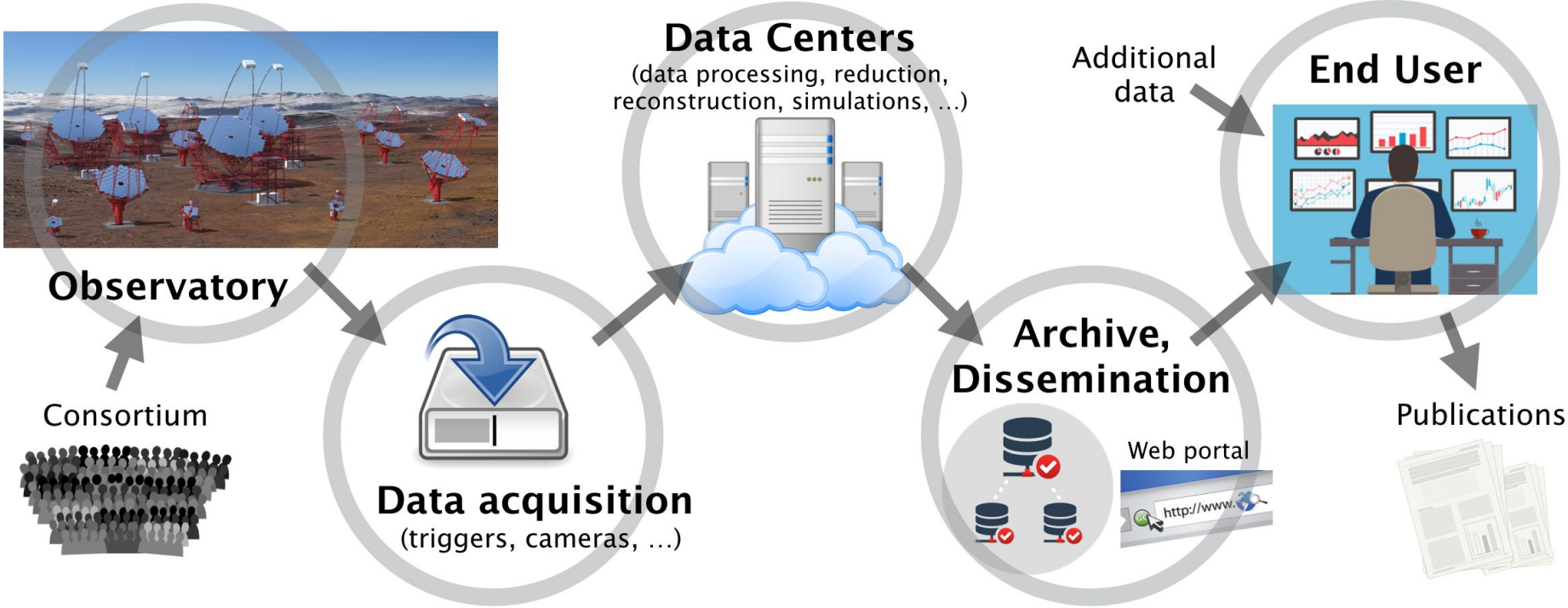
Implementations of the IVOA Provenance Data Model

Mathieu Servillat, Catherine Boisson,
François Bonnarel, Mireille Louys, Michèle Sanguillon
Jean-François Sornay

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/

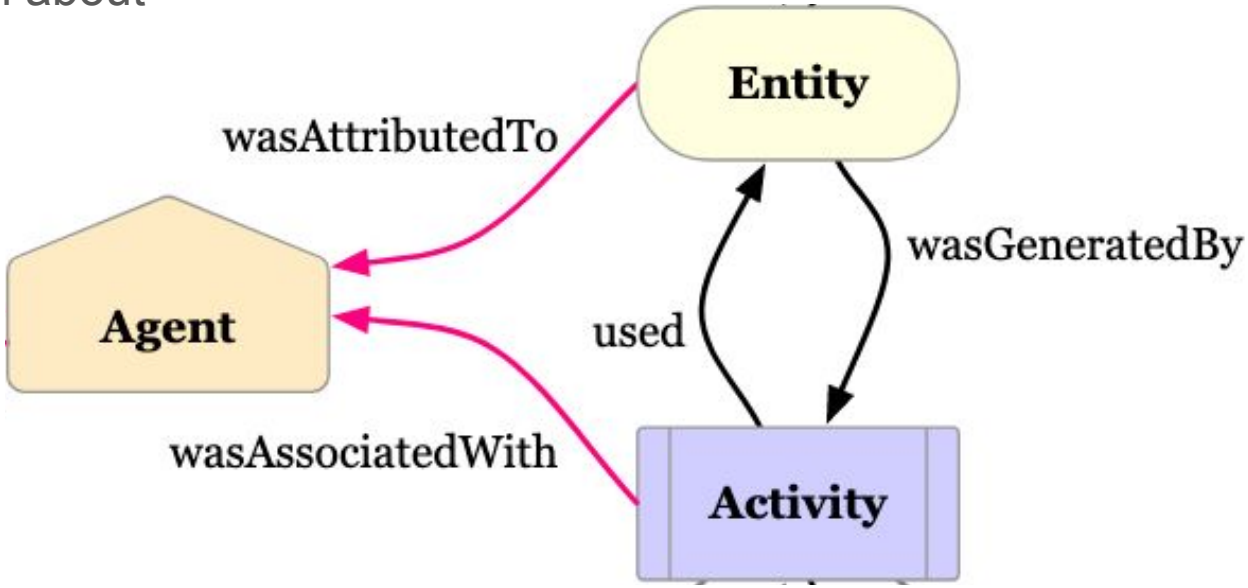
http://www.w3.org/TR/prov-overview/

W3C Provenance definition

Word Wide Web Consortium

W3C PROV (PROV-DM, 2013)

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**.



