Implementation of the IVOA Provenance Data Model: capture in gammapy

Mathieu Servillat

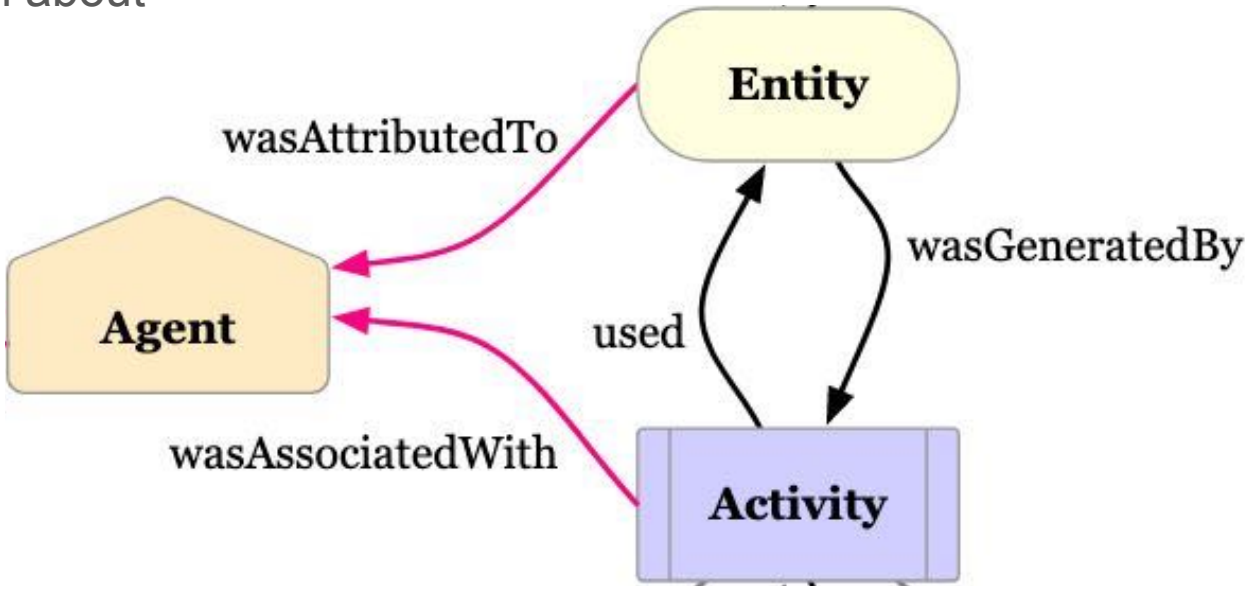
l'Observatoire de Paris — LUTH
Laboratoire Univers et Théories
cnrs
PSL RESEARCH UNIVERSITY PARIS
PADC Paris Astronomical Data Centre

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

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.

