IVOA Provenance Data Model
Version 1.0

IVOA Working Draft 2018-05-30

Working group
DM

This version
http://www.ivoa.net/documents/ProvenanceDM/20180530

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

Previous versions
WD-ProvenanceDM-1.0-20170921.pdf
WD-ProvenanceDM-1.0-20161121.pdf
ProvDM-0.2-20160428.pdf
ProvDM-0.1-20141008.pdf

Author(s)
Kristin Riebe, Mathieu Servillat, François Bonnarel, Anastasia Galkin, Mireille Louys, Markus Nullmeier, Florian Rothmaier, Michèle Sanguillon, Ole Streicher, IVOA Data Model Working Group

Editor(s)
Kristin Riebe, Mathieu Servillat

Previous draft content

1 Introduction

1.1 Goal of the provenance model

1.2 Minimum requirements for provenance

1.3 Role within the VO architecture

1.4 Previous efforts

2 The provenance data model

2.1 Overview: Conceptional UML class diagram and introduction to core classes

2.2 Model description

2.2.1 Class diagram and VO-DML compatibility

2.2.2 Entity and EntityDescription

2.2.3 Collection

2.2.4 Activity and ActivityDescription

2.2.5 ActivityFlow

2.2.6 Entity-Activity relations → WasDerivedFrom? WasInformedBy?

2.2.7 Parameters

2.2.8 Agent

3 Links to other data models

3.1 Links with Dataset/ObsCore Model

3.2 Links with Simulation Data Model

Moved to the Appendices

4 Serialization of the provenance data model

4.1 Introduction

4.2 Serialization formats: PROV-N, PROV-JSON and PROV-XML

~~4.3 PROV-VOTable format~~ comes with ProvTAP

4.4 Serialization of description classes in the data processing context

4.5 W3C PROV-DM compatible serializations

5 Accessing provenance information

5.1 Access protocols

5.2 ProvDAL

5.2.1 ProvDAL example use cases

5.3 ProvTAP

5.4 VOSI availability and capabilities

—> DAL document

6 Use cases – applying the data model

6.1 How to use the data model

6.2 voprov Python package

6.3 Provenance of RAVE database tables

6.4 Provenance for CTA

6.5 Provenance for the POLLUX database

6.6 Provenance of HiPS datasets

Appendices —> Implementation note

+ consistent VO vocabulary, map with external ID

New draft content

1 Introduction

1.1 Goal of the provenance model

1.2 Minimum requirements for provenance

1.3 Role within the VO architecture

1.4 Previous efforts

2 The provenance data model

2.1 Overview: Conceptional UML class diagram and introduction to core classes

2.2 Model description

2.2.1 Class diagram and VO-DML compatibility

2.2.2 Entity and EntityDescription

2.2.3 Collection

2.2.4 Activity and ActivityDescription

2.2.5 ActivityFlow → **Postponed to next version**

2.2.6 Entity-Activity relations

2.2.7 Parameters → **Open question on the modelling**

2.2.8 Agent

3 Serialization of the provenance data model

3.1 Introduction

3.2 Serialization formats: PROV-N, PROV-JSON and PROV-XML → **Really needed?**

3.3 PROV-VOTable format

3.4 Serialization of description classes for web services

3.5 W3C PROV-DM compatible serializations → **Can all IVOA contents fit in W3C serializations?**

4 Accessing provenance information

Appendix A Serialization Examples

Appendix B Links to other data models

B.1 Links with Dataset/ObsCore Model

B.2 Links with Simulation Data Model

Use cases

- CTA (Cherenkov Telescope Array) pipeline and data access
- RAVE (Radial Velocity Experiment)
- POLLUX (synthetic stellar spectra service)
- SVOM gamma ray burst / transients
- TAP-based API for images in an archive @CDS
- MuseWise pipeline

⇒ Different aspects of Provenance

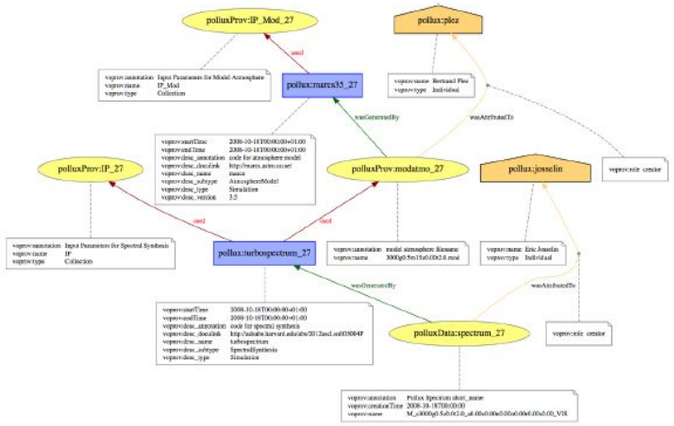
- How to **collect** the provenance information
- How to **store** this information
- How to **access** and **visualize** the provenance

Use cases and implementations

VOPROV LIBRARY

The voprov package is an open source Python library which allows users to serialize their provenance information in different formats: PROV-N, JSON, XML, VOTable or in graphical ones: PNG, SVG,PDF.

(Cf https://github.com/sanguillon/voprov/)



This package is used in the context of **Pollux**.

Pollux is a stellar spectra database proposing access to high resolution synthetic spectra computed using the best available models of atmosphere and efficient spectral synthesis codes.

DJANGO PACKAGE

The Django provenance package is an open source Python package that can be reused in Django web applications for serving provenance information via a ProvDAL and a REST interface. The data model classes are directly mapped to tables in a relational database tables. It supports IVOA as well as W3C serializations into PROV-JSON and PROV-N formats.

(Cf https://github .com/kristinriebe/django-prov_vo)

This package is used in the context of **RAVE**.

The RAVE (RAdial Velocity Experiment) is a survey that observed the spectra of half a million stars from the southern hemisphere.

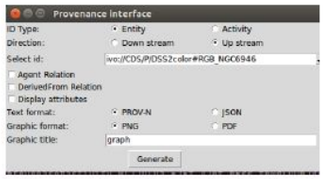
In a pipeline of several steps the data were calibrated, reduced and stellar properties were determined, which were then released in the form of star catalogues.

Prototype PostGreSQL database at CDS

In the CDS prototype we implemented a PostGreSQL database for Provenance information attached to image datasets. A database schema has been designed from the IVOA Provenance DM and implemented.

A set of images, together with their digitization and extraction steps, RGB color composition and HiPS generation activities are fed to the database. Various scenarii for querying and displaying the Provenance information have been tested. PROV-N, PROV-Json and PROV-VOTable formats for the response are provided for the query response.