Example: carbon footprint estimation of IVOA meetings

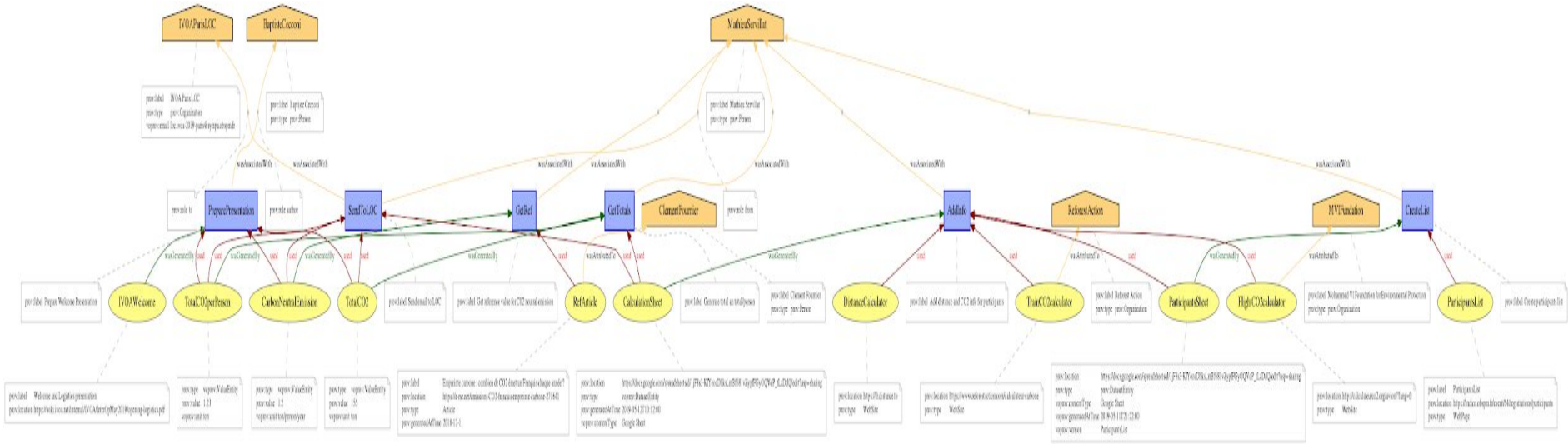
IVOA Paris 2019

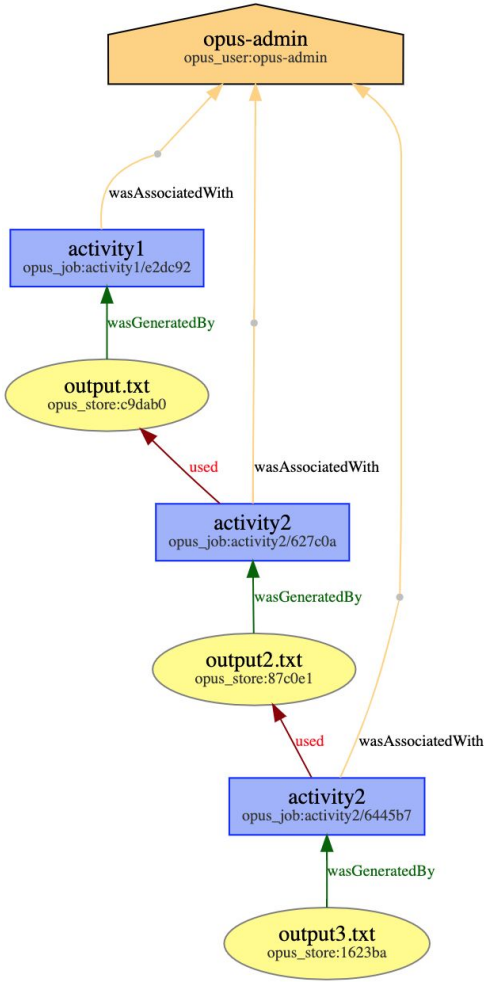
155 tons of CO2 for travels

Check the provenance of the calculations:
https://frama.link/CO2prov

IVOA Sydney 2020 virtual

300 grams of CO2 for visioconferences
(~1500 h.person of visio estimated)
https://greenspector.com/en/which-video-conferencing-mobile-application-to-reduce-your-impact/





Provenance graph

Provenance is :

- a **chain** of activities and entities (used and generated)
- that occured in the **past**

Is it enough to trust the processing of a product?

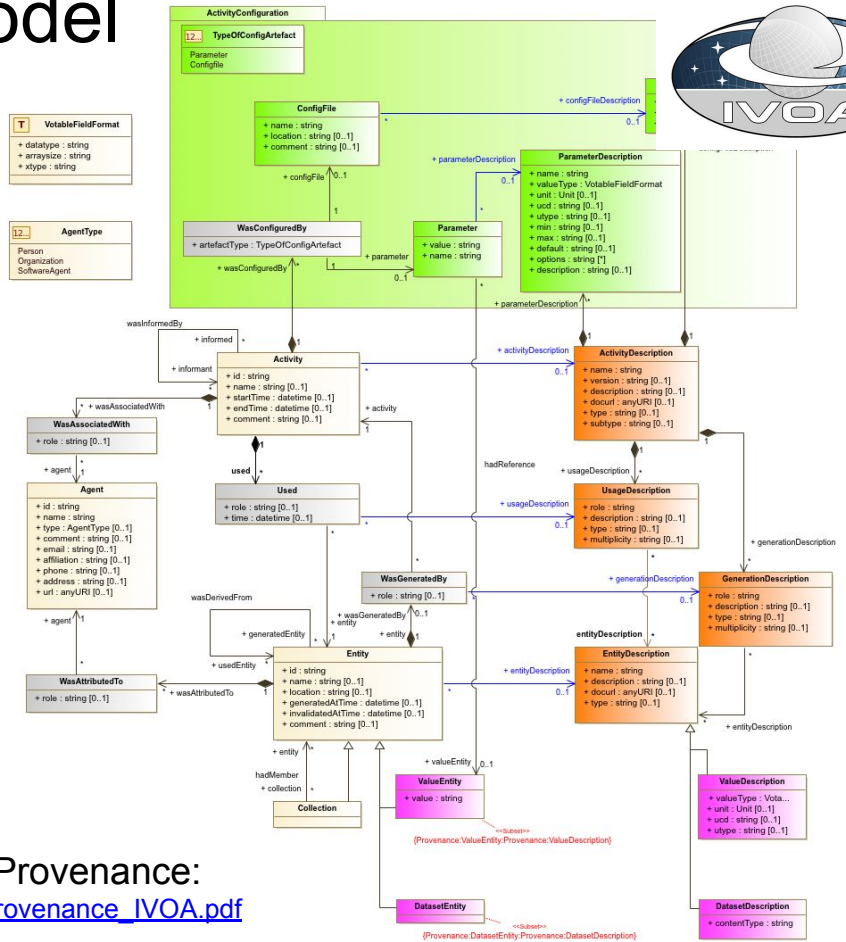
- What **happened** during each **activity**?
- How was the **activity** **tuned** to be executed properly?
- What **kind of content** is in the **entities** ?

IVOA Provenance Data Model

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

Recommendation 2020 !

- Adds “**Description**” classes
- Adds “**Configuration**” classes
- Plugged in with
  - **VO** data models and concepts
    (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

Descriptions bring VO specific attributes:

Usage/Generation

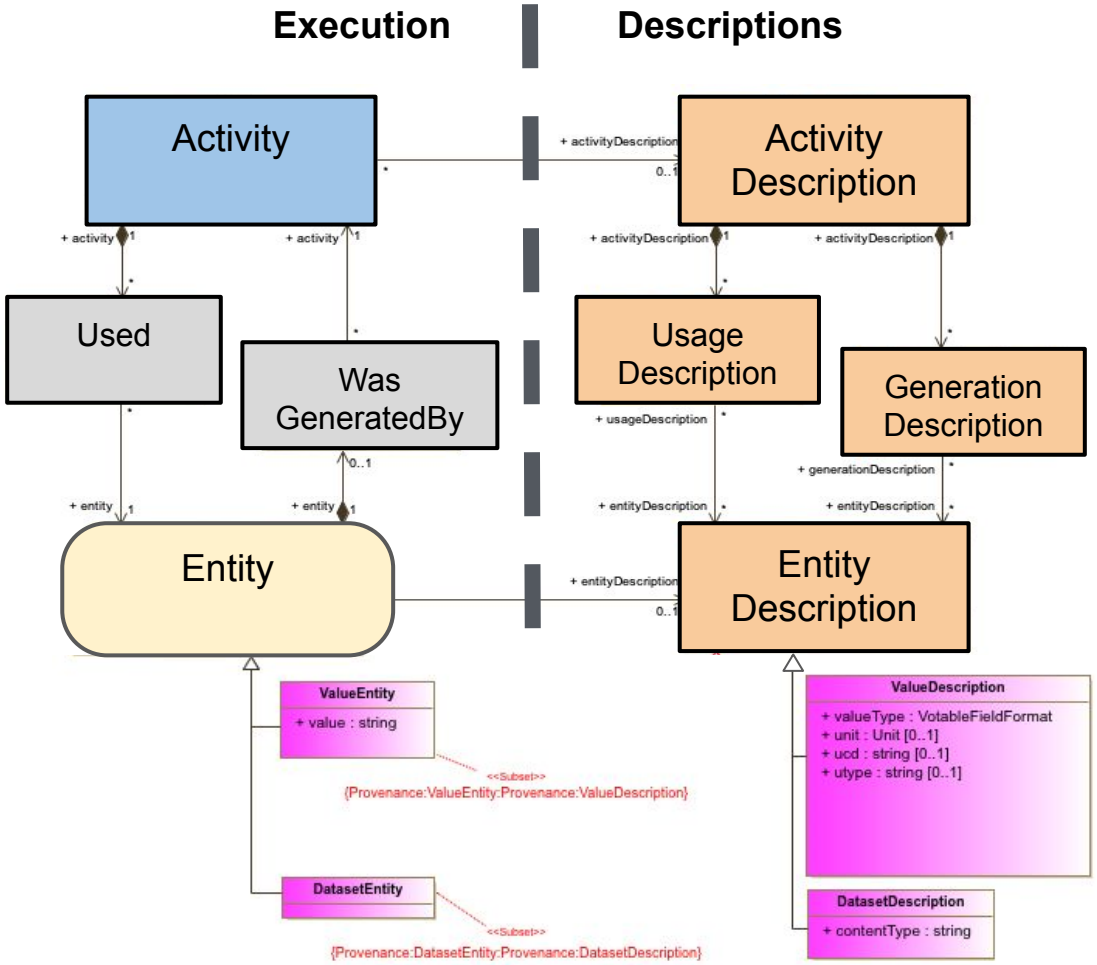
- **role** (master_bias, IRF, eventlist, ...)
- **type** (main, calibration, preview, quality, log, context)

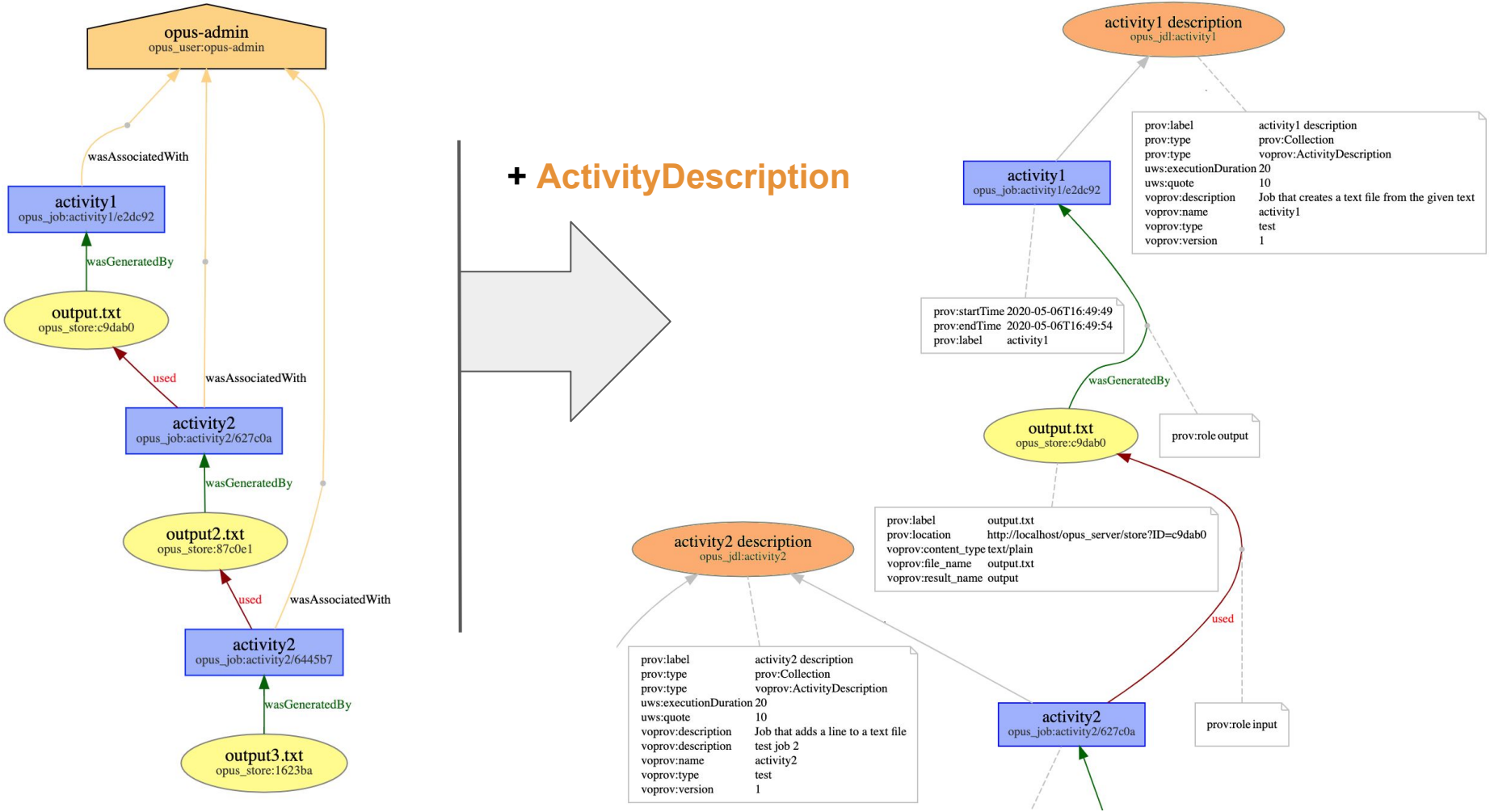
DatasetEntity (e.g. a file)

