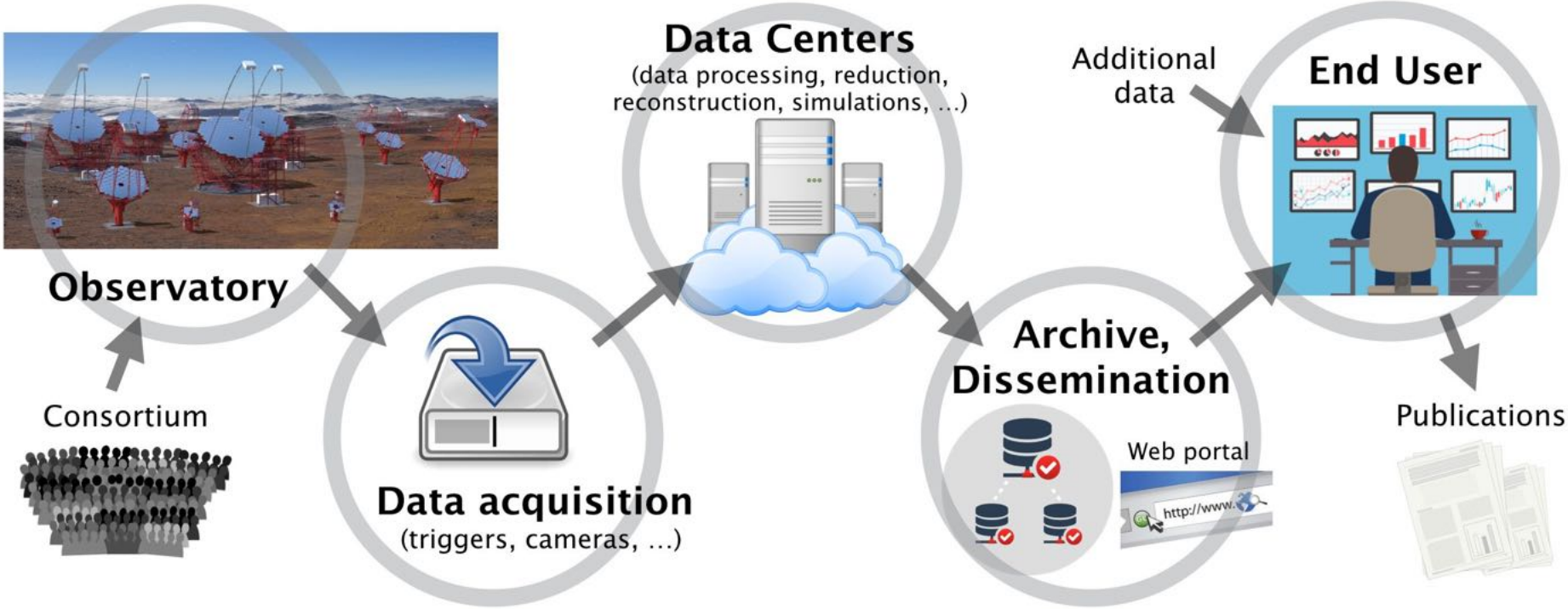
Implementations of the IVOA Provenance Data Model

Mathieu Servillat, Catherine Boisson,
François Bonnarel, Mireille Louys, Michèle Sanguillon

Need for homogenous, structured provenance



FAIR principles for data sharing

Findable

F1. (Meta)data are assigned a globally unique and persistent identifier
F2. Data are described with rich metadata
F3. Metadata clearly and explicitly include the identifier of the data they describe
F4. (Meta)data are registered or indexed in a searchable resource

Accessible

A1. (Meta)data are retrievable by their identifier using a standardised communications protocol
A1.1. The protocol is open, free, and universally implementable
A1.2. The protocol allows for an authentication and authorisation procedure, where necessary
A2. Metadata are accessible, even when the data are no longer available

Interoperable

I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
I2. (Meta)data use vocabularies that follow FAIR principles
I3. (Meta)data include qualified references to other (meta)data

Reusable (+ Reproducible?)

R1. Meta(data) are richly described with a plurality of accurate and relevant attributes
R1.1. (Meta)data are released with a clear and accessible data usage license
R1.2. (Meta)data are associated with detailed provenance
R1.3. (Meta)data meet domain-relevant community standards

https://www.go-fair.org/fair-principles/

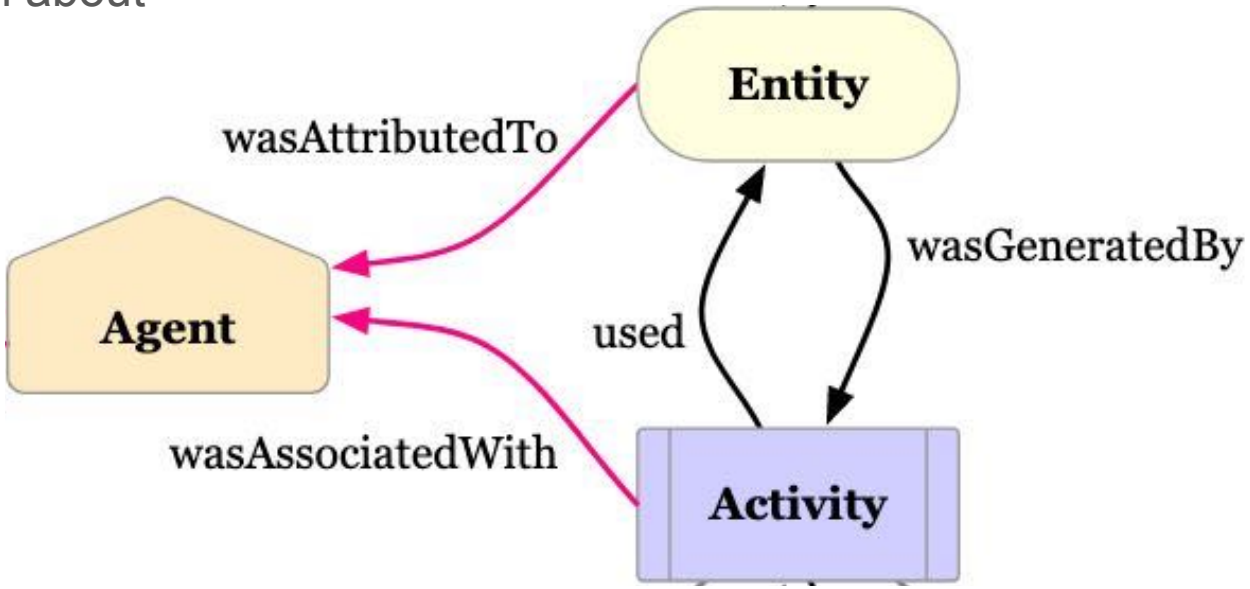
http://www.w3.org/TR/prov-overview/

W3C Provenance definition

W3C PROV (PROV-DM, 2013)

Word Wide Web Consortium

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

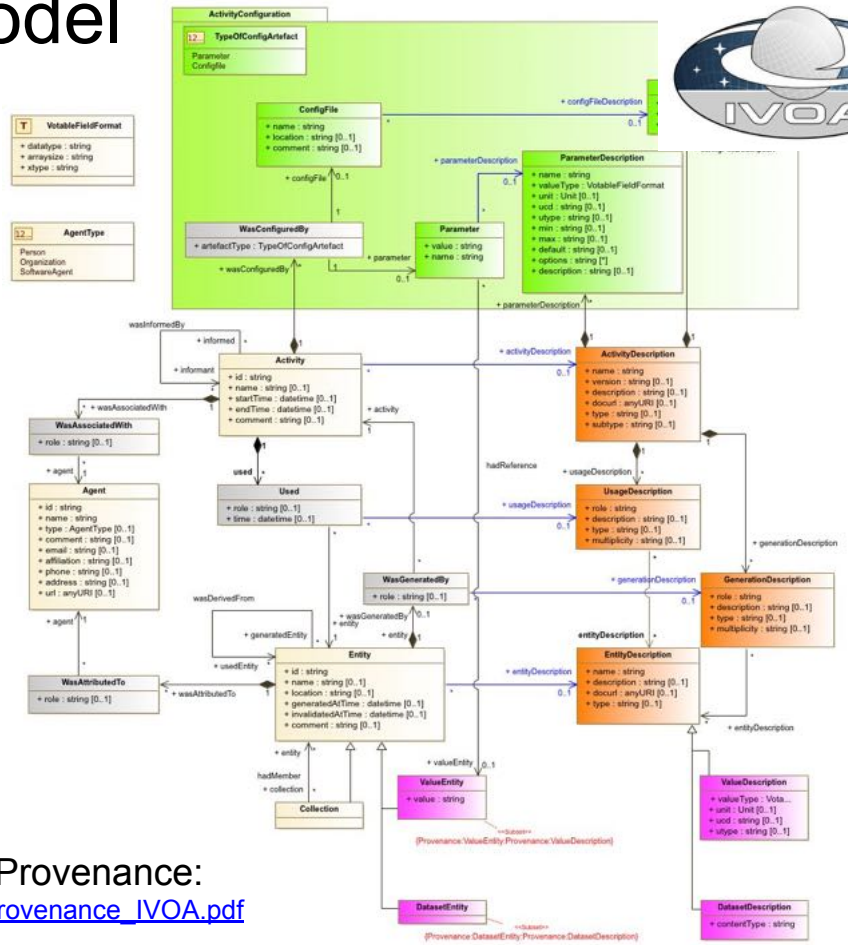


http://www.ivoa.net/documents/ProvenanceDM/

IVOA Provenance Data Model

Recommendation 2020 !

