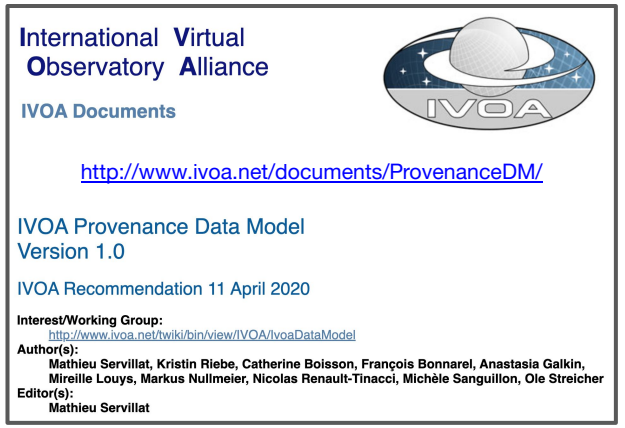
One-Step Provenance

IVOA virtual 2022-10

Mathieu Servillat (LUTH - Observatoire de Paris / CNRS)
Catherine Boisson, François Bonnarel, Mireille Louys
+ ESCAPE participants
+ CTA members

IVOA Provenance


International Virtual Observatory Alliance

IVOA Documents

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

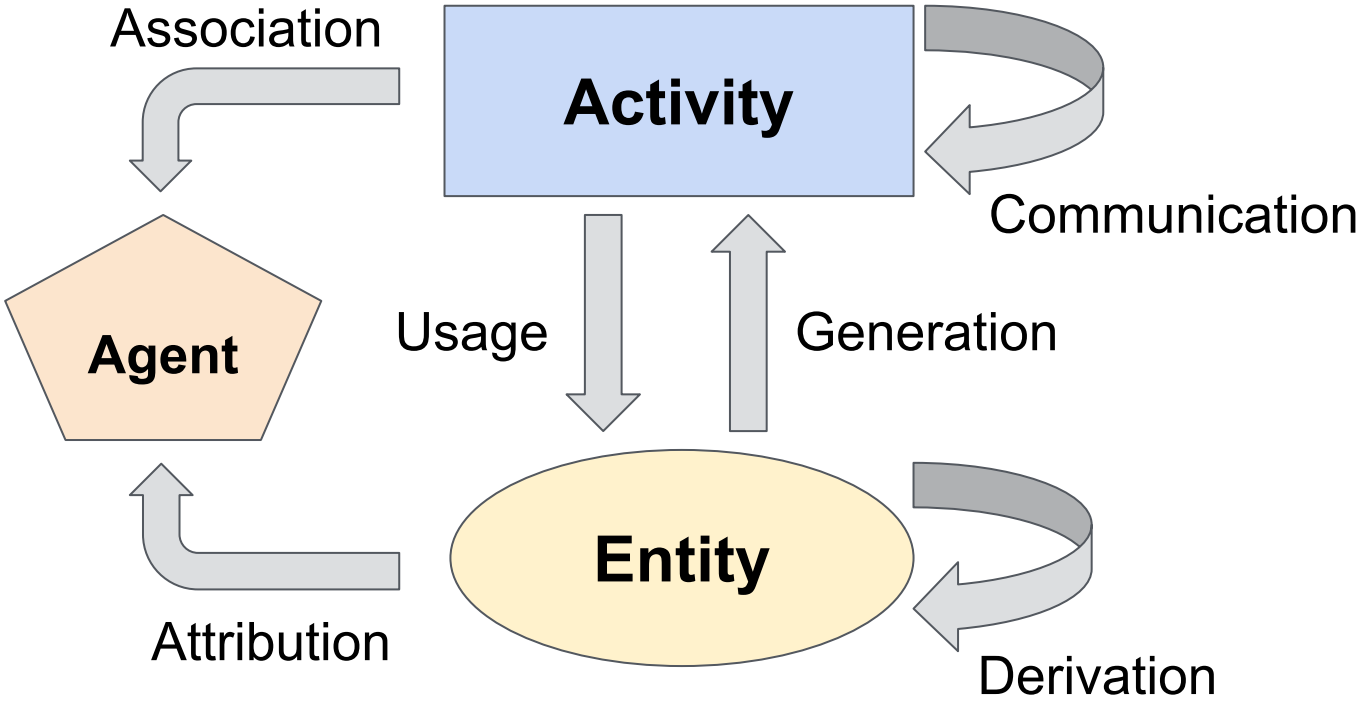
IVOA Provenance Data Model
Version 1.0

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 in April 2020, based on W3C PROV
- **ProvSAP**: simple access protocole
  - Request provenance graph for an identifier (entity/activity)
  - Takes advantage of the W3C serializations (JSON, XML, SVG, PNG…)
  - `voprov` Python package
  - Implementations : Pollux, OPUS (CTA, MASER, CompOSE), ...
    https://voparis-uws-test.obspm.fr/client/proxy/provsap?ID=a6018b&DESCRIPTIONS=1&CONFIGURATION=1&ATTRIBUTES=0&DEPTH=ALL
- **ProvTAP**: table access protocole
  - TAP Schema based on ProvenanceDM
  - Adding a few simplified views
  - Implementations : HiPS tiles provenance, …

