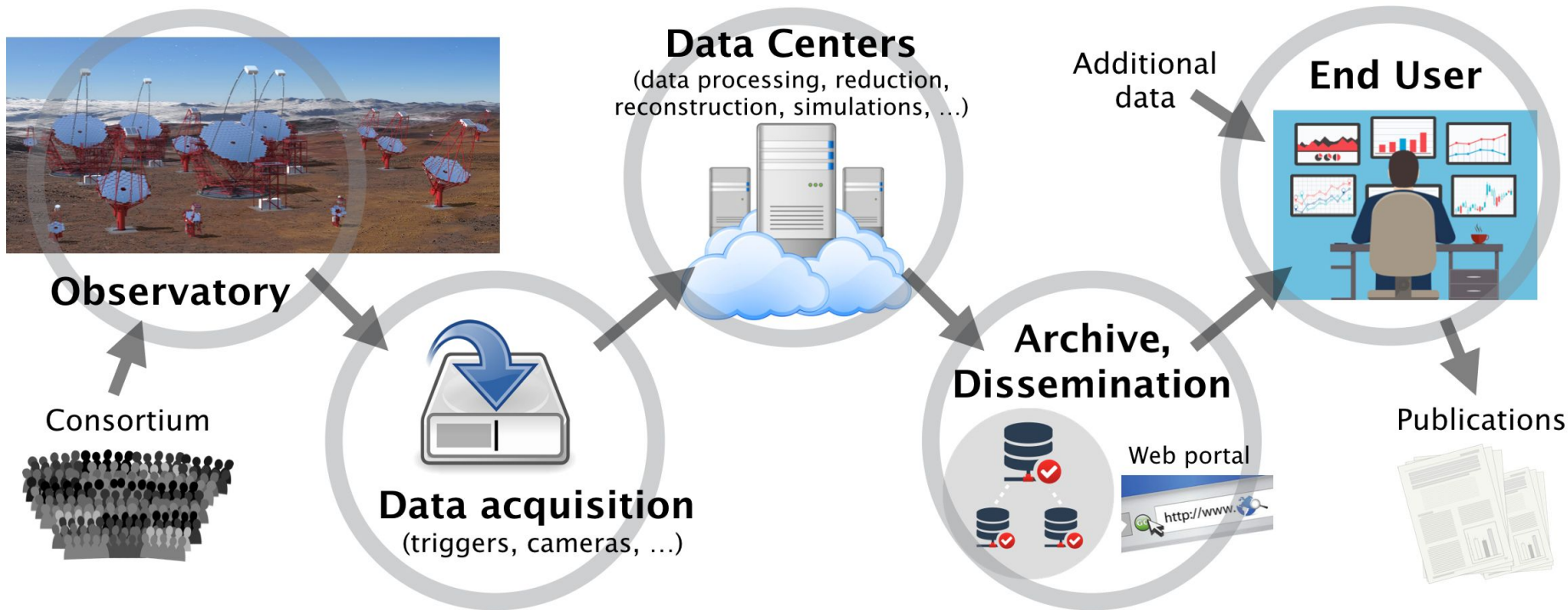


Implementations of the IVOA Provenance Data Model

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?)

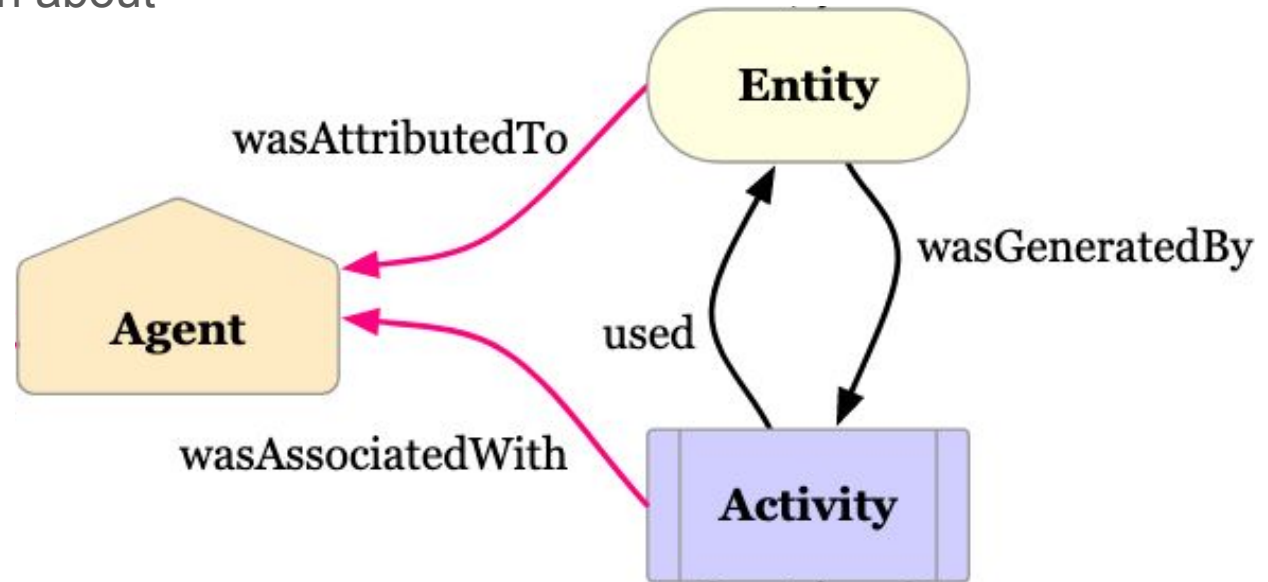
- R1. Meta(data) are richly described with a plurality of accurate and relevant attributes
 - R1.1. (Meta)data are released with a clear and accessible data usage license
 - R1.2. (Meta)data are associated with detailed provenance
 - R1.3. (Meta)data meet domain-relevant community standards

