CTA Data Diffusion

The Cherenkov Telescope Array

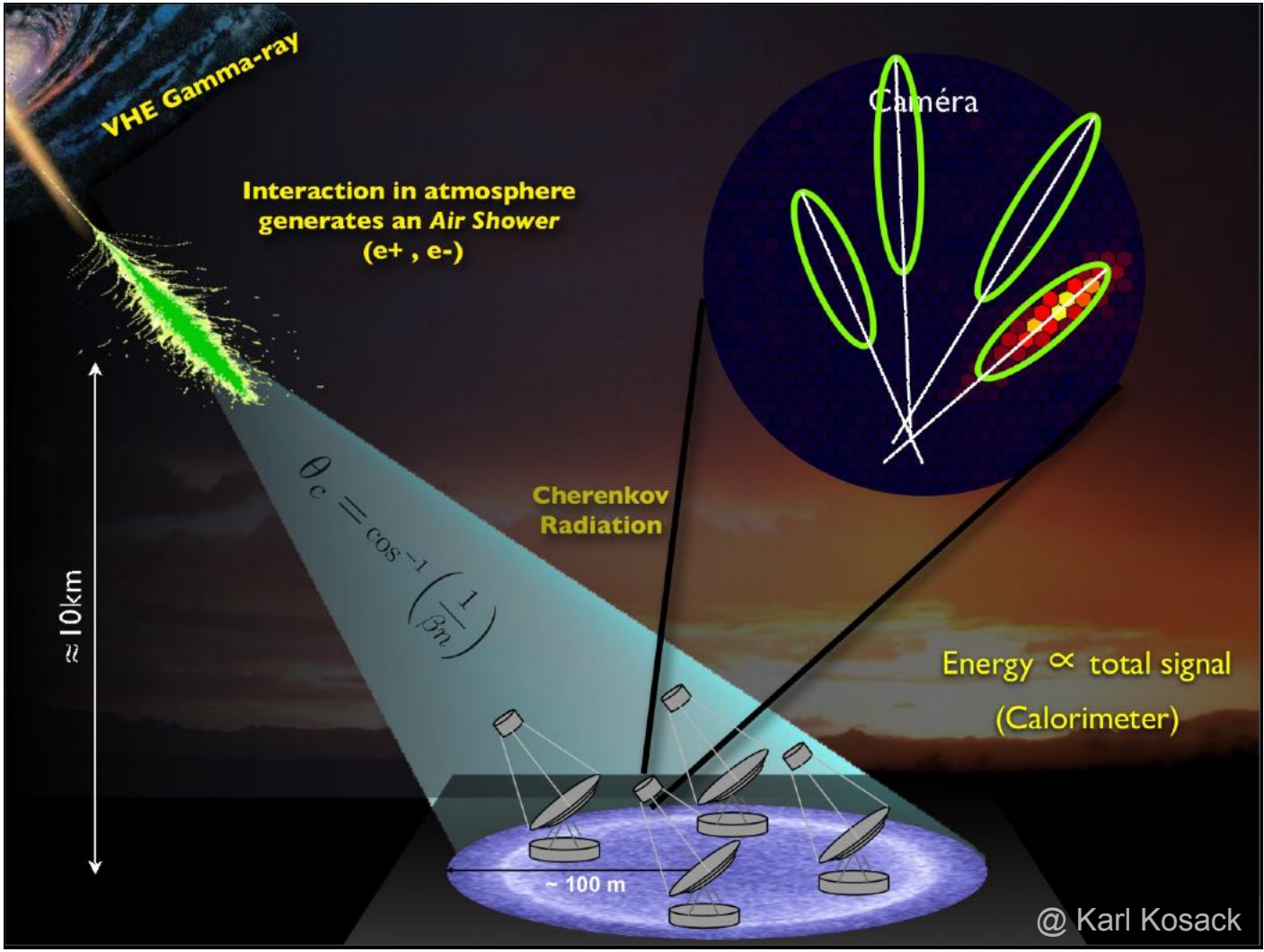
Mathieu Servillat

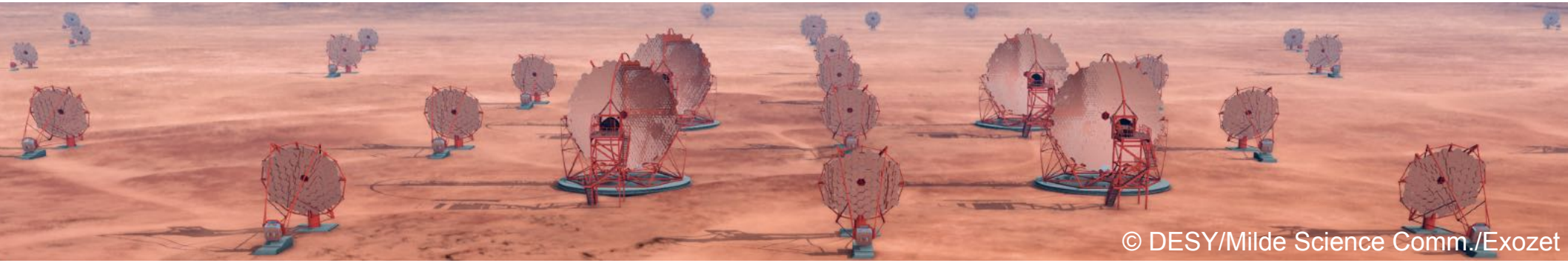
Laboratoire Univers et Théories
Observatoire de Paris
PSL Research University

3rd ASTERICS DADI Tech Forum, Strasbourg

Cherenkov Imaging

- **Dark nights** (small duty cycle)
- Field of view: 5-8 degrees
- **Event Reconstruction**: photon, particle shower, Cherenkov light (faint, few nanoseconds)
- **Atmosphere** = calorimetre Simulations, assumptions
- **Complex Metadata**, need to be structured