Provenance glossary

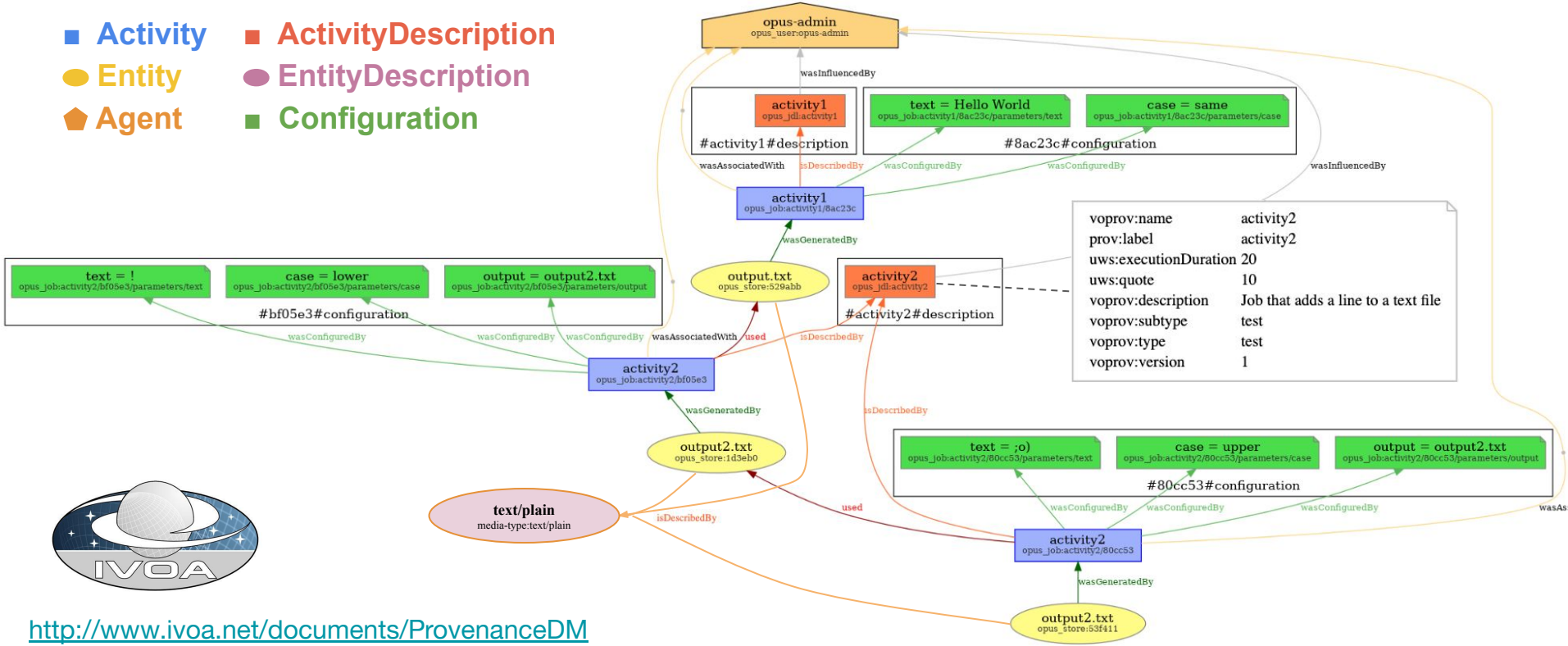


Word Wide Web Consortium

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

Full IVOA Provenance graph

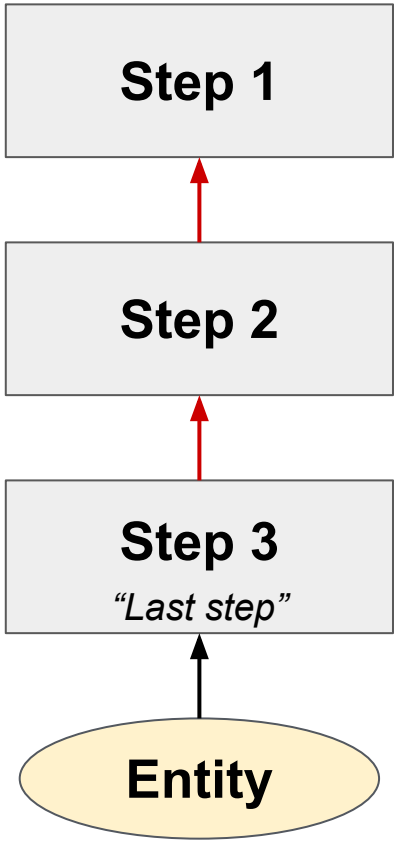
- Activity
- ActivityDescription
- Entity
- EntityDescription
- Agent
- Configuration