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

ValueEntity

- **valueType** (as in **VOTable**) **(datatype+arraysize+xtype)**
- unit, ucd, utype




opus-admin
opus_user:opus-admin
wasAssociatedWith
activity1
opus_job:activity1/e2dc92
wasGeneratedBy
output.txt
opus_store:c9dab0
used
wasAssociatedWith
activity2
opus_job:activity2/627c0a
wasGeneratedBy
output2.txt
opus_store:87c0e1
used
wasAssociatedWith
activity2
opus_job:activity2/6445b7
wasGeneratedBy
output3.txt
opus_store:1623ba
+ ActivityDescription
activity1 description
opus_jdl:activity1
prov:label activity1 description
prov:type prov:Collection
prov:type voprov:ActivityDescription
uws:executionDuration 20
uws:quote 10
voprov:description Job that creates a text file from the given text
voprov:name activity1
voprov:type test
voprov:version 1
activity1
opus_job:activity1/e2dc92
prov:startTime 2020-05-06T16:49:49
prov:endTime 2020-05-06T16:49:54
prov:label activity1
wasGeneratedBy
output.txt
opus_store:c9dab0
prov:role output
prov:label output.txt
prov:location http://localhost/opus_server/store?ID=c9dab0
voprov:content_type text/plain
voprov:file_name output.txt
voprov:result_name output
activity2 description
opus_jdl:activity2
prov:label activity2 description
prov:type prov:Collection
prov:type voprov:ActivityDescription
uws:executionDuration 20
uws:quote 10
voprov:description Job that adds a line to a text file
voprov:description test job 2
voprov:name activity2
voprov:type test
voprov:version 1
used
activity2
opus_job:activity2/627c0a
prov:role input

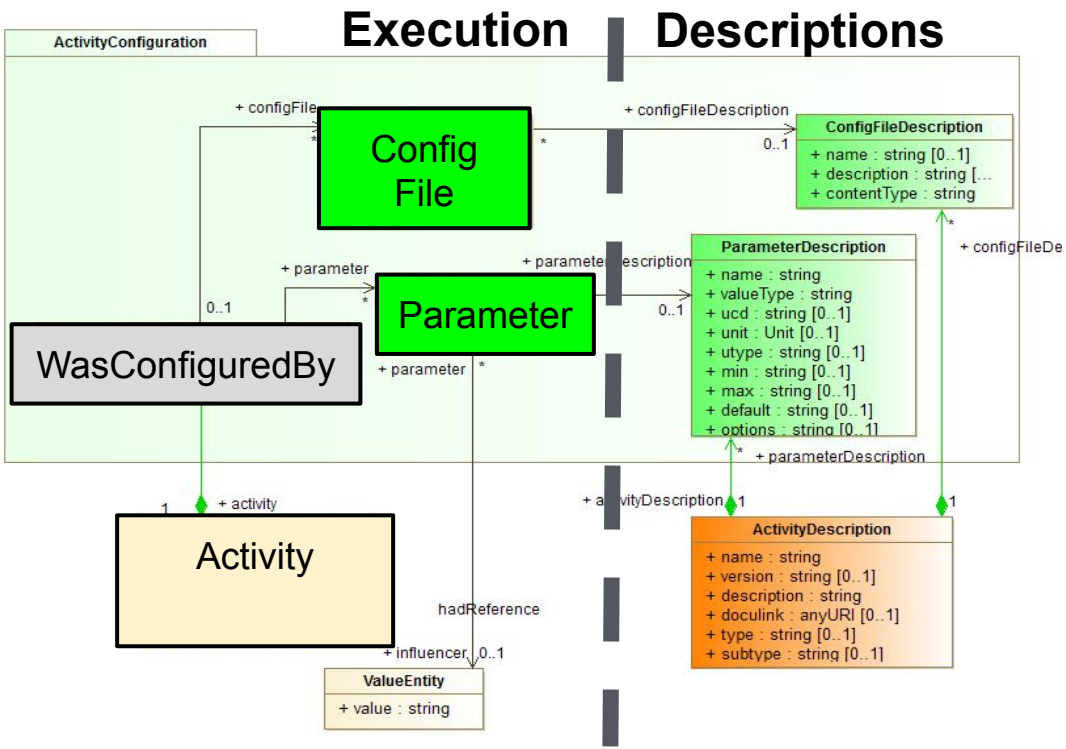
- Identify **the history track of** a data product ⇒ **Provenance (skeleton)**
- Identify what **detailed options** were used ⇒ **Configuration (flesh)**

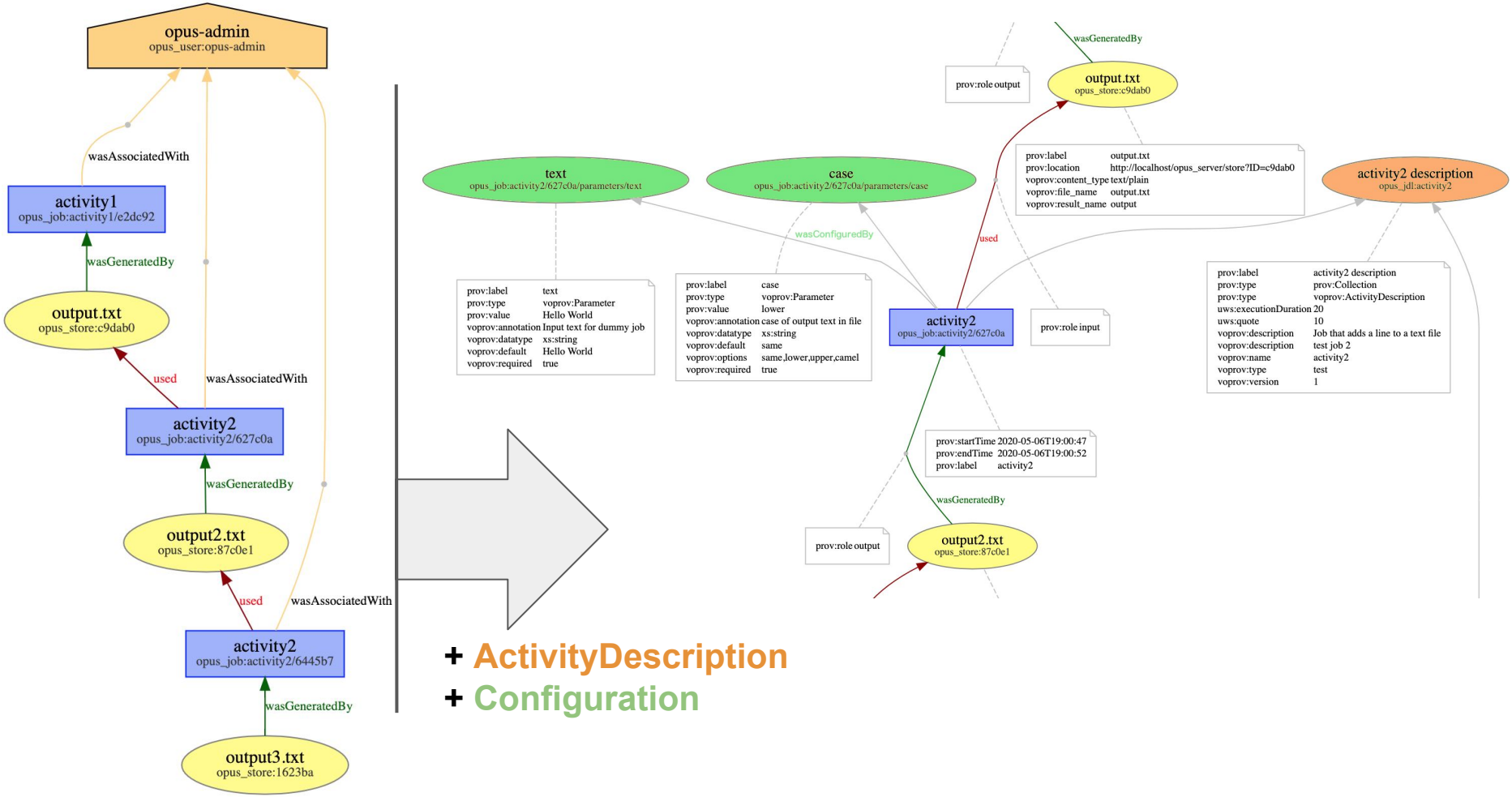
WasConfiguredBy

- **Parameter**
  - **name = value**
  - no identifier (part of activity)
- **ConfigFile**
  - name & location
- + their descriptions
  - valueType
  - unit, ucd, utype
  - min, max, options, default

