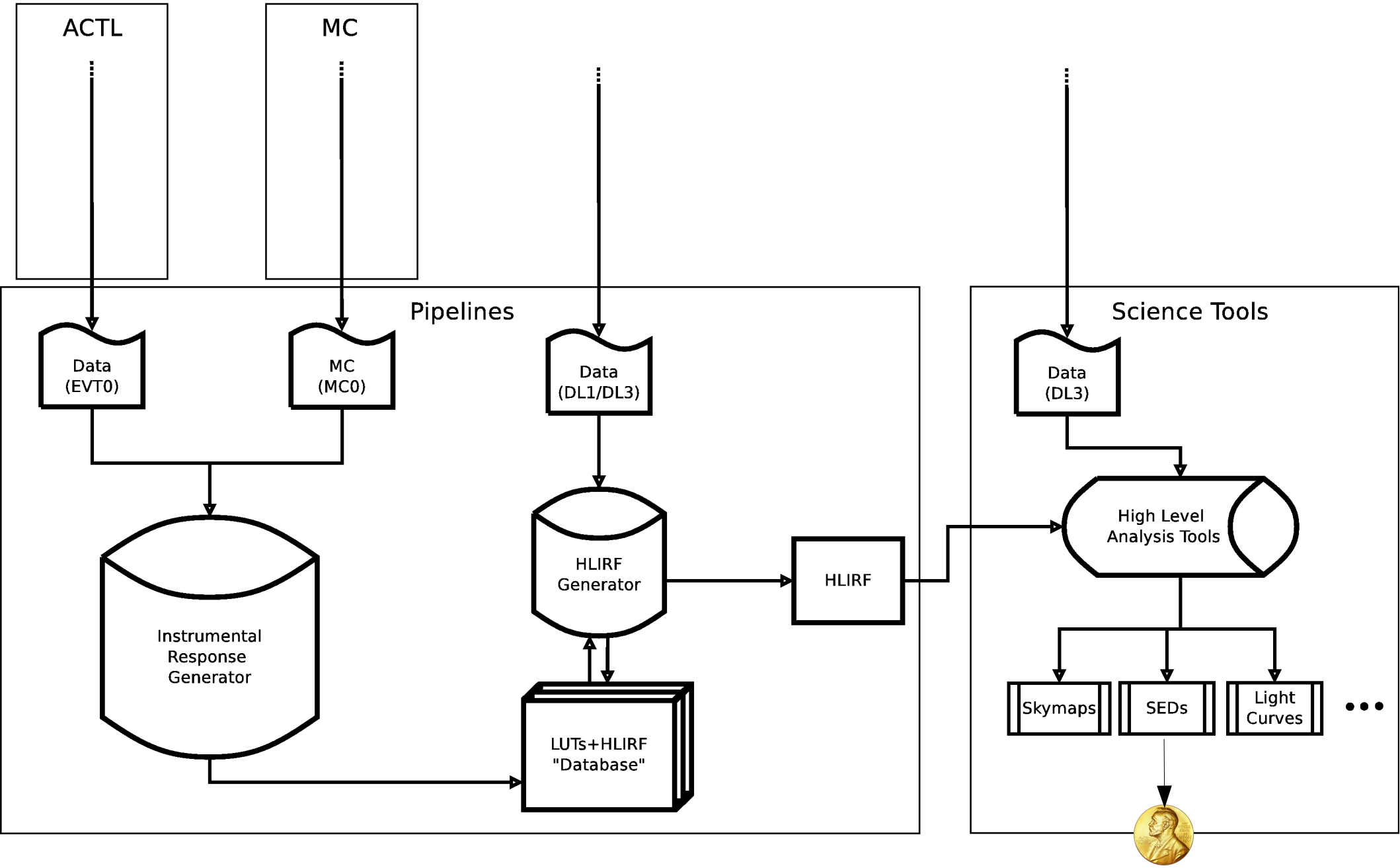
CTA IRF Model

T. Hassan, J. E. Ward and J. Rico

The IRM

- The Instrument Response Model (IRM) is divided into 2 parts:
    - Lookup tables (LUTs): Store low level relations, such as direction LUTs, trained RF, etc...
    - High-Level IRFs (HLIRFs): reconstructed performance
- FITS format:
    - Standard format within astronomy
    - Allows high flexibility
    - Provides the perfect framework to use/store Metadata

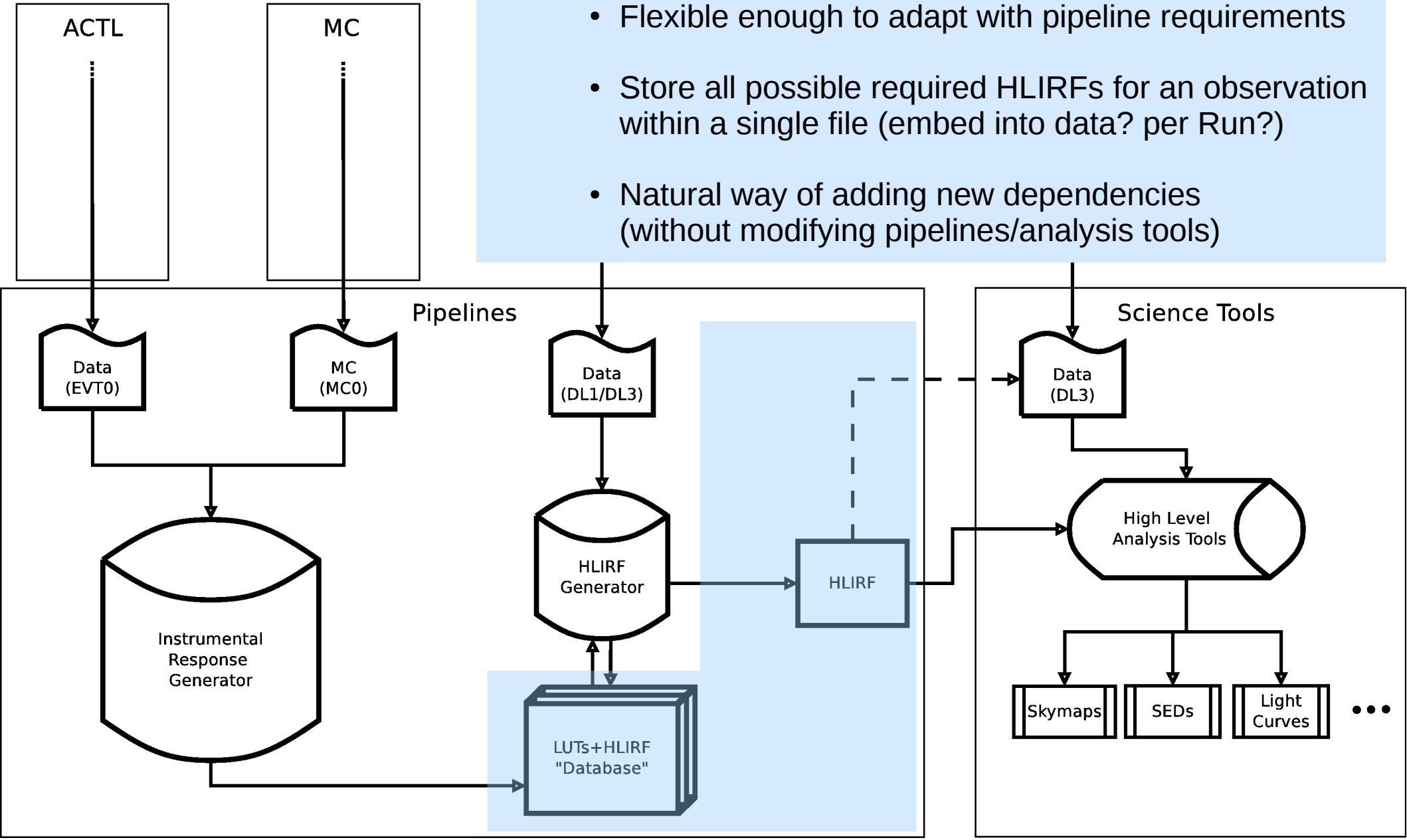
The IRM


ACTL
MC
Pipelines
Data
(EVT0)
MC
(MC0)
Data
(DL1/DL3)
Instrumental
Response
Generator
HLIRF
Generator
HLIRF
LUTs+HLIRF
"Database"
Science Tools
Data
(DL3)
High Level
Analysis Tools
Skymaps
SEDs
Light
Curves

The IRM

A common format is desired:

- Flexible enough to adapt with pipeline requirements
- Store all possible required HLIRFs for an observation within a single file (embed into data? per Run?)
- Natural way of adding new dependencies (without modifying pipelines/analysis tools)



The IRM

- FITS files are a collection of Header Data Units (HDUs)