A simple user interface allowing to select the main types of requests and to display the responses via W3C Prov software has been designed. It allows querying for various combinations of Provenance relationships in the database.



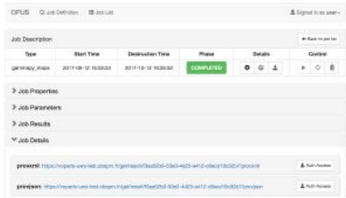
UWS Server at Observatoire de Paris

OPUS (Observatoire de Paris UWS System) is an open source job control system based on the IVOA UWS pattern.

It is developed in the context of the Cherenkov Telescope Array (CTA) project to test the execution of CTA data analysis tools on a work cluster.

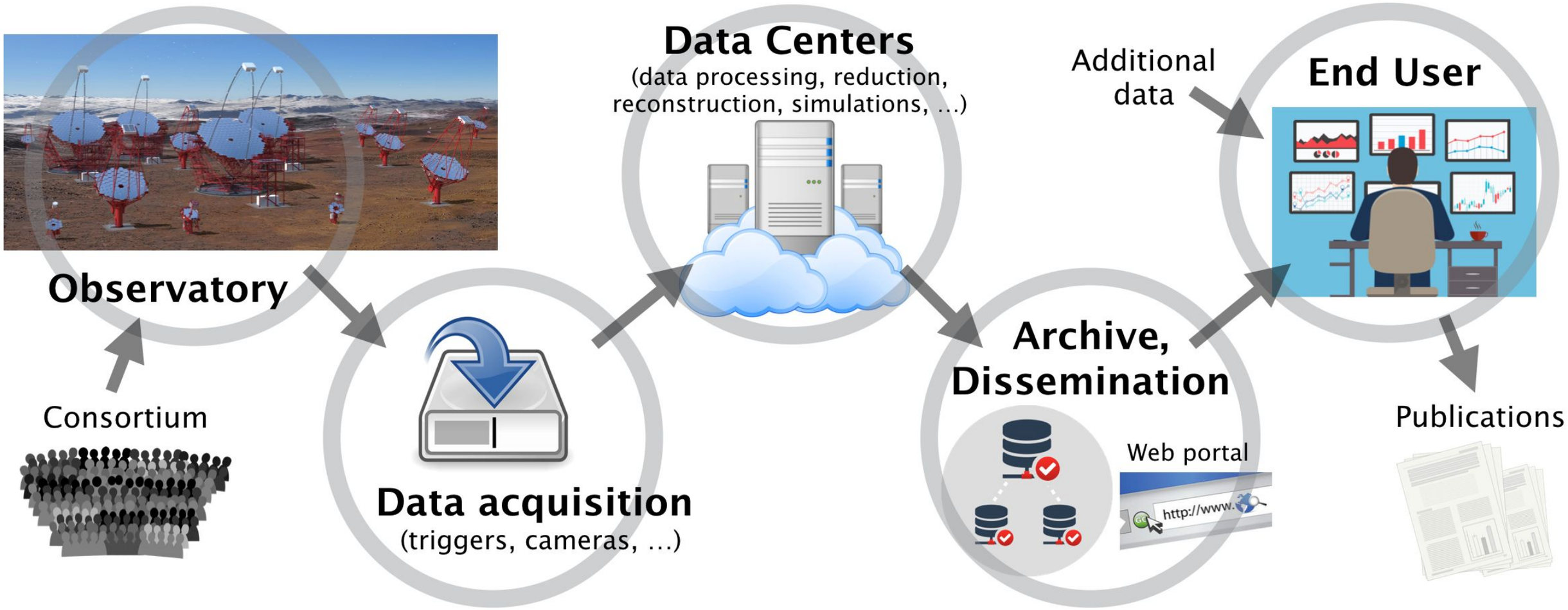
It implements the concept of ActivityDescription files and provides the serialized provenance information as files for each executed job (see also ADASS Poster p3822).

(Cf https://github.com/mservillat/OPUS)



The **CTA** is the next generation ground-based very high energy gamma-ray instrument. It will serve as an open observatory providing data to a wide astrophysics community, with the requirement to propose self-described data products to users with detailed provenance information.

Objectives and context



- Data product generation **obscure** to end user
- **Quality**, **reliability**, **trustworthiness**?
- **Usefulness** of the data?

Need **structured** and **detailed** provenance information

Goals

A: Tracking the production history

Find out which steps were taken to produce a dataset and list the methods/tools/software that was involved.

B: Attribution and contact information

Find the people involved in the production of a dataset, that need to be cited or can be asked for more information.

C: Locate error sources

Find the location of possible error sources in the generation of a dataset.

D: Quality assessment

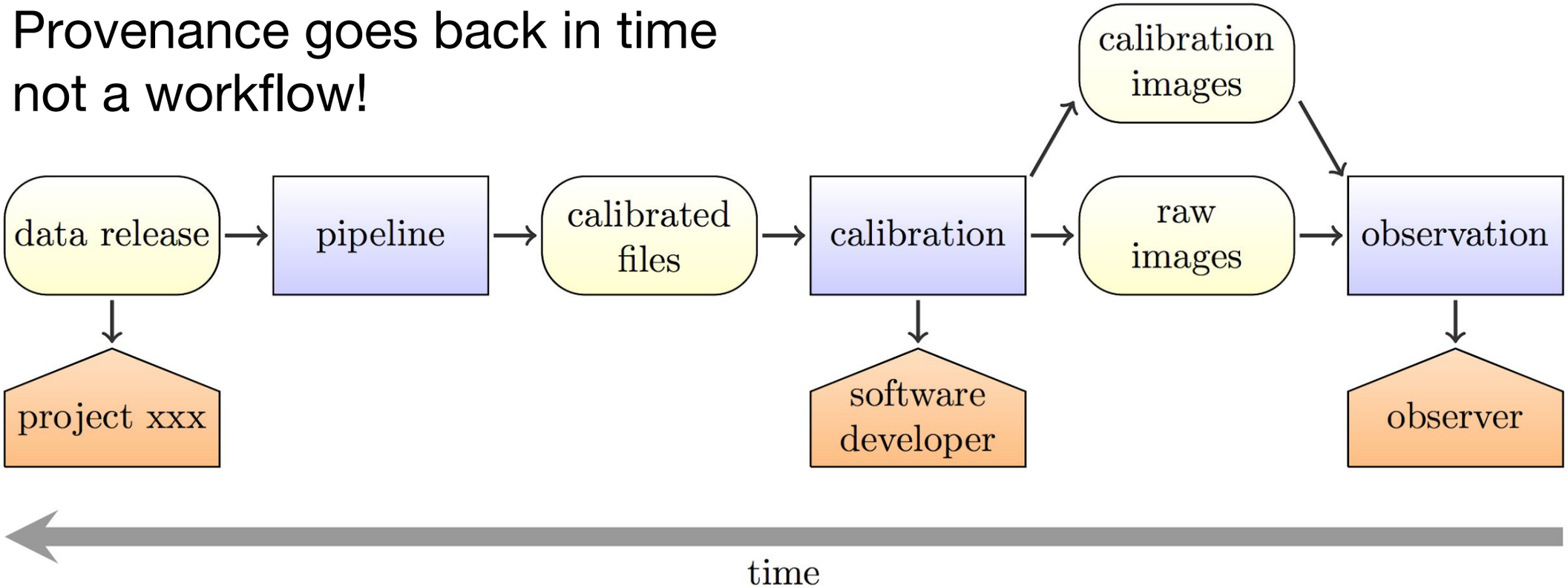
Judge the quality of an observation, production step or dataset.

