

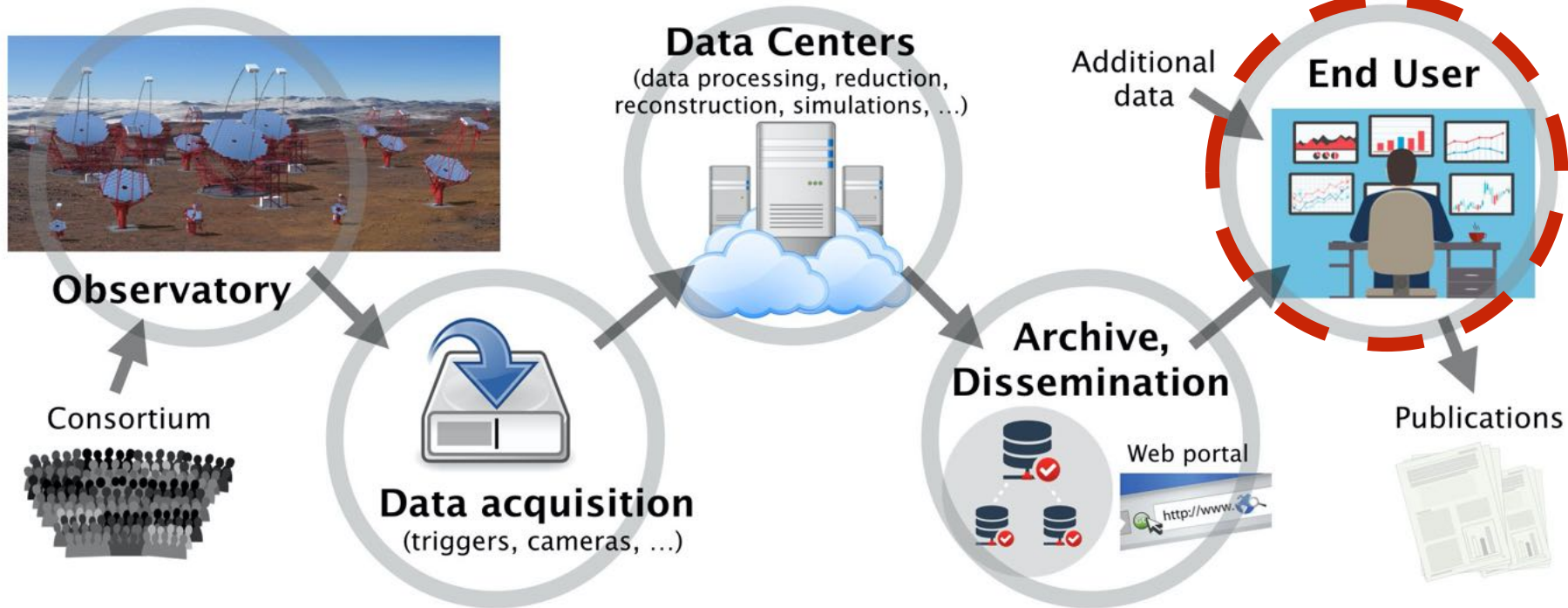
Provenance en pratique



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

Astronomy context for data generation

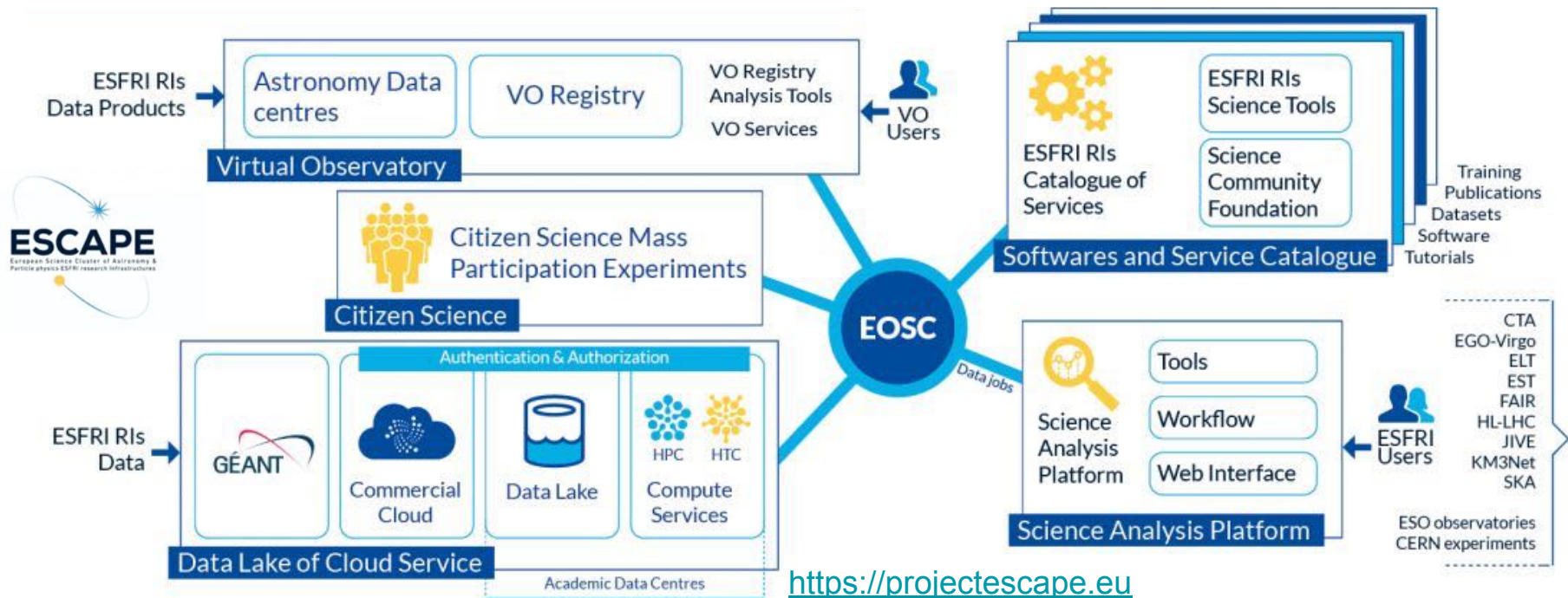
Data product generation **obscure** to end user



Need structured and detailed provenance information!

EOSC – The **E**uropean **O**pen **S**cience **C**loud is a cloud for research data in Europe allowing for universal access to data;

ESCAPE – « The **E**uropean **S**cience **C**luster of **A**stronomy & **P**article Physics **E**SFRI Research Infrastructures » answers the EOSC ambition in bringing People, Data, Services, Training, Publications, Projects & Organisations, all together in an integrated and federated environment.



FAIR Principles

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

FINDABLE

Unique identifiers and metadata are used to allow data to be located quickly and efficiently



ACCESSIBLE

Data is open, free and universally available for research discovery efforts



INTER-OPERABLE

A common programming language is used to allow use in a broad range of applications



REUSABLE

All data is clearly described and outlines associated data-use standards



FAIR principles for data sharing

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

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**

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

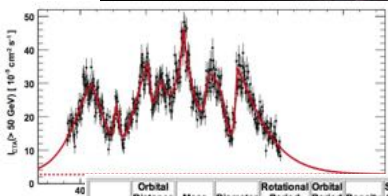
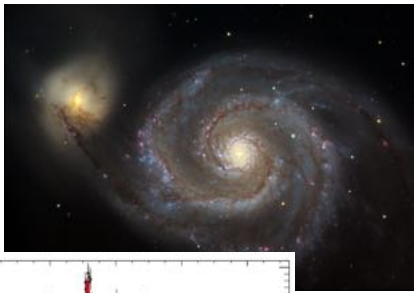
Editor(s):

Mathieu Servillat

Thing

Provenance of a thing? history, trace, chain of custody, location, ownership...

provenance = **origin** (where does it come from?) and **path** (what has been done?)
gives confidence on the quality/reliability/trustworthiness of an entity (e.g. a data product).