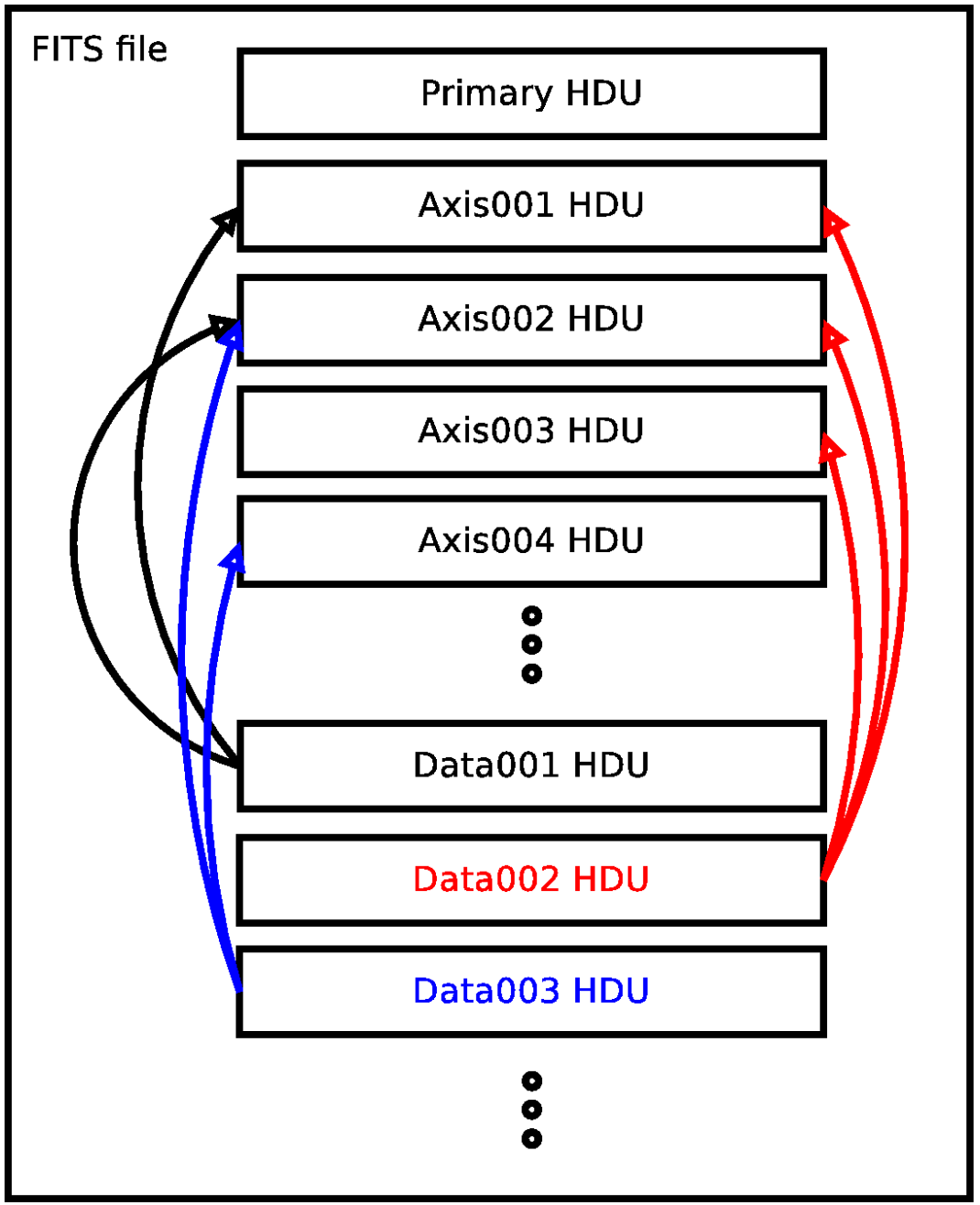
- We propose two types of HDUs:

    - The “Axis” HDUs: Containing all the information regarding IRF dependencies

    - The “Data” HDUs: Containing the instrument performance as a function of referred “Axis” HDUs

- This approach allows any binning, parameterization or adding any dependency to the IRFs within a single FITS file with no data duplicity

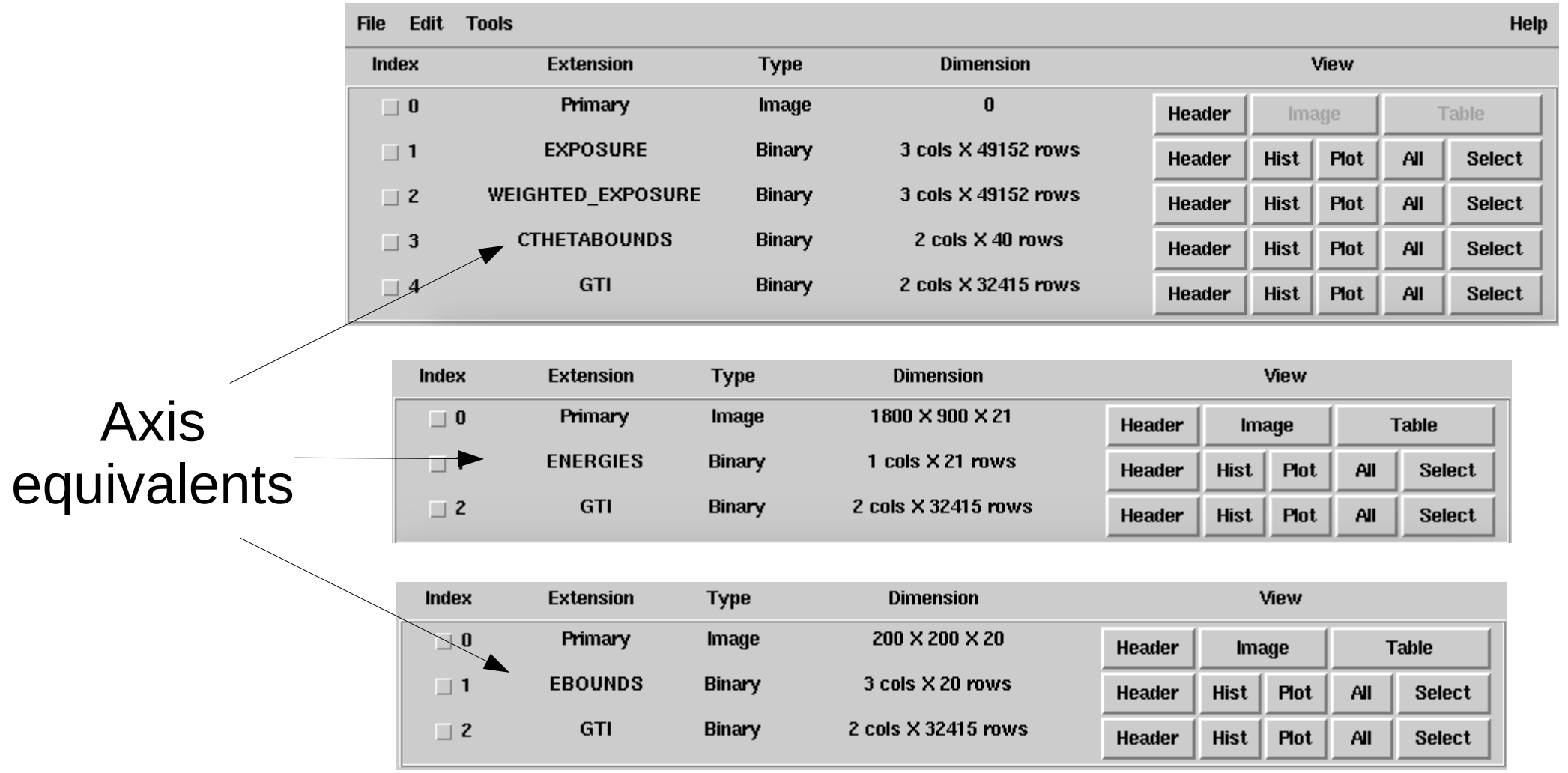
The IRM



- Primary HDU: Metadata
- Axis00*: IRF dependencies (E, θ, φ, ID, NSB, weather...)
- Data00*: IRF n-dimensional data cubes. (Eff. Area, Mig. Matrix... etc..)

