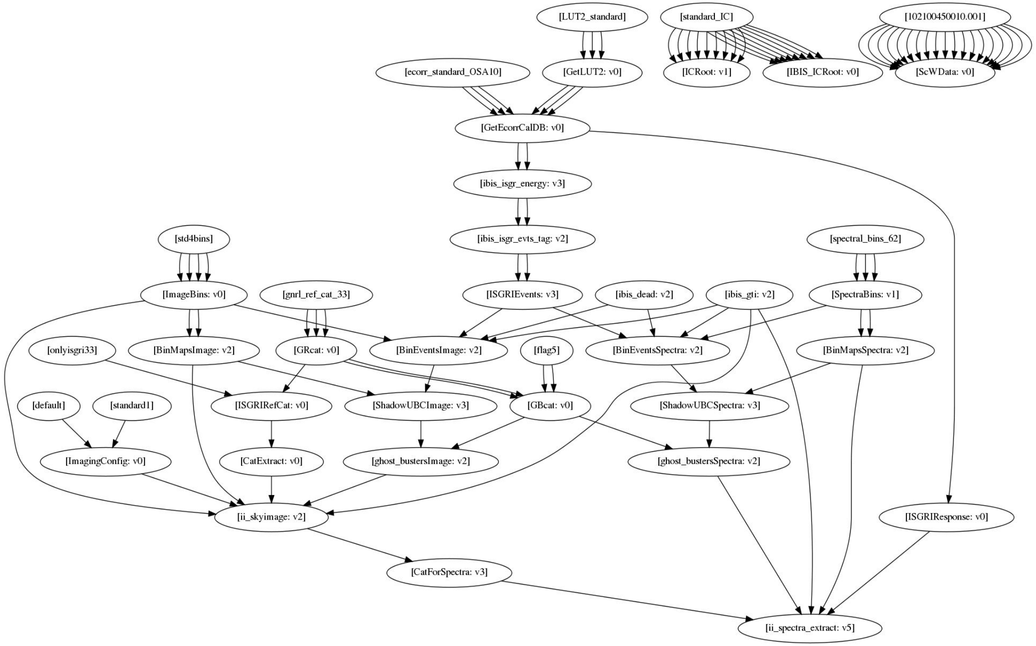
  …*why not?*

INTEGRAL/ISGRI Calibration pipeline

Try to **avoid shared writable data** structure like DAL structures

Instead of doing some manipulations to a given structure, take competed pipeline elements as input and result in some data.

Elements resemble pure functions - leave no side effects and hence can be **run in parallel**



Data as cached analysis

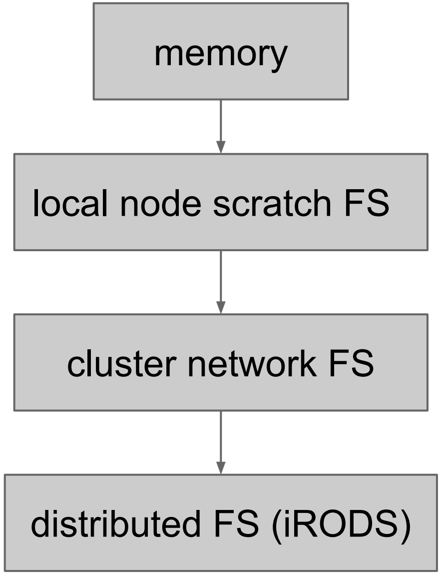
Data is identified by the graph of pipeline elements that lead to it

Structure of the graph naturally translates into filesystem, local or distributed

Different results are stored at different levels, possibly replicated

Cost of transport to the different level caches is taken into account in selecting it.