Configuration is **dependent (part of)** on Activity / ActivityDescription

⇒ **fosters reproducibility**




opus-admin
opus_user:opus-admin
wasAssociatedWith
activity1
opus_job:activity1/e2dc92
wasGeneratedBy
output.txt
opus_store:c9dab0
used
wasAssociatedWith
activity2
opus_job:activity2/627c0a
wasGeneratedBy
output2.txt
opus_store:87c0e1
used
wasAssociatedWith
activity2
opus_job:activity2/6445b7
wasGeneratedBy
output3.txt
opus_store:1623ba
+ ActivityDescription
+ Configuration
wasGeneratedBy
prov:role output
output.txt
opus_store:c9dab0
prov:label output.txt
prov:location http://localhost/opus_server/store?ID=c9dab0
voprov:content_type text/plain
voprov:file_name output.txt
voprov:result_name output
activity2 description
opus_jdl:activity2
text
opus_job:activity2/627c0a/parameters/text
case
opus_job:activity2/627c0a/parameters/case
wasConfiguredBy
used
prov:label text
prov:type voprov:Parameter
prov:value Hello World
voprov:annotation Input text for dummy job
voprov:datatype xs:string
voprov:default Hello World
voprov:required true
prov:label case
prov:type voprov:Parameter
prov:value lower
voprov:annotation case of output text in file
voprov:datatype xs:string
voprov:default same
voprov:options same,lower,upper,camel
voprov:required true
activity2
opus_job:activity2/627c0a
prov:role input
prov:label activity2 description
prov:type prov:Collection
prov:type voprov:ActivityDescription
uws:executionDuration 20
uws:quote 10
voprov:description Job that adds a line to a text file
voprov:description test job 2
voprov:name activity2
voprov:type test
voprov:version 1
prov:startTime 2020-05-06T19:00:47
prov:endTime 2020-05-06T19:00:52
prov:label activity2
wasGeneratedBy
prov:role output
output2.txt
opus_store:87c0e1

Applying the model

Different contexts in use cases

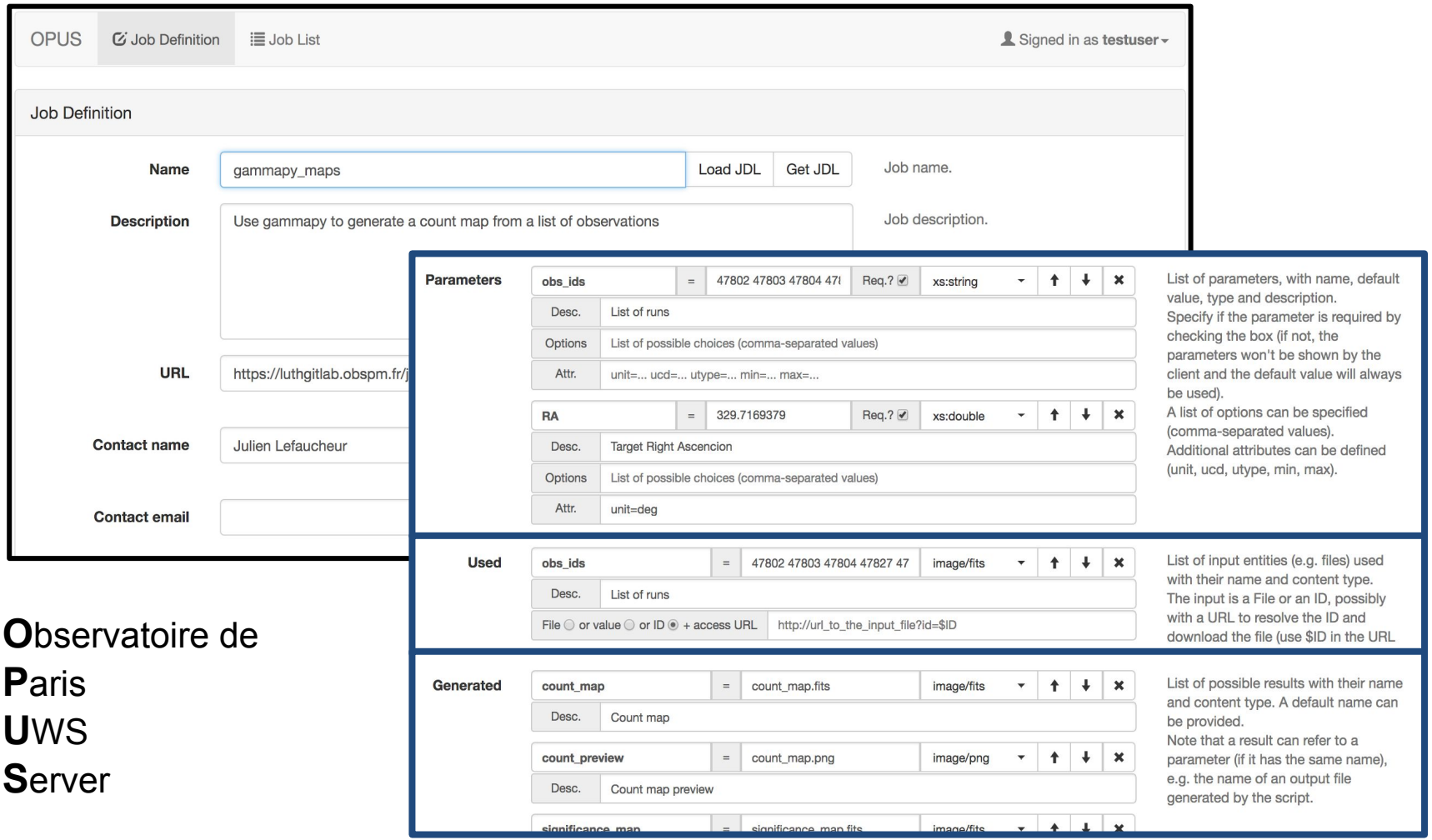
- Two flavours:
  - **on-top** (data products/collection already exist with some provenance metadata)
  - **inside** (a project plans to save provenance information during the processing)
- **Granularity** defined by the project (what steps? what objects?)
- **Level of details** required by project (descriptions? configuration?)
- **Identifiers:** unique and without meaning (if possible)

Different steps in provenance management

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

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**
  - **OPUS :** UWS job + entity store → W3C files
  - **CTADIRAC + `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**
  - **voprov** based on prov (W3C python package)