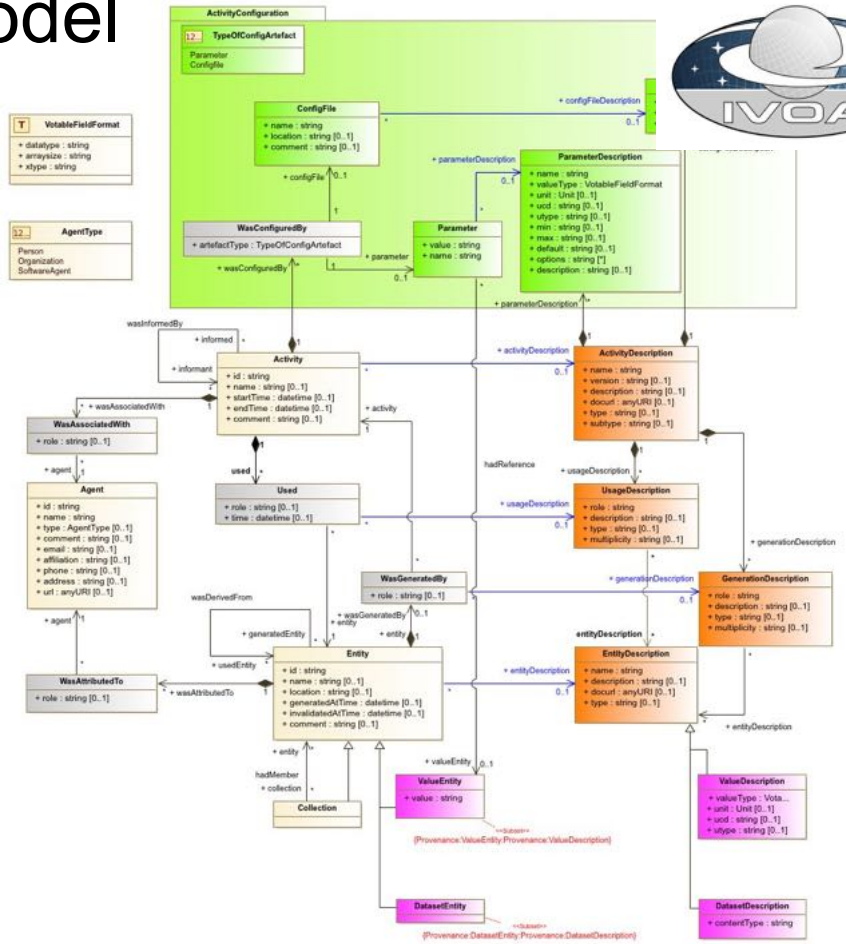


IVOA Provenance Data Model

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

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



Usage/Generation

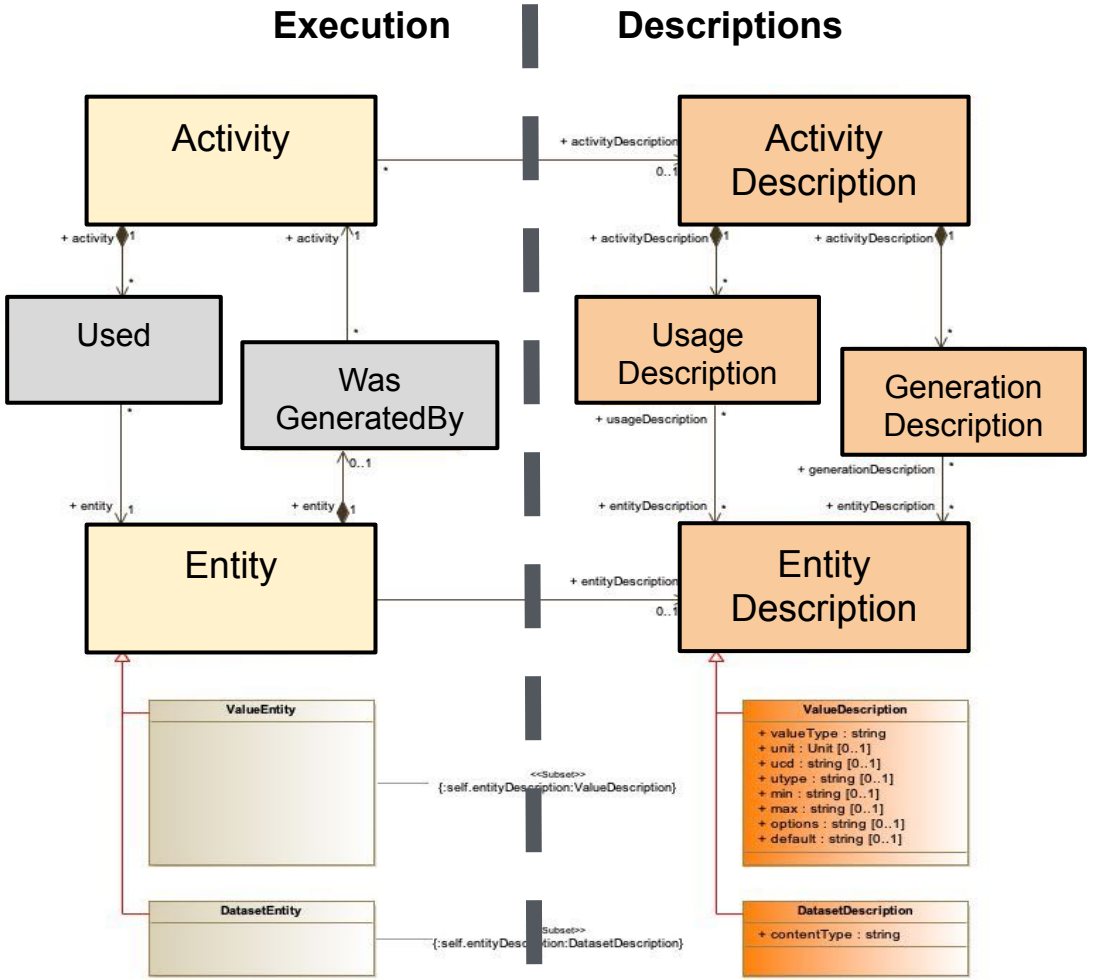
- **role** (master_bias, IRF, eventlist, ...)
- **type** (main, calibration, preview, quality, log, context)