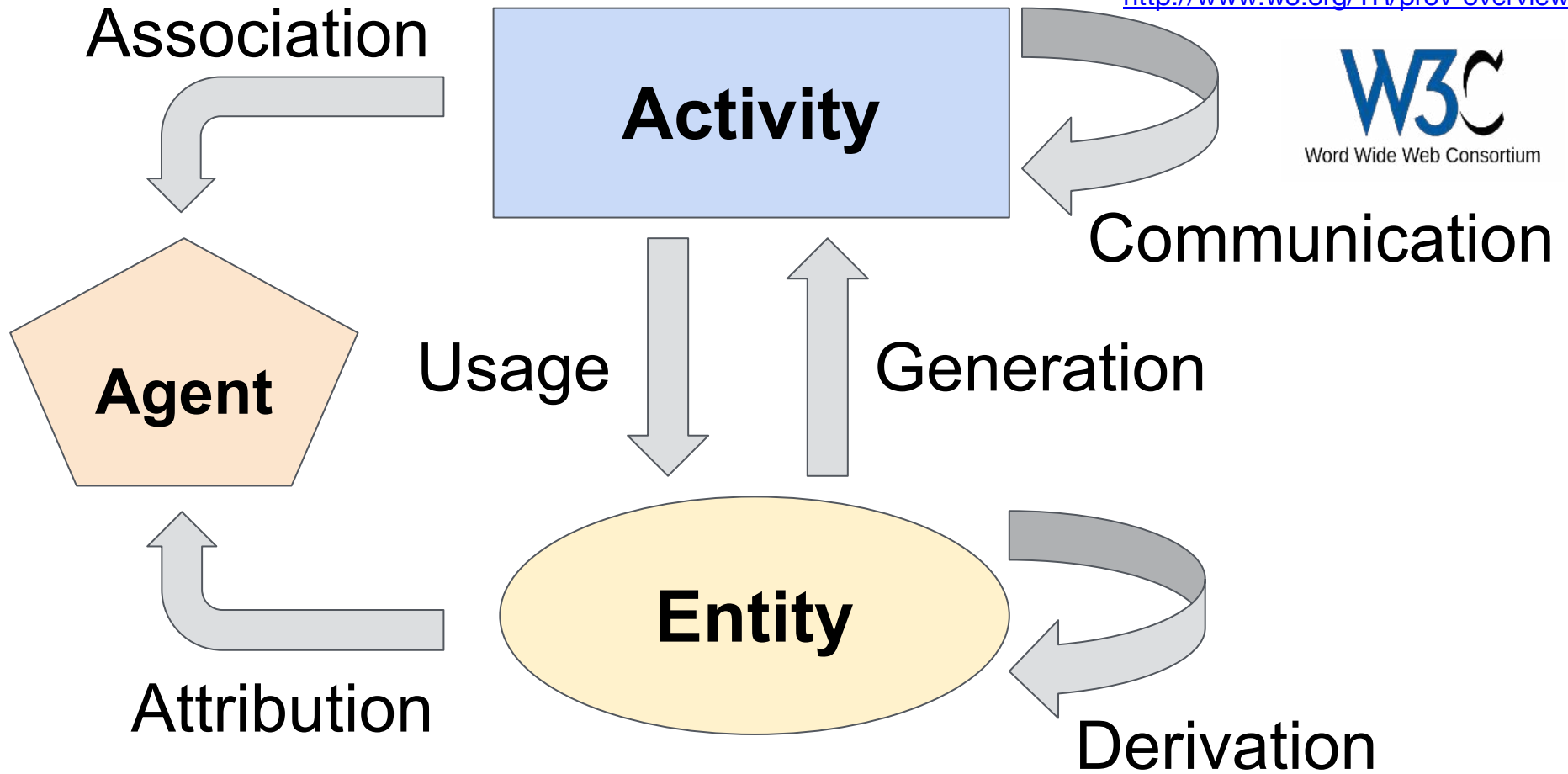
	Orbital Distance (AU)	Mass (earths)	Diameter (earths)	Rotational Period (days)	Orbital Period (years)	Orbital Density (earths)	Surface Gravity (earths)	Moons
Sol	0.0	330,000	109.2	25.4	—	1.42	28	—
Mercury	0.4	0.06	0.38	59	0.24	0.98	0.38	0
Venus	0.7	0.81	0.95	243	0.62	0.95	0.90	0
Earth	1.0	1.00	1.00	1.00	1.0	1.00	1.00	1
Mars	1.5	0.11	0.53	1.03	1.9	0.71	0.38	2
(Ceres*)	2.8	0.00015	0.07	0.38	4.6	0.38	0.03	0
Jupiter	5.2	317.8	11.2	0.42	11.9	0.24	2.34	63
Saturn	9.5	95.2	9.4	0.44	29.4	0.12	1.16	60
Uranus	19.2	14.5	4.0	0.72	83.7	0.23	1.15	27
Neptune	30.1	17.2	3.9	0.67	163.7	0.30	1.19	13
(Pluto*)	39.4	0.002	0.18	6.40	248.0	0.37	0.04	3
(Eris*)	67.7	0.0027	0.18	—8	557	?	?	1