Observatoire de
Paris
UWS
Server

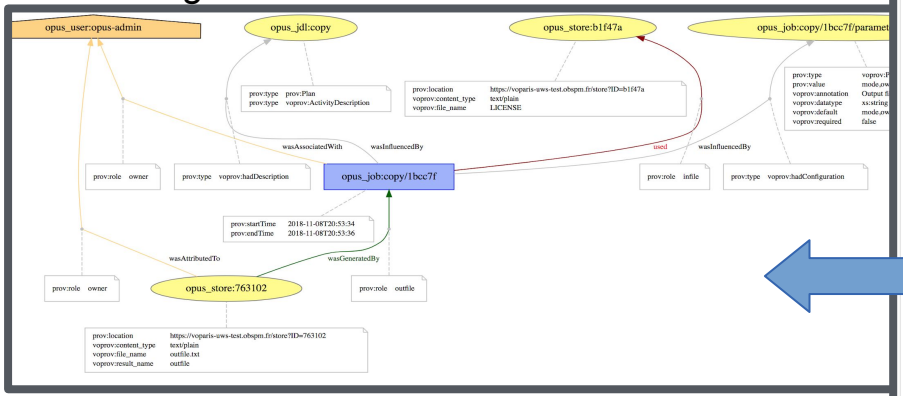
OPUS | Job Definition | Job List | Signed in as **user**

Job List for gammapy_spectra

Refresh Job List | Create Test Job | Create New Job

Type	Start Time	Destruction Time	Phase	Details	Control
gammapy_spectra	2017-10-02 10:47:07	2017-11-01 10:47:05	COMPLETED	Properties / Parameters / Results	Start / Abort / Delete
gammapy_spectra		2017-11-01 10:47:03	PENDING	Properti	
gammapy_spectra	2017-09-29 15:07:52	2017-10-29 15:07:51	COMPLETED	Properti	
gammapy_spectra	2017-09-29 14:55:10	2017-10-29 14:55:09	ABORTED	Properti	
gammapy_spectra	2017-09-29 14:21:20	2017-10-29 14:21:19	COMPLETED	Properti	

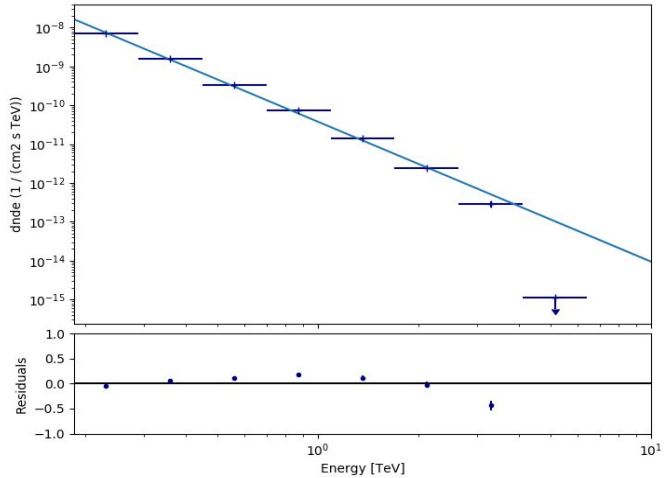
Tracking of Provenance informations

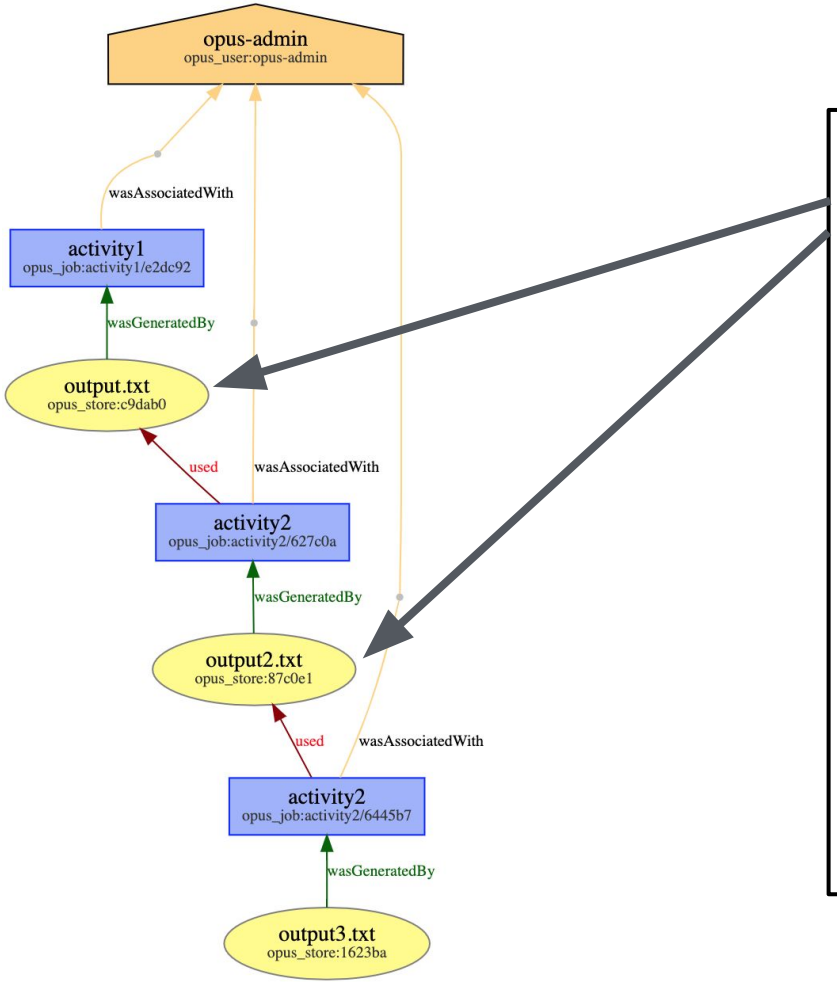


Job Results

spectrum: Download [image/fits]

spectrum_preview: Download [image/png]





- **OPUS registered entities**
  - Unknown input files
  - Results

  → stores: **id** + **hash** + **filename**

- **OPUS checks if already exist in the system**
  - **filename**?
    → may change...
  - **hash**?
    → integrity?
    → unicity?
  - **identifier**?
    → kept inside the files
    → more info in a database