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

W3C Provenance definition

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

IVOA Sydney 2020 virtual

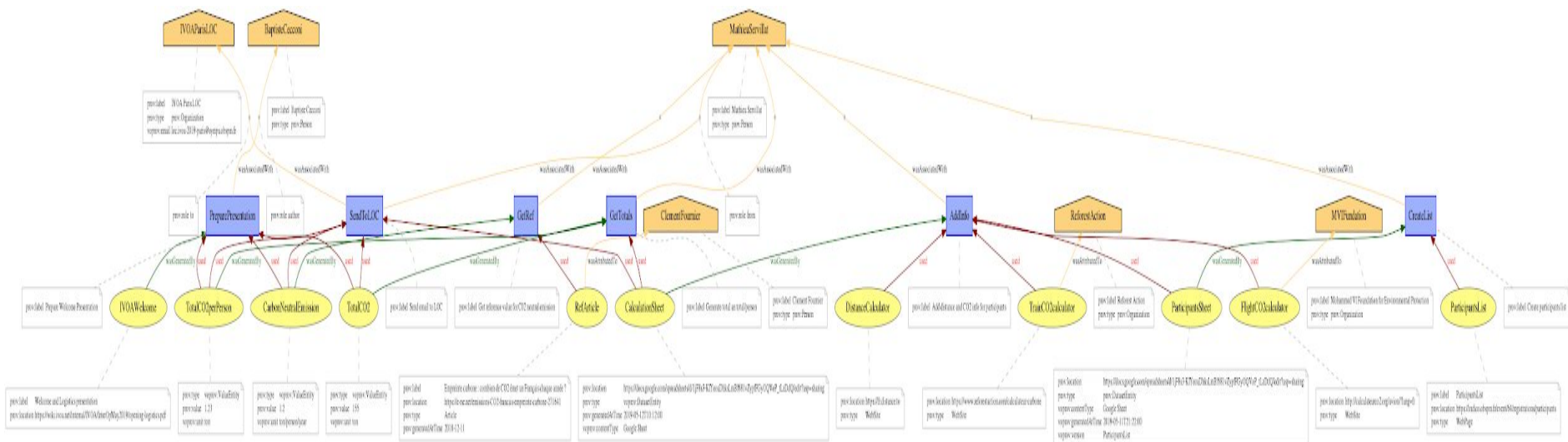
300 grams of CO2 for visioconferences

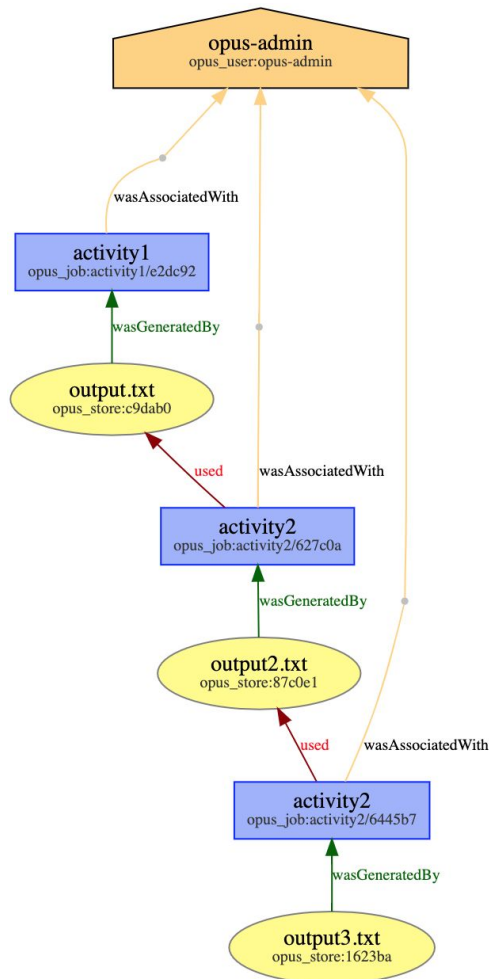
(~1500 h.person of visio estimated)

Check the provenance of the calculations:

<https://frama.link/CO2prov>

<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 occurred 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** ?