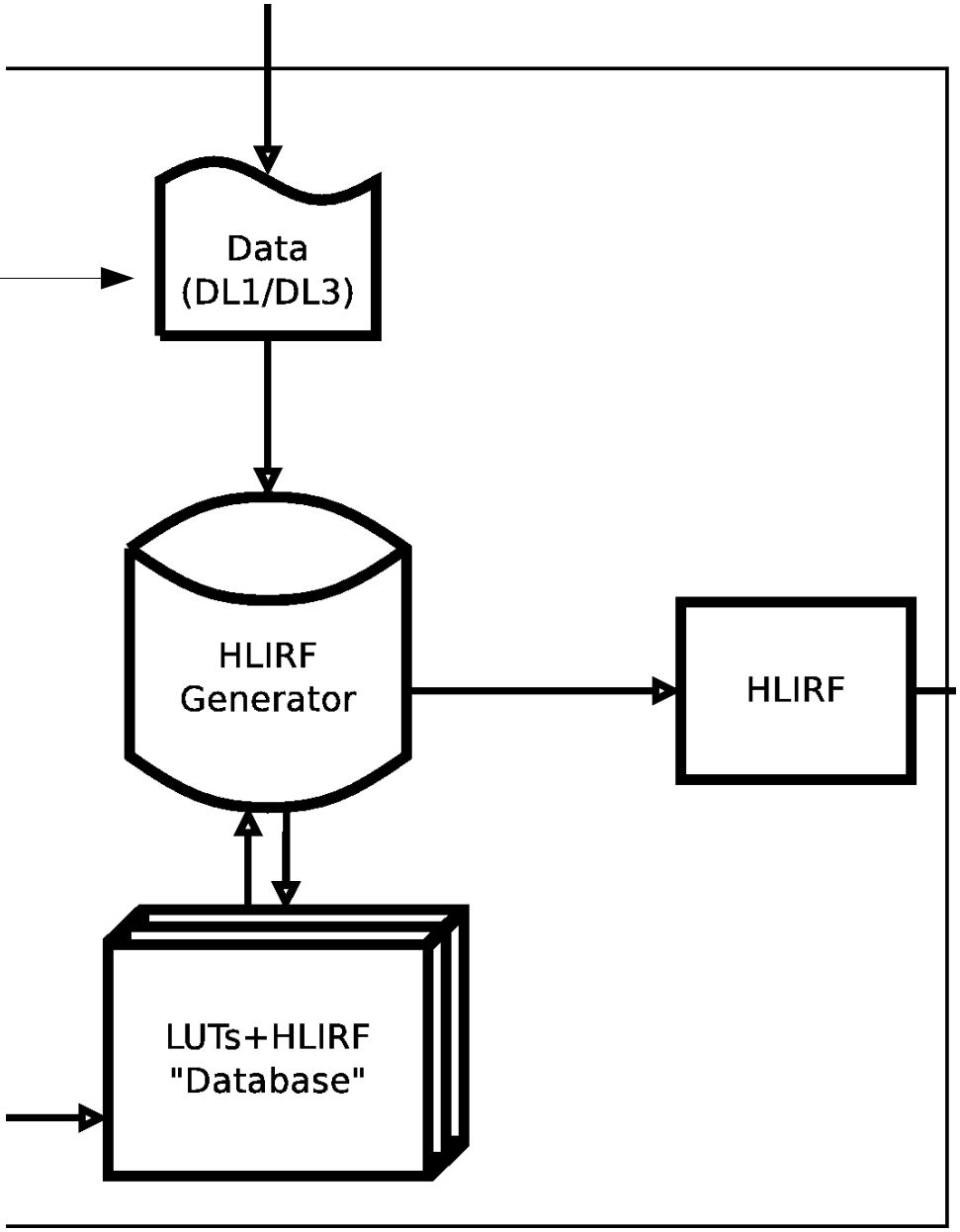
The IRM

- The approach is not new: Fermi-LAT data

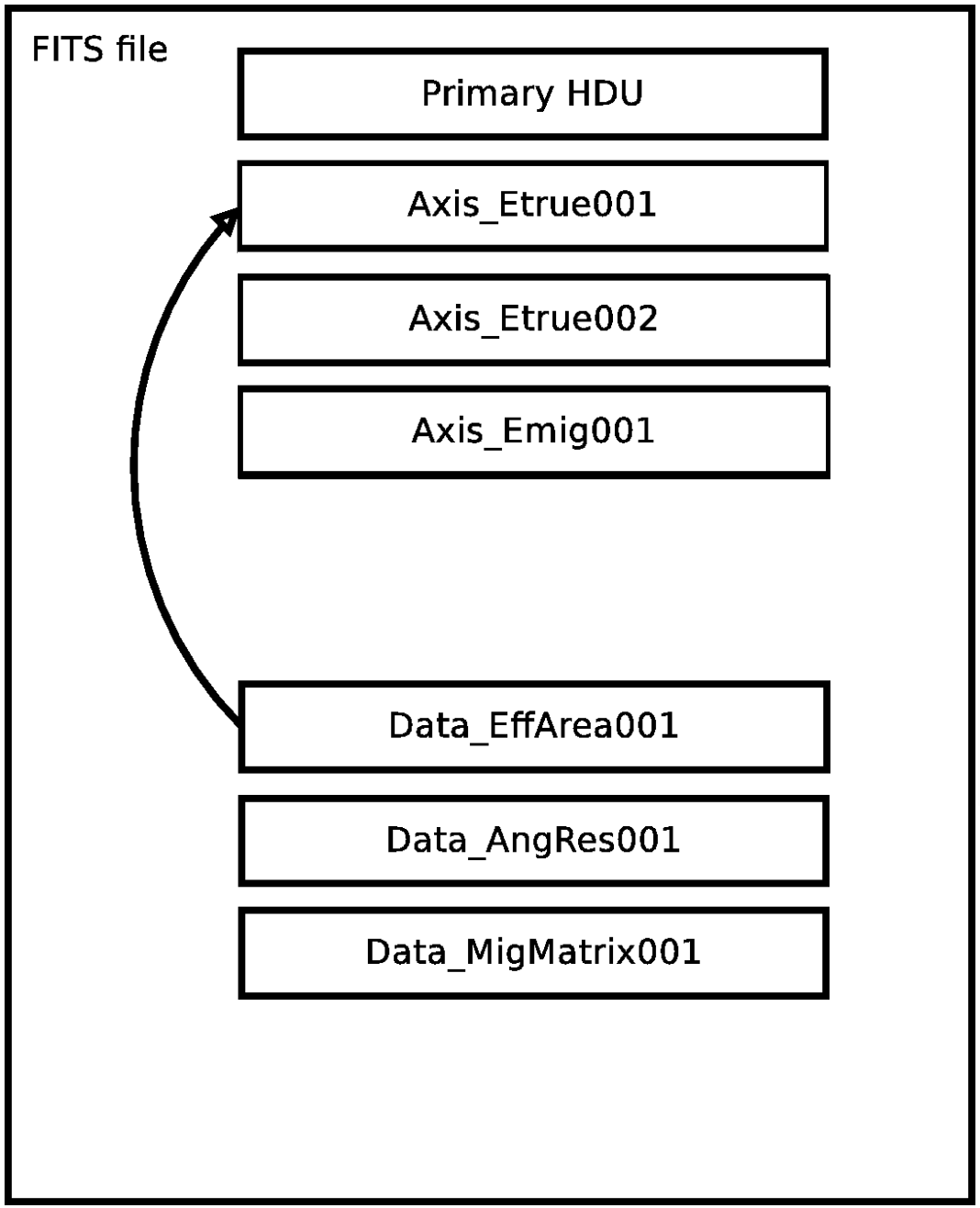


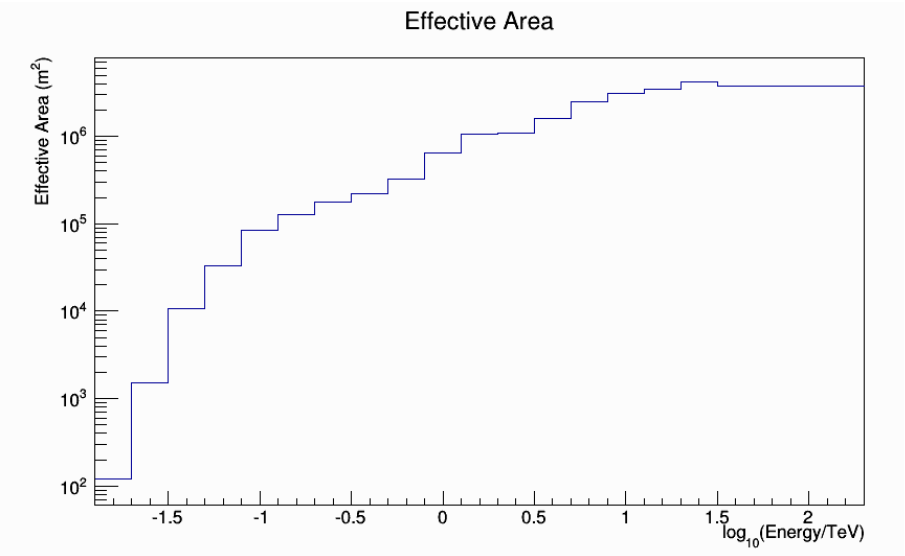
The IRM – Example: Obs A