Entity



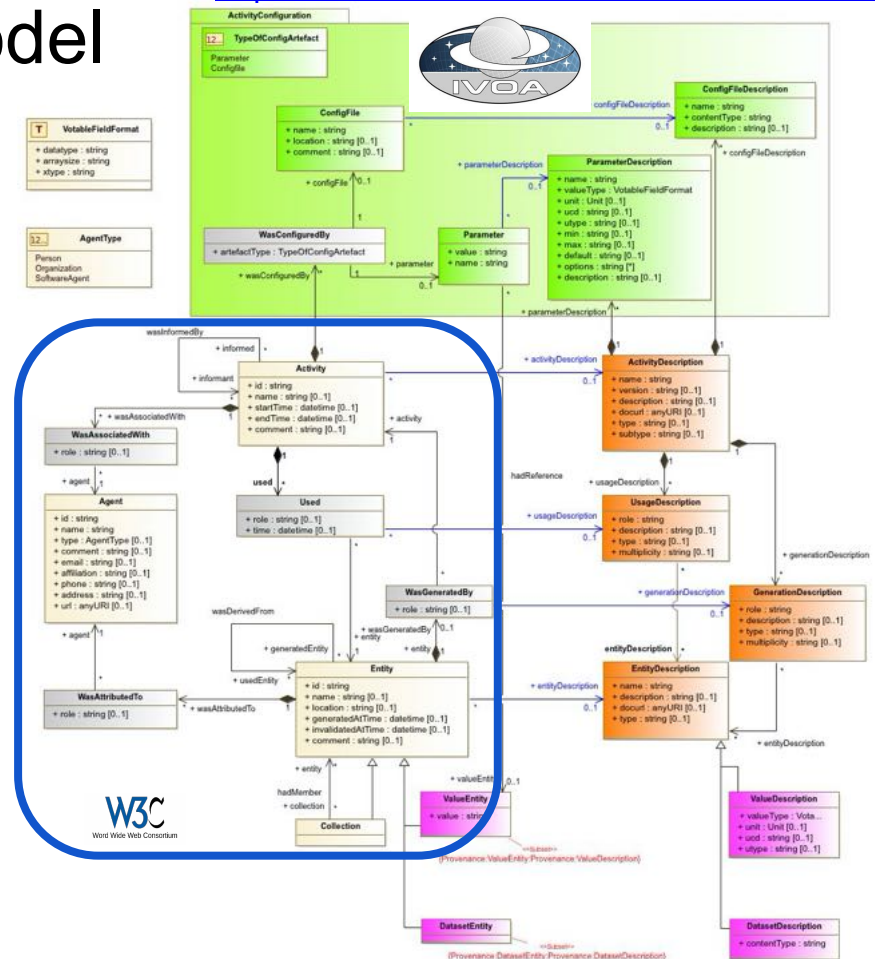
Derivation



IVOA Provenance Data Model

Recommendation in April 2020

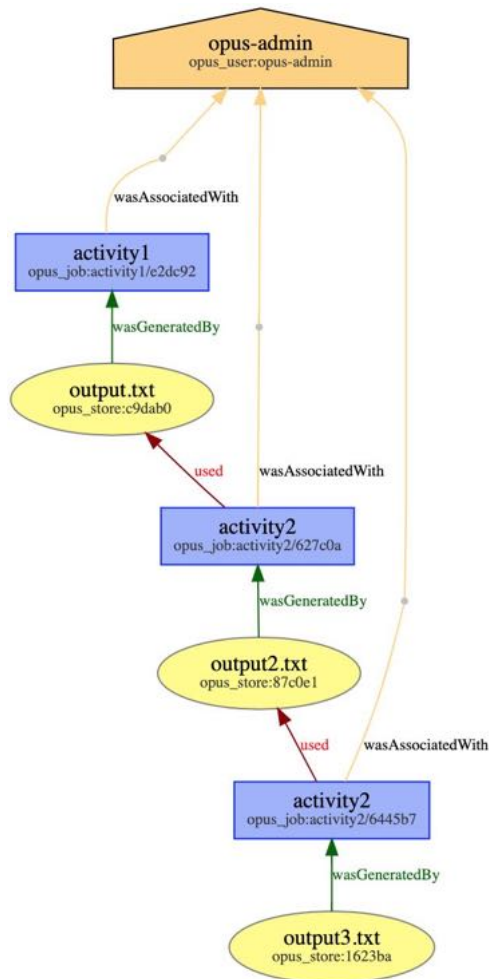
- Adds “**Description**” classes
- Adds “**Configuration**” classes
- Plugged in with
 - VO data models and concepts (UCD, VOUnit, VOTable...)
 - VO access protocols (ProvTAP, ProvSAP)
- Serializations
 - W3C PROV (XML, JSON, SVG...)
 - VO specific (VOTable)



Why recording structured provenance?