International Virtual Observatory Alliance

IVOA Documents



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

IVOA Provenance Data Model Version 1.0

IVOA Recommendation 11 April 2020

Interest/Working Group:

<http://www.ivoa.net/twiki/bin/view/IVOA/IvoaDataModel>

Author(s):

**Mathieu Servillat, Kristin Riebe, Catherine Boisson, François Bonnarel, Anastasia Galkin,
Mireille Louys, Markus Nullmeier, Nicolas Renault-Tinacci, Michèle Sanguillon, Ole Streicher**

Editor(s):

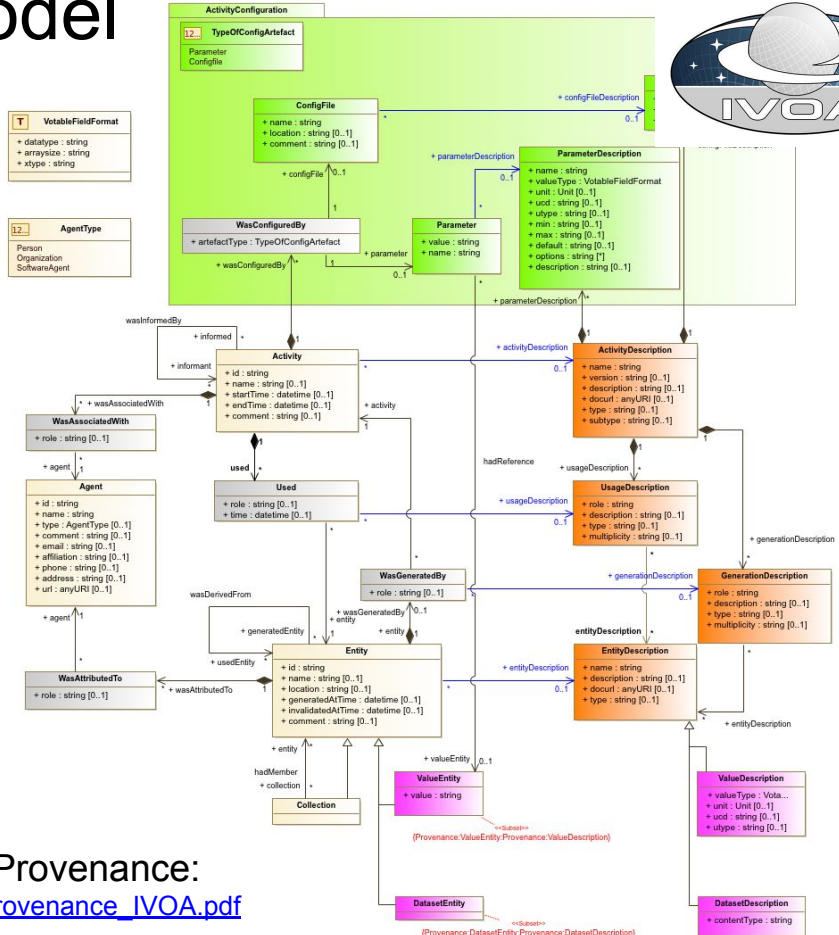
Mathieu Servillat



IVOA Provenance Data Model

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

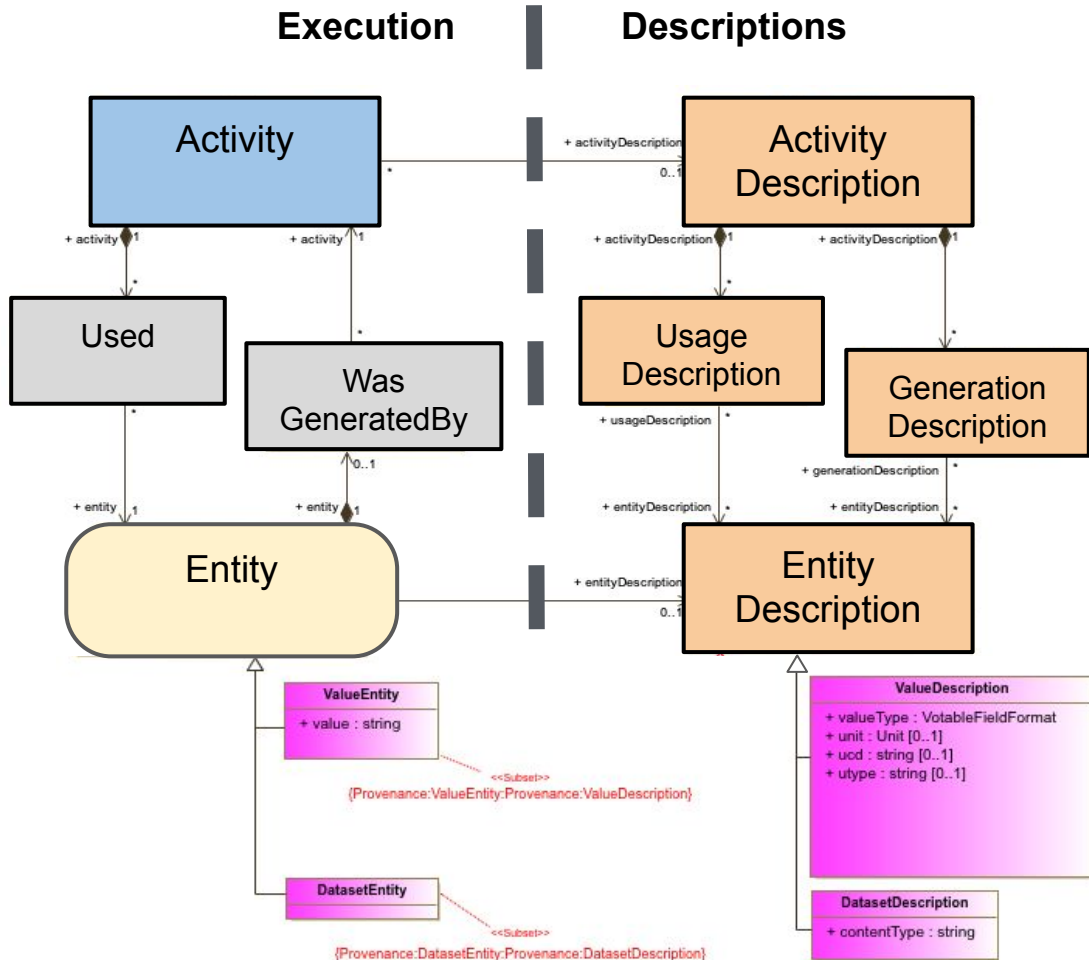
- **role** (master_bias, IRF, eventlist, ...)
- **type** (main, calibration, preview, quality, log, context)