- Adds “Description” classes
- Adds “Configuration” classes
- Plugged in with
  - VO data models
    (UCD, VOUnit, VOTable…)
  - VO access protocols
    (ProvTAP, ProvSAP)
- Serializations
  - W3C PROV
  - VO specific



See a general presentation of the IVOA Data Model for Provenance:
https://wiki.ivoa.net/internal/IVOA/InterOpMay2019DM/2019-05-15_servillat_provenance_IVOA.pdf

Usage/Generation

- **role** (master_bias, IRF, eventlist, ...)
- **type** (main, calibration, preview, quality, log, context)

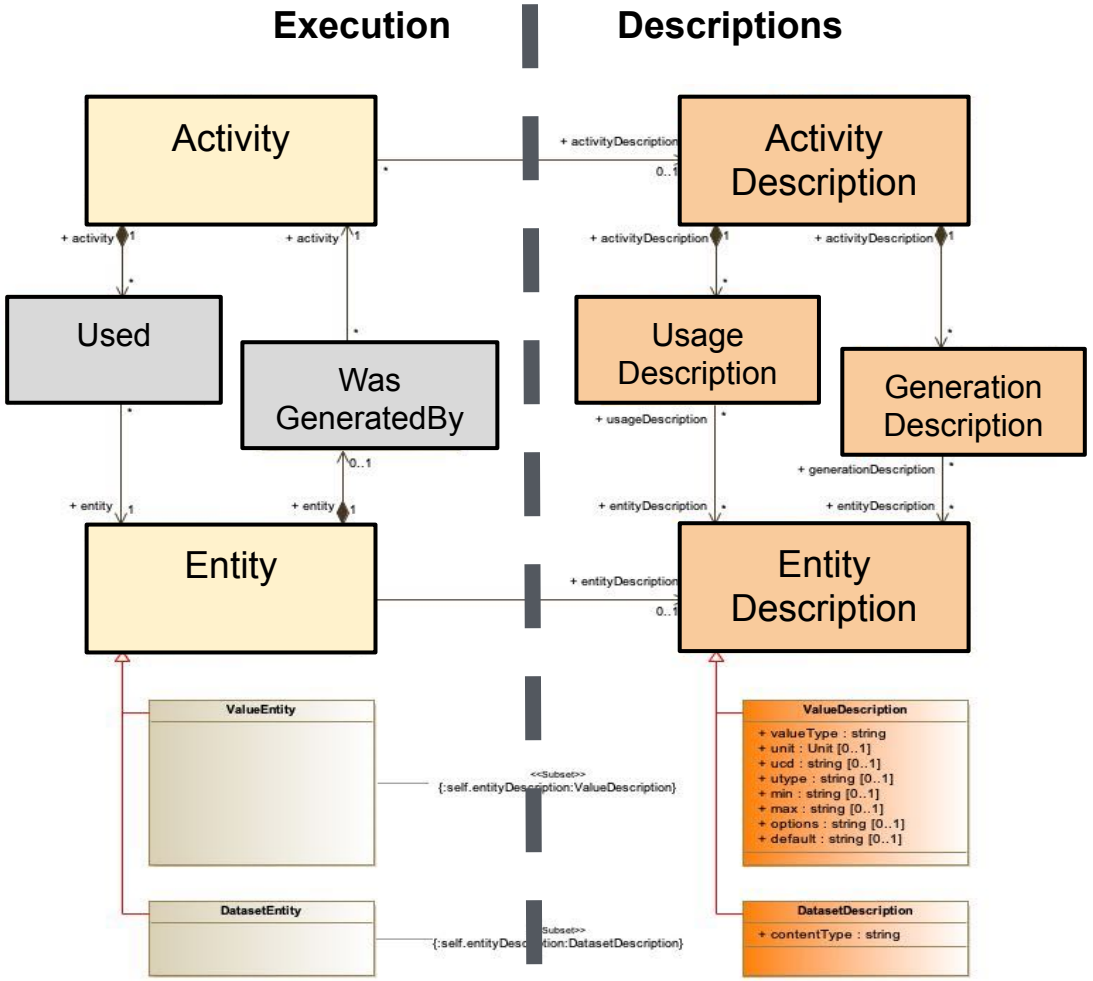
DatasetEntity

- **contentType** (similar to access_format in ObsCore)

ValueEntity

- **valueType**
- unit/ucd/utype
- min/max/options

+ Descriptions



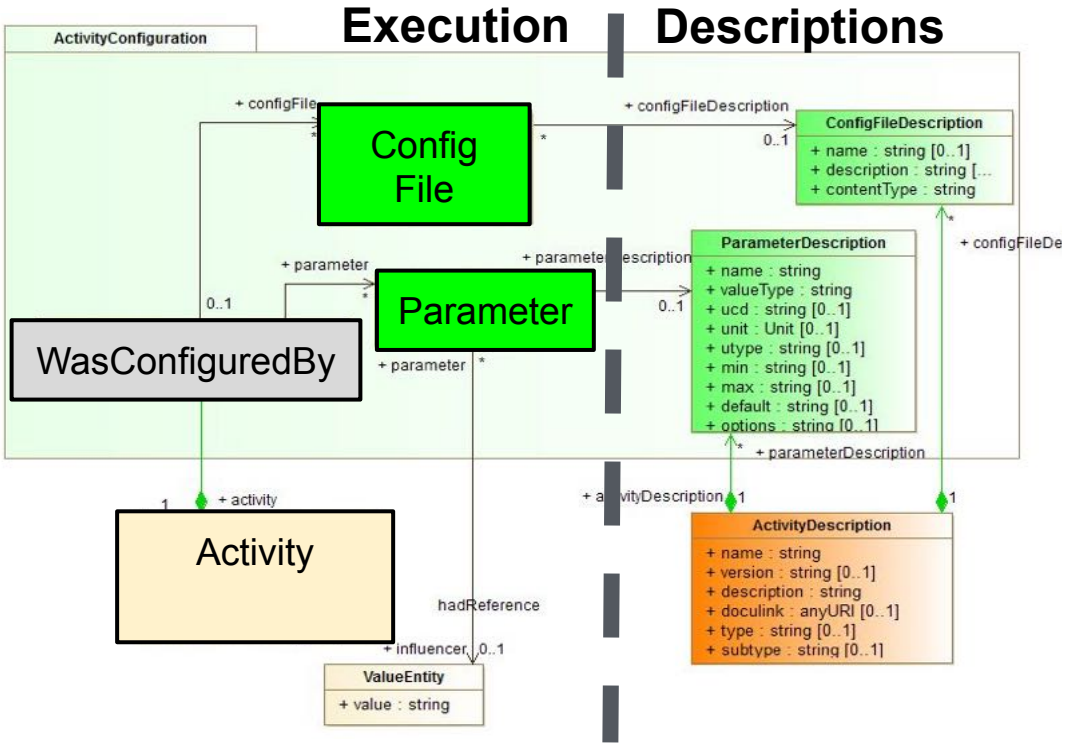
- Identify **how** a data product was produced ⇒ **Provenance (skeleton)**
- Identify what **detailed options** were used ⇒ **Configuration (flesh)**

WasConfiguredBy

- Parameter
  - name = **value**
  - no identifier (part of activity)
- ConfigFile
  - name & location
- + Descriptions