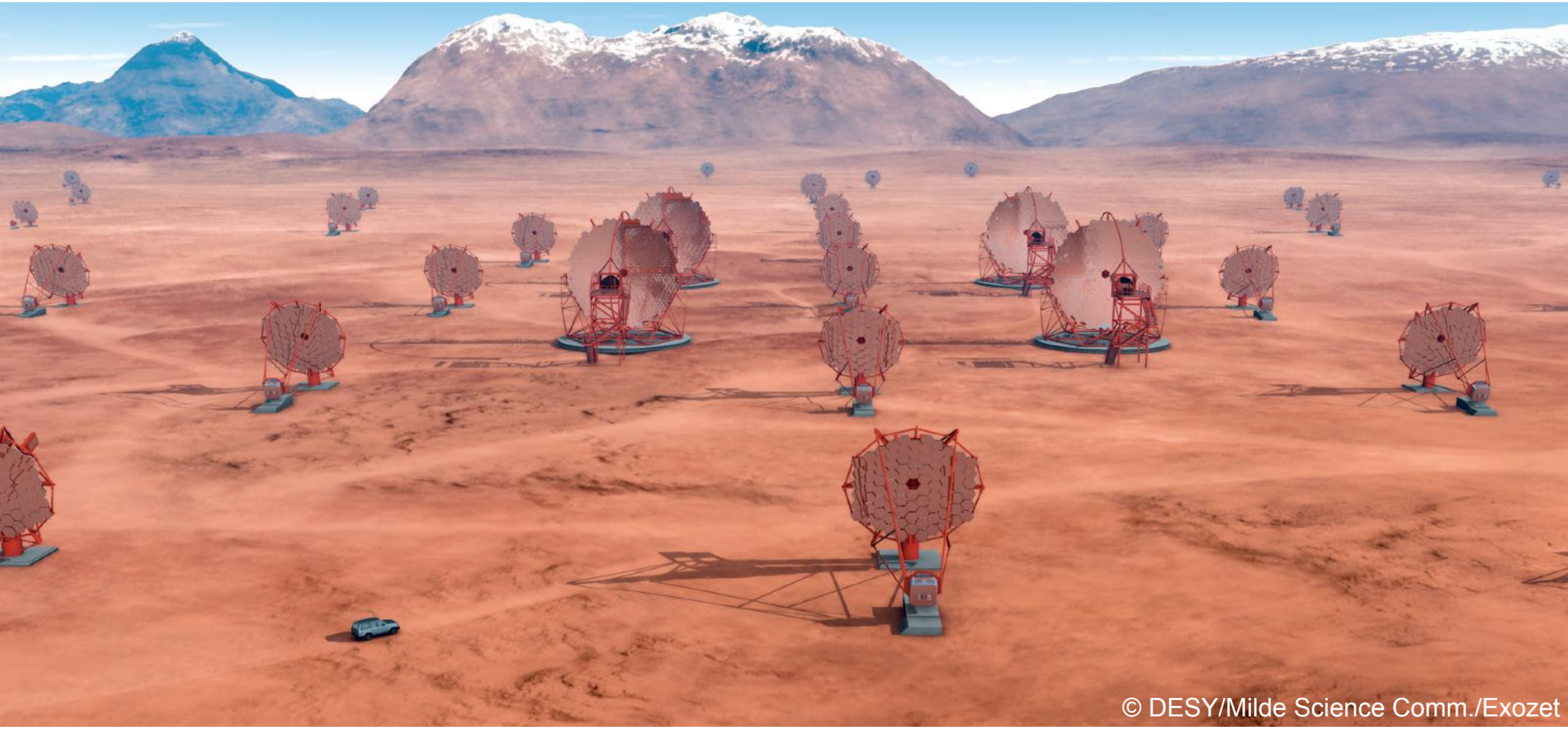




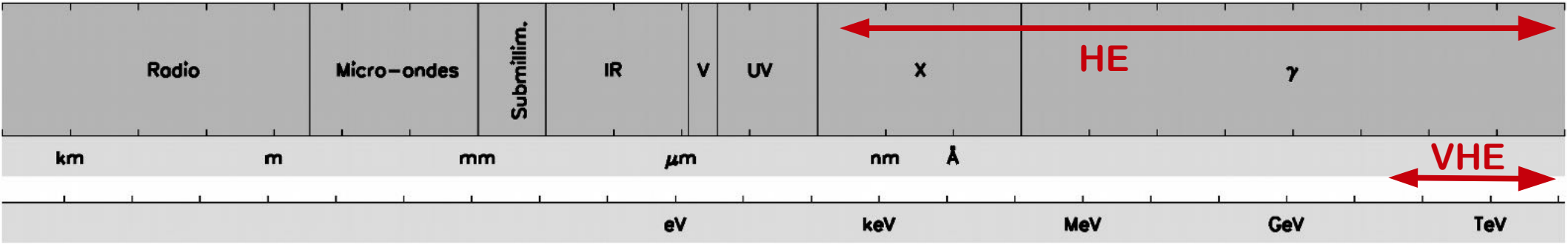
cherenkov telescope array

Observatory

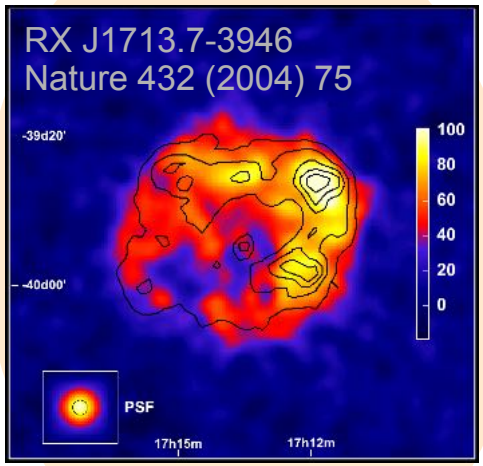
- Two arrays of **100 (South)** et **20 (North)** Cherenkov telescopes (4, 12 et 24 m in diametre)
- July 2015: site selection, Chile (ESO) and La Palma
- 2016: pre-production phase
- 2018-2013: production phase
- Observatory **open** to the Astronomy community



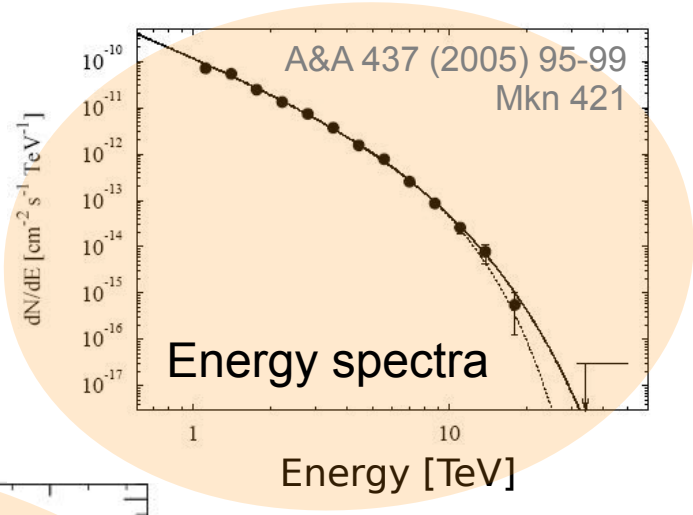
Very high energy (VHE) data

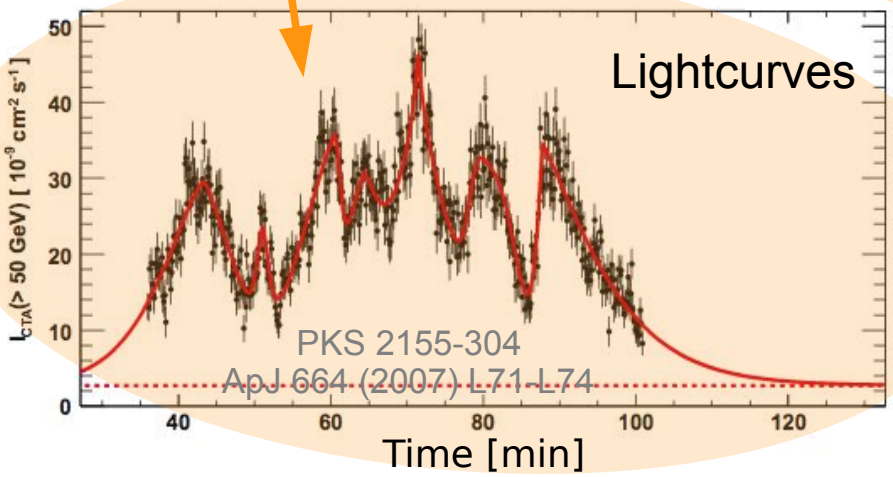


- Several orders of magnitude
- Photon counting
- Low count statistics, high background
- **Event lists** (coordinates, time, energy)



Images

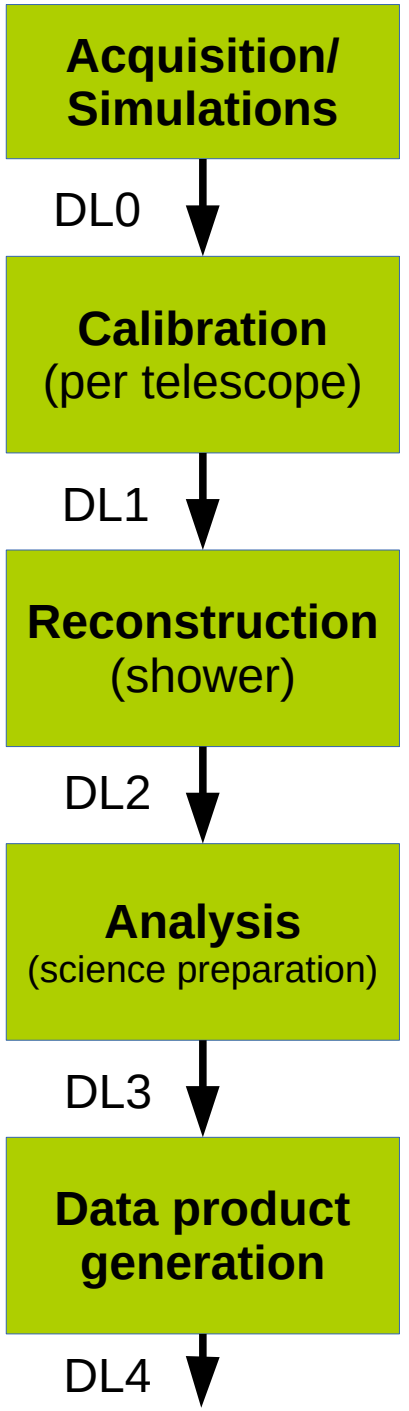




Data levels and workflow

Data Level	Short Name	Description	Data reduction factor
Level 0 (DL0)	DAQ-RAW	Data from the Data Acquisition hardware/software.	
Level 1 (DL1)	CALIBRATED	Physical quantities measured in each separate camera: photons, arrival times, etc., and per-telescope parameters derived from those quantities.	1-0.2
Level 2 (DL2)	RECONSTRUCTED	Reconstructed shower parameters (per event, no longer per-telescope) such as energy, direction, particle ID, and related signal discrimination parameters.	10^{-1}
Level 3 (DL3)	REDUCED	Sets of selected (e.g. gamma-ray-candidate) events, along with associated instrumental response characterizations and any technical data needed for science analysis.	10^{-2}
Level 4 (DL4)	SCIENCE	High Level binned data products like spectra, sky maps, or light curves.	10^{-3}
Level 5 (DL5)	OBSERVATORY	Legacy observatory data, such as CTA survey sky maps or the CTA source catalog.	10^{-5} - 10^{-3}