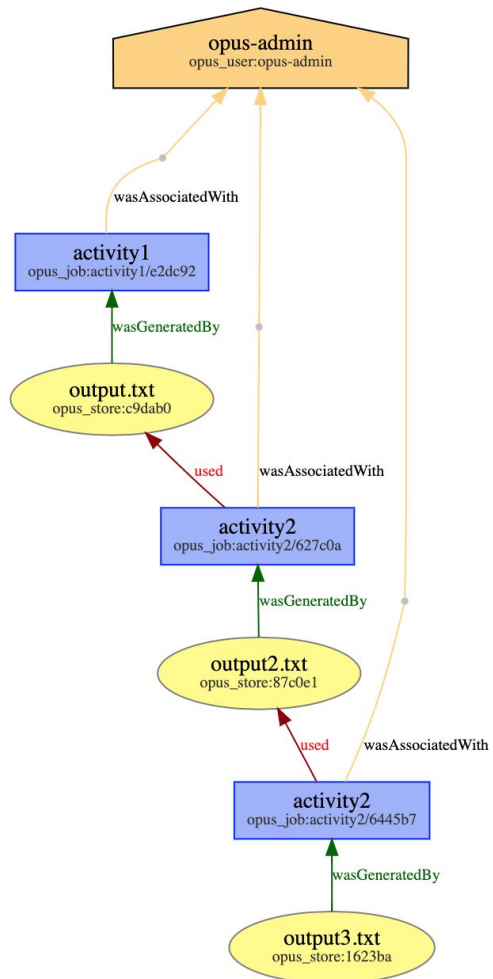
DatasetEntity (e.g. a file)

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

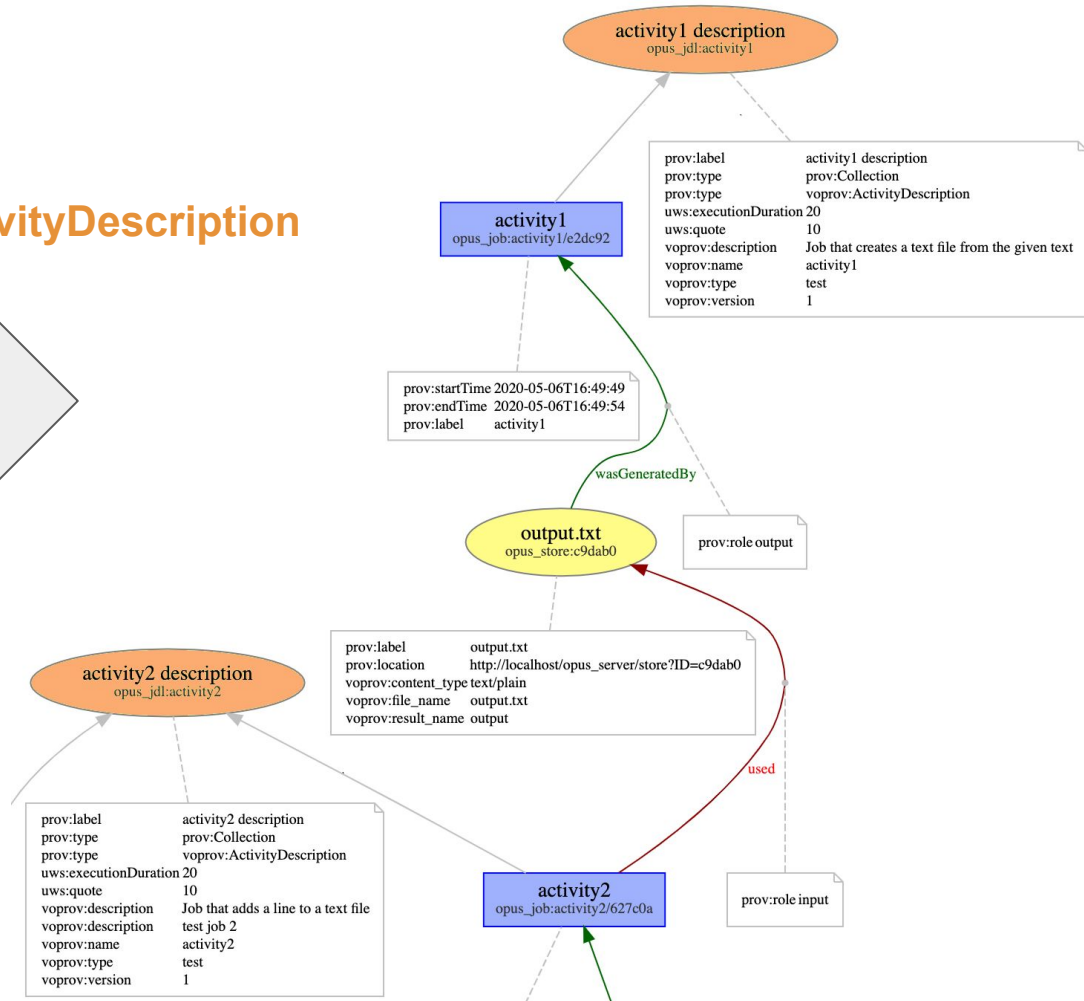
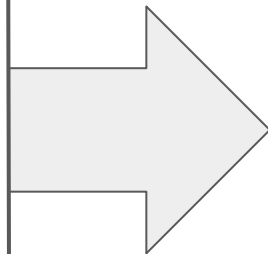
ValueEntity