E: Search in structured provenance metadata

This would allow one to also do a “forward search”, i.e. locate derived datasets or outputs.

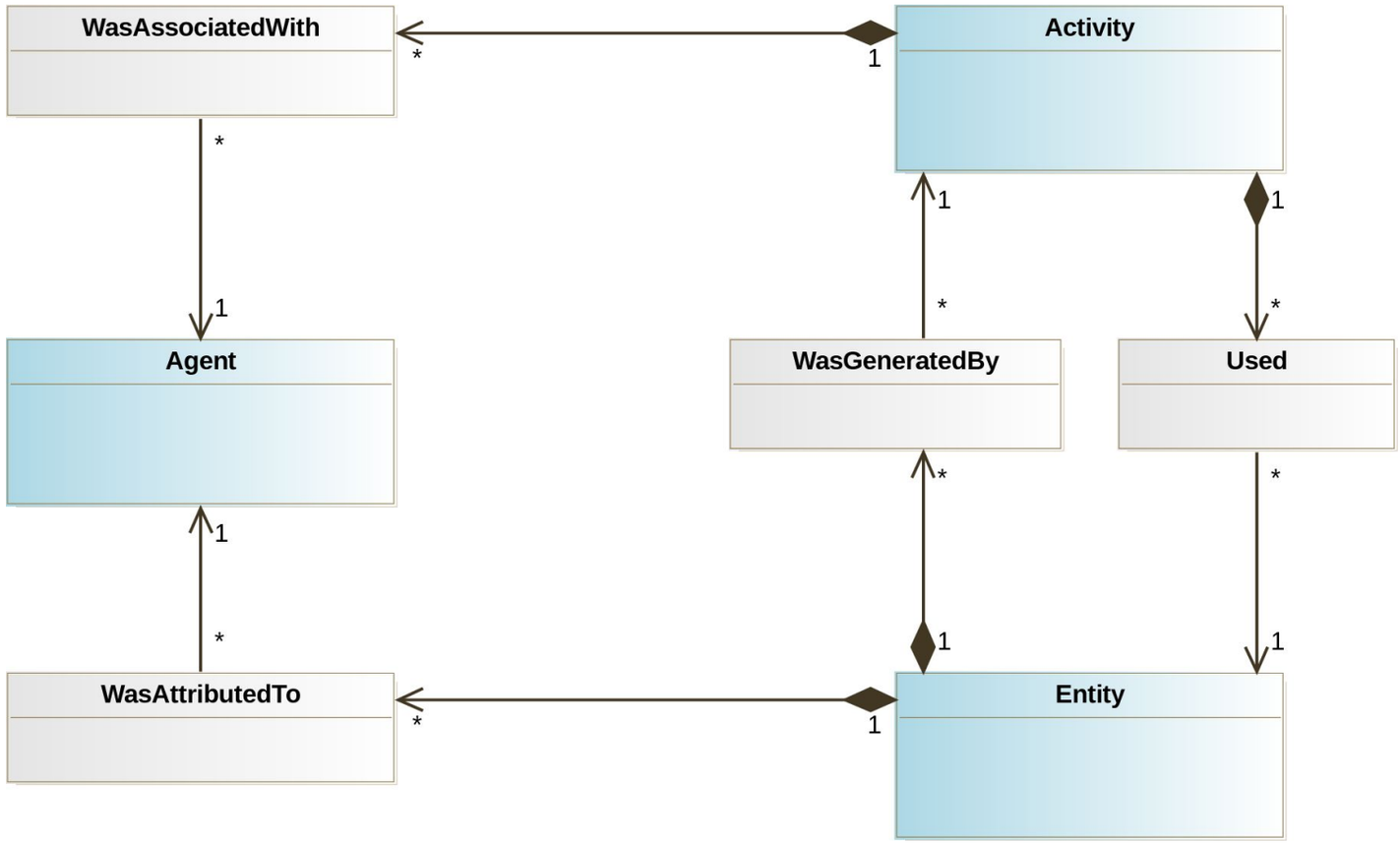
What is provenance?

Provenance goes back in time
not a workflow!



- **Provenance** = Identify how a data product was produced
- **Configuration** = Identify what detailed options were used
- **Contextual** information:
  - Instrument Configuration
  - Ambient Conditions
  - Software environment

Core Provenance Data Model



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

30 April 2013

Word Wide Web Consortium

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

- Core concepts from the W3C PROV recommendations
  - **Entity - Activity - Agent**
  - **Relations** and **roles** = provenance information
  - W3C PROV has many more relations
  - IVOA Provenance connected to **VO concepts** and **astronomy needs**