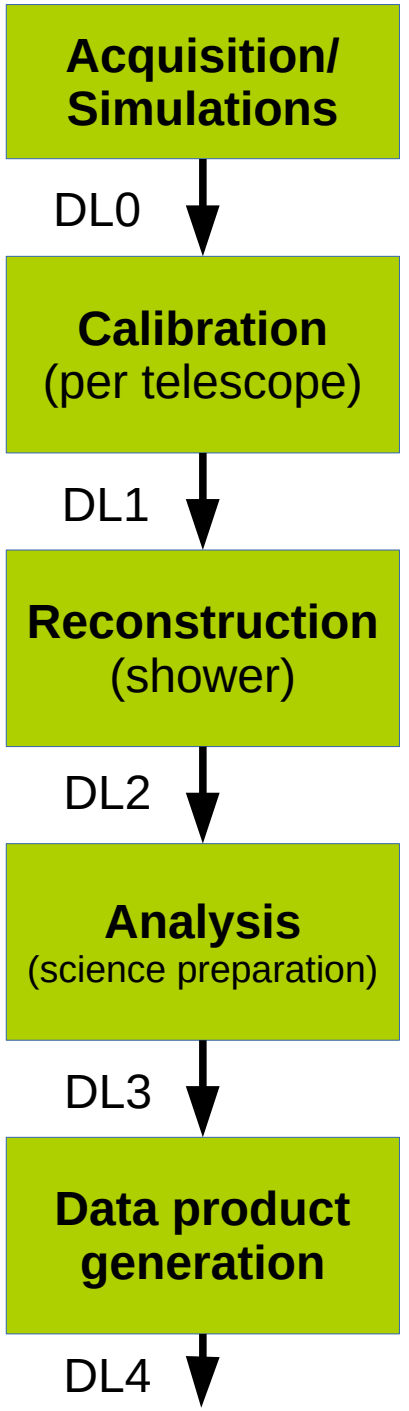
Published & Archived



Data Processing Pipeline

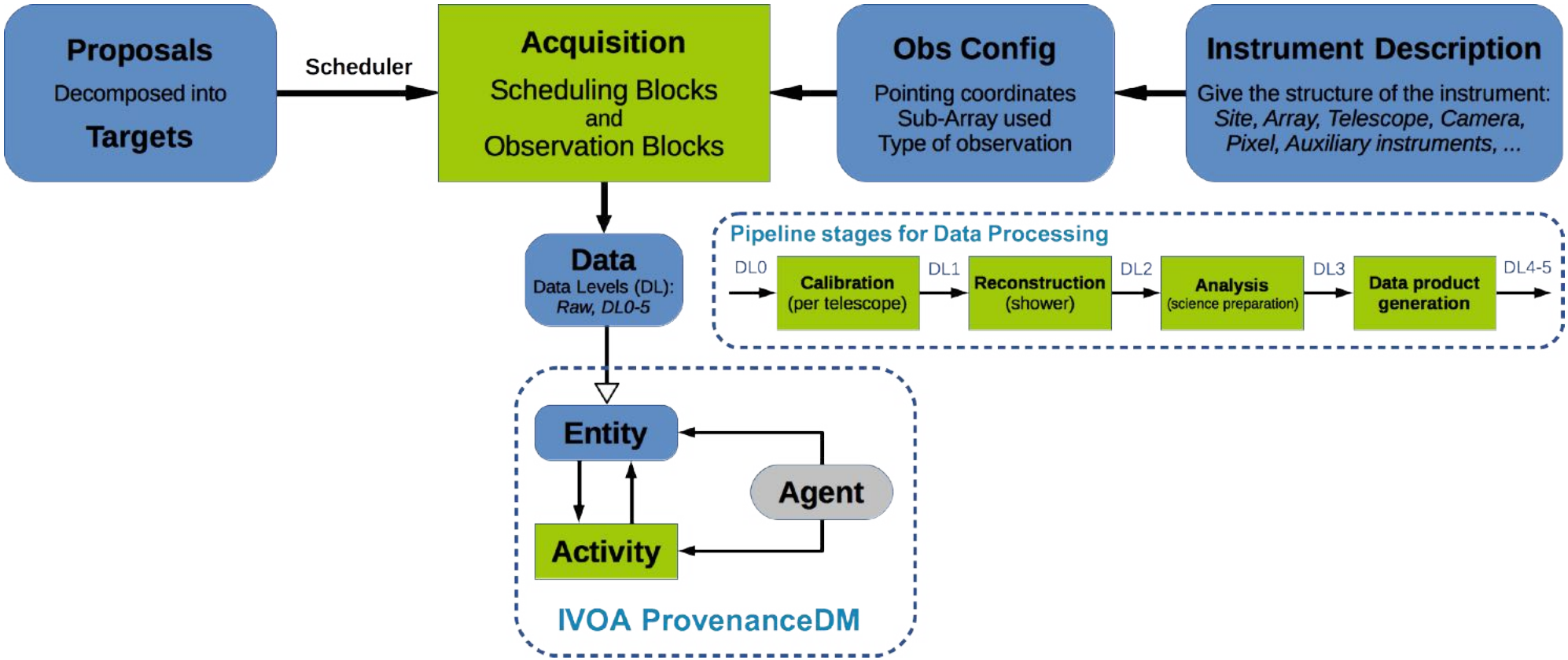
- **Open** observatory
- Must ensure that data processing is **traceable** and **reproducible** (A-USER-0110)
- **Inform** user on processing steps performed
- Link to progenitor to regenerate data (DL3 to DL4)

- Identify how a data product was produced
  ⇒ Provenance
- Identify what detailed options were used
  ⇒ Configuration



High level data model

- Defines **structure** of services, content and context of data
- Can be seen as a **global interface**



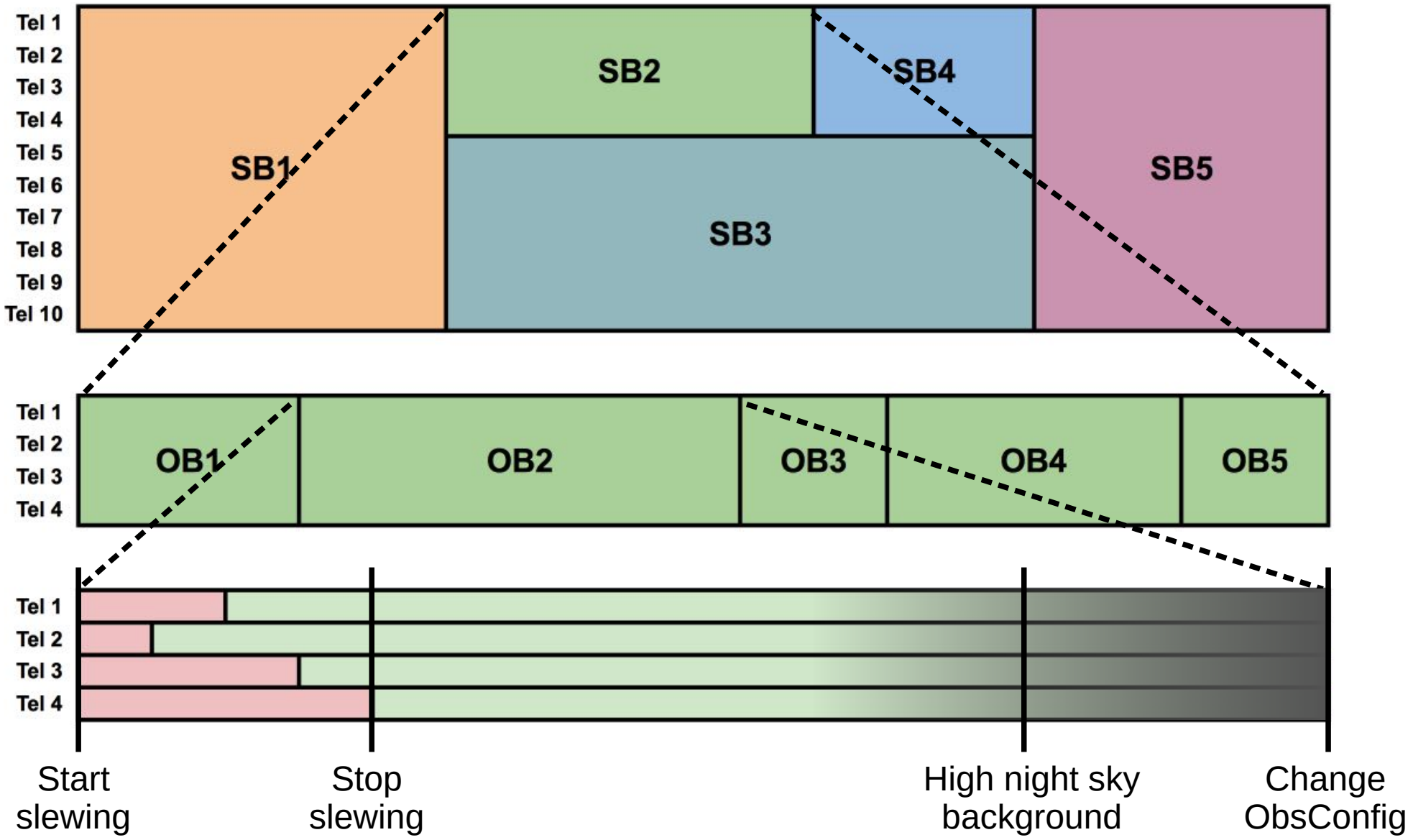
High level data model

- **Proposals → Targets** + requirements and constraints
  - **Scheduling** Blocks
    (sequence of observations planned for a given Target)
  - **Observation** Blocks
    (effective start and stop times with a given configuration)
- **ObsConfig**
  - Defines sky positions (set of coordinates), strategy, sub-array, type of observation, pointing and trigger modes...
- **InstrumentDescription**
  - Static part of the ObsConfig → simply point to a description file
  - SubArray: fixed set of telescopes, list of active telescopes
- **Acquisition**
  - Raw Data then processed to higher Data Levels