Configuration is **dependent** on Activity/ActivityDescription

⇒ **fosters reproducibility**



Applying the model

Different context in use cases

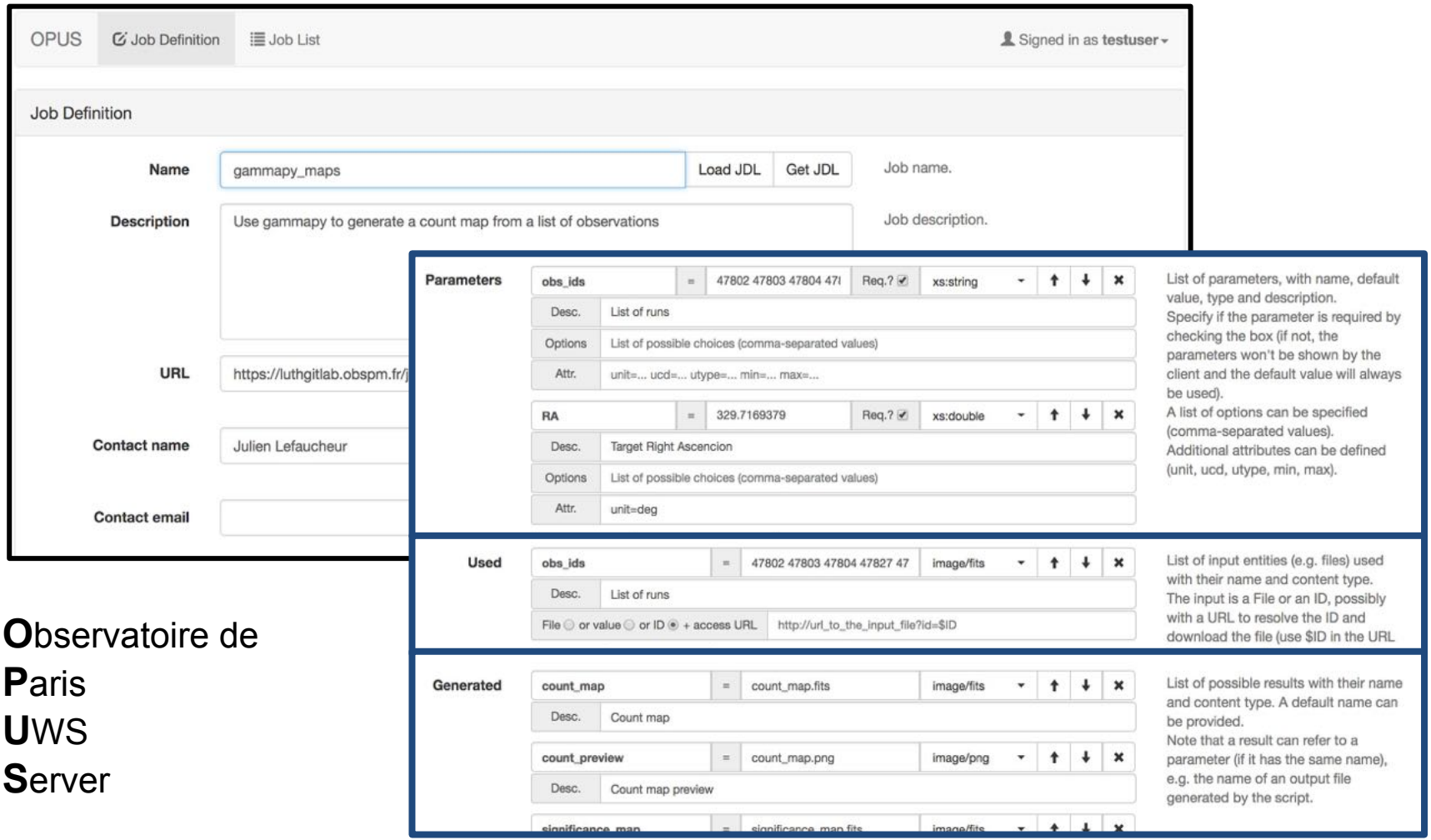
- Provenance **on-top** or **inside**
- Granularity
- Level of details
- Identifiers

Different steps in provenance management

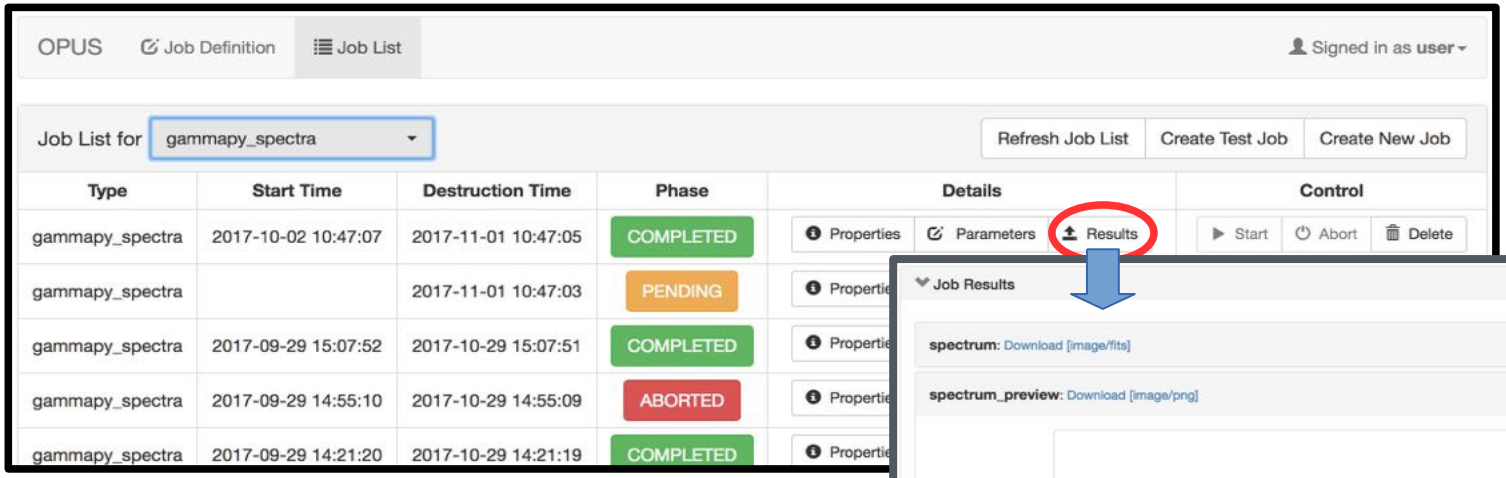
- How to **capture** the provenance information
- How to **store** this information
- How to **access**
- How to **visualize** the provenance

Examples of implementations

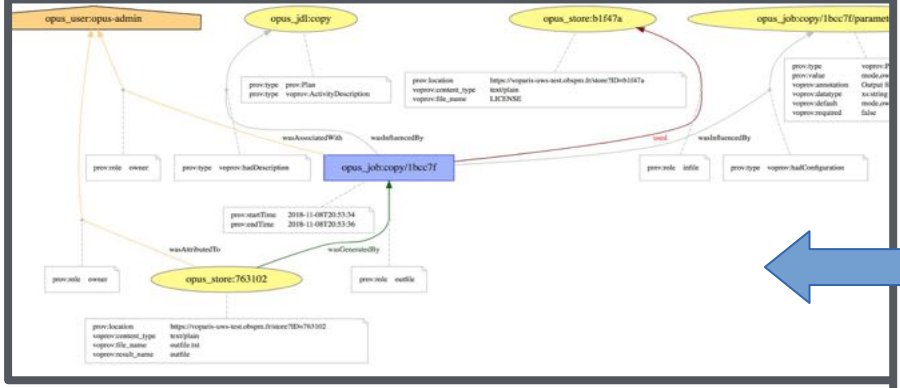
- **Capture**
  - **OPUS**: 1 = 1 job, returns W3C files and graphs
  - **`ctapipe`**: 1 activity = 1 Tool, returns a dictionary (JSON)
  - **`gammapy`**: 1 activity = 1 high level interface function, returns a structured log
  - **OPUS + `gammapy`**: granularity mix ! transfer of identifiers !
- **Storage**
  - **Mostly W3C files for now**
  - **OPUS :** UWS job + entity store
  - **DIRAC + `ctapipe` :** ProvDB (sqlachemy and ingest scripts)
  - ProvStore / Triple Store
- **Access**
  - **OPUS & Pollux : ProvSAP** - application specific, extract a graph (W3C compatibility)
  - **ProvHiPS: ProvTAP** - easy to deploy, complex queries, VOTable output
- **Visualization**
  - **Only W3C graphs for now...**