- **Make data FAIR** (Findable, Accessible, Interoperable, Reusable)
 - <https://www.go-fair.org/fair-principles/>
 - “rich” metadata, following standard data model, protocols and formats
 - “detailed provenance”
- **Quality / Reliability / Trustworthiness** of the products
- **Reproducibility requirement** in projects
 - Be able to rerun each activity (maybe testing and improving each step)
 - Not necessary to keep every intermediate file that is easily reproducible (possible gain on disk space and costs)
- **Debugging**
 - Not necessary to restart from scratch: locate in the provenance tree the faulty parts or the products to be discarded

→ We often realize too late that there are missing elements or links in the provenance. The capture of the provenance should be as detailed as possible and as naive as possible (simply record what happens).



Provenance graph

Provenance is :

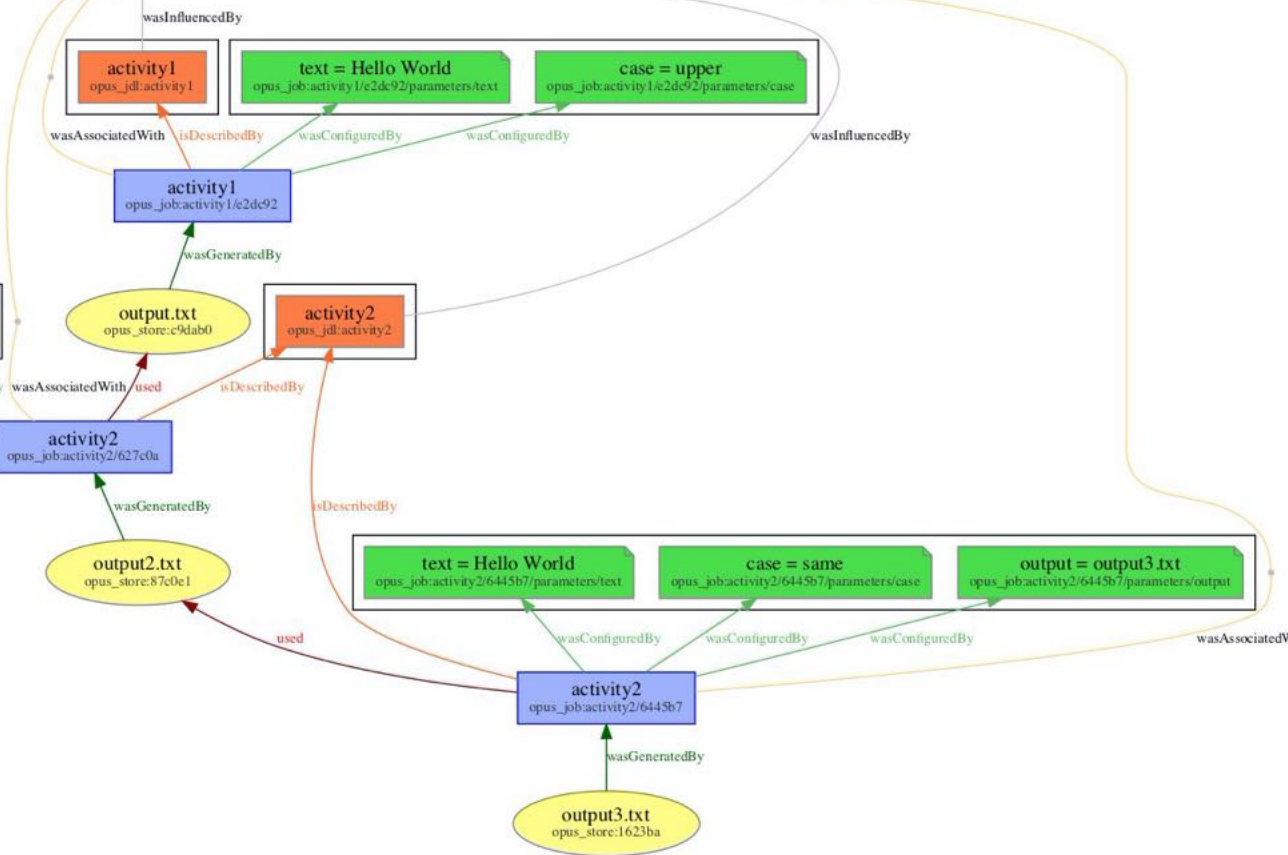
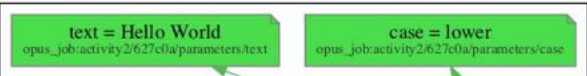
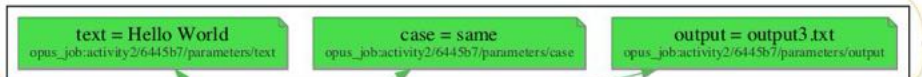
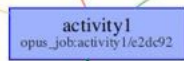
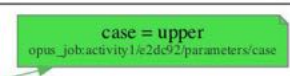
- a **chain** of activities and entities (used and generated)
- that occurred in the **past**