- Obs. A contains default on-axis data

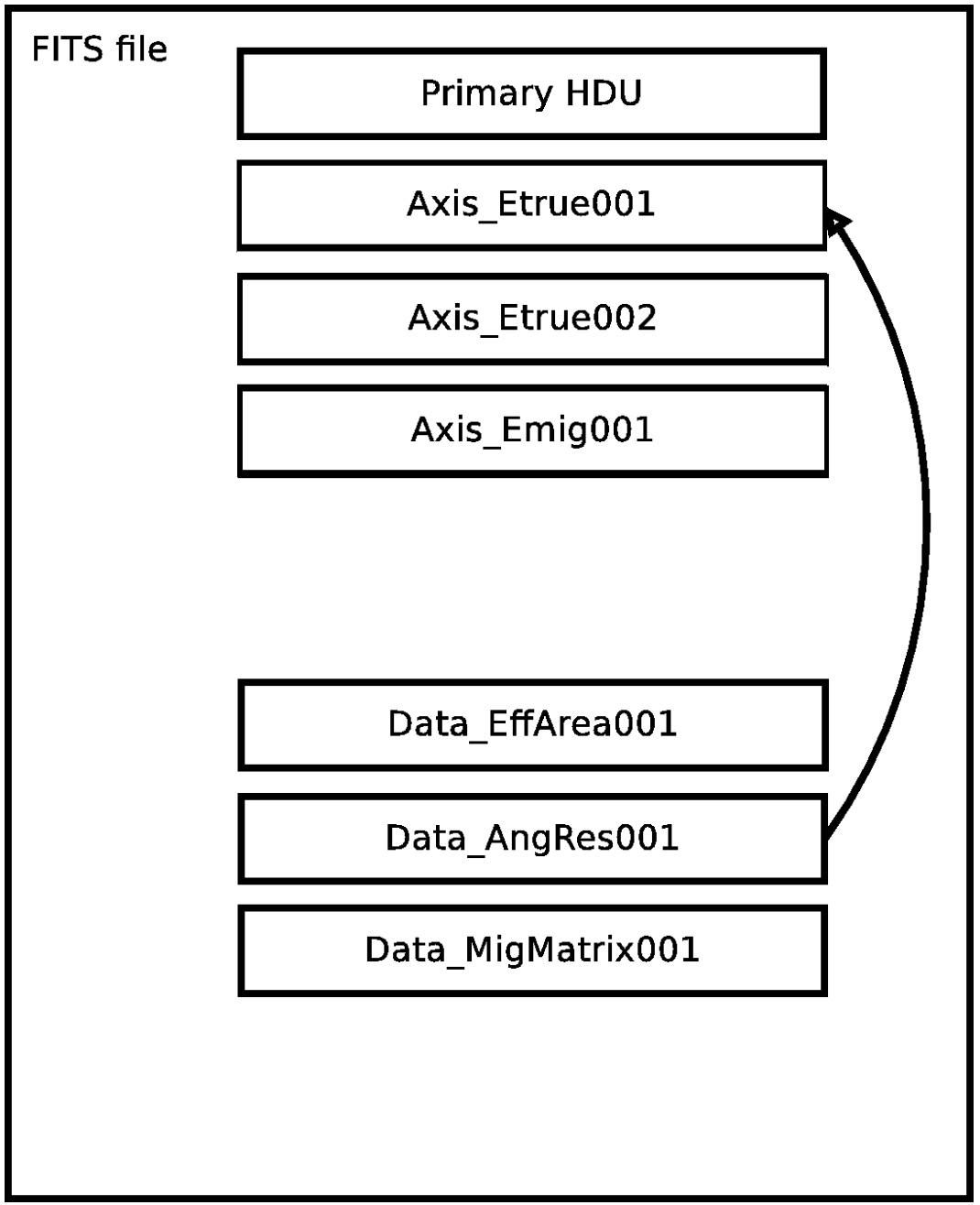


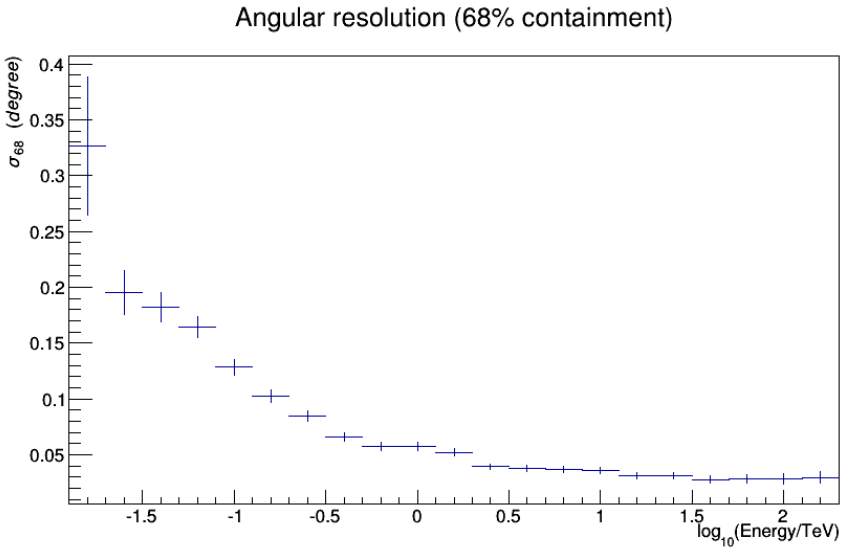
The IRM – Example: Obs A



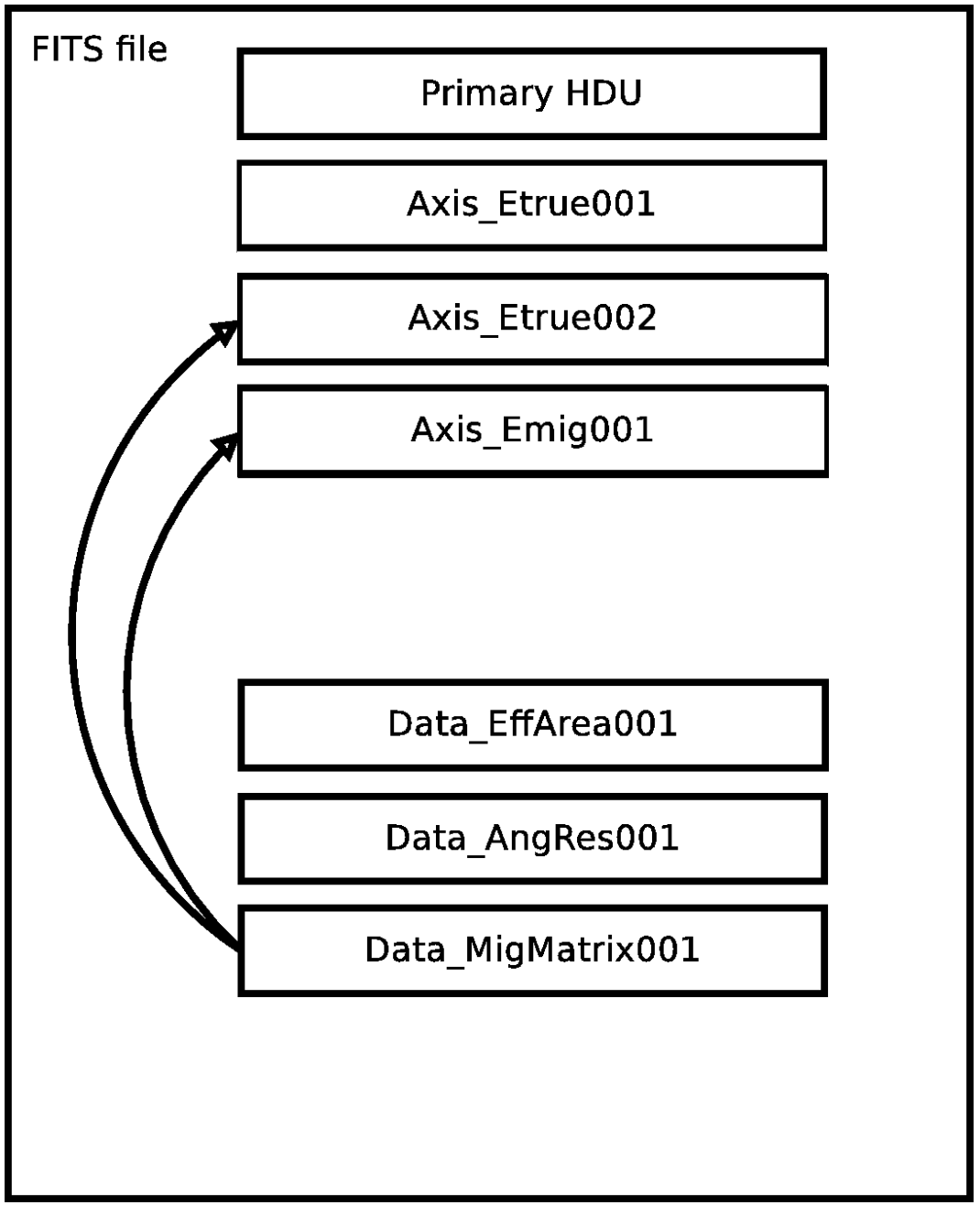


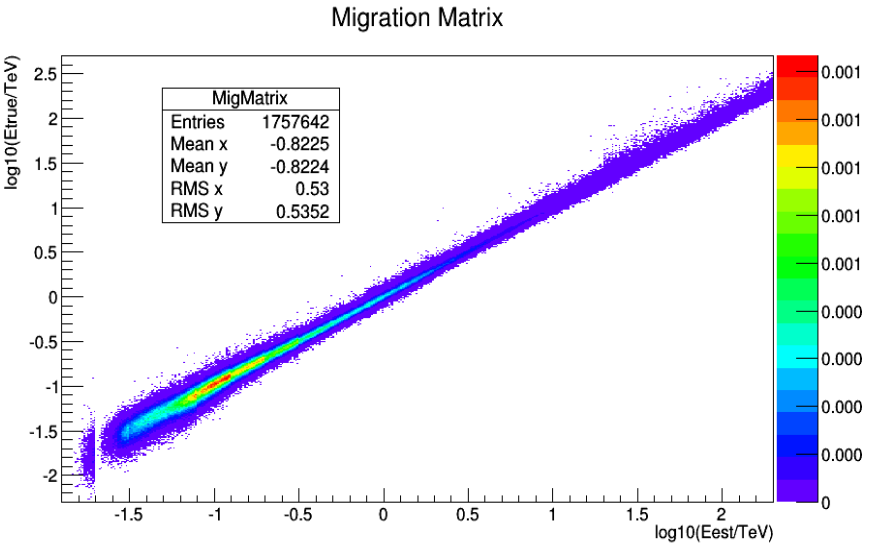