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





+ ActivityDescription



- ◆ 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

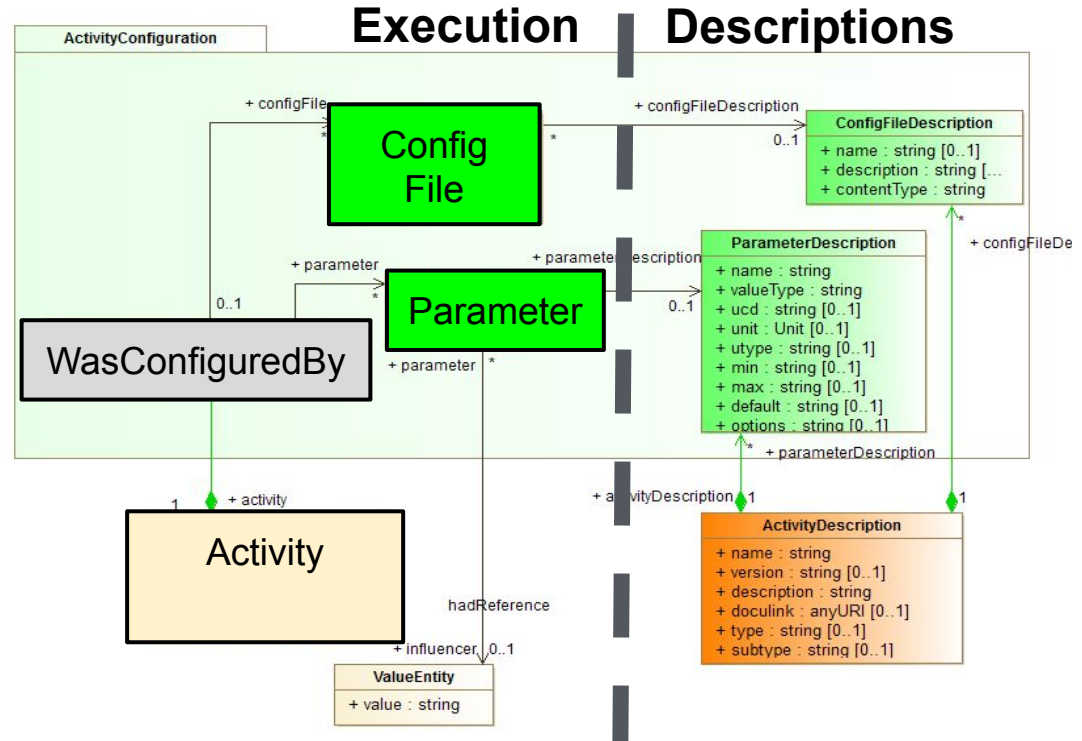
- name & location

◆ + their descriptions

- valueType
- unit, ucd, utype
- min, max, options, default

Configuration is dependent (part of) on Activity / ActivityDescription

⇒ **fosters reproducibility**



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 (sqlalchemy 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)

Job Definition

Name

gammapy_maps

Load JDL

Get JDL

Job name.

Description

Use gammapy to generate a count map from a list of observations

Job description.

URL

https://luthgitlab.obspm.fr/j

Contact name

Julien Lefaucheur

Contact email

Parameters

obs_ids	=	47802 47803 47804 471	Req.? <input checked="" type="checkbox"/>	xs:string	↑	↓	×
Desc.	List of runs						
Options	List of possible choices (comma-separated values)						
Attr.	unit=... ucd=... utype=... min=... max=...						
RA	=	329.7169379	Req.? <input checked="" type="checkbox"/>	xs:double	↑	↓	×
Desc.	Target Right Ascension						
Options	List of possible choices (comma-separated values)						
Attr.	unit=deg						

List of parameters, with name, default value, type and description.

Specify if the parameter is required by checking the box (if not, the parameters won't be shown by the client and the default value will always be used).

A list of options can be specified (comma-separated values). Additional attributes can be defined (unit, ucd, utype, min, max).

Used

obs_ids	=	47802 47803 47804 47827 47	image/fits	↑	↓	×
Desc.	List of runs					
File <input type="radio"/> or value <input type="radio"/> or ID <input checked="" type="radio"/> + access URL	http://url_to_the_input_file?id=\$ID					

List of input entities (e.g. files) used with their name and content type. The input is a File or an ID, possibly with a URL to resolve the ID and download the file (use \$ID in the URL).