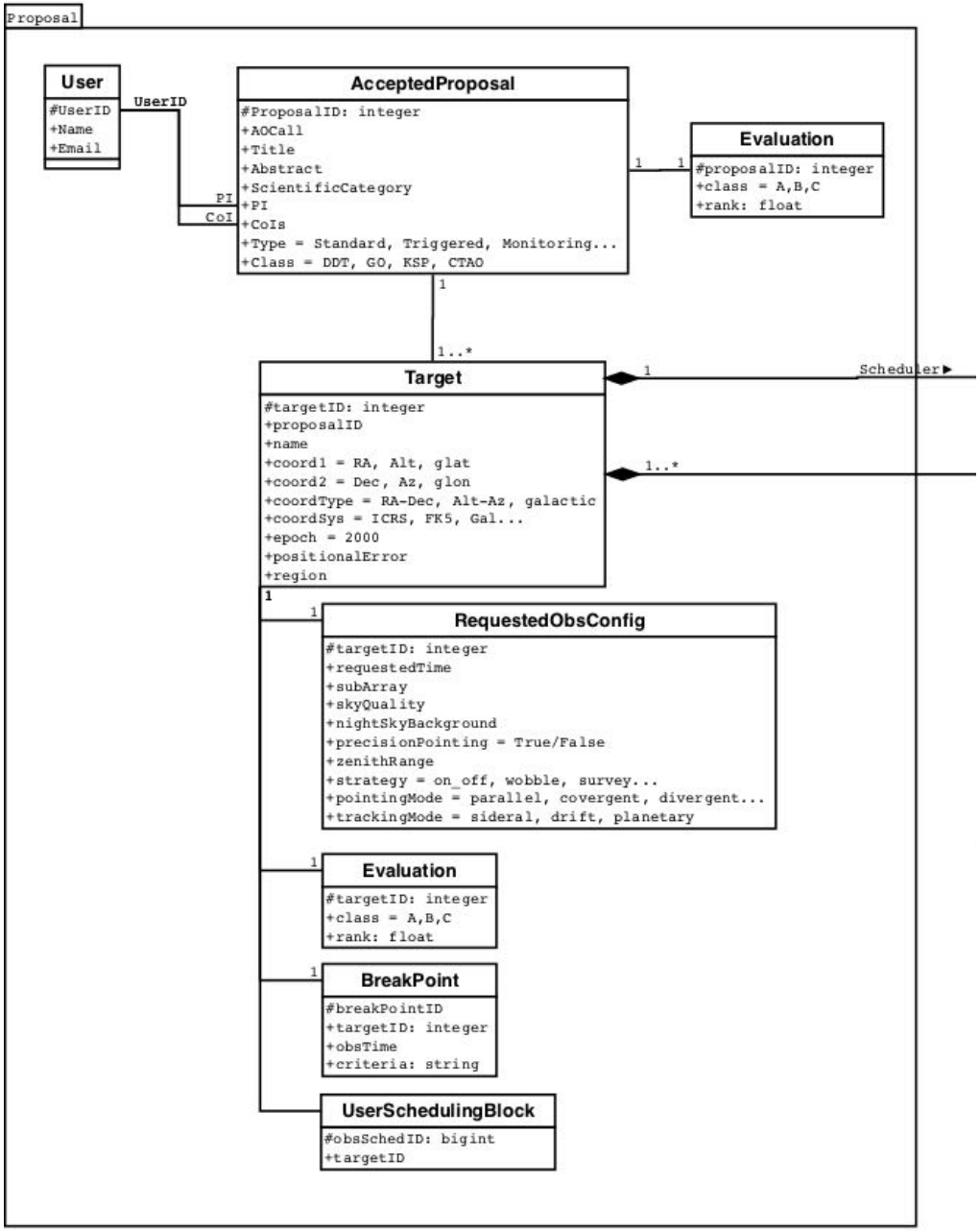
Acquisition as a stream of data

- **Scheduling Blocks (SB) definition**
  - a **unit of observation** that includes all necessary **calibration observations/procedures** for the Observatory and the Guest Observers needed for reduction and analysis. They include descriptions of configurations and calibrations.
- **Observation Block (OB) definition**
  - a part of the acquisition data stream with a **start** time, a **stop** time and a **persistent unique identifier**. An OB uses one and just one **ObsConfig** (sky position, sub-array, pointing mode, ...).
- **Time Intervals (TI) definition**
  - a part of an OB with a **start** time, a **stop** time, and **common characteristics** (slewing, calibration, high background, ...). Different TIs could require different processing or extra MC simulations.
    TIs may be defined from a **list of events** occurring during the OB, e.g.: start slewing, stop slewing, hardware failure, high trigger rate suggesting high NSB...