The IRM – Example: Obs A





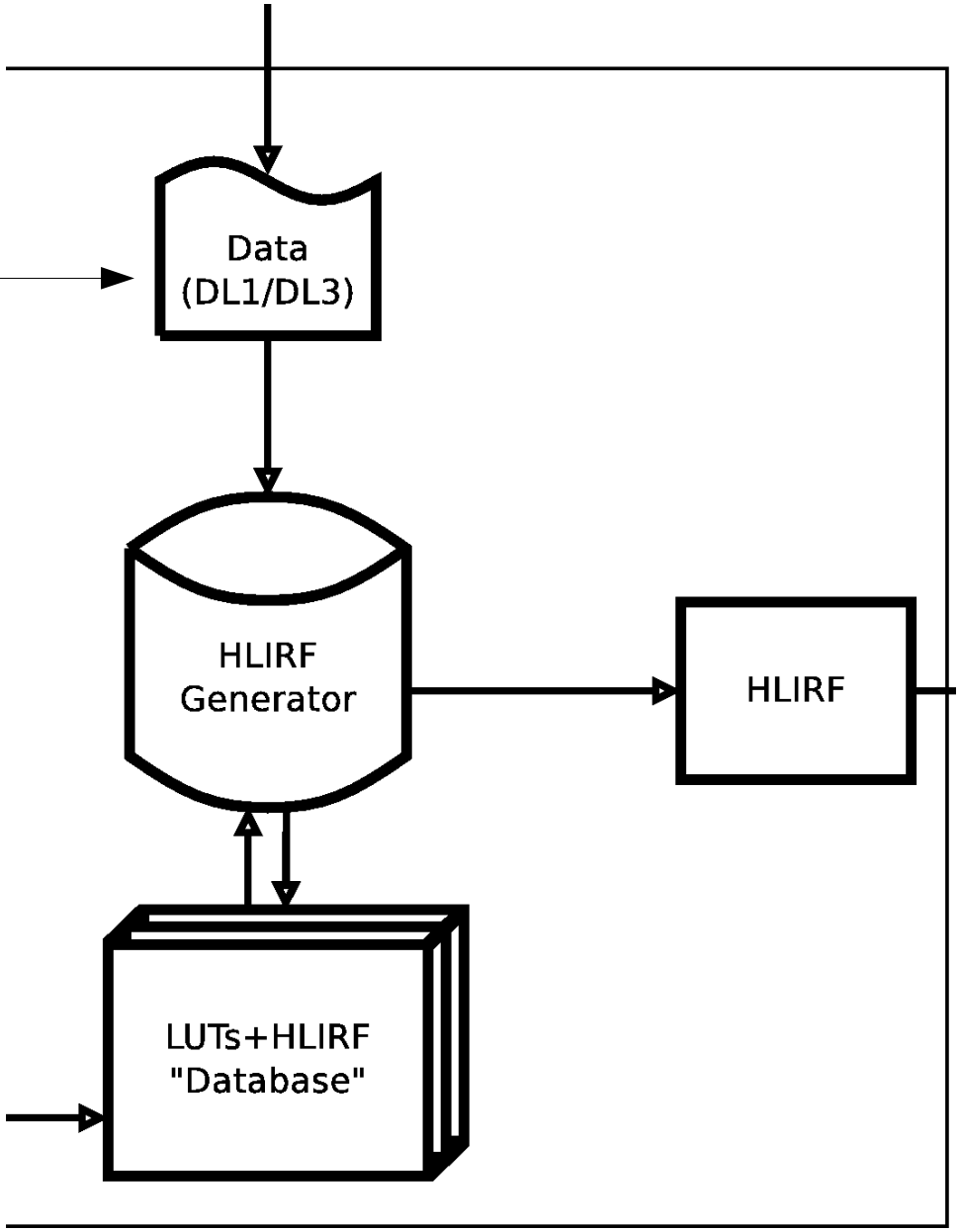
The IRM – Example: Obs A



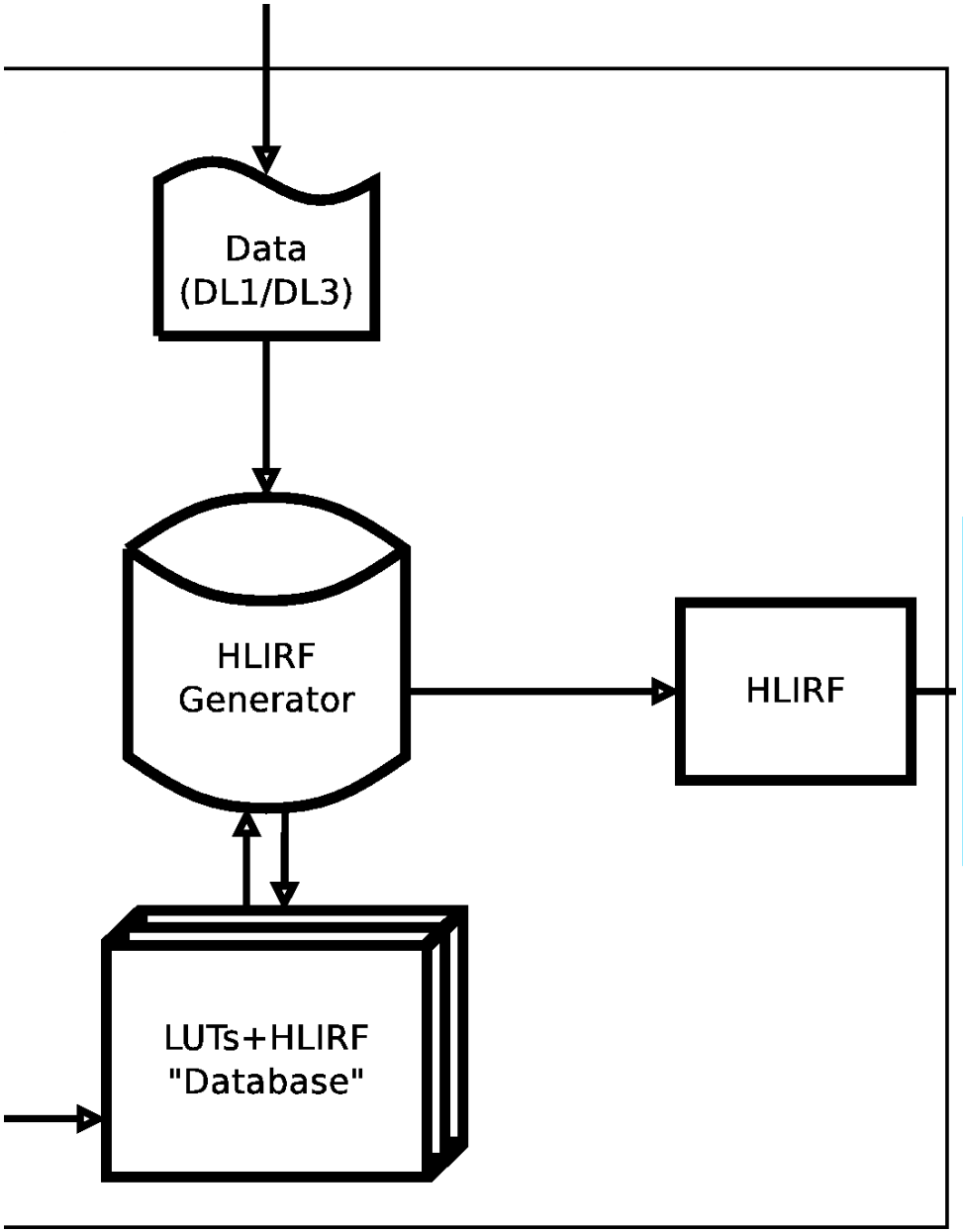


The IRM – Example: Obs B

• Obs. B contains data with 3 different weather conditions