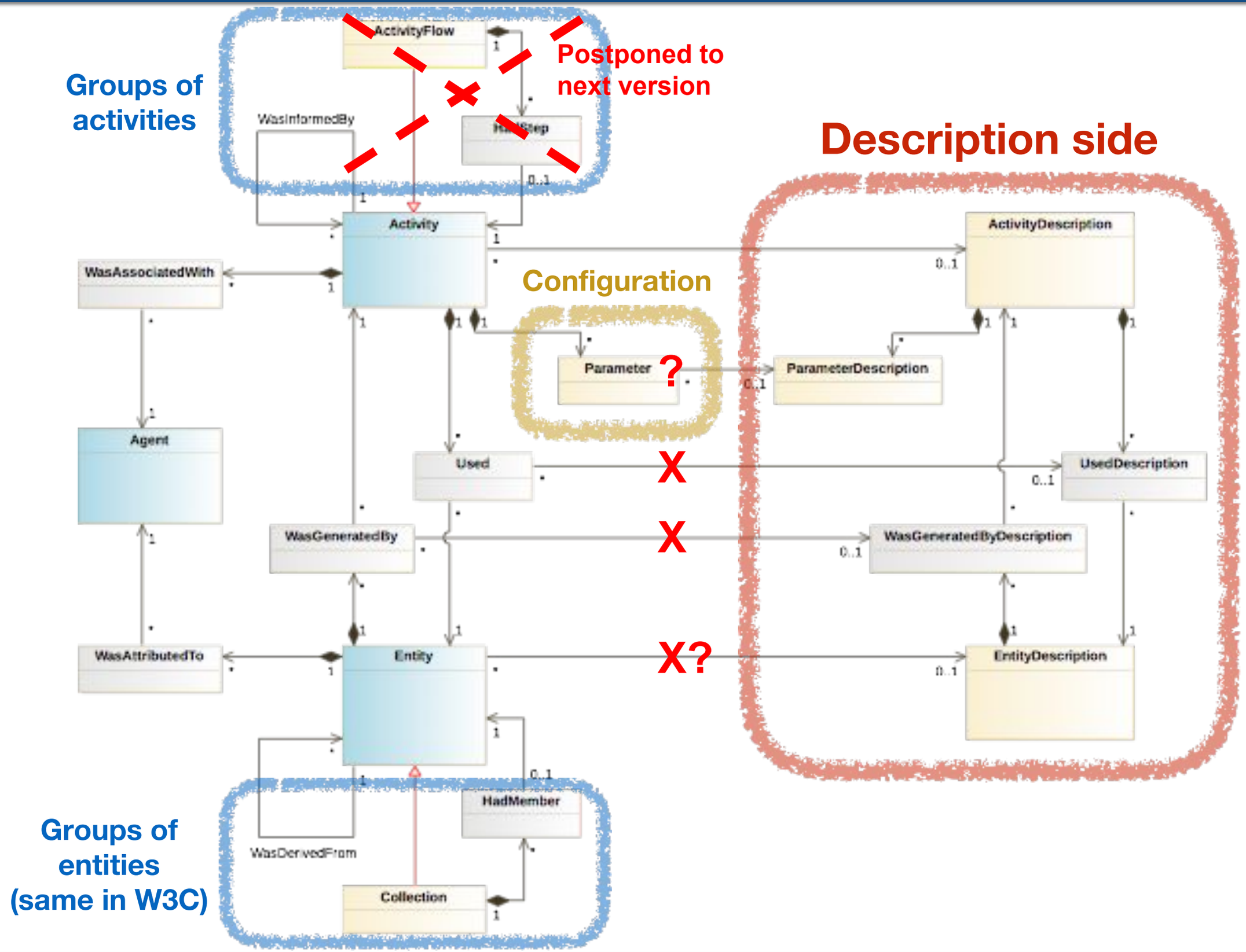
Concepts In Astronomy

- **Entities**: datasets composed of VOTables, FITS files or database tables, or files containing logs, values (spectra, lightcurves), parameters, etc.
- **Activities**: an observation, a simulation, or processing steps (image stacking, object extraction, etc.).
- **Agents**: the people involved can be individual persons (observer, publisher...), groups or organisations.
- **Connections to existing VO concepts**
  - Entity <—> Dataset (Curation, DataID), ObsCore, SimDM DataObject
  - Activity <—> SimDM (Resource, Experiment)
  - Agent <—> Party, Contact
- **Connections to external concepts** (PROV, DOI, ORCID, …)