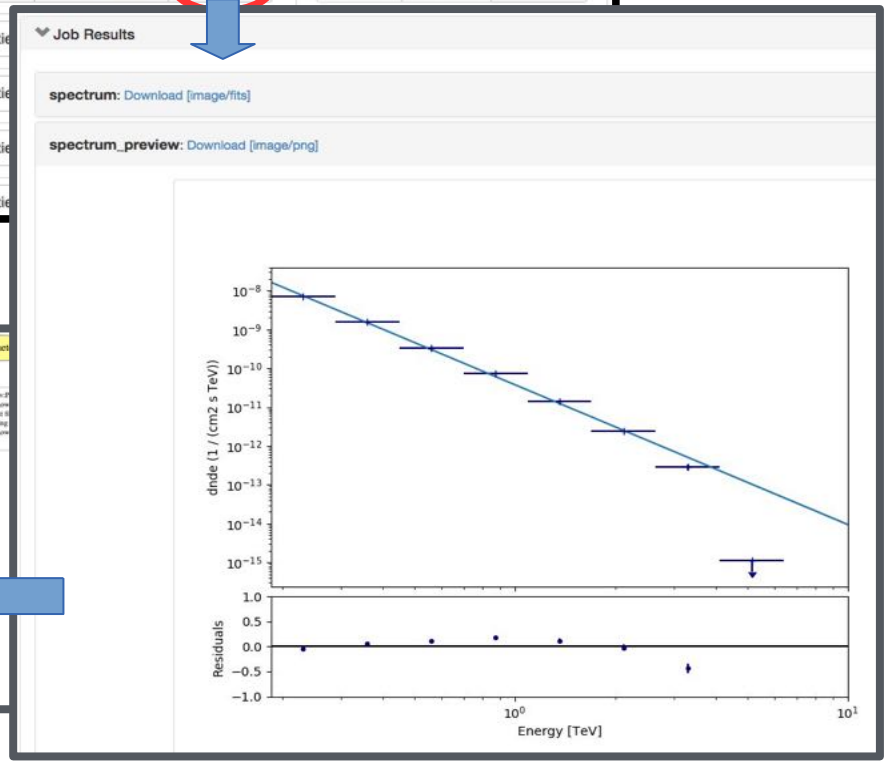


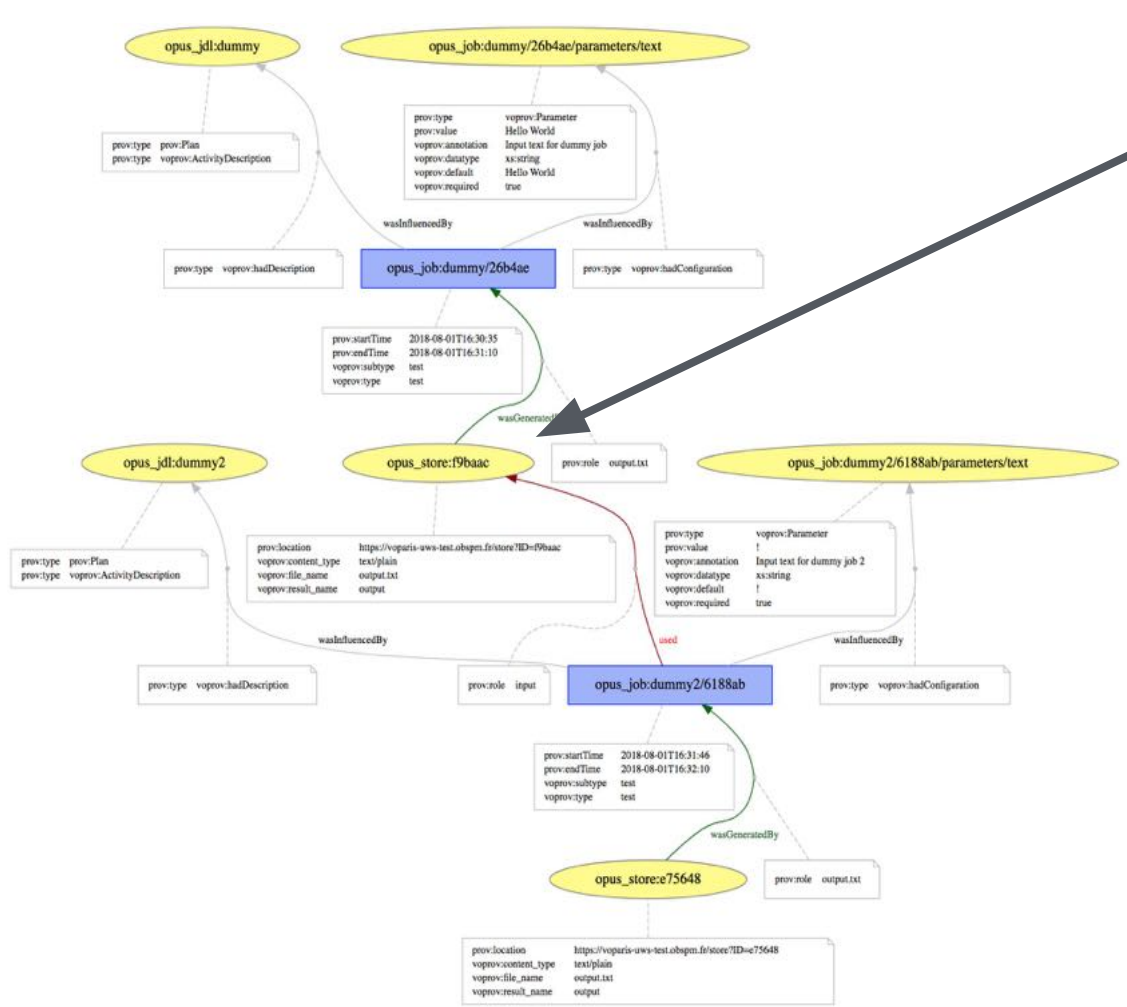
Observatoire de
Paris
UWS
Server



Tracking of Provenance informations







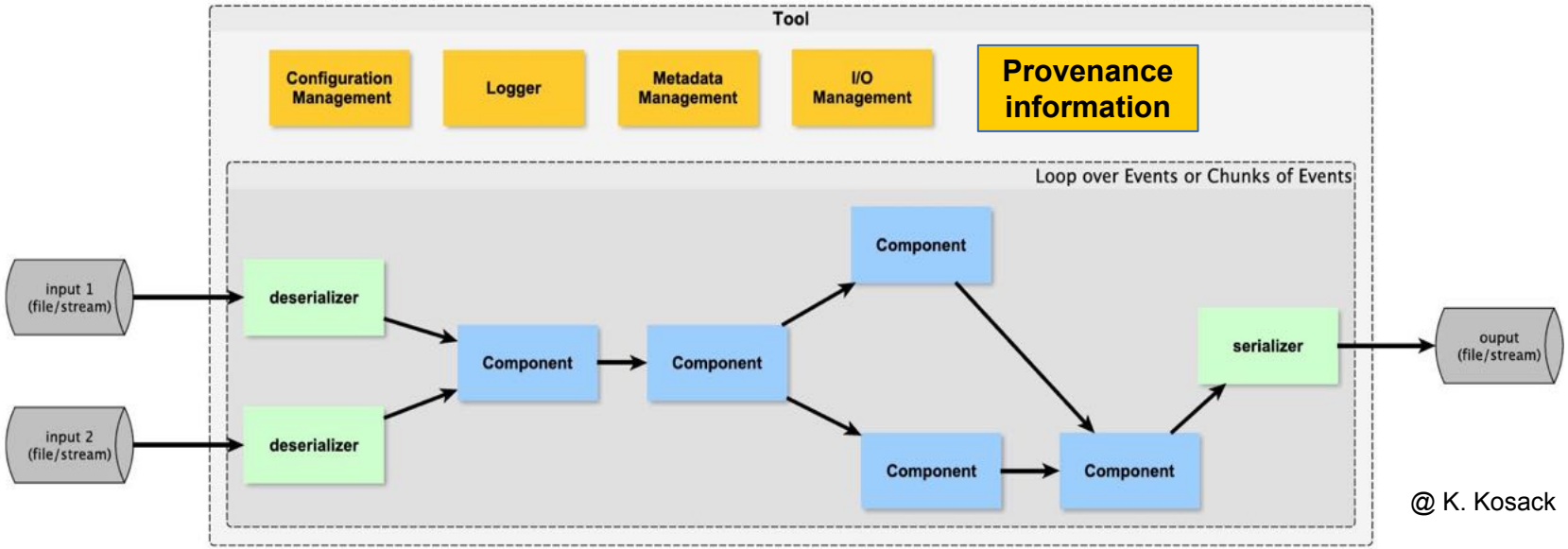
- Register all entities
  - unknown input data
  - results