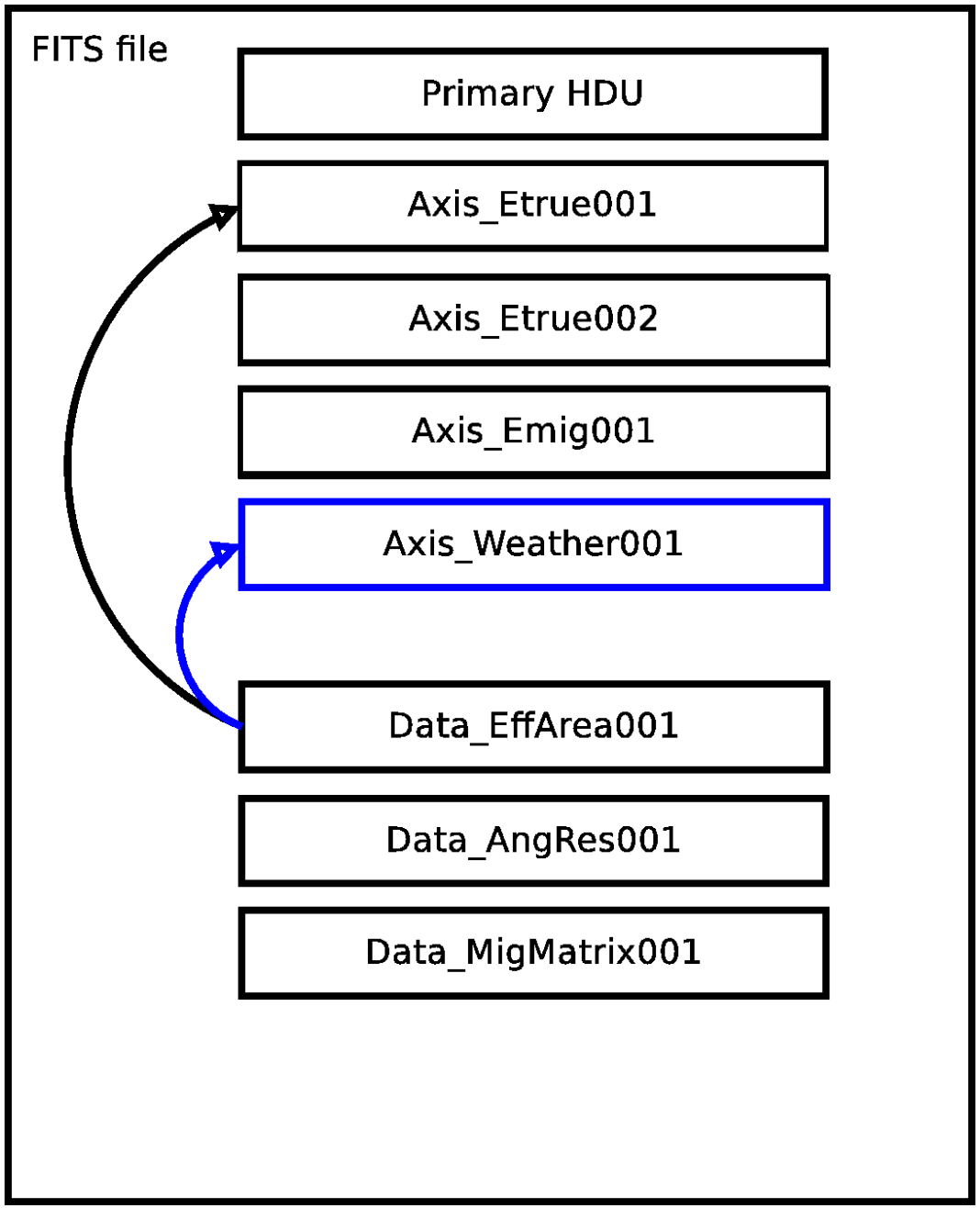


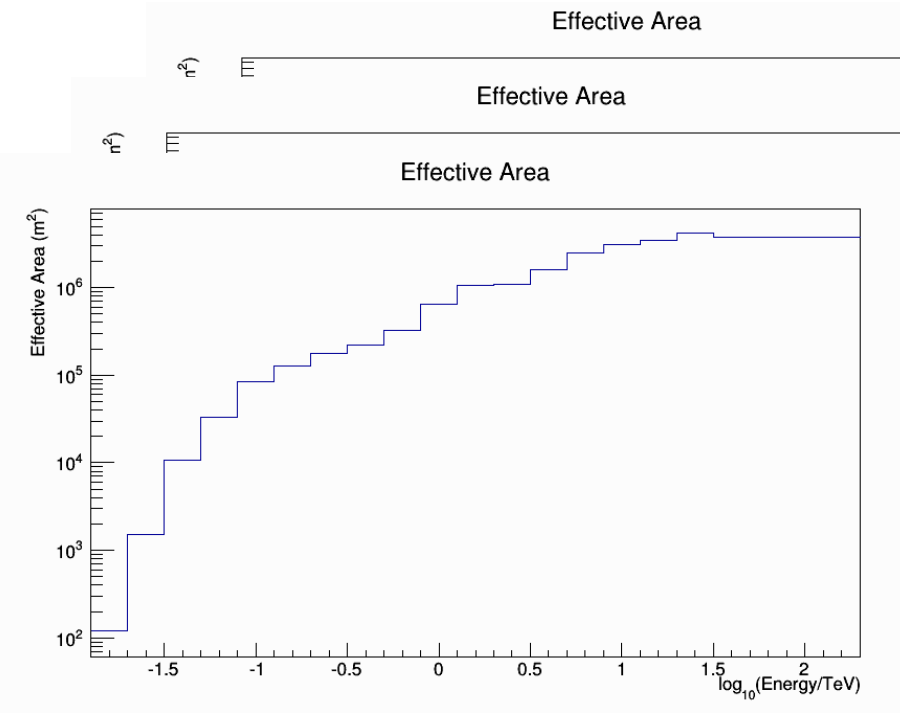
The IRM – Example: Obs B



- HLIRFs need to contain these IRF dependencies (and data analysis to use them accordingly)