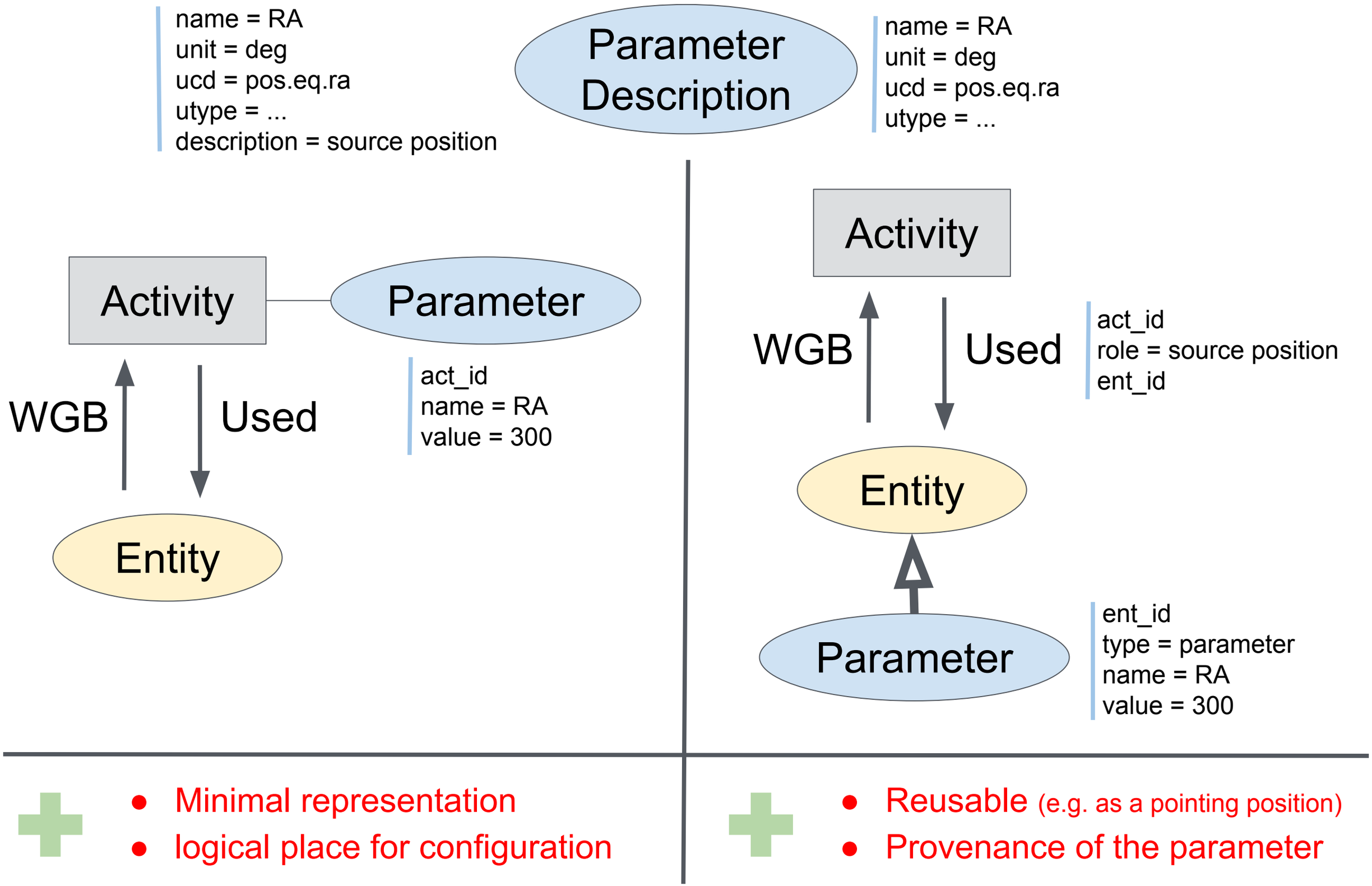
Minimum requirements

1. Provenance information must be stored in a **standard model**, with **standard serialization formats**.
2. Provenance information must be **machine readable**.
3. Provenance data model classes and attributes should be **linked to other IVOA concepts** when relevant (DatasetDM, ObsCoreDM, SimDM, VOTable, UCDs...).
4. Provenance information should be **serializable into the W3C provenance standard formats** (PROV-N, PROV-XML, PROV-JSON) with minimum information loss.
5. Provenance metadata must contain information to find immediate **progenitor(s)** (if existing) for a given entity, i.e. a dataset.
6. An entity must point to **the activity that generated it** (if the activity is recorded).
7. Activities must point to **input entities** (if applicable).
8. Activities may point to **output entities**.
9. Provenance information should make it possible to derive the **chronological** sequence of activities.
10. Provenance information can only be given for **uniquely identifiable entities**, at least inside their domain.
11. Released entities should have a **main contact**.
12. It is recommended that all activities and entities have **contact information** and contain a (short) **description** or link to a description.

IVOA Provenance Data Model diagram



Parameter (section 2.2.7)