Value

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

Dataset

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



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

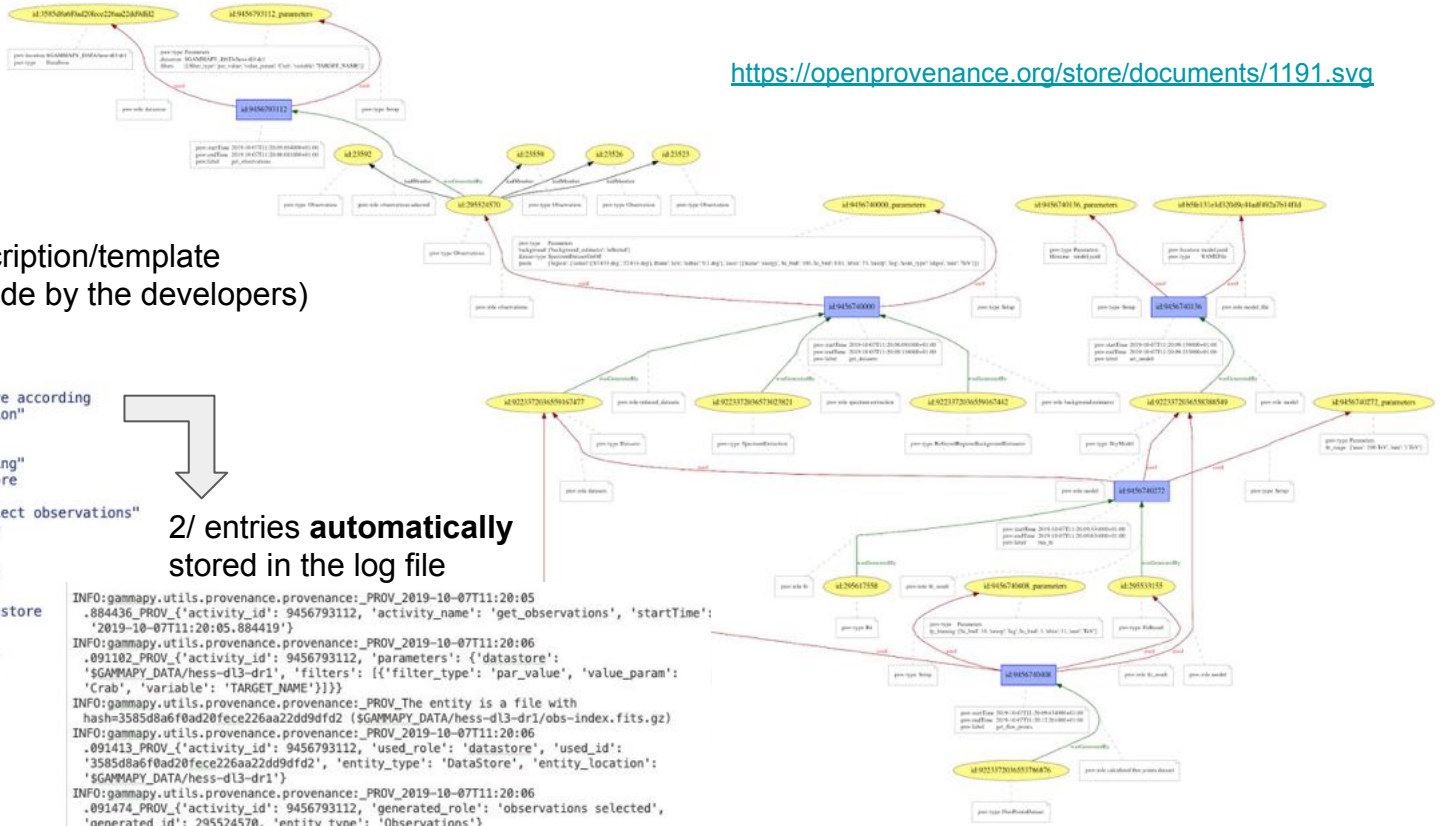
- **Capture**
  - **OPUS**: 1 = 1 job, returns W3C files and graphs
  - `ctapipe`: 1 activity = 1 Tool, returns a dictionary
  - **`gammapy`: 1 activity = 1 high level interface function, returns a structured log**
  - **OPUS + `gammapy`**: granularity mix ! transfer of identifiers !
- **Storage**
 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

Gammapy

- A community-developed, open-source Python package for gamma-ray astronomy: https://docs.gammapy.org/
- **High Level Interface**
  - Analysis class with methods
    - get_observations()
    - set_model()
    - run_fit()
    - …
- **Provenance capture: https://github.com/Bultako/gammapy/tree/prov/gammapy/utils/provenance**
  - non-intrusive: class decorator to trace all methods of the Analysis class
  - log each prov events as a structured dictionary
  - descriptions in a `definition.yaml` file (activity description, parameters, usage, generation…)
- Démo...

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

Implementation details

- 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 ?