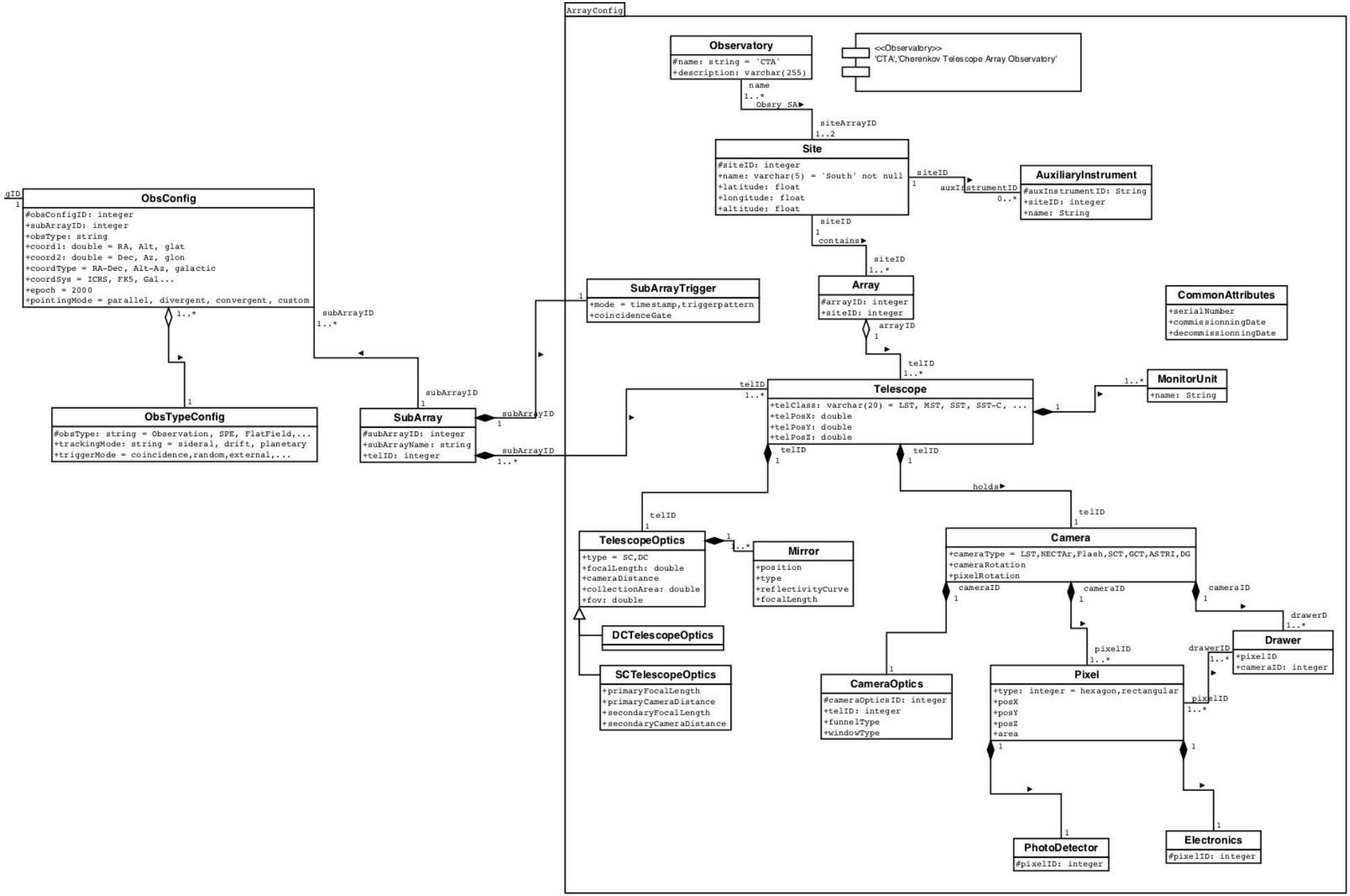
Acquisition as a stream of data



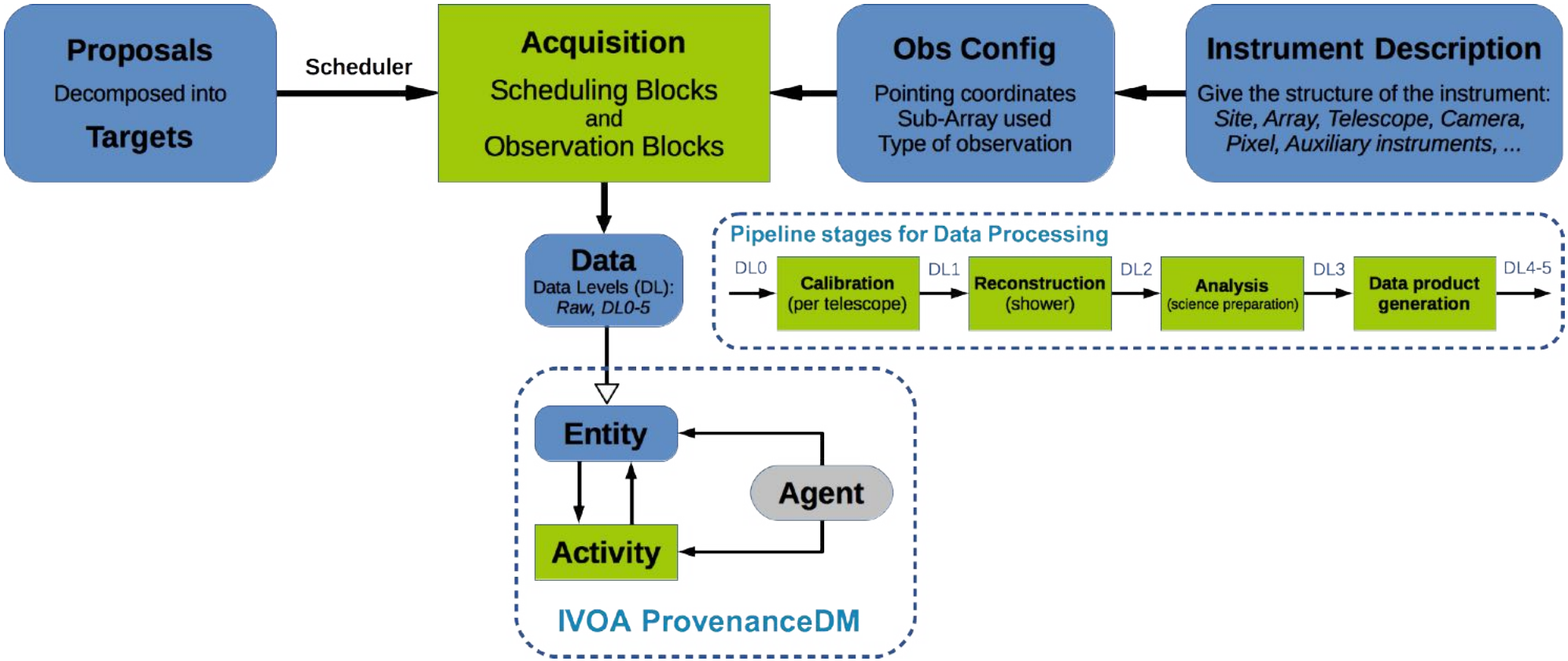
Proposal and Targets



ObsConfig and InstrumentDescription



Acquisition and Data Processing



Extended ObsCore fields for CTA

- **Optional ObsCore fields:**
  - **dataproduct_subtype**: show DL0-5?
  - **obs_release_date**
  - **data_rights** (Public/Secure/Proprietary)
  - **s_resolution min**, **s_resolution max** (as it is dependent on energy)
  - **proposal_id**
- **ObsConfig (project specific):**
  - **site**: North or South site.
  - **sub_array_name** (or directly in instrument_name)
  - **pointing_mode**: parallel, divergent, convergent, custom...
  - **obs_mode**: wobble, scan, on, off
  - **obs_type**: flatfield, science, SPE…
- **Provenance keywords (project specific):**
  - **data_quality**: flag giving information on the data quality
  - **calib_version**: version of the calibration stage of the Pipeline
  - **reco_version**: version of the reconstruction stage of the Pipeline
  - **reco_method**: reconstruction method used to obtain DL2 data
  - **applied_cuts**: selection criteria used to obtain e.g. a DL3 photon event list
  - **spectral_model**: spectral model assumed to obtain spectrum

Data mining use cases for CTA

Use case	Description
Cone Search	**Search data available for a given Target**
ObsCore search	**Search data available corresponding to ObsCore keywords** (target_name, time interval, ...), e.g.: • search data for a given **target** at a given **time** • search data in a given **region of the sky** • search data that contain events at **energy** higher than 50 TeV
ObsCore optional search	**Search data available corresponding to ObsCore optional keywords** (target_class, data_rights, ...), e.g.: • search **public** data for all **blazars** • search data for a given **proposal_id**
ObsConfig search	**Search data available corresponding to ObsConfig keywords** (sub_array_name, pointing_mode, obs_mode ...), e.g.: • search data that include the **Large Size Telescopes** (LSTs) • search data for a given target, that do not include the divergent **pointing mode**
Provenance search	**Search data available corresponding to Provenance keywords** (calib_version, creation_date ...), e.g.: • search data produced by a given **version of the pipeline** and for a given target • search data produced using a given **reconstruction method** • search data for a given target produced with **loose cuts**

CTA Data Distiller https://voparis-cta-test.obspm.fr

CTA Data Distiller | Search Form | Job List | Sign out **user**

Cone Search

Target Name	Crab Nebula	Used to query Simbad with Sesame and set RA/Dec.
Source RA (deg)	83.633	
Source Dec (deg)	22.514	
Search radius (deg)	0.001	

Submit Reset

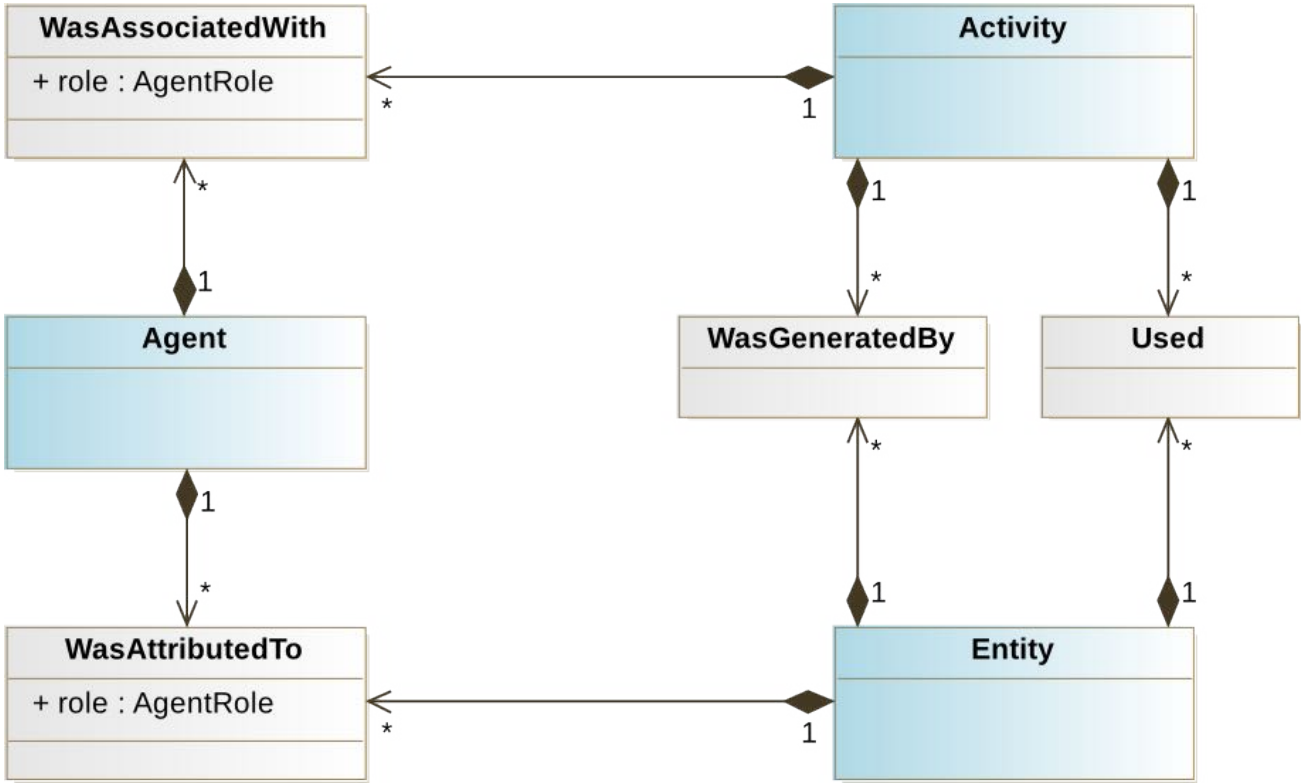
- Django, jQuery, BootStrap3
- Name resolver
  Simbad through Sesame
- Builds and Sends the ADQL query

ObsCore Search

proposal_id		Proposal ID
dataproduct_type	Nothing selected	Data product (file content) primary type
dataproduct_level	Nothing selected	DL0-5

Provenance from W3C PROV

Provenance is “information about **entities**, **activities**, and **people** involved in producing a piece of data or thing, which can be used to form assessments about its **quality**, **reliability** or **trustworthiness**”.