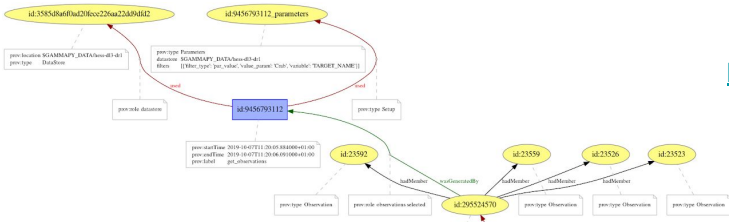
Capture Python package in `gammapy`

- **gammapy** is a Python package for gamma-ray data analysis (with a focus on CTA - the Cherenkov Telescope Array)
  - https://gammapy.org/
- A **provenance capture class** is in development
  - https://github.com/Bultako/gammapy/tree/prov/gammapy/utils/provenance
  - Based on the **high level interface** (i.e. not any base function)
  - **Non-intrusive** (class decorator) → developers don't see it
  - **Transparent** → users don't see it
  - Send each provenance event to the **logger** as a **structured dictionary**

prov.log:

```
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'}]}}
```

Provenance in `gammapy`



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

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

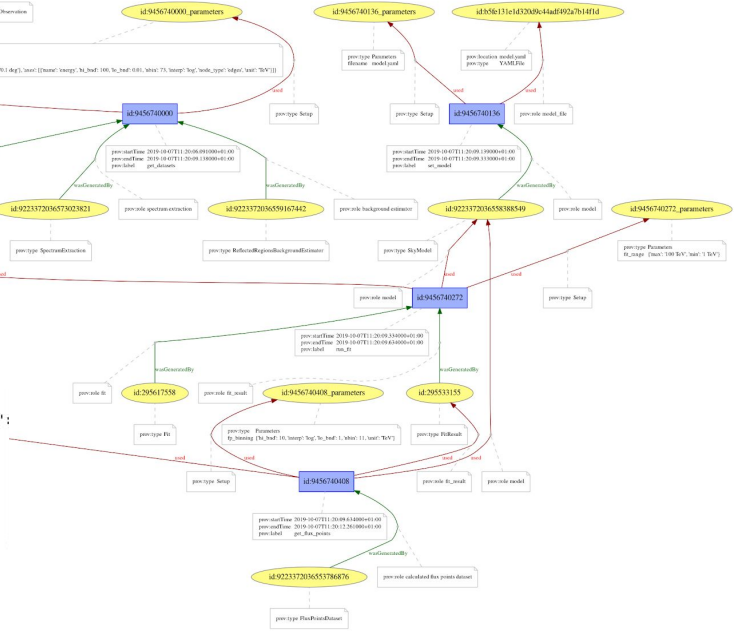
```yaml
activities:

    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

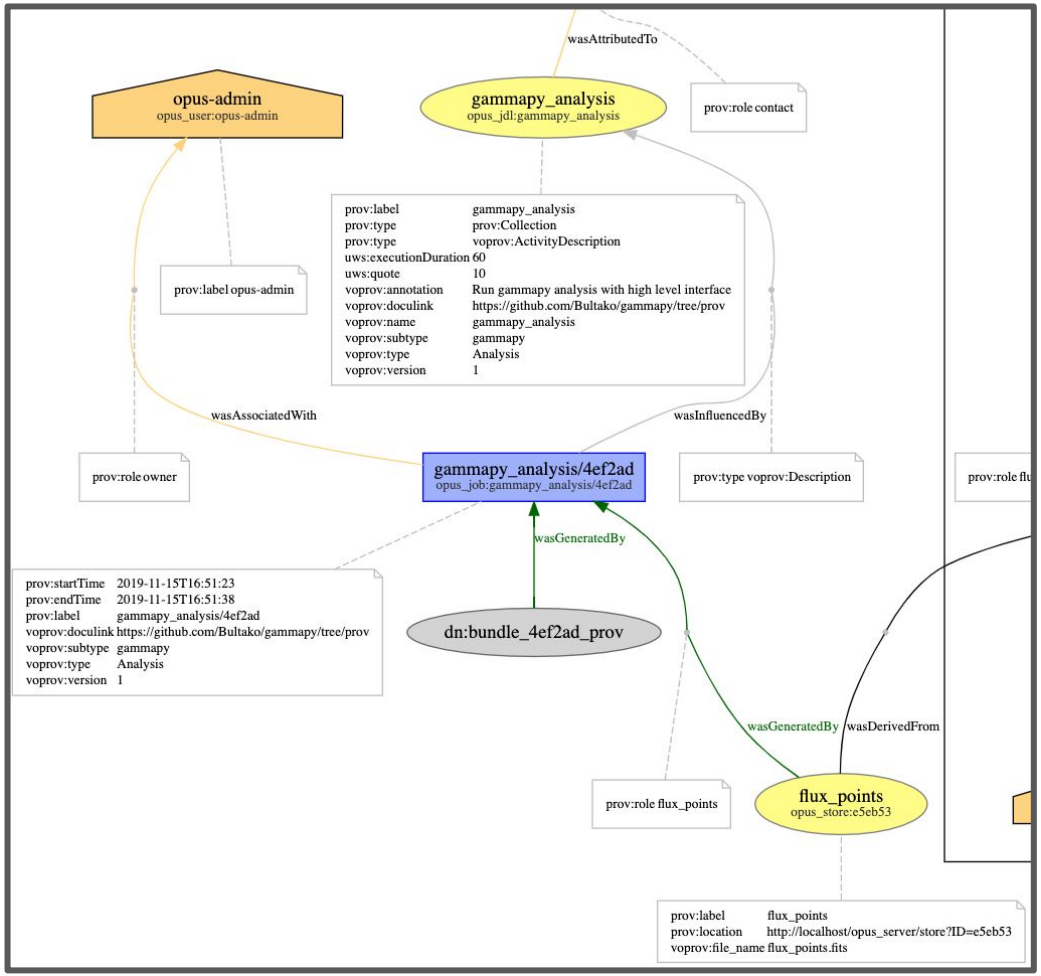
@ M. Servillat & J. E. Ruiz

Capture Python package in `gammapy`

- Identifiers and unicity...
  - the identifier is recomputed from the entity each time it is used
  - id(), hash(), str(), file hash (md5, sha1…)
- Parameter values directly attached to the activity
- Check for entity modifications before usage
  - indicate a derivation if entity has change (no more information about what happened)
- session identifier
  - when the Analysis class is instanciated
  - place to keep the system information, the global configuration
- granularity
  - 1 super-activity = run a gammapy analysis
  - internal provenance is accessible through a bundle generated by the super-activity
  - démo with OPUS (next slide)

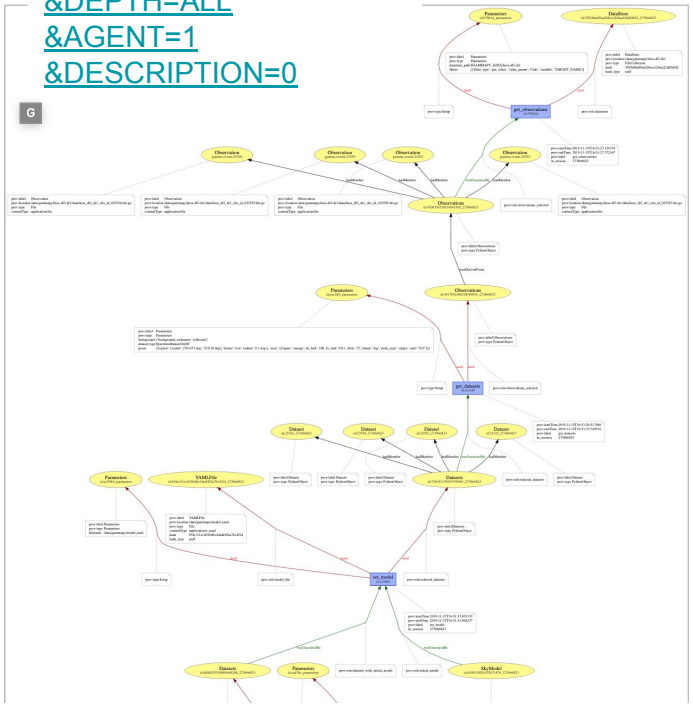
OPUS + `gammapy`

- 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 ?



ProvSAP & ProvTAP

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



TOPCAT(15): Table Browser

Window Subsets Help

Table Browser for 15: TAP_23_(SELECT,parameter,parameterdescription,ac...

	a_id	a_name	a_starttime	pd_name	pd_ucd	p_value
1	act:CDS/P/C0	Generation of C0 composite survey HiPS	2012-05-29T21:35Z	hips_frame	pos.frame	galactic
2	act:CDS/P/Finkbeiner	Generation of Finkbeiner Halpha composite s...	2013-06-28T11:09Z	hips_frame	pos.frame	galactic
3	act:CDS/P/HI	Generation of HI composite survey HiPS		hips_frame	pos.frame	galactic
4	act:CDS/P/HI4PI/NHI	Generation of HI4PI NHI survey (full-sky HI colu...	2011-02-14T12:00Z	hips_frame	pos.frame	galactic
5	act:CDS/P/Haslam408	Generation of Haslam 408MHz HiPS	2017-06-08T23:47Z	hips_frame	pos.frame	galactic
6	act:CDS/P/Haslam408/v2	Generation of Haslam 408Mhz reprocessed Hi...	2015-04-10T13:58Z	hips_frame	pos.frame	galactic
7	act:CDS/P/IRIS/color	Generation of IRAS-IRIS HEALPix survey, color ...		hips_frame	pos.frame	galactic
8	act:CDS/P/Mellinger/color	Generation of Mercury MESSENGER-MDIS-L0I-1...	2018-01-27T17:16Z	hips_frame	pos.frame	galactic
9	act:CDS/P/PLANCK/R2/CMB	Generation of PLANCK R2 HFI color compositio...		hips_frame	pos.frame	galactic
10	act:CDS/P/PLANCK/R2/HFI/color	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
11	act:CDS/P/PLANCK/R2/HFI100	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
12	act:CDS/P/PLANCK/R2/HFI143	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
13	act:CDS/P/PLANCK/R2/HFI217	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
14	act:CDS/P/PLANCK/R2/HFI353	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
15	act:CDS/P/PLANCK/R2/HFI545	Generation of PLANCK R2 nominal frequency H...		hips_frame	pos.frame	galactic
16	act:CDS/P/PLANCK/R2/HFI857	Generation of PLANCK R2 LFI color compositio...		hips_frame	pos.frame	galactic
17	act:CDS/P/PLANCK/R2/LFI/color	Generation of PLANCK R2 nominal frequency L...		hips_frame	pos.frame	galactic
18	act:CDS/P/PLANCK/R2/LFI030	Generation of PLANCK R2 nominal frequency L...		hips_frame	pos.frame	galactic
19	act:CDS/P/PLANCK/R2/LFI044	Generation of PLANCK R2 nominal frequency L...		hips_frame	pos.frame	galactic

TAP_SCHEMA.col
TAP_SCHEMA.key
TAP_SCHEMA.key
TAP_SCHEMA.sch
TAP_SCHEMA.tab
provenance (14)

a_starttime	VARCHAR		time.start	voprov:Activity.startTime
a_endtime	VARCHAR		time.end	voprov:Activity.endTime
a_annotation	VARCHAR		meta.description	voprov:Activity.annotation
a_description	VARCHAR		meta.id	voprov:Activity.description

Service Capabilities

Query Language: ADQL-2.0 Max Rows: 1000000 (default) Uploads: unavailable

ADQL Text

Mode: Synchronous

1