Using the **core data model**, some goals are achieved:

- Unique identifiers
- Traceability
- Contact and Acknowledgement

By using the **full IVOA data model**, more questions are answered:

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



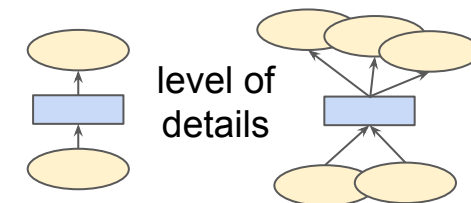
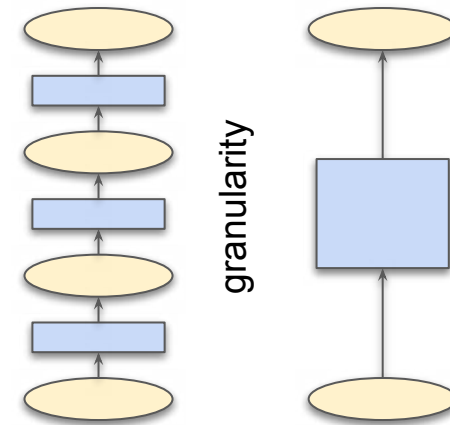
Applying the model

Different contexts in use cases

- Two flavours:
 - **on-top** (data products/collection already exist)
 - **inside** (save provenance information during the processing)
- **Identifiers**: unique and without meaning (if possible)
- **Granularity** (what steps? what objects?)
- **Level of details** (descriptions? configuration?)

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



Some terminology

See ADASS XXX proceedings : <https://arxiv.org/abs/2101.08691>

- **full provenance**: graph/tree/chain of activities and entities up to the raw data. This information is not hosted by the entities themselves (stored on an external server? as separate files?)
- **minimum/one-step provenance**: attached to an entity, list of keywords that gives some context and info on **last activity** (general process/workflow, software versions, contacts...).

Note: it would be interesting to include used entities, so that a full provenance may be reconstructed from each minimum provenance. However, such information on what was used may not be kept, or may not be complete.

- **end-user/specific “provenance”**: attached to an entity, list of keywords or data that provides **key information to use/analyse** the entity (e.g. for CTA: event class, event type, telescope configuration, sky conditions, reco method...)

Note: may be extracted from full provenance (some parameters or entities generated at a given step), but it is considered as data here. Reversely, this specific “provenance” information may be a source of information to be mapped in the standard in order to fill the full provenance graph with more details.

1. Defining the content of a **minimum provenance**
 - List of keywords related to the last activity and context
2. **Serializing** provenance
 - Both human and machine readable
 - Explore W3C formats, and YAML / VOTABLE / VOEvent formats
3. Provenance and **workflows**
4. From provenance "**on-top**" to provenance "**inside**"
 - How to map specific provenance information into the IVOA model?
 - How to introduce provenance **capture** inside a pipeline?
5. Provenance **storage**
6. Provenance **exploration** and **visualization**
 - Access protocols (ProvTAP, ProvSAP)
 - **voprov** Python package, or other tools

voprov Python package

- `prov` is a Python package that implements W3C PROV
 - <https://github.com/trungdong/prov>
- `voprov` extends `prov` to implement the IVOA ProvenanceDM
 - <https://github.com/sanguillon/voprov>
 - Development Lead: Jean-François Sornay, Michèle Sanguillon
 - Contributors: Mathieu Servillat, Mireille Louys, François Bonnarel, Catherine Boisson
- Main features
 - Description and Configuration classes and relations
 - Coloured graph to visualize IVOA objects
 - Conversion to a W3C graph