Generated

count_map	=	count_map.fits	image/fits	↑	↓	×
Desc.	Count map					
count_preview	=	count_map.png	image/png	↑	↓	×
Desc.	Count map preview					
significance_map	=	significance_map.fits	image/fits	↑	↓	×

List of possible results with their name and content type. A default name can be provided.

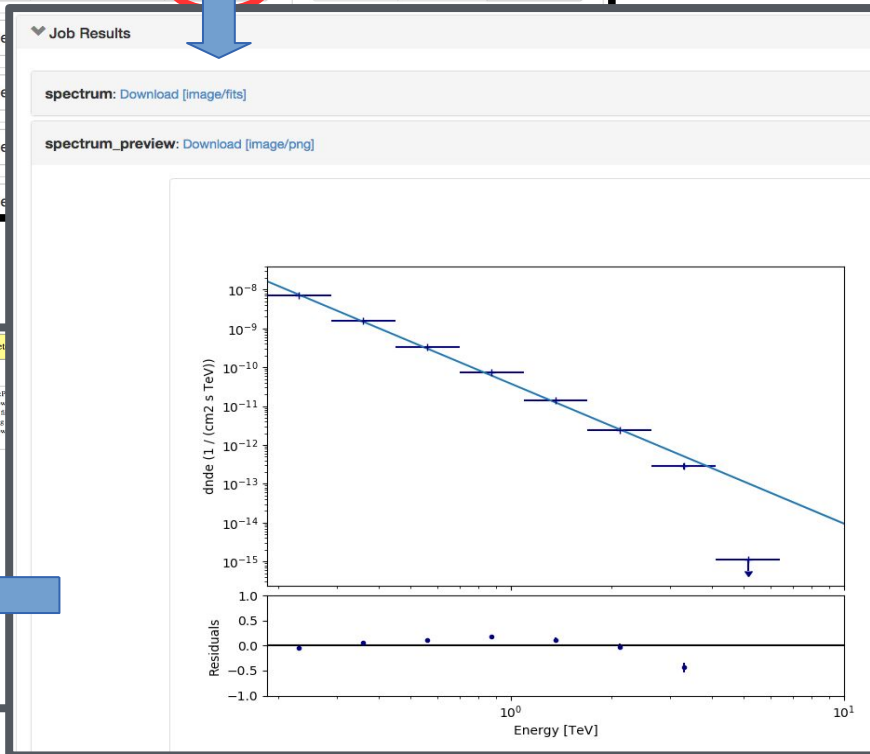
Note that a result can refer to a parameter (if it has the same name), e.g. the name of an output file generated by the script.

Observatoire de
Paris
UWS
Server

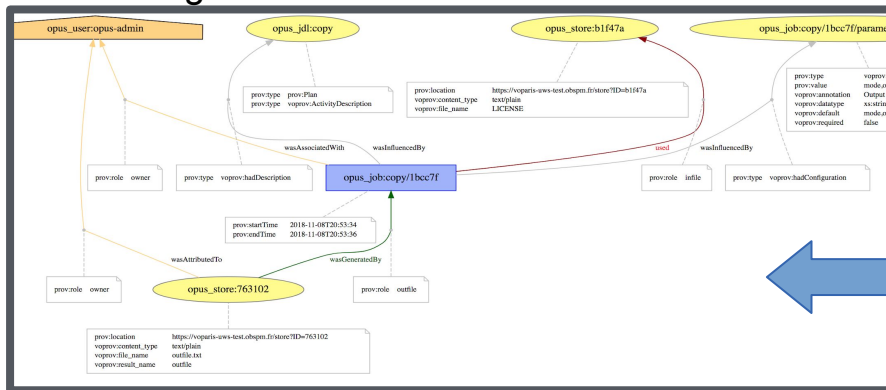
OPUS Job Definition Job List Signed in as user

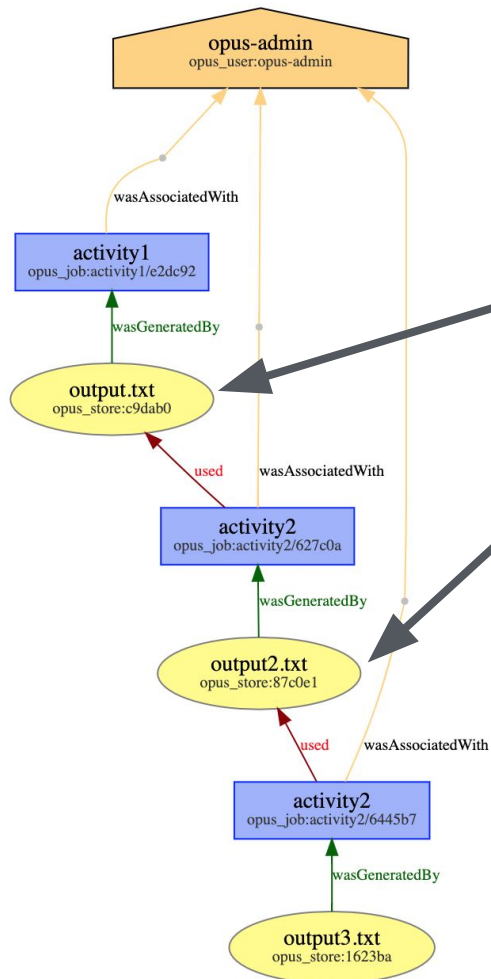
Job List for **gammapy_spectra** Refresh Job List Create Test Job Create New Job