W3C PROV Ontology : https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

IVOA Provenance

http://www.ivoa.net/documents/ProvenanceDM/

International
Virtual
Observatory
Alliance

IVOA Provenance Data Model

Version 1.0

IVOA Working Draft 2017-02-17

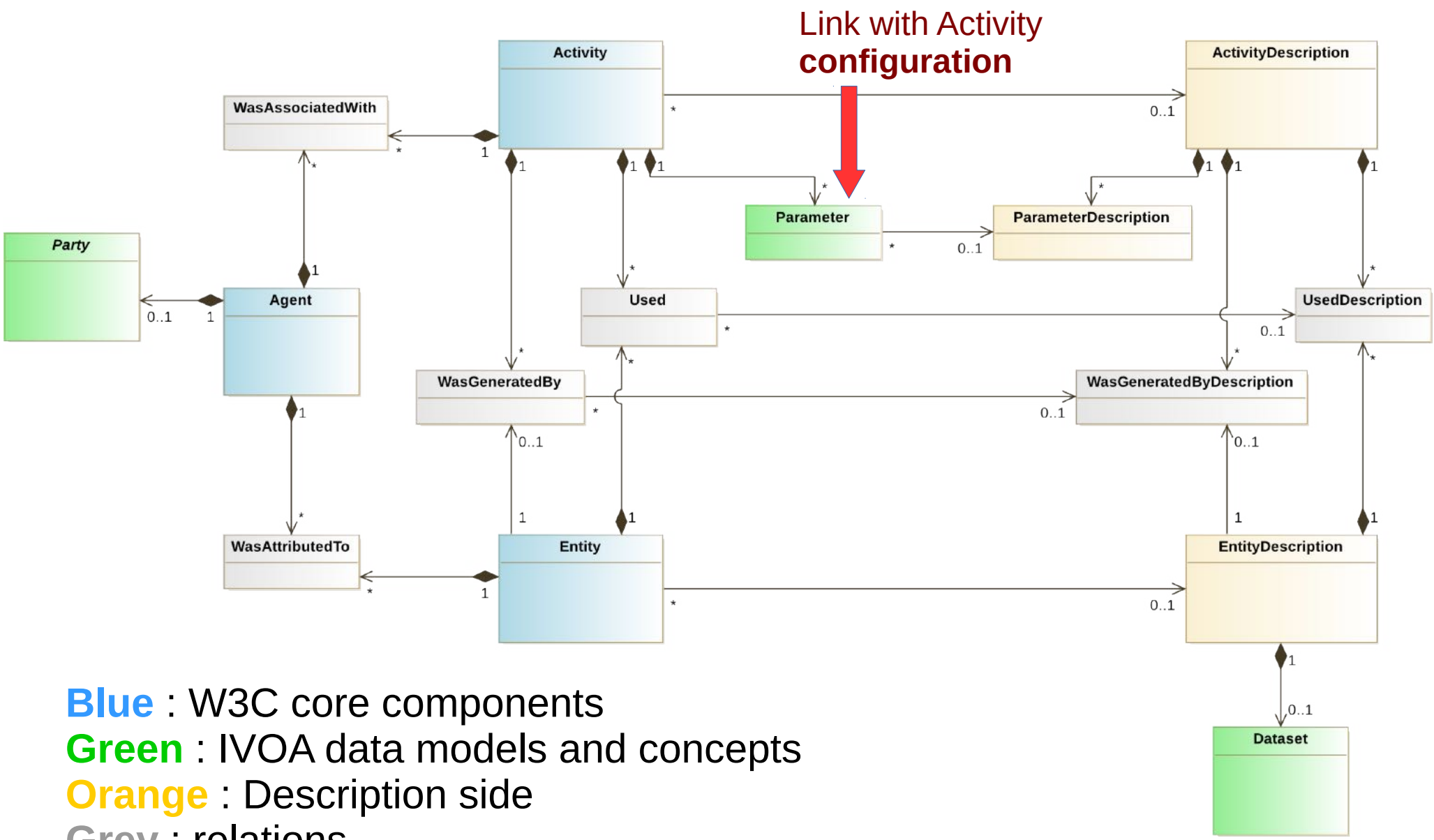
Author(s)

Kristin Riebe, Mathieu Servillat, François Bonnarel, Mireille Louys, Florian Rothmaier, Michèle Sanguillon, IVOA Data Model Working Group

Editor(s)

Kristin Riebe, Mathieu Servillat

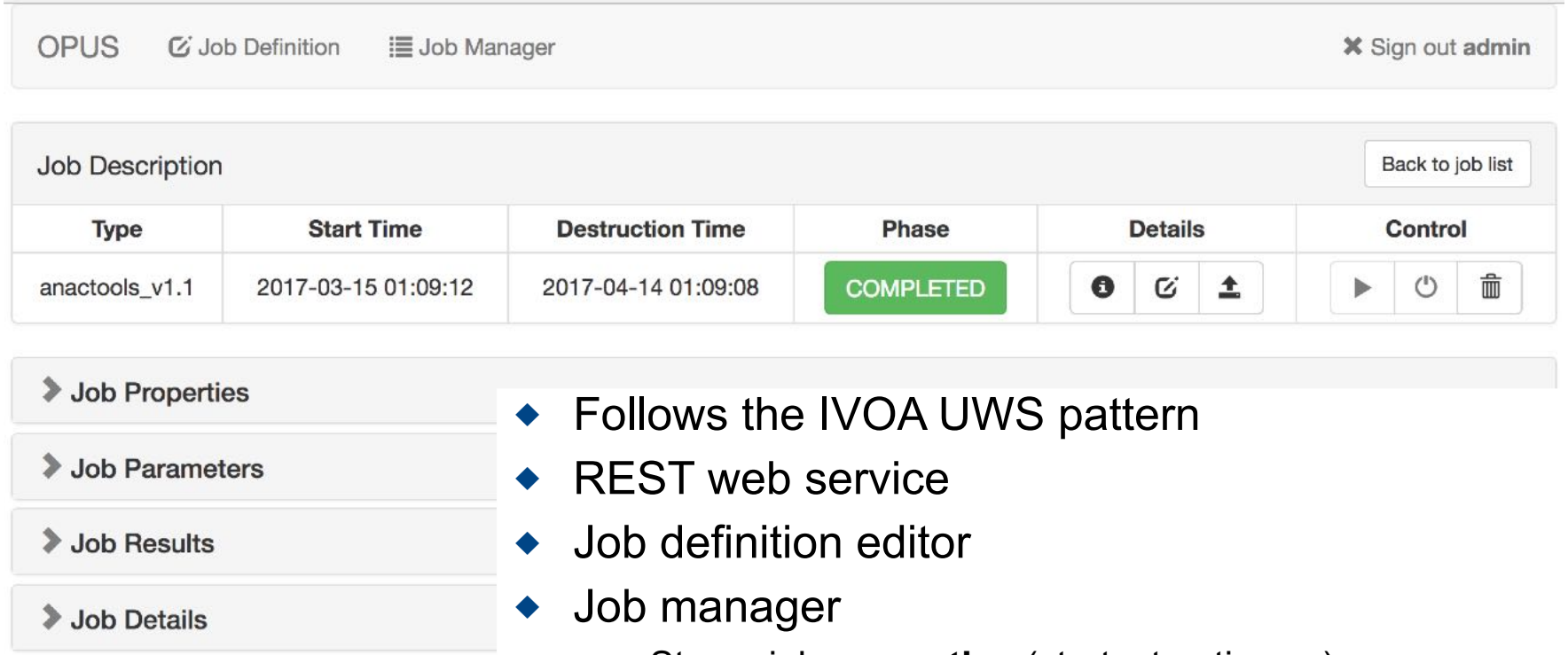
IVOA Provenance data model



Blue : W3C core components
Green : IVOA data models and concepts
Orange : Description side
Grey : relations

Example 1: analysis step with OPUS

- **OPUS** (Observatoire de Paris UWS Server) is a light job controller for the Paris Observatory work cluster developped in Python : https://github.com/ParisAstronomicalDataCentre/OPUS



- Follows the IVOA UWS pattern
- REST web service
- Job definition editor
- Job manager
  - Stores job **properties** (start, stop time…)
  - **Parameter** also kept
  - Access to **results**
  - Visualization of **logs** and **Provenance information**