ProvSAP: Simple Access Protocol

- Simple HTTP endpoint
- Query string with arguments:
 - **ID** = a9b7e2 (activity or entity)
 - **DEPTH** = ALL / 1..
 - **DIRECTION** = FORWARD / BACKWARD
 - **RESPONSEFORMAT** = PROV-SVG / PROV-JSON / PROV-XML
 - **MODEL** = IVOA / W3C
 - **AGENTS** = 0 / 1
 - **CONFIGURATION** = 0 / 1
 - **DESCRIPTIONS** = 0 / 1 / 2
 - **ATTRIBUTES** = 0 / 1
- <https://voparis-uws-test.obspm.fr/provsap?ID=a9b7e2&DESCRIPTIONS=1&CONFIGURATION=1&ATTRIBUTES=0&DEPTH=ALL&MODEL=IVOA>

See ADASS XXX proceedings on OPUS : <https://arxiv.org/abs/2101.08683>

logprov Python package

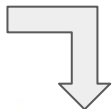
- **logprov**: <https://github.com/mservillat/logprov>
- Capture provenance transparently
 - For Python tools (in particular data analysis on a laptop)
 - Make provenance info compatible between different software (ctapipe, gammapy, ...)
 - Minimum effort from the developers, reuse existing tools
- Development drivers
 - Use **logging** system
 - known by developers, already integrated (base Python package)
 - **log dictionaries** to structure the information
 - **Non-intrusive** addition of code
 - use **decorators** for classes and methods
 - Fill with useful information
 - **descriptions** in a `definition.yaml` file
 - log parameters, arguments...