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

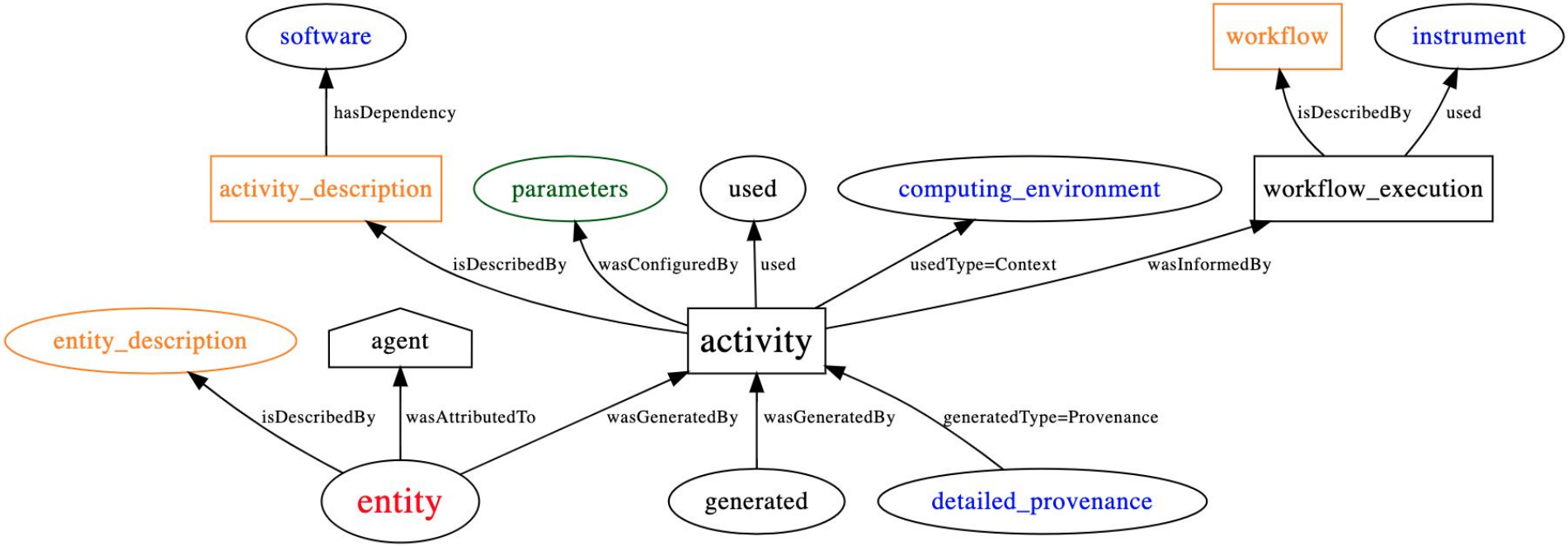
Provenance in practice

- Problematic
  - Provenance graphs are **complex**
  - Is there a **minimum**/**optimal** provenance?
- Use cases
  - CTA data products header
  - Workshops with ESCAPE partners, within ASOV
  - ADASS BoF session
- Requirements
  - Cite **software**
  - Record the **context** of execution
  - Include provenance attributes **inside** an entity
  - Handle different levels of **details**
- Definition of a “One Step Provenance”
  - **List of attributes** to describe one step of data generation
  - Links between steps using **identifiers**
  - The “Last step” may be embedded in the entity



One-Step Provenance Data Model

- **Subgraph** designed with IVOA Provenance concepts
- Applied to **digital object generation**
- Introduces **specialized entities** (software, computing environment, instrument)



Serialisation of a One-Step Provenance record

- Flat list of keywords
  - A few mandatory keywords (id)
  - other optional information to provide “detailed” provenance (see FAI**R** principles)
- Proposed FITS keywords
  - “Last step” provenance
  - embedded in the generated entity
- Identifiers are key
  - Resolved via a ProvSAP service

keyword	UType	FITS keyword
entity_id	Entity.id	ENT_ID
entity_location	Entity.location	ENT_LOC
entity_generatedAtTime	Entity.generatedAtTime	ENT_GTIM
entity_name	EntityDescription.name	ENT_NAME
entity_type	EntityDescription.type	ENT_TYPE
entity_content_type	EntityDescription.content_type	ENT_CTYP
entity_docurl	EntityDescription.docurl	ENT_DURL
entity_comment	Entity.comment	ENT_COMM
agent_id	Agent.id	AGT_ID
agent_name	Agent.name	AGT_NAME
agent_type	Agent.type	AGT_TYPE
agent_email	Agent.email	AGT_MAIL

...

Specialized entities: context and links

- entityType = Instrument
  - name
  - identifier
  - but maybe more advanced model
- entityType = Software
  - name
  - version
  - identifier (e.g. doi)