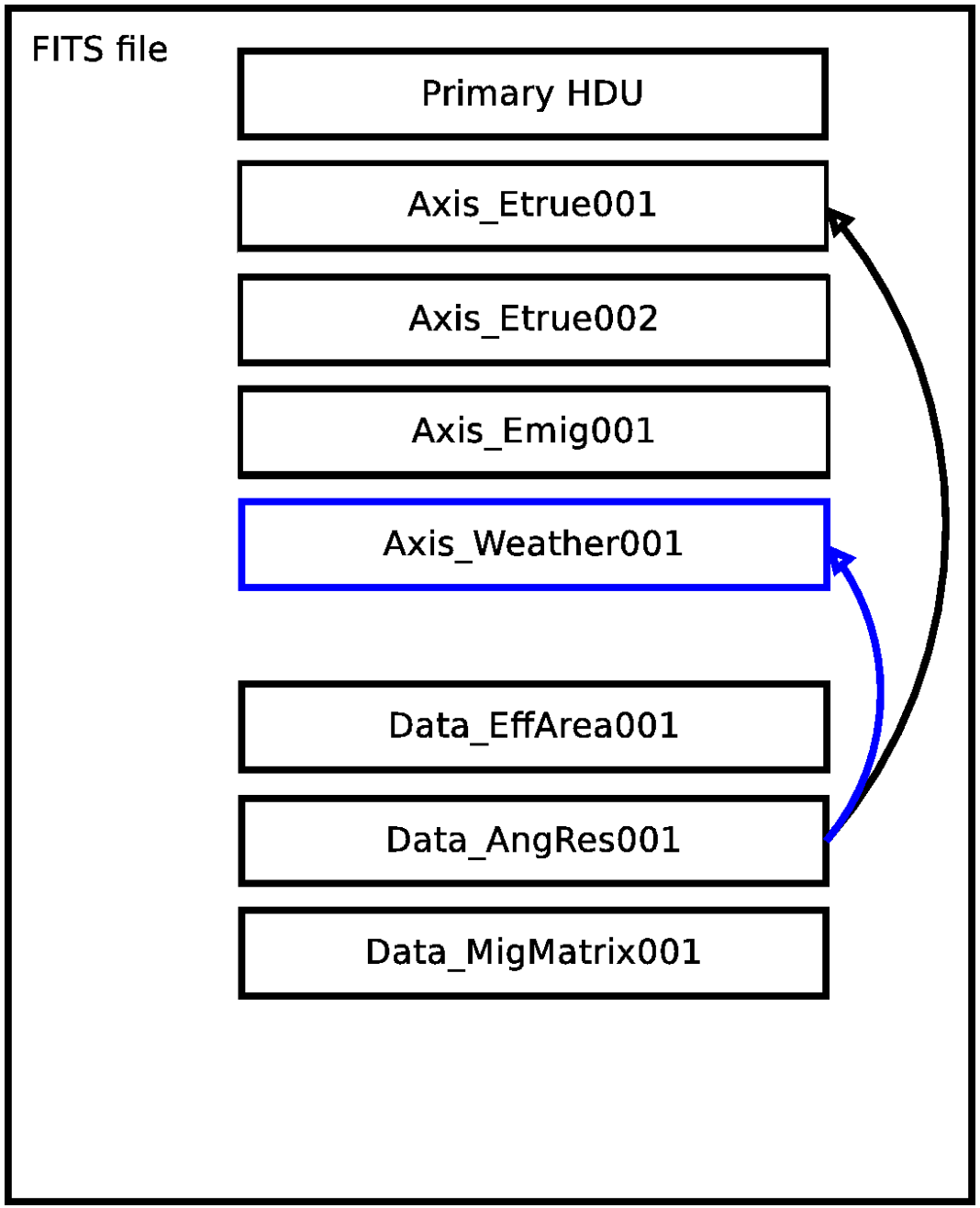
The IRM – Example: Obs B

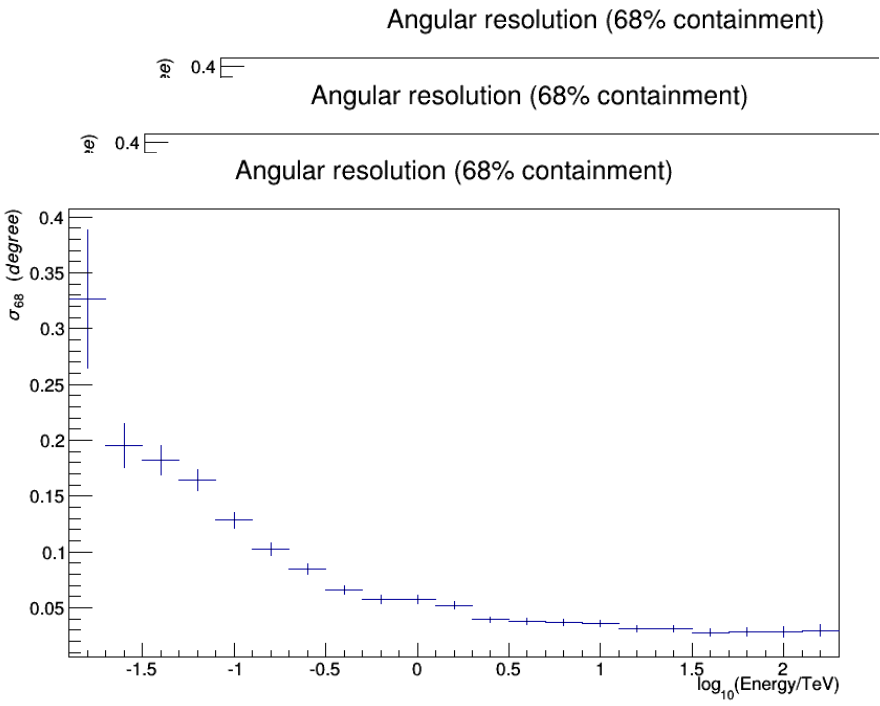




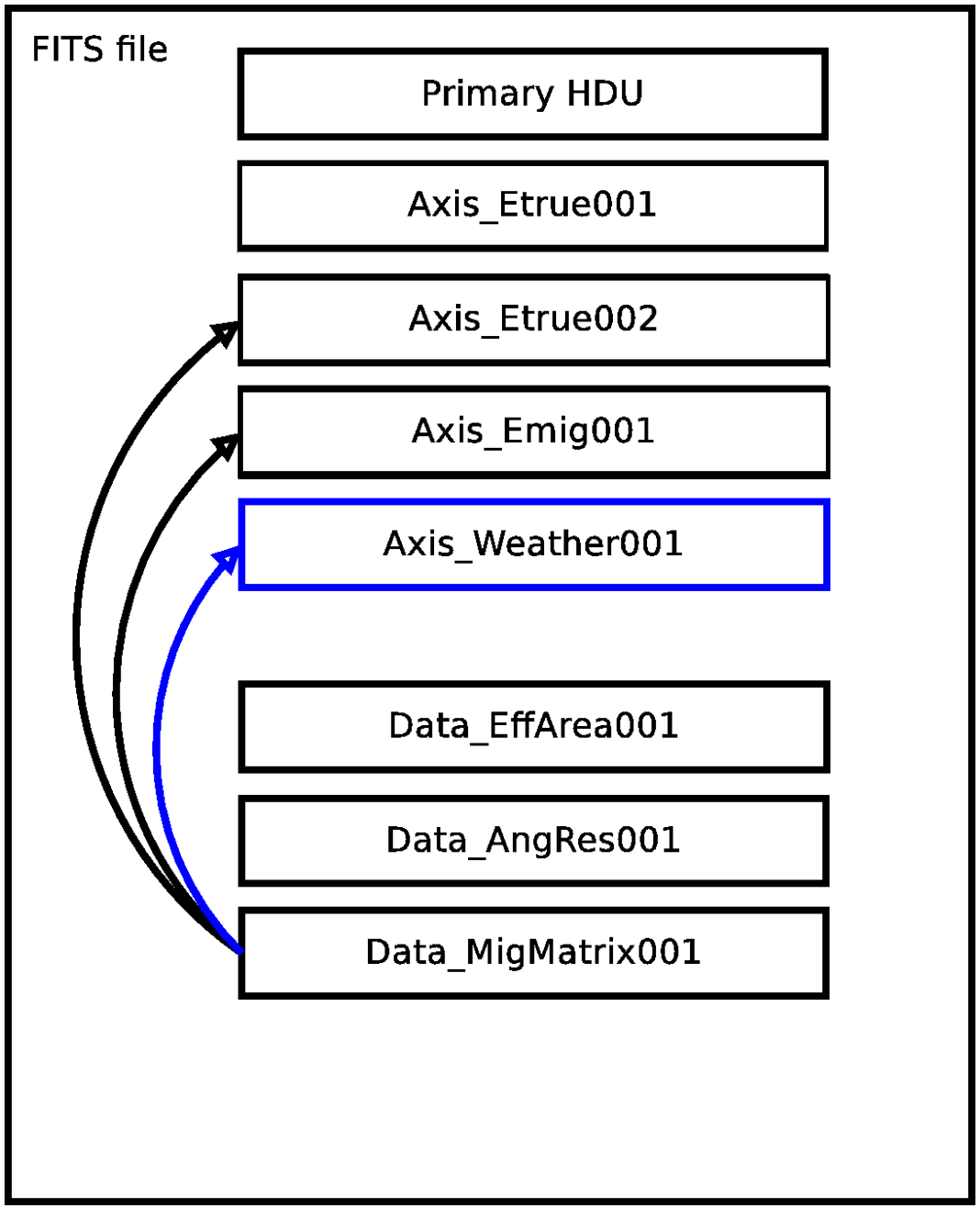
Axis_Weather001 could also be defined as a parameterization, of the EffArea for different weather conditions