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	Properties	
gammapy_spectra	2017-09-29 15:07:52	2017-10-29 15:07:51	COMPLETED	Properties	
gammapy_spectra	2017-09-29 14:55:10	2017-10-29 14:55:09	ABORTED	Properties	
gammapy_spectra	2017-09-29 14:21:20	2017-10-29 14:21:19	COMPLETED	Properties	



Tracking of Provenance informations





- **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

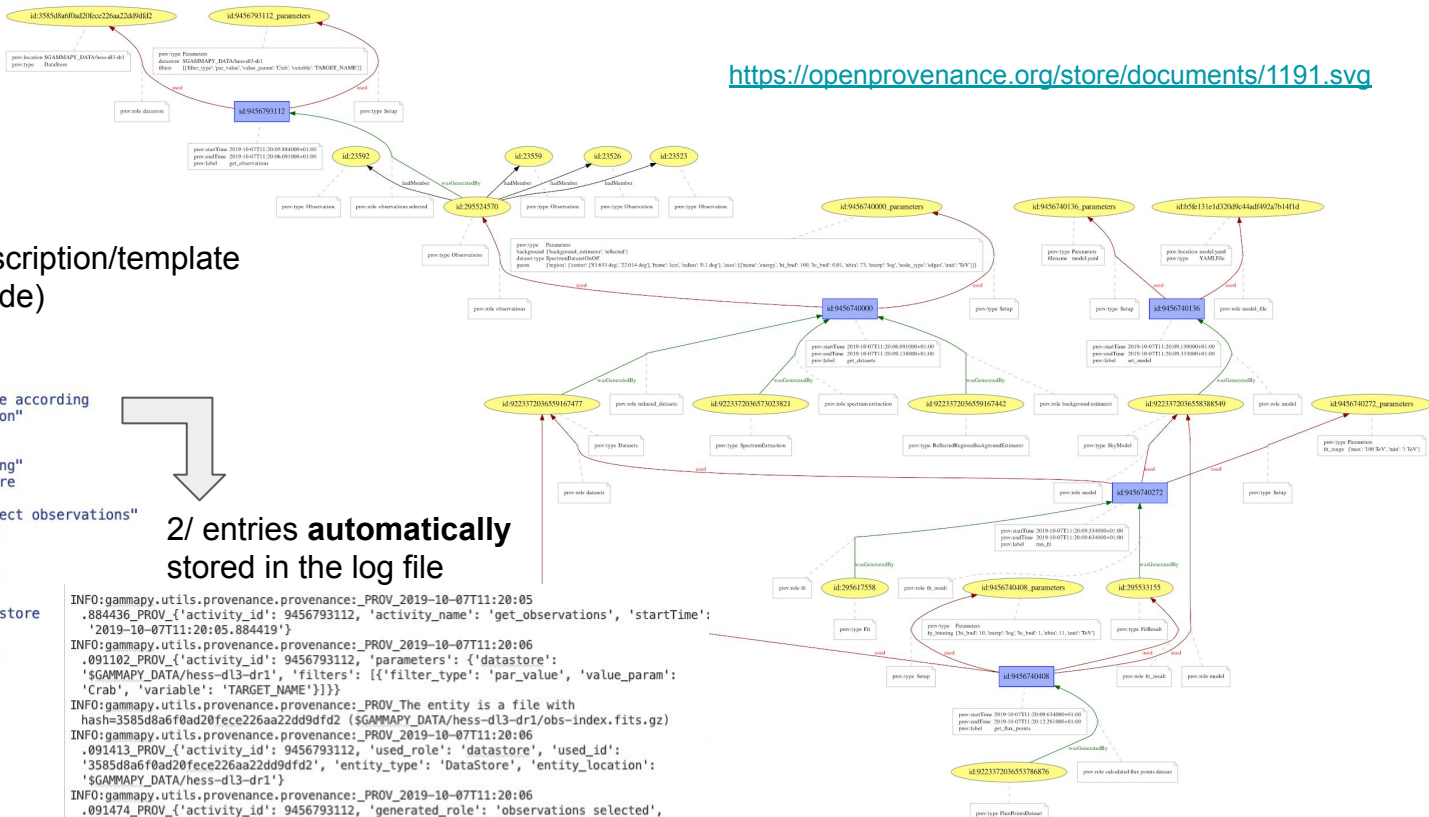
<https://openprovenance.org/store/documents/1191.svg>

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

```
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': 'SGMMAPY_DATA/hess-dl3-dr1', 'filters': [{'filter_type': 'par_value', 'value_param': 'Crab', 'variable': 'TARGET_NAME'}]}}
INFO:gammapy.utils.provenance.provenance:_PROV_2019-10-07T11:20:06.091413_PROV_{'activity_id': '9456793112', 'used_role': 'datastore', 'used_id': '3585d8a6f0ad20fce226aa22dd9dfd2', 'entity_type': 'DataStore', 'entity_location': 'SGMMAPY_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'}
```