Hierarchical data cache structure

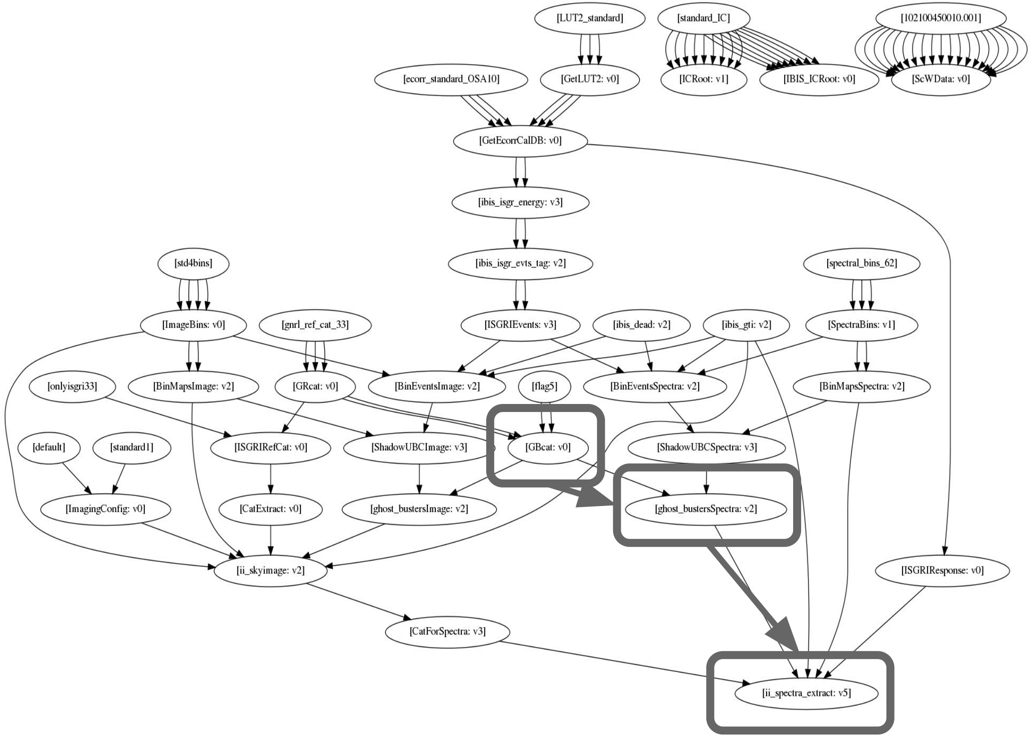


Reusable results and rapid development

Change in the analysis parameters or (equivalently) in the implementation of a pipeline element changes the signature of the **results that depend on it, but only these**!

This also allows to **easily develop the pipeline**, changing elements and introducing new connections, while never re-processing unaffected results.

The results are acquired and deposited in a **variety of local and distributed data storages** transparently



How does it look like

Analysis elements are implemented by python classes: allowing the use the **OO machinery** to construct them.

Dynamic features of python are used to make the implementation of new elements natural

Code is stored in the same caching manner in the top distributed storage - iRODS, with replication to the cluster FS

```
class BAnalysis(da.DataAnalysis):$
    version="newversion"$
    def main(self):$
        print "test"$
        self.data="data"$
$
class Analysis(da.DataAnalysis):$
    input_b=BAnalysis$
    cached=True$
    cache=MyCache$

    def main(self):$
        print "test"$
        self.data=self.input_b.data$

A=Analysis()$
A.get()$
    $
self.assertEqual(A.data, 'data')$
```

How does it look like

We build up a database of preprocessed results at different levels:

10 Tb at CC-Lyon (1M CPUh/year)

30 Tb in the FACe (1M CPUh/year)

20 Gb of the most costly results in iRODS

Transient storage in **cloud** (stratuslab, OpenStack) depositing to iRODs

DIRAC/frangrille for GEANT simualtions

Approaching **a fraction of a billion** of diverse individual data results

The simplicity of use and high performance allowed us:

- make a major revision of instrument energy calibration
- design a variety of new checks to re-evaluate instrument performance through the mission and monitor it in near real time

Conclusions

INTEGRAL accumulates data and demands new ways to analyse and calibrate it.

Reality of modern systems require different approach to larger analysis than was anticipated 20 years ago.

Hopefully this experience can be used in designing new experiments, at least coded mask like SVOM