```
SELECT a_id, a_name, a_starttime, pd_name, pd_ucd, p_value
 FROM
    (SELECT p_isaparamof, pd_name, pd_ucd, p_value
     FROM parameter INNER JOIN parameterdescription
     ON  p_parameterdescription = pd_id
     WHERE pd_ucd = 'pos.frame' and p_value ='galactic')
     AS temp1
 INNER JOIN
     activity
 ON activity.a_id = temp1.p_isaparamof
```

Examples Info

Conclusions

- Capture → Storage → Access → Visualization
- Unique **identifiers**!
  - Level of uniqueness?
  - Internal and external identifiers? namespaces?
  - 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: Bundles for provenance of provenance (W3C)
- Storage
  - Full provenance should be centralized in a **database**
  - Partial provenance inside data products
- Visualization/Serialization
  - Mostly W3C graphs for now
  - Dedicated VO tool ? need use cases !

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

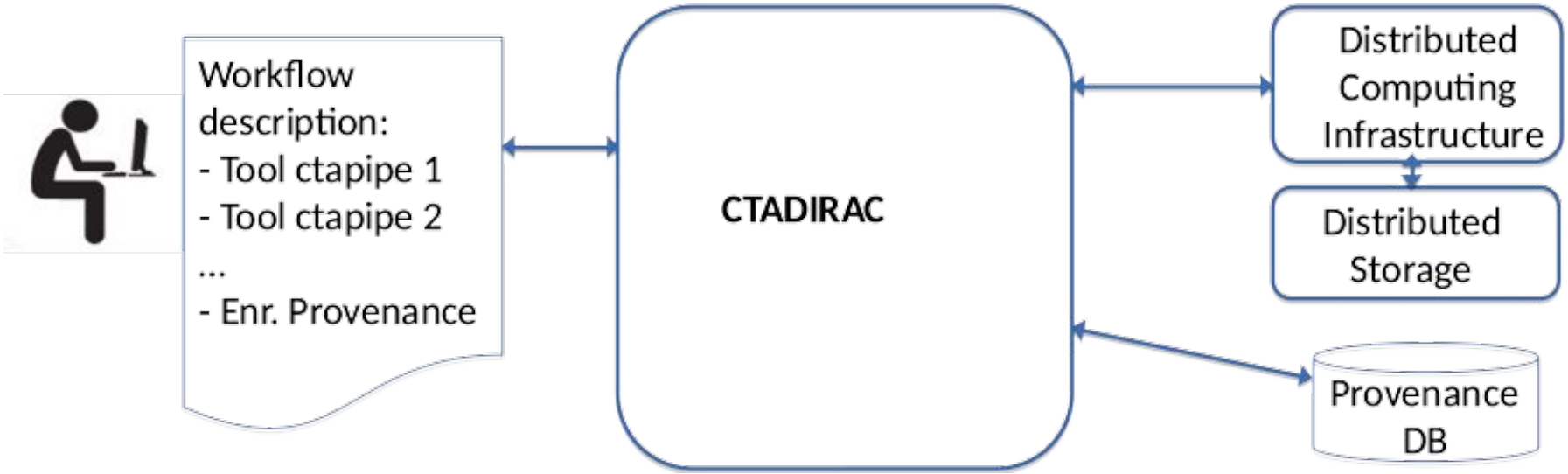
```
provBase = declarative_base()

# 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

ProvDB storage: DIRAC + `ctapipe`

- `ctapipe` produces provenance information
- CTADIRAC read and ingests this information in a database



@ M. Sanguillon