  → **id** + **hash** + **name**

- Check if already exist in the system
  - **name**?
    - may change...
  - **hash**?
    - integrity
    - unicity?
  - **id**?
    - inside the files
    - prov database

Provenance in `ctapipe`

- **Tool** Python class providing configuration, logger, I/O management… and Provenance information
  https://cta-observatory.github.io/ctapipe/examples/provenance.html



@ K. Kosack

Provenance in `ctapipe`

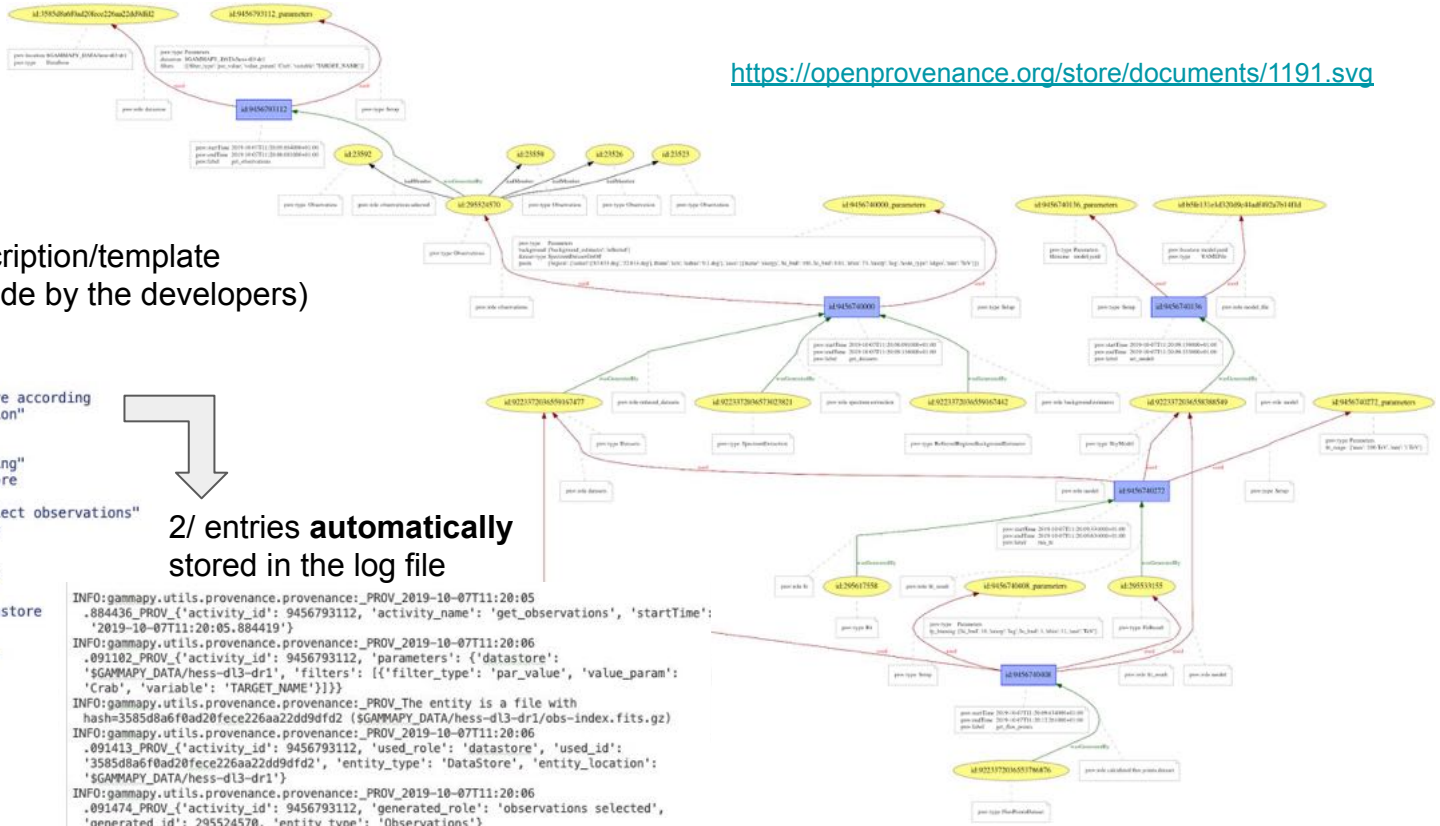
- Integrated to the **framework** (almost transparent to the users/developers)
- Tracks start and end of a **Tool** execution
- Records the entities (files) used (input) and generated (output)
- Also records system configuration, state, and software versions
  → contextual information

```python
from ctapipe.core import Provenance

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

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
```

- **However**:
  - returns a dictionary at the end of the process (not accessible before)
  - tends to use 1 big activity and structured set of parameters

Provenance in `gammapy`

https://openprovenance.org/store/documents/1191.svg



1/ definition.yaml file for description/template
(already **integrated** to the code by the developers)

```
activities:

    get_observations:
        description:
            "Fetch observations from the data store according
            to criteria defined in the configuration"
        parameters:
            - name: datastore
              description: "DataStore path as string"
              value: settings.observations.datastore
            - name: filters
              description: "Filter criteria to select observations"
              value: settings.observations.filters
        usage:
            - role: datastore
              description: "DataStore object file"
              entityType: DataStore
              location: settings.observations.datastore
        generation:
            - role: observations selected
              description: "Observations selected"
              entityType: Observations
              value: observations
              has_members:
                  entityType: Observation
                  list: observations.list
                  id: obs_id
                  namespace: ""

    get_datasets:
        description: "Produce reduced datasets"
        parameters:
```

2/ entries **automatically** stored in the log file

```
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:05
 .884436_PROV_{'activity_id': 9456793112, 'activity_name': 'get_observations', 'startTime':
  '2019-10-07T11:20:05.884419'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091102_PROV_{'activity_id': 9456793112, 'parameters': {'datastore':
 '$GAMMAPY_DATA/hess-dl3-dr1', 'filters': [{'filter_type': 'par_value', 'value_param':
 'Crab', 'variable': 'TARGET_NAME'}]}}
INFO:gammapy.utils.provenance.provenance:_PROV_The entity is a file with
  hash=3585d8a6f0ad20fece226aa22dd9dfd2 ($GAMMAPY_DATA/hess-dl3-dr1/obs-index.fits.gz)
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091413_PROV_{'activity_id': 9456793112, 'used_role': 'datastore', 'used_id':
 '3585d8a6f0ad20fece226aa22dd9dfd2', 'entity_type': 'DataStore', 'entity_location':
 '$GAMMAPY_DATA/hess-dl3-dr1'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091474_PROV_{'activity_id': 9456793112, 'generated_role': 'observations selected',
 'generated_id': 295524570, 'entity_type': 'Observations'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091527_PROV_{'entity_id': 295524570, 'member_id': 23592, 'member_type': 'Observation'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091571_PROV_{'entity_id': 295524570, 'member_id': 23523, 'member_type': 'Observation'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091613_PROV_{'entity_id': 295524570, 'member_id': 23526, 'member_type': 'Observation'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091653_PROV_{'entity_id': 295524570, 'member_id': 23559, 'member_type': 'Observation'}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06
 .091691_PROV_{'activity_id': 9456793112, 'endTime': '2019-10-07T11:20:06.091068'}
```