- entityType = Computing Environment
  - Operating System
  - Nodes, memory…
  - Variables (path, python, …)

- Granularity of provenance
  - Link to more detailed provenance
  - Level of detail to be defined
- Link to Full Provenance
  - Identifiers should be resolved through a ProvSAP service
  - URL to service to be provided with the one-step record
  - **Reconstruct** Full provenance from several one-step records

Example of implementation

- Analysis of Cherenkov data with gammapy using OPUS
  - OPUS: job manager based on UWS (https://opus-job-manager.readthedocs.io)
  - job definition = expected input/output + configuration parameters + description
- Automatic generation of **coarse** provenance
  - 1 job = 1 step
  - information from job definition and job execution (UWS attributes)
  - May not be complete !
- Link with **detailed** provenance ?
  - things happen “inside” the executed script
  - need to provide an **internal provenance** using the `voprov` Python package
- **Exchange** of provenance information from script to OPUS
  - Ensure that the coarse provenance provided by OPUS is **optimal**

Structured Serialisation of Provenance

- W3C formats
  - XML, JSON
  - structure by blocks
  - not easy to find an information
  - “storage only”
- Proposed YAML format
  - machine **and** human readable
  - include relations with entities/activities
  - Makes it easy to **manipulate** provenance

```
activity:
   <activity_id>:
      name: <activity_name>
      startTime: <activity_startTime>
      endTime: <activity_endTime>
      activity_description: <activity_name>
      parameters:
         <name>: <value>  # from <activity_parameters>
         ...
      used:
         - <used_id>  # from <used_ids>
         - ...
      generated:
         - <generated_id>  # from <generated_ids>
         - ...
      informed:
         - <workflow_id>
   <workflow_id>:
      comment: <workflow_comment>
      activity_description: <workflow_name>
...
```