Provenance in gammapy

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

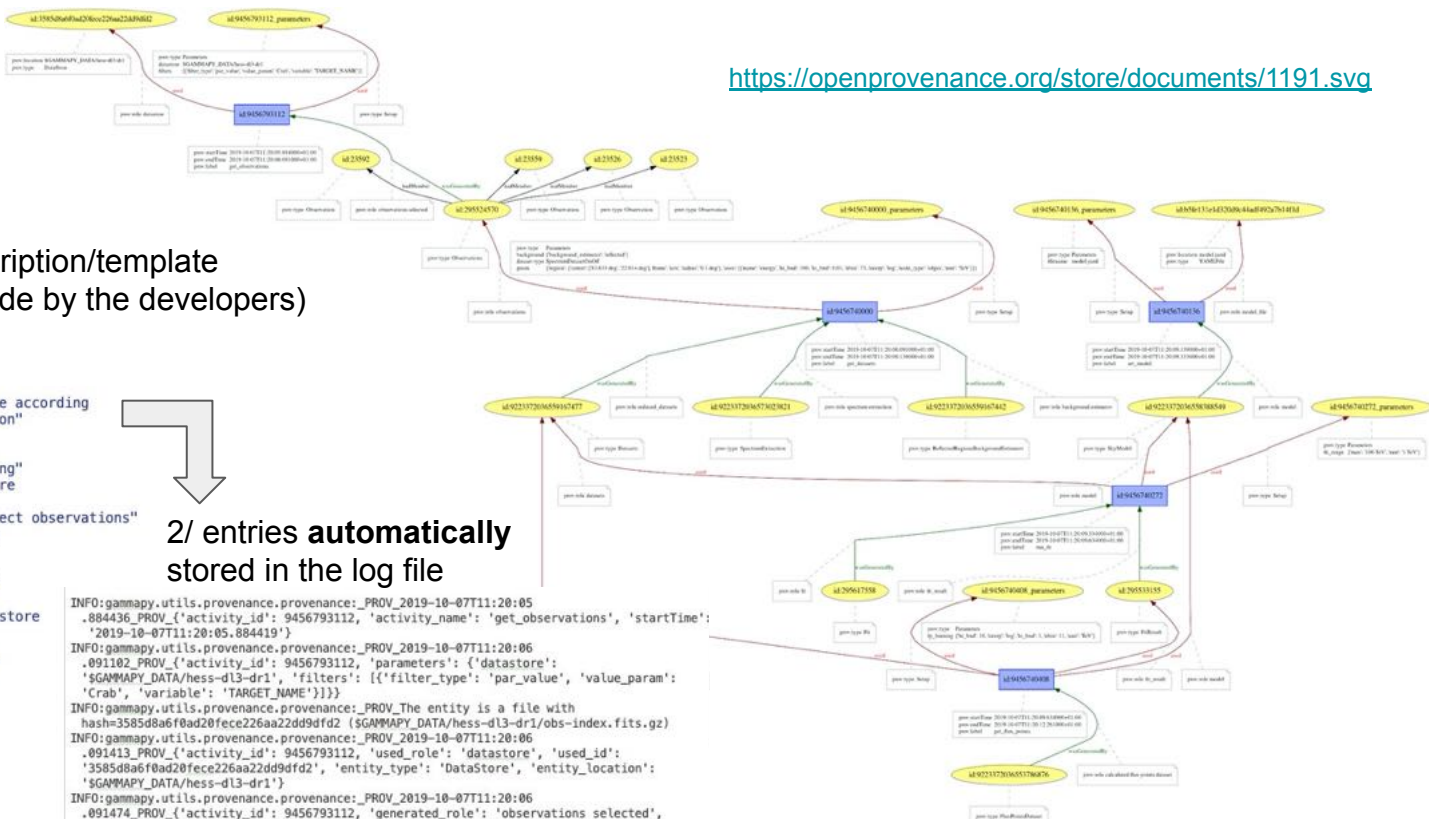
1/ definition.yaml file for description/template (already **integrated** to the code by the developers)

```
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':
'SGAMMAPY_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':
'3585d8a6f0ad20fece226aa22dd9dfd2', 'entity_type': 'DataStore', 'entity_location':
'SGAMMAPY_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_role': 'observations selected', '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



Serializations

- **W3C PROV standards + prov/voprov**
 - XML & JSON, not easy to read by humans : lists of classes (entity, activity, agent, used, ...)
 - But visualizations possible : SVG, PNG, PDF
- **YAML proposed solution** 
- Machine readable **and** human readable
- Structure with only 3 main groups :
 - activities : contains the links to reconstruct the chain of activities-entities
 - entities : contains information on how to find the entities and what they are
 - agents : contains contact information only (no roles)
- **Relations** are attached to the above objects
- **See example of Vizier** (talk by Gilles Landais)
 - Model Mapping in VOTable with similar structure

```
agents:
  mservillat:
    name: MS
    email: m..@obspm.fr
entities:
  1234:
    name: input.txt
  5678:
    name: output.txt
    generatedAtTime: 2021-...
activities:
  9876:
    name: transform
    startTime: 2021-...
    endTime: 2021-...
    parameters:
      <name>: <value>
    used:
      - entity_id: 1234
        role: input
    generated:
      - entity_id: 5678
        role: output
    associated:
      - agent_id: mservillat
        role: operator
```

Minimum/One-step provenance

- List of keywords
- Attached to an entity
- Gives info on last activity and context
- Enables full provenance reconstruction

⇒ Such a list is a restriction of the full provenance information, that is more easily stored in a header (FITS) or a flat table (same as ObsCore)

⇒ Identifiers have to be “resolvable” to provide further provenance information

- entity
 - id
 - creation_time
 - name
 - location
 - comment
 - content_type
 - description
- contact
 - name
 - email
 - affiliation
- activity_description
 - name
 - version
 - docurl
- activity
 - id
 - name
 - type
 - start_time
 - stop_time
- used
 - [id]
- generated
 - [id]

ProvTAP and last step provenance

François Bonnarel @ IVOA Nov 2020:

<https://wiki.ivoa.net/internal/IVOA/InterOpNov2020DM/ProvTAPevolution.pdf>

- Provenance information can be attached to data in various ways:
 - Embedded in the data « header » itself
 - Linked to the data record via DataLink or URL
 - Retrievable via ProvSAP via data id
- ProvTAP allows to discover « data » by constraining provenance features
 - Table Access Protocol using ADQL for queries and a TAP Schema
 - It's a « reverse » mechanism
- Last step provenance
 - `select * from last_step_provenance where entity_name = "..."`
 - Provides the Minimum/One-step provenance for an entity