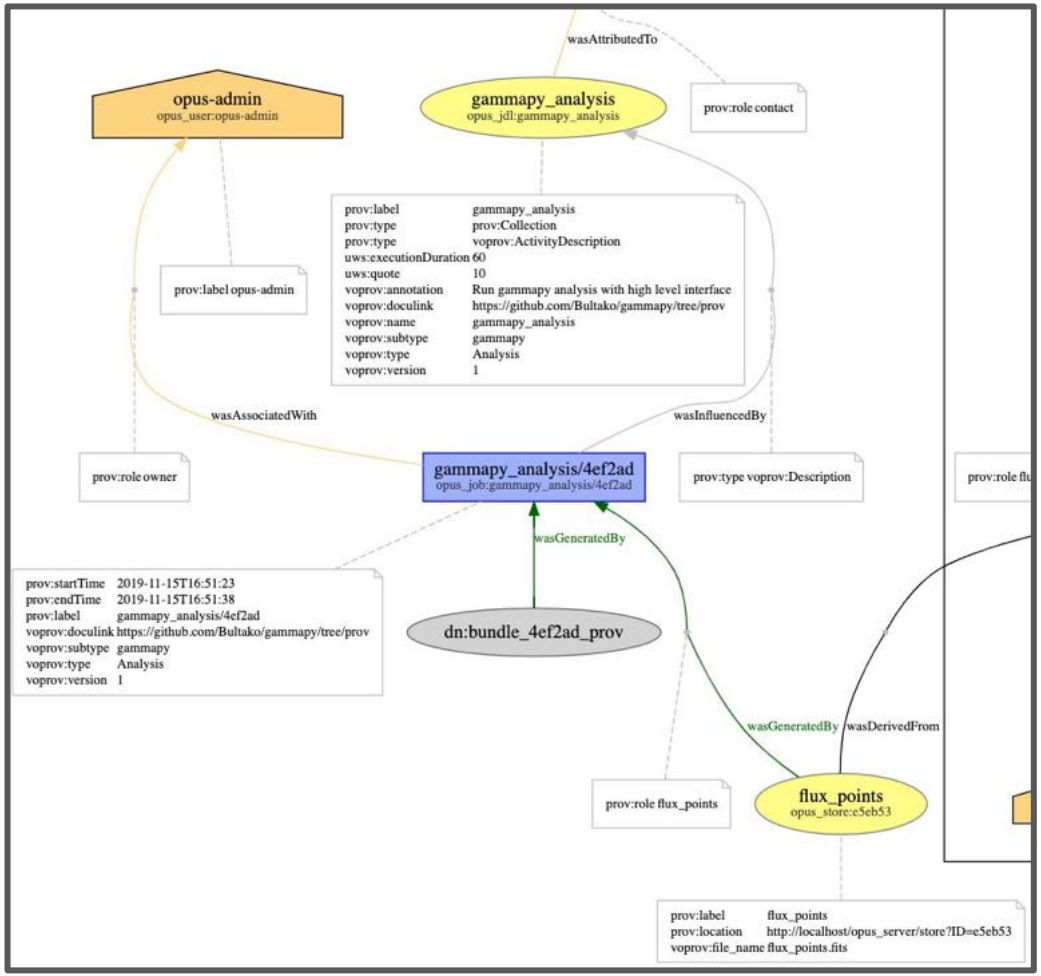
3/ export to W3C PROV
or search in provenance records

@ M. Servillat & J. E. Ruiz

OPUS + `gammapy`

- 1 OPUS job
  - Runs several `gammapy` functions
  - Stores result entities

- Internal provenance
  - Store objects
  - **Link to OPUS job**
    - Sub-activities ?
    - W3C PROV Bundle ?
  - **link to result**
    - Stored in OPUS archive
    - Derivation ?
    - Copy ?



ProvDB storage: DIRAC + `ctapipe`

- Capture done by `ctapipe` while executing a job
- Returns a JSON dictionary
- Ingested by DIRAC on a dedicated PostgreSQL server
  - https://github.com/cta-observatory/CTADIRAC

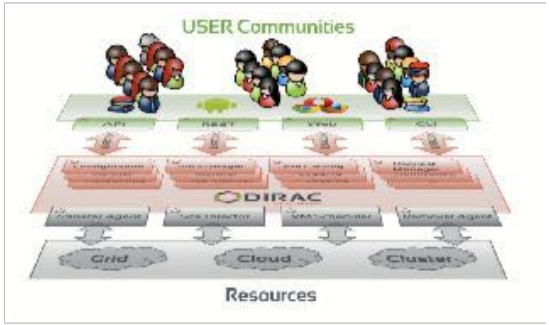
```
provBase = declarative_base()

# Define the Activity class mapped to the activities table
class Activity(provBase):
    __tablename__ = 'activities'
    ordered_attribute_list = ['id','name','startTime','endTime','comment','activityDescription_id']
    id        = Column(String, primary_key=True)
    name      = Column(String)
    startTime = Column(String)
    endTime   = Column(String)
    comment   = Column(String)
    activityDescription_id = Column(String, ForeignKey("activityDescriptions.id"))
    activityDescription    = relationship("ActivityDescription")
```

@ M. Sanguillon, L. Arrabito, M. Servillat

Intégration de la provenance dans CTADIRAC

- Contexte :
  - pipeline DL0-DL3, qui utilise les outils de la bibliothèque ctapipe (dont le module Provenance qui capture les informations de provenance - K. Kosack - CEA)

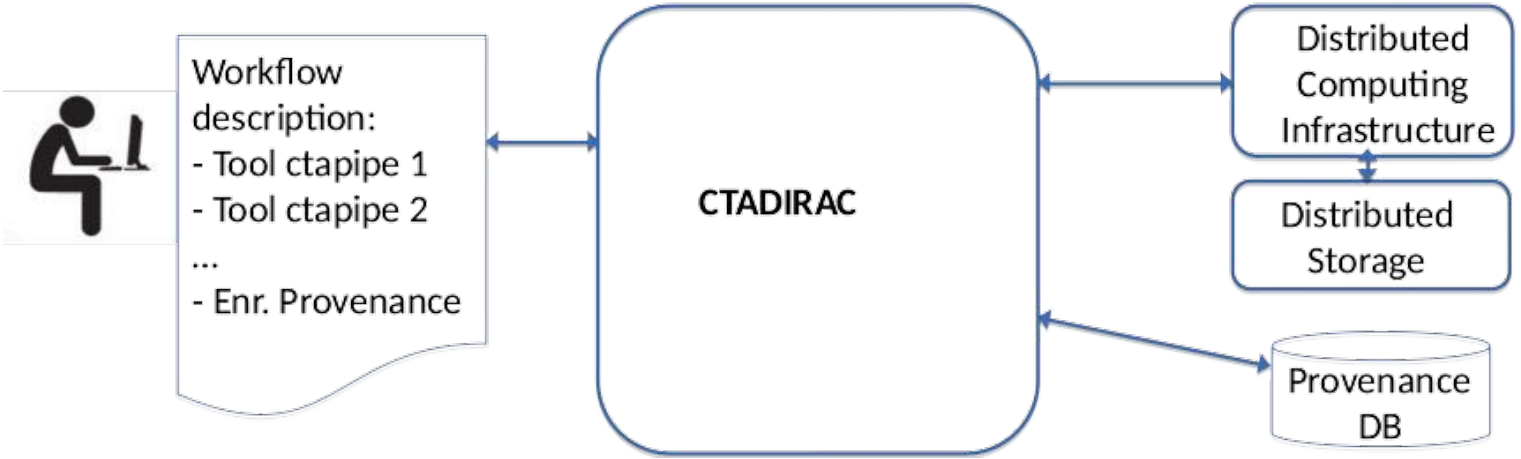
- CTADIRAC(Prototype basé sur DIRAC - L. Arrabito - LUPM) utilisé pour la gestion des calculs distribués sur des ressources hétérogènes (grille, cloud, clusters)



@ M. Sanguillon

Intégration de la provenance dans CTADIRAC

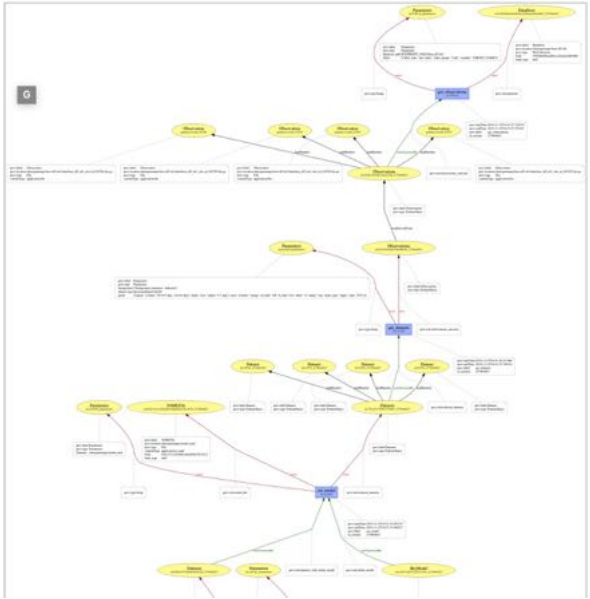
- L'enregistrement de la provenance dans la base de données Provenance se fait automatiquement par CTADIRAC à partir des informations de provenance capturées lors de l'exécution de chaque outil de la bibliothèque ctapipe.