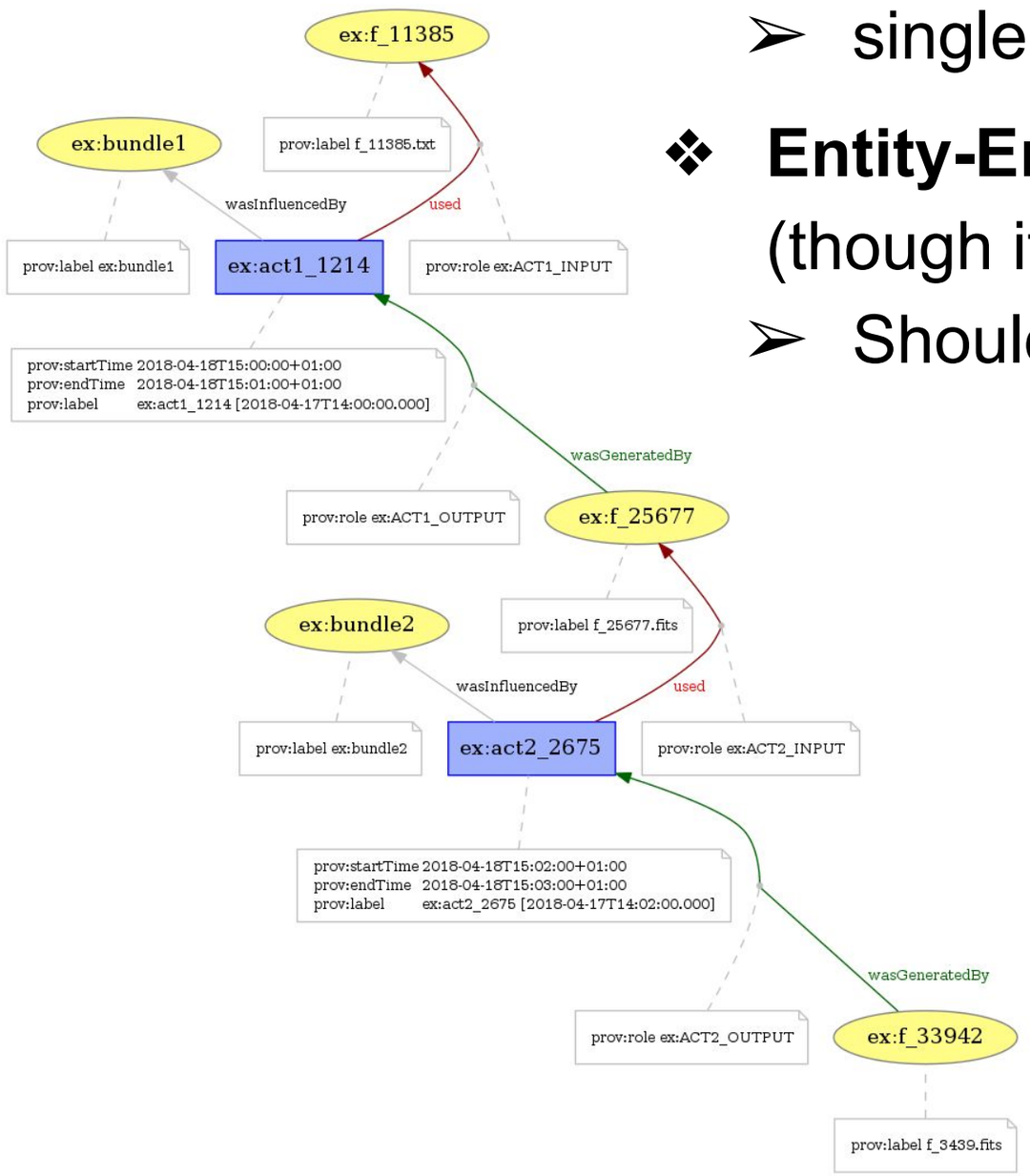


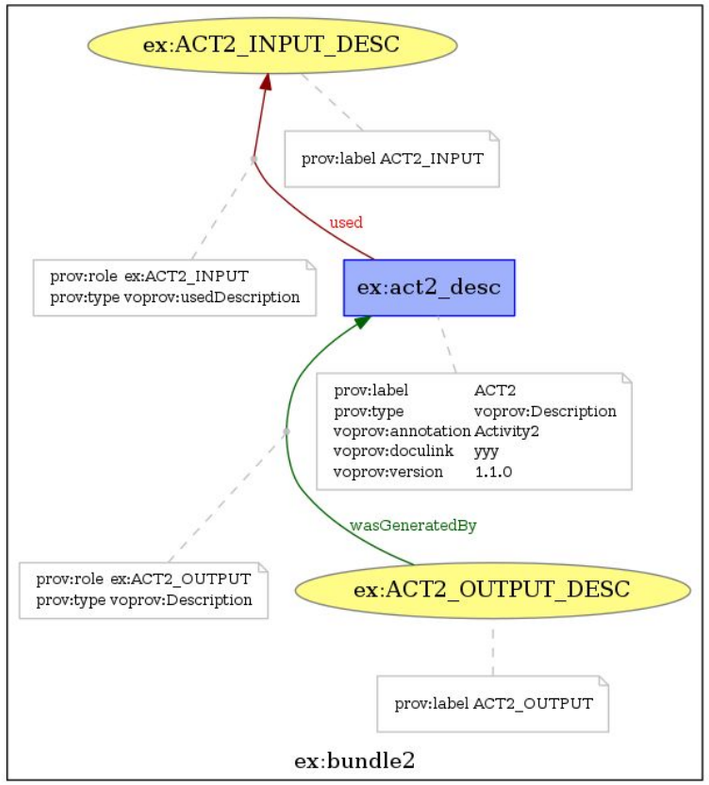
- Minimal representation
- logical place for configuration

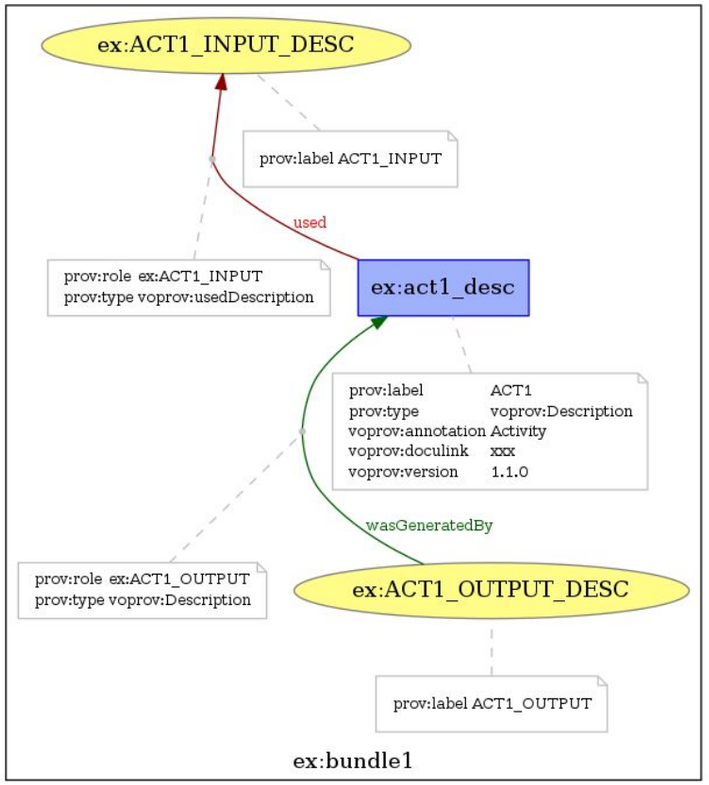
- Reusable (e.g. as a pointing position)
- Provenance of the parameter

Grouping description classes

- The **description side** has a meaning only in the context of an Activity
  - single relation **Activity-ActivityDescription**
- **Entity-EntityDescription** may still be useful alone (though it is not provenance, but data **structuration**)
  - Should we keep this option?







Serializations - W3C PROV formats

```
<prov:document xmlns:ctadata="ivo://vopdc.obspm/cta#" xmlns:ctajob
  <prov:activity prov:id="ctajobs:ctbin">
    <prov:startTime> 2016-03-13T23:44:46 </prov:startTime>
    <prov:endTime> 2016-03-13T23:44:56 </prov:endTime>
  </prov:activity>
  <prov:agent prov:id="cta:consortium">
    <prov:type xsi:type="xsd:string">Organization </prov:type>
  </prov:agent>
  <prov:wasAssociatedWith>
    <prov:activity prov:ref="ctajobs:ctbin" />
    <prov:agent prov:ref="cta:consortium" />
  </prov:wasAssociatedWith>
  <prov:entity prov:id="uwsdata:parameters/inobs" />
  <prov:used>
    <prov:activity prov:ref="ctajobs:ctbin" />
    <prov:entity prov:ref="uwsdata:parameters/inobs" />
  </prov:used>
  <prov:entity prov:id="uwsdata:results/outcube" />
  <prov:wasGeneratedBy>
    <prov:entity prov:ref="uwsdata:results/outcube" />
    <prov:activity prov:ref="ctajobs:ctbin" />
  </prov:wasGeneratedBy>
  <prov:wasDerivedFrom>
    <prov:generatedEntity prov:ref="uwsdata:results/outcube" />
    <prov:usedEntity prov:ref="uwsdata:parameters/inobs" />
  </prov:wasDerivedFrom>
  <prov:entity prov:id="uwsdata:results/logfile" />
  <prov:wasGeneratedBy>
    <prov:entity prov:ref="uwsdata:results/logfile" />
    <prov:activity prov:ref="ctajobs:ctbin" />
  </prov:wasGeneratedBy>
  <prov:wasDerivedFrom>
    <prov:generatedEntity prov:ref="uwsdata:results/logfile" />
    <prov:usedEntity prov:ref="uwsdata:parameters/inobs" />
  </prov:wasDerivedFrom>
</prov:document>
```