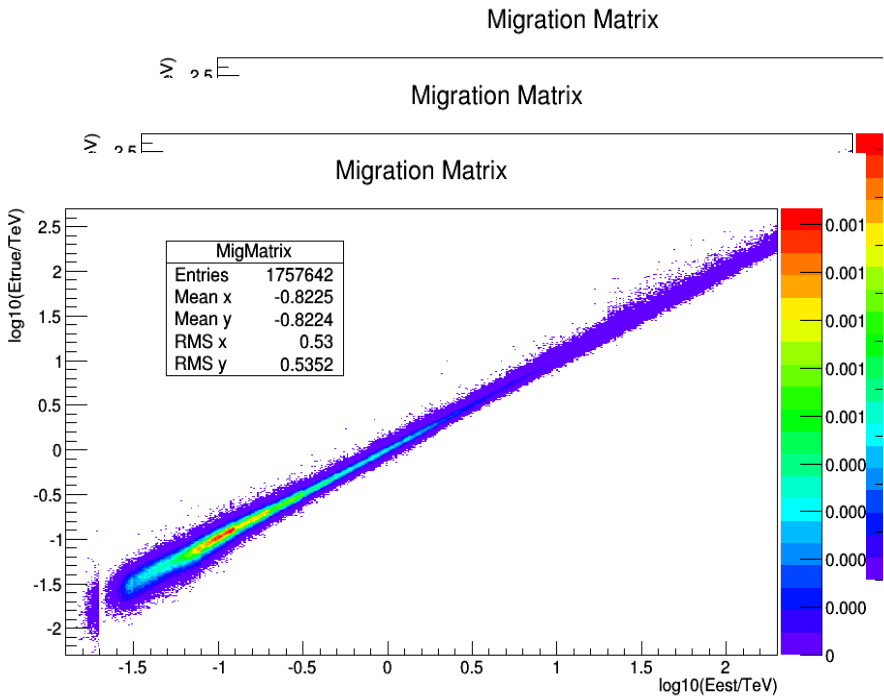
The IRM – Example: Obs B





The IRM – Example: Obs B





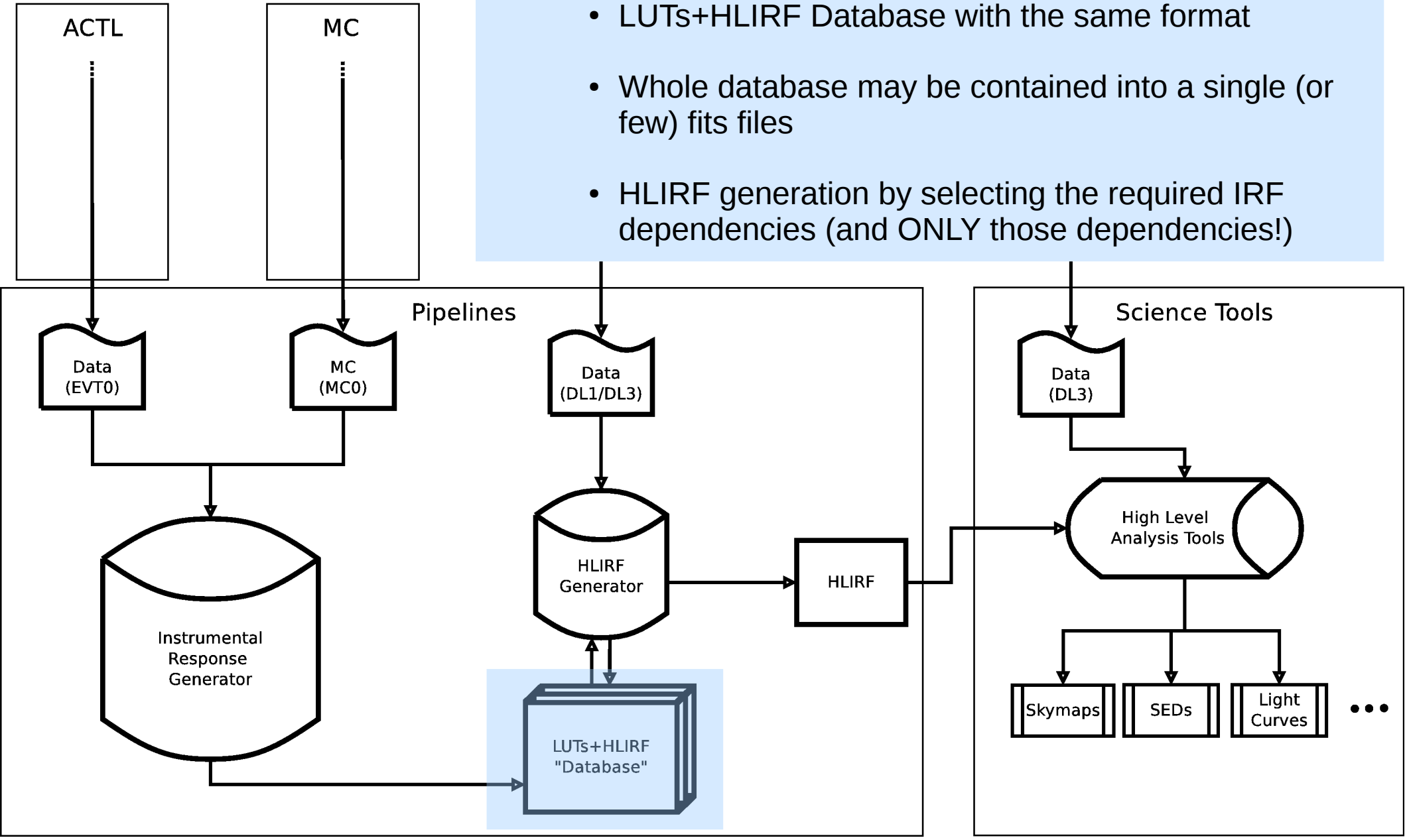
Identify dependencies of PDF on estimators

- **E:** Energy
- **Theta, Phi** (**r** assuming az sym.): Shower arrival direction
- **ID:** Particle nature (gamma, proton, electron or “signal” and “background”)
- **S/BG:** S/N maximization (hadroness, mean-scaled width etc.). Function of analysis types, aims (similar concept to Fermi-LAT DataClean, Diffuse Class events)
- **THETA/PHI:** Telescope pointing coordinates
- **ArrCon:** Array configuration - Full, sub-array, pointing scheme, trigger modes
- **NSB:** Night Sky Background levels (dark, medium, high, moon)
- **W:** Weather -> A, B or C grade? (previously defined), Transmittance values?
- **HWStatus:** Hardware status describing mirror reflectivity, camera window transmittance, optical PSF, PMT gains
- **SpecShape:** Assumed spectral shape, depends on fineness of binning

The IRM

The proposed format allows:

- LUTs+HLIRF Database with the same format
- Whole database may be contained into a single (or few) fits files
- HLIRF generation by selecting the required IRF dependencies (and ONLY those dependencies!)



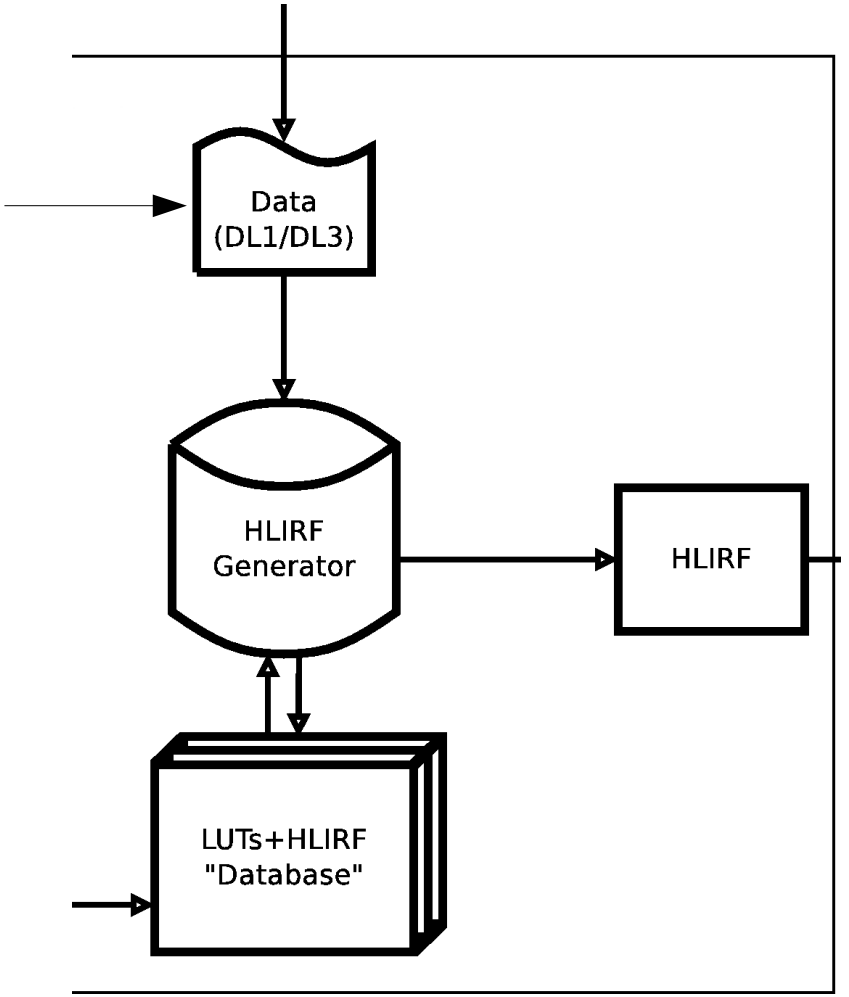
The IRM – Example: Obs C

- Obs. C

  30º<Zd<60º

  Full-Array
  +
  Partial

  W = 1



- HLIRF

  30º<Zd<60º

  Full-Array
  +
  Partial

  W = 1

Not required to be stored →

0º<Zd<30º
60º<Zd
...

IRM: Current Status

- Already working examples: (By J.E. Ward and J. Rico)
    - Preliminary FITS keywords for Data/Axis HDUs
    - ROOT (IFAE/DESY) IRFs → IRM translator
    - FITS files merger
- As soon as I catch up:
    - Define/document in detail the best implementation of the IRM (define keywords, FITS extensions used, etc...)
    - Develop required software:
        - Proper libraries to read/write IRM
        - Implement it into pipelines and science tools

IRM: Current Status

Thank you