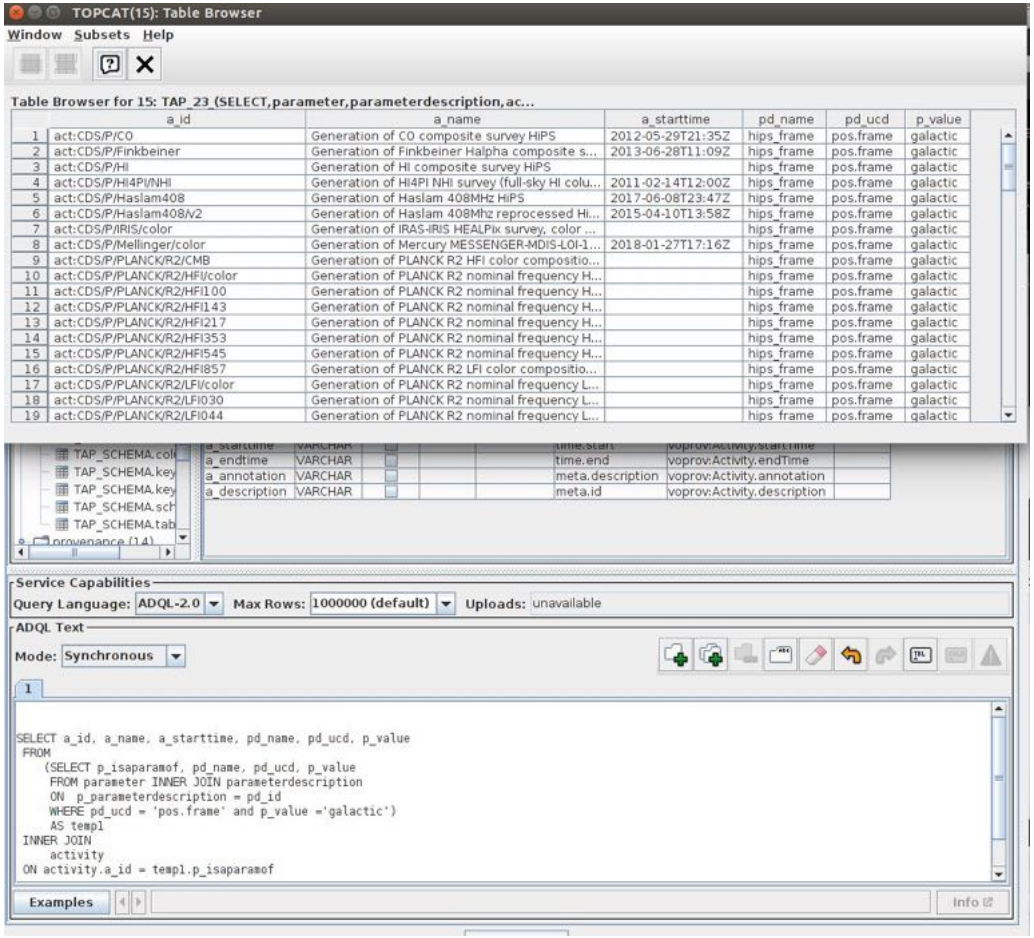


@ M. Sanguillon

ProvSAP & ProvTAP

opus_server/provsap?
ID=4ef2ad
&DEPTH=ALL
&AGENT=1
&DESCRIPTION=0
&RESPONSEFORMAT=PROV-SVG





Pollux database & Provenance
ProvSAP implementation

Parameter	Values	Description
ID	qualified ID	a valid qualified identifier for an entity, activity or agent (can occur multiple times)
DEPTH	0,1,2,..., ALL	number of relations to be followed or ALL for everything. independent of the relation type
RESPONSEFORMAT	PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE	serialisation format of the response
DIRECTION	BACK, FORTH	BACK = track the provenance history, FORTH = explore the results of activities and where entities have been used
MEMBERS	true (1) or false (0)	if true/1, retrieve and track members of collections
STEPS	true (1) or false (0)	if true/1, retrieve and track steps of activityFlows
AGENT	true (1) or false (0)	if true/1, explore all relations for agents, i.e. find out what an agent is responsible for
MODEL	IVOA or W3C	compatibility of the serialization to the IVOA or W3C provenance data model

DRAFT

ID: Multiple ID
spectra ids only

DEPTH: 0,1,...,ALL

RESPONSEFORMAT: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE, SVG

DIRECTION: BACK only

MEMBERS: not implemented

STEPS: not implemented

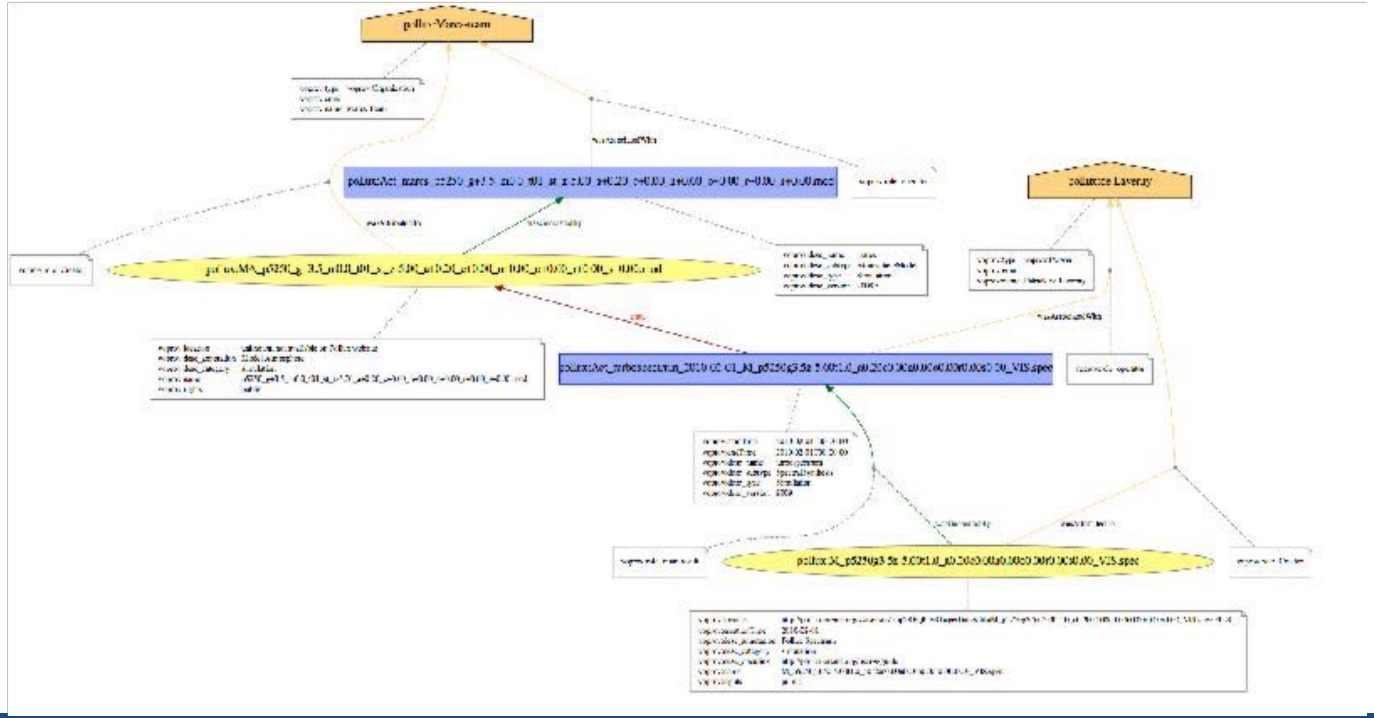
AGENT: true or false

MODEL: W3C

@ M. Sanguillon

Pollux database & Provenance
ProvSAP webservice

http://pollux.oreme.org/vo/provsap?id=M_p5250g3.5z-5.00t1.0_a0.20c0.00n0.00o0.00r0.00s0.00_VIS.spec.FITS&RESPONSEFORMAT=SVG&DEPTH=ALL&AGENT=true