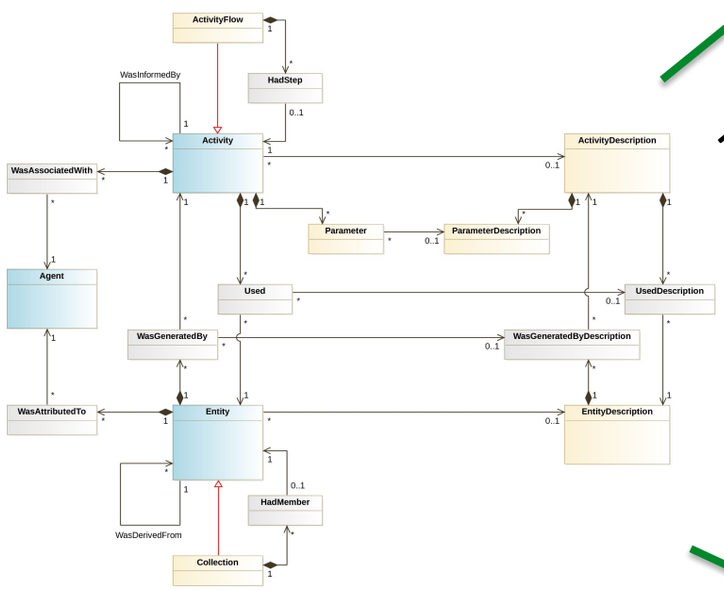
```
{
  - wasAssociatedWith: {
      - _:idl: {
            prov:agent: "cta:consortium",
            prov:activity: "cta:anactools_v1.1"
        }
    },
  - agent: {
      - cta:consortium: {
            prov:type: "Organization"
        }
    },
  - entity: {
        uwsdata:results/fit_results: { },
        uwsdata:results/configfile: { },
        uwsdata:results/butterfly: { },
        uwsdata:results/spectrum_plot: { },
        uwsdata:results/spectrum: { }
    },
  - prefix: {
        uwsdata: "https://voparis-uws-test.obspm.fr/rest
        cta: "http://www.cta-observatory.org#",
        voprov: "http://www.ivoa.net/ns/voprov#"
    },
  - activity: {
      - cta:anactools_v1.1: {
            prov:startTime: "2016-04-07T00:26:00",
            prov:endTime: "2016-04-07T00:27:15"
        }
    },
  - wasGeneratedBy: {
      - _:id5: {
            prov:entity: "uwsdata:results/butterfly",
            prov:activity: "cta:anactools_v1.1"
        },
      - _:id4: {
            prov:entity: "uwsdata:results/fit_results",
            prov:activity: "cta:anactools_v1.1"
        },
```

Serializations - VOTable

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2" xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
    xmlns:ex="http://www.example.com/provenance"
    xmlns:ivo="http://www.ivoa.net/documents/rer/ivo/"
    xmlns:voprov="http://www.ivoa.net/documents/dm/provdm/voprov/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.2 http://www.ivoa.net/xml/VOTable/VOTable-1.2.xsd">
<RESOURCE type="provenance">
    <DESCRIPTION>Provenance VOTable</DESCRIPTION>
    <TABLE name="Usage" utype="voprov:used">
        <FIELD arraysize="*" datatype="char" name="activity" ucd="meta.id" utype="voprov:Usage.activity"/>
        <FIELD arraysize="*" datatype="char" name="entity" ucd="meta.id" utype="voprov:Usage.entity"/>
        <DATA>
            <TABLEDATA>
                <TR>
                    <TD>ex:Process1</TD>
                    <TD>ivo://example#DSS2.143</TD>
                </TR>
            </TABLEDATA>
        </DATA>
    </TABLE>
    <TABLE name="Generation" utype="voprov:wasGeneratedBy">
        <FIELD arraysize="*" datatype="char" name="entity" ucd="meta.id" utype="voprov:Generation.entity"/>
        <FIELD arraysize="*" datatype="char" name="activity" ucd="meta.id" utype="voprov:Generation.activity"/>
        <DATA>
            <TABLEDATA>
                <TR>
                    <TD>ivo://example#Public_NGC6946</TD>
                    <TD>ex:Process1</TD>
                </TR>
            </TABLEDATA>
        </DATA>
    </TABLE>
    <TABLE name="Activity" utype="voprov:Activity">
        <FIELD arraysize="*" datatype="char" name="id" ucd="meta.id" utype="voprov:Activity.id"/>
        <FIELD arraysize="*" datatype="char" name="name" ucd="meta.title" utype="voprov:Activity.name"/>
        <FIELD arraysize="*" datatype="char" name="start" ucd="" utype="voprov:Activity.startTime"/>
        <FIELD arraysize="*" datatype="char" name="stop" ucd="" utype="voprov:Activity.endTime"/>
        <DATA>
            <TABLEDATA>
                <TR>
                    <TD>ex:Process1</TD>
```

Serialization context

IVOA DM

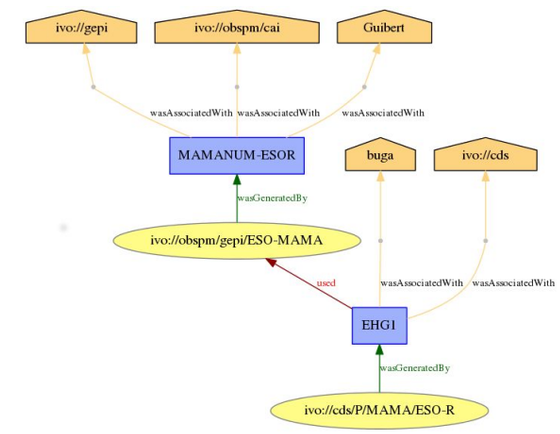


All classes defined in the specification

Translation

W3C PROV View

PROV-JSON
PROV-N
PROV-XML



ProvSAP

W3C Tools
Provenance suite,
Prov-Python

Translation

IVOA View

VOTable
JSON
FITS

ProvTAP

TopCat
TapHandle

Specific classes Translation

- **In IVOA ecosystem**
  - Use one table for each defined class, fully extensible
- **In W3C**
  - No direct W3C structure for added IVOA classes
  - Need some transcription to the W3C existing constructs
    - Parameter → simplified **Entity** with `prov:type=voprov:Parameter`
    - Descriptions → **Bundle** with **wasInfluencedBy** relation
    - ActivityFlow (postponed due to modeling issues)
  - Should this affect the **data model** or just the **serialization**?

Serializations - ActivityDescription

```
<RESOURCE ID="gammapy_maps" name="gammapy_maps" type="meta" utype="voprov:ActivityDescription">

    <DESCRIPTION>Use gammapy to generate a count map from a list of observations</DESCRIPTION>
    <!-- Service Descriptor -->
    <PARAM name="accessURL" datatype="char" arraysize="*" value="https://voparis-uws-test/rest/gammapy_maps" />
    <PARAM name="standardID" datatype="char" arraysize="*" value="ivo://ivoa.net/std/SODA#1.0" />
    <!-- Activity Description -->
    <PARAM name="type" datatype="char" arraysize="*" value="None" utype="voprov:ActivityDescription.type"/>
    <PARAM name="subtype" datatype="char" arraysize="*" value="None" utype="voprov:ActivityDescription.subtype"/>
    <PARAM name="annotation" datatype="char" arraysize="*" value="Use gammapy to generate a count map from a list of
    <PARAM name="version" datatype="char" arraysize="*" value="None" utype="voprov:ActivityDescription.version"/>
    <PARAM name="doculink" datatype="char" arraysize="*" value="https://luthgitlab.obspm.fr/jlefaucheur/hess_release
    <PARAM name="contact_name" datatype="char" arraysize="*" value="Julien Lefaucheur" utype="voprov:Agent.name"/>
    <PARAM name="contact_email" datatype="char" arraysize="*" value="" utype="voprov:Agent.email"/>
    <!-- UWS job attributes -->
    <PARAM name="executionDuration" datatype="int" value="600" utype="uws:Job.executionDuration"/
    <PARAM name="quote" datatype="int" value="120" utype="uws:Job.quote"/>