Collecting Provenance information

- Using **UWS**
- Database
  - Jobs
  - Parameters
  - Results
- Need a **job description** to expose Provenance information

```
<uws:job xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0" xmlns:xli
  <uws:jobId> 3745c408-8f39-404b-9982-d5b1116ad639 </uws:jobId>
  <uws:phase> COMPLETED </uws:phase>
  <uws:executionDuration> 300 </uws:executionDuration>
  <uws:quote> 120 </uws:quote>
  <uws:error xsi:nil="true" />
  <uws:startTime> 2017-03-15T01:09:12 </uws:startTime>
  <uws:endTime> 2017-03-15T01:10:05 </uws:endTime>
  <uws:destruction> 2017-04-14T01:09:08 </uws:destruction>
  <uws:ownerId> admin </uws:ownerId>
  <uws:parameters>
    <uws:parameter byReference="false" id="anatype"> unbinned </
    <uws:parameter byReference="false" id="run_numbers"> 23523+
    <uws:parameter byReference="false" id="edisp"> true </uws:pa
  </uws:parameters>
  <uws:results>
    <uws:result id="butterfly" xlink:href="https://voparis-uws-
    <uws:result id="stdout" xlink:href="https://voparis-uws-tes
    <uws:result id="spectrum" xlink:href="https://voparis-uws-t
    <uws:result id="fit_results" xlink:href="https://voparis-uw
    <uws:result id="configfile" xlink:href="https://voparis-uws
```

http://www.ivoa.net/documents/UWS/

ActivityDescription serialization

- VOTable based on Datalink service descriptor

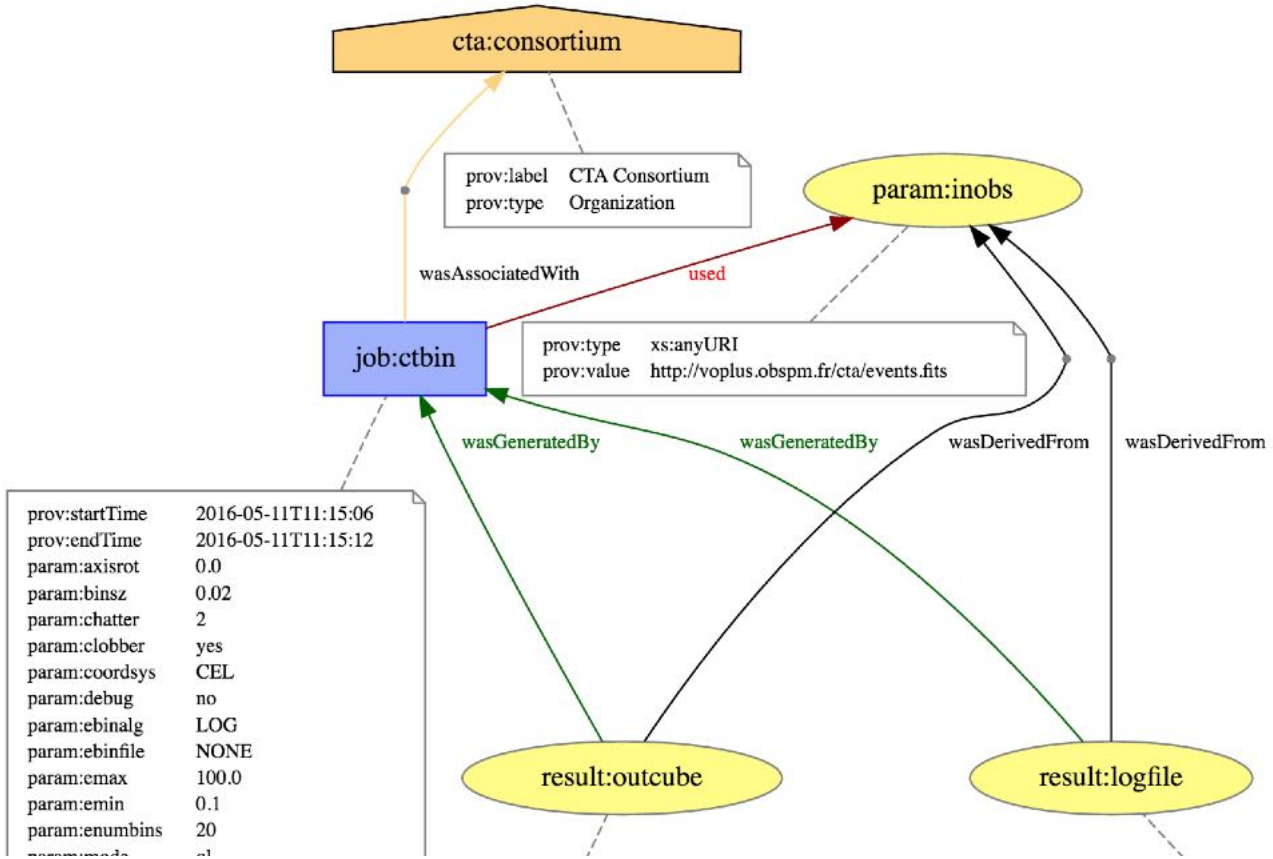
```
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.ivoa.net/xml/VOTable/v1.3" version="1.3"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.3 http://www.ivoa.net/xml/VOTable/v1.3">
 <RESOURCE ID="ctbin" name="ctbin" type="meta" utype="voprov:ActivityDescription">
   <!--  Job description  -->
  <DESCRIPTION>...</DESCRIPTION>
   <PARAM name="label" datatype="char" arraysize="*" value="CTOOLS ctbin job" utype="voprov:ActivityDescription.label"/>
   <PARAM name="type" datatype="char" arraysize="*" value="Analysis" utype="voprov:ActivityDescription.type"/>
   <PARAM name="subtype" datatype="char" arraysize="*" value="Binning" utype="voprov:ActivityDescription.subtype"/>
   <PARAM name="version" datatype="float" value="1.0" utype="voprov:ActivityDescription.version"/>
   <PARAM name="doculink" datatype="char" arraysize="*" value="http://cta.irap.omp.eu/ctools/reference_manual/ctbin.html"
   utype="voprov:ActivityDescription.doculink"/>
   <PARAM name="contact_name" datatype="char" arraysize="*" value="CTOOLS Helpdesk" utype="voprov:Agent.name"/>
   <PARAM name="contact_email" datatype="char" arraysize="*" value="ctools@irap.omp.eu" utype="voprov:Agent.email"/>
   <PARAM name="executionduration" datatype="int" value="5" utype="uws:Job.executionduration"/>
   <PARAM name="quote" datatype="int" value="5" utype="uws:Job.quote"/>
   <!--  Job parameters  -->
  <GROUP name="InputParams" utype="voprov:Parameter">
    <!--  General parameters  -->
```

- Adding information on used/generated entities

```
   <!--  Used entities  -->
  <GROUP name="Used" utype="voprov:Used">
    <PARAM name="inobs" ref="inobs" datatype="char" arraysize="*" value="" xtype="image/fits" utype="voprc
    <PARAM name="ebinfile" ref="ebinfile" datatype="char" arraysize="*" value="" xtype="plain/text" utype=
   </GROUP>
   <!--  Generated entities / UWS results  -->
  <GROUP name="Generated" utype="voprov:WasGeneratedBy">
    <PARAM name="outcube" ref="outcube" datatype="char" arraysize="*" value="" xtype="image/fits" utype="v
    <PARAM name="logfile" ref="logfile" datatype="char" arraysize="*" value="" xtype="plain/text" utype="v
   </GROUP>
```

Provides Provenance files

UWS database + Activity Description (VOTable) = Provenance Files (VOTable, XML, JSON, SVG)



prov and voprov packages

```
    from prov.model import ProvDocument
    from prov.dot import prov_to_dot

     pdoc = ProvDocument()

    # Declaring namespaces for various prefixes used in the example
    pdoc.add_namespace('prov', 'http://www.w3.org/ns/prov#')
    pdoc.add_namespace('voprov', 'http://www.ivoa.net/ns/voprov#')
    pdoc.add_namespace('cta', 'http://www.cta-observatory.org#')
    pdoc.add_namespace('uwsdata', 'https://voparis-uws-test.obspm.fr/rest/' +
                          job.jobname + '/' + job.jobid + '/')
    pdoc.add_namespace('ctajobs', 'http://www.cta-observatory.org#')

    # Adding an activity
    ctbin = pdoc.activity('ctajobs:' + job.jobname, job.start_time, job.end_time)

    # Agent
    pdoc.agent('cta:consortium', other_attributes={'prov:type': "Organization"})
    pdoc.wasAssociatedWith(ctbin, 'cta:consortium')
. . .

    pdoc.serialize(fname, format='json')
    pdoc.serialize(fname, format='xml')
    dot = prov_to_dot(pdoc)  # make the fancy diagrams as in previous slides.
```

- **prov** Follows the W3C standard
- **voprov** adds VOTable and ActivityDescription features