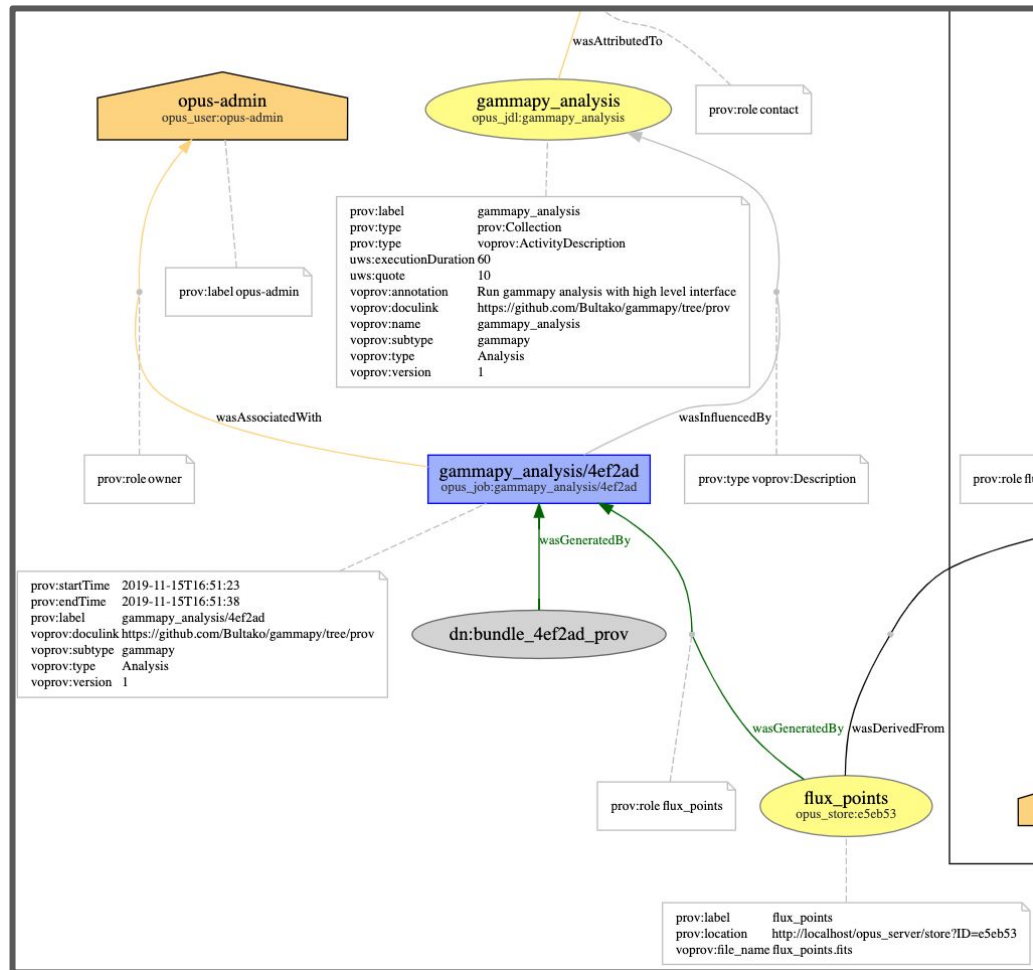
3/ export to W3C PROV or search in provenance records

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?](#)

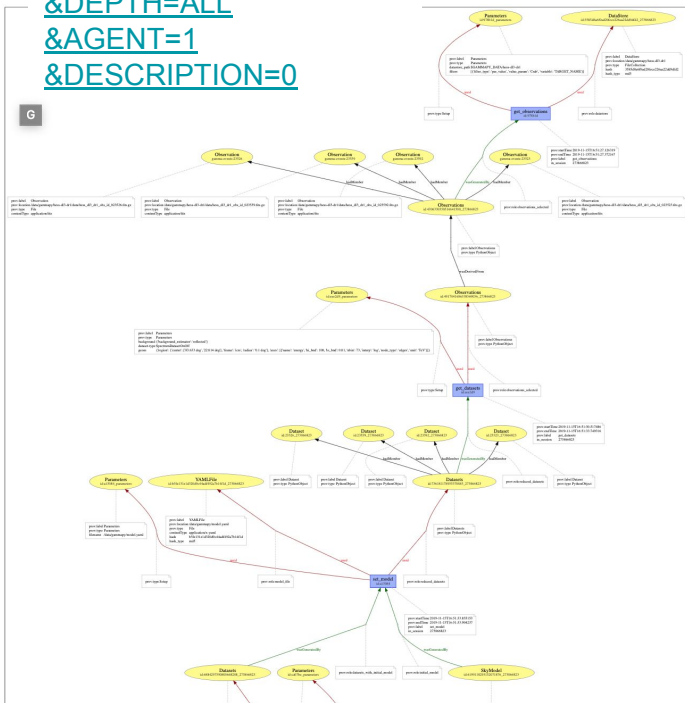
[ID=4ef2ad](#)

[&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/CO	Generation of CO 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/Haslam408V2	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-LOI-1...	2018-01-27T17:16Z	hips_frame	pos.frame	galactic
9	act:CDS/P/PLANCK/R2/CMB	Generation of PLANCK R2 HF1 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

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 templ
INNER JOIN
activity
ON activity_a_id = templ.p_isaparamof
```

Examples

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")
```

ProvDB storage: DIRAC + ctapipe

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