    <!-- UWS parameters (Provenance Entities or Parameters) -->
    <GROUP name="InputParams">
        <PARAM ID="obs_ids" arraysize="*" datatype="char" name="obs_ids" value="47802 47803 47804
            <DESCRIPTION>List of runs</DESCRIPTION>
        </PARAM>
        <PARAM ID="RA" datatype="double" name="RA" value="329.7169379" unit="deg"...>
        <PARAM ID="Dec" datatype="double"
        <PARAM ID="nxpix" arraysize="*" da
            <DESCRIPTION>Number of pixels
            <VALUES>
                <MIN value="0"/>
                <MAX value="1000"/>
            </VALUES>
        </PARAM>
        <PARAM ID="nypix" arraysize="*" da
        <PARAM ID="binsz" datatype="float"
    </GROUP>
```

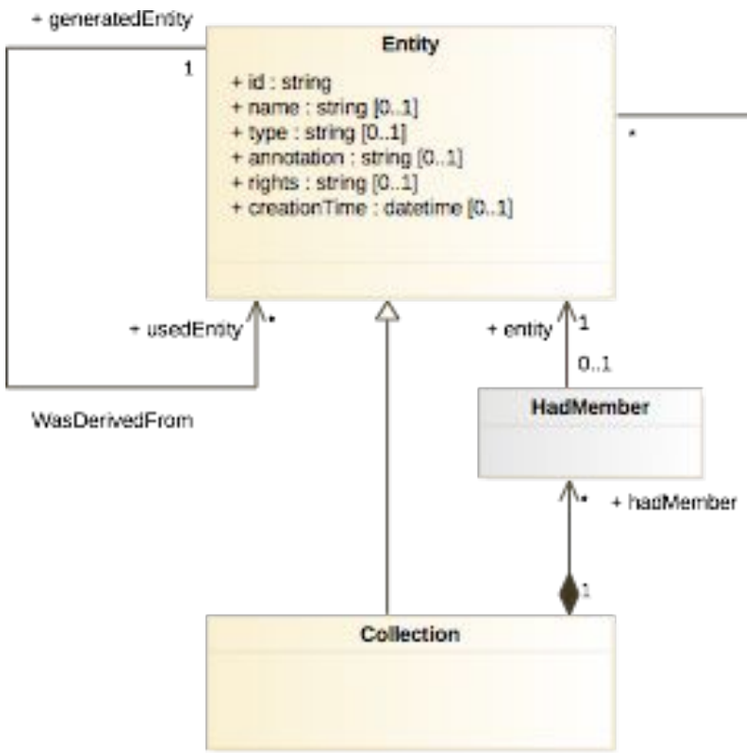
VOTable
DataLink Service Descriptor
UWS Job Description Language
Provenance ActivityDescription

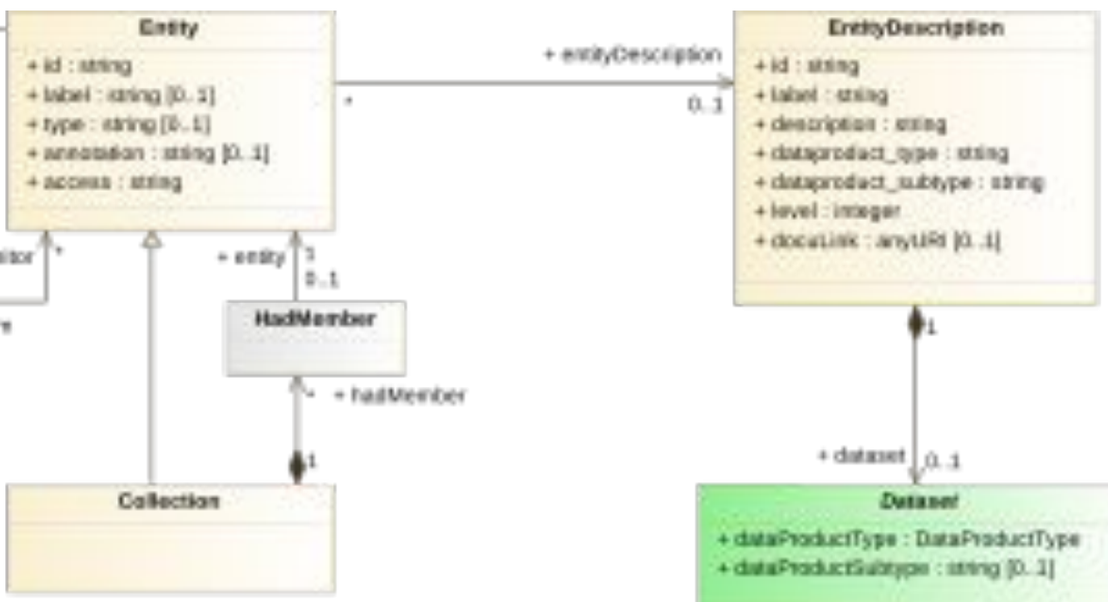
```
<!-- Used Entities -->
<GROUP name="Used">
    <GROUP name="obs_ids" utype="voprov:UsedDescription" ref="obs_ids">
        <PARAM arraysize="*" datatype="char" name="role" utype="voprov:UsedDescription.role" value="DL3"/>
        <PARAM arraysize="*" datatype="char" name="location" utype="voprov:EntityDescription.location" value=""/
        <PARAM arraysize="*" datatype="char" name="content_type" utype="voprov:EntityDescription.content_type" v
    </GROUP>
</GROUP>

<!-- Generated Entities / UWS results -->
<GROUP name="Generated" utype="voprov:WasGeneratedBy">
    <GROUP name="count_map" utype="voprov:EntityDescription">
        <DESCRIPTION>Count map</DESCRIPTION>
        <PARAM arraysize="*" datatype="char" name="role" utype="voprov:UsedDescription.role" value="DL4 image"/>
        <PARAM arraysize="*" datatype="char" name="default" utype="voprov:Entity.id" value="count_map.fits"/>
        <PARAM arraysize="*" datatype="char" name="content_type" utype="voprov:EntityDescription.content_type" v
    </GROUP>
    <GROUP name="count_preview" utype="voprov:EntityDescription">
        <DESCRIPTION>Count map preview</DESCRIPTION>
```