See on Github

Output files (PROV-XML and PROV-JSON)

```
<prov:document xmlns:ctadata="ivo://vopdc.obspm/cta#" xmlns:ctajob
  <prov:activity prov:id="ctajobs:ctbin">
    <prov:startTime> 2016-03-13T23:44:46 </prov:startTime>
    <prov:endTime> 2016-03-13T23:44:56 </prov:endTime>
  </prov:activity>
  <prov:agent prov:id="cta:consortium">
    <prov:type xsi:type="xsd:string"> Organization </prov:type>
  </prov:agent>
  <prov:wasAssociatedWith>
    <prov:activity prov:ref="ctajobs:ctbin" />
    <prov:agent prov:ref="cta:consortium" />
  </prov:wasAssociatedWith>
  <prov:entity prov:id="uwsdata:parameters/inobs" />
  <prov:used>
    <prov:activity prov:ref="ctajobs:ctbin" />
    <prov:entity prov:ref="uwsdata:parameters/inobs" />
  </prov:used>
  <prov:entity prov:id="uwsdata:results/outcube" />
  <prov:wasGeneratedBy>
    <prov:entity prov:ref="uwsdata:results/outcube" />
    <prov:activity prov:ref="ctajobs:ctbin" />
  </prov:wasGeneratedBy>
  <prov:wasDerivedFrom>
    <prov:generatedEntity prov:ref="uwsdata:results/outcube" />
    <prov:usedEntity prov:ref="uwsdata:parameters/inobs" />
  </prov:wasDerivedFrom>
  <prov:entity prov:id="uwsdata:results/logfile" />
  <prov:wasGeneratedBy>
    <prov:entity prov:ref="uwsdata:results/logfile" />
    <prov:activity prov:ref="ctajobs:ctbin" />
  </prov:wasGeneratedBy>
  <prov:wasDerivedFrom>
    <prov:generatedEntity prov:ref="uwsdata:results/logfile" />
    <prov:usedEntity prov:ref="uwsdata:parameters/inobs" />
  </prov:wasDerivedFrom>
</prov:document>
```

```
{
  - wasAssociatedWith: {
      - _:id1: {
            prov:agent: "cta:consortium",
            prov:activity: "cta:anactools_v1.1"
        }
    },
  - agent: {
      - cta:consortium: {
            prov:type: "Organization"
        }
    },
  - entity: {
        uwsdata:results/fit_results: { },
        uwsdata:results/configfile: { },
        uwsdata:results/butterfly: { },
        uwsdata:results/spectrum_plot: { },
        uwsdata:results/spectrum: { }
    },
  - prefix: {
        uwsdata: "https://voparis-uws-test.obspm.fr/rest
        cta: "http://www.cta-observatory.org#",
        voprov: "http://www.ivoa.net/ns/voprov#"
    },
  - activity: {
      - cta:anactools_v1.1: {
            prov:startTime: "2016-04-07T00:26:00",
            prov:endTime: "2016-04-07T00:27:15"
        }
    },
  - wasGeneratedBy: {
      - _:id5: {
            prov:entity: "uwsdata:results/butterfly",
            prov:activity: "cta:anactools_v1.1"
        },
      - _:id4: {
            prov:entity: "uwsdata:results/fit_results",
            prov:activity: "cta:anactools_v1.1"
        },
```

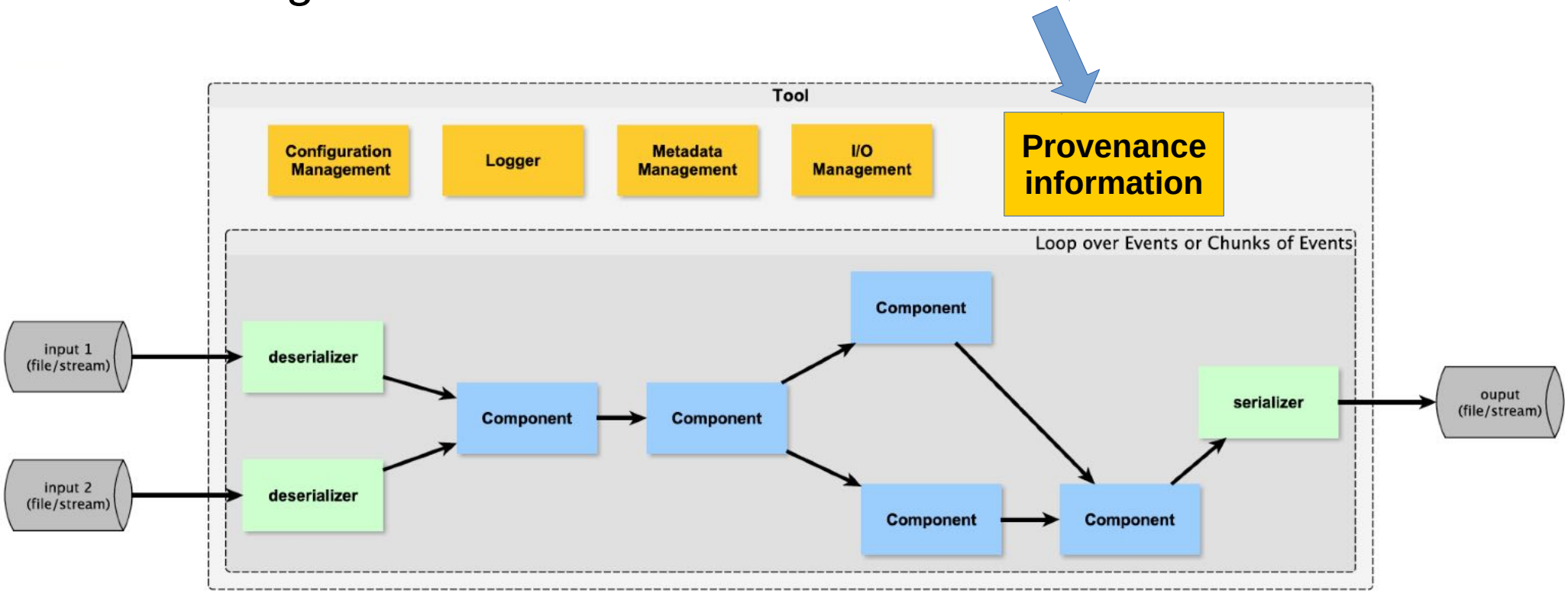
VOTable serialization

```
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.ivoa.net/xml/VOTable/v1.2" version="1.2"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.2 http://www.ivoa.net/xml/VOTable/v1.2">
 <RESOURCE name="Stage1">
  <TABLE name="activities" utype="prov:activity">
    <FIELD name="name" utype="prov:activity.name" datatype="char" arraysize="*"/>
    <FIELD name="start" utype="prov:startTime" datatype="char" arraysize="*" xtype="ISO8601"/>
    <FIELD name="stop" utype="prov:endTime" datatype="char" arraysize="*" xtype="ISO8601"/>
    <FIELD name="methodname" utype="voprov:method_name" dataype="char" arraysize="*"/>
    <FIELD name="version" utype="voprov:method_version" datatype="char" arraysize="*"/>
   <DATA>
    <TABLEDATA>
     <TR>
      <TD>cta:telescope_stage_520</TD>
      <TD>2015-07-30T09:45:00</TD>
      <TD>2015-07-30T10:00:00</TD>
      <TD>Telescope_stage</TD>
      <TD>1.0</TD>
     </TR>
    </TABLEDATA>
   </DATA>
  </TABLE>
  <TABLE name="entities" utype="prov:entity">
    <FIELD name="name" utype="prov:entitity.name" datatype="char" arraysize="*"/>
    <FIELD name="label" utype="prov:label" datatype="char" arraysize="*"/>
    <FIELD name="type" utype="prov:type" datatype="char" arraysize="*"/>
    <FIELD name="run" utype="cta:runNumber" datatype="int"/>
    <FIELD name="tel" utype="cta:telescope" datatype="char" arraysize="*"/>
   <DATA>...</DATA>
  </TABLE>
  <TABLE name="usedRelationship" utype="voprov:used">
    <FIELD name="head" datatype="char" arraysize="*"/>
    <FIELD name="tail" datatype="char" arraysize="*"/>
   <DATA>...</DATA>
  </TABLE>
  <TABLE name="wasGeneratedByRelationship" utype="voprov:wasGeneratedBy">
    <FIELD name="head" datatype="char" arraysize="*"/>
    <FIELD name="tail" datatype="char" arraysize="*"/>
   <DATA>...</DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

Example 2: CTA Pipeline

cherenkov telescope array

- **Ctapipe**: a CTA data processing framework (prototype, not official, not recommended for use!) https://github.com/cta-observatory/ctapipe
- **Tool Python class** providing configuration, logger, I/O management… and **Provenance information**



Provenance class for ctapipe

```
from ctapipe.core import Provenance

prov = Provenance()
 # prov a singleton, so this gives you the same provenance class

prov.start_activitity("some_activity")

... # do things
prov.add_input_file("test.txt")
prov.add_output_file("out.txt")

prov.start_activity("some_sub_activity")

# do more things
prov.add_output_file("out2.txt")

prov.finish_activity() # finish some_activity
prov.finish_activity() # finish some_sub_activity
```

- Importance of **persistent identifiers**
- Also records **system configuration**, **state**, and **software versions**

Manipulating Provenance

Storing Provenance:

- Write to files 
- Store with data product (header, fits-plus...) 
- Store in a database (using data model) 

Retrieving Provenance:

- Request Provenance path
  - From files 
  - From database (API) 
- Search data products based on Provenance 
  - A given Activity was performed (with given version)...
  - A given input parameter was set to...

Next steps

- High level data model to be completed
  - Interactions with CTA working groups
- Use ProvenanceDM to define a database
  - UWS pattern
  - Project specific
- I/O package for this database
  - Using descriptions: activity/data/parameters
  - Based on prov: voprov
- Will be included in the CTA framework
  - ctapipe project in Python
  - Fills the Provenance info from DL0 to DL3