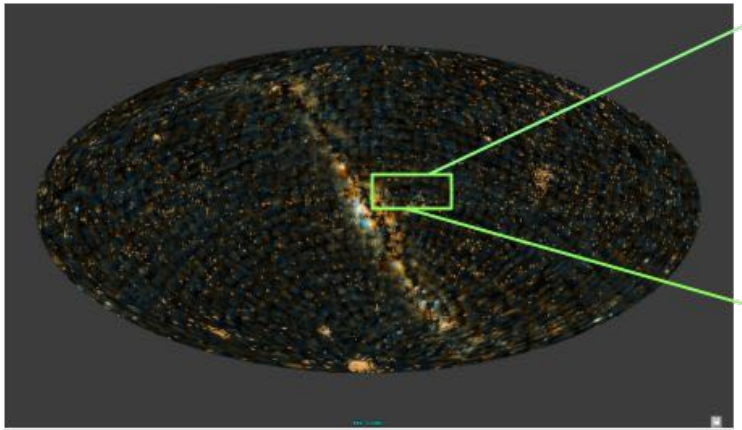


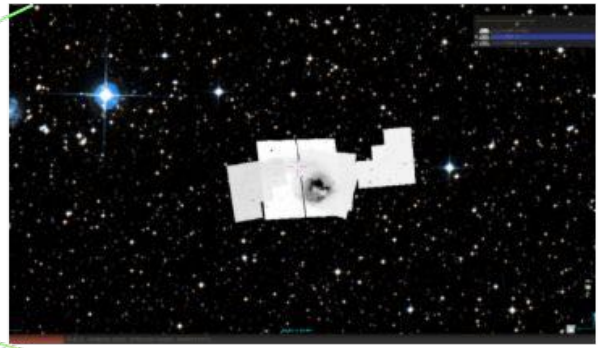
@ M. Sanguillon

ProvHiPS project: tracing provenance of HiPS and HiPS tiles

HiPS All sky hierarchical View

from HST images





HST V HiPS on top of DSS HiPS : detail

HST V HiPS on top of DSS HiPS : AllSky

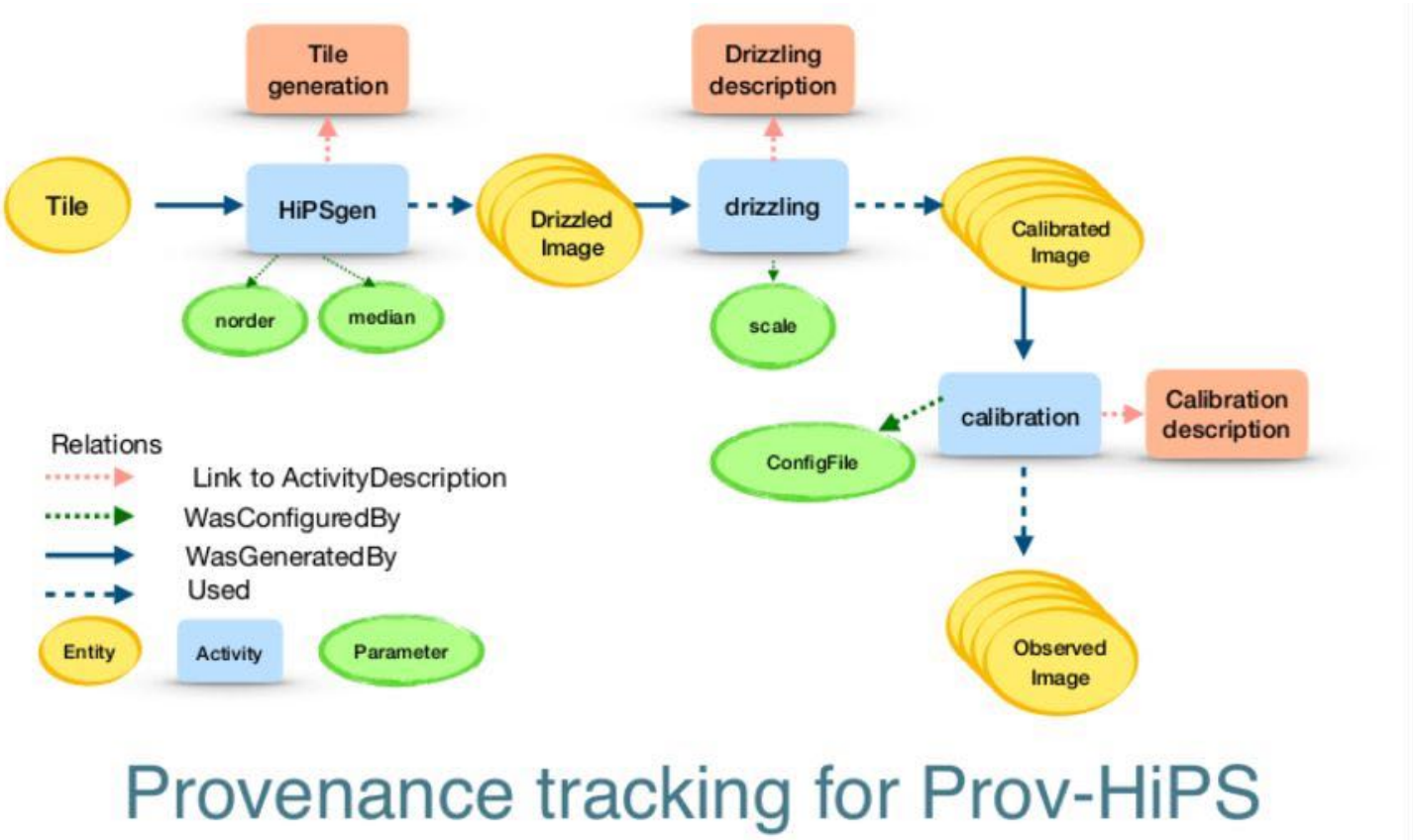
Provenance Metadata of **HiPS tiles** extracted from HST FITS headers

Mapped onto **IVOA Provenance DM**

Served by **IVOA TAP --> ProvTAP prototype service**

@ F. Bonnarel

«HiPS» Provenance diagram



@ F. Bonnarel

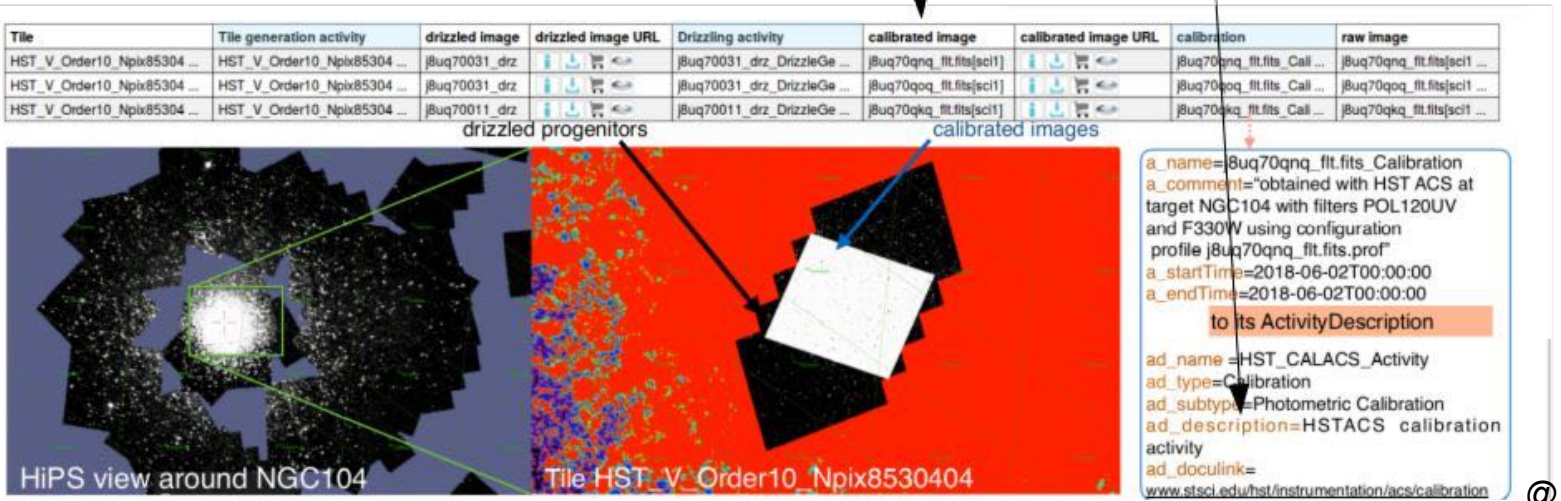
ProvHiPS in action

- From HiPS tiles back to raw HST

images, via drizzled images and single calibrated images

---> 13 join ADQL (= sql-like) query

- Many calibration activities share the same « activity description »



@ F. Bonnarel

Conclusions

- Capture → Storage → Access → Visualization
- Unique **identifiers**!
  - Level of uniqueness
  - Internal and external identifiers ?
  - When does an entity really changes and becomes another entity ?
- **Granularity** choice
  - Adapted to the project (as is the level of details, but remind FAIR F2 and R1!)
  - How can we mix different granularities ?
- Storage
  - Full provenance centralized in a database
  - Partial provenance inside data products ?
- Visualization/Serialization
  - mostly W3C graphs
  - Dedicated VO tool ? need use cases !

Next:
ProvTAP IVOA working draft
ASOV prov meeting in June
ESCAPE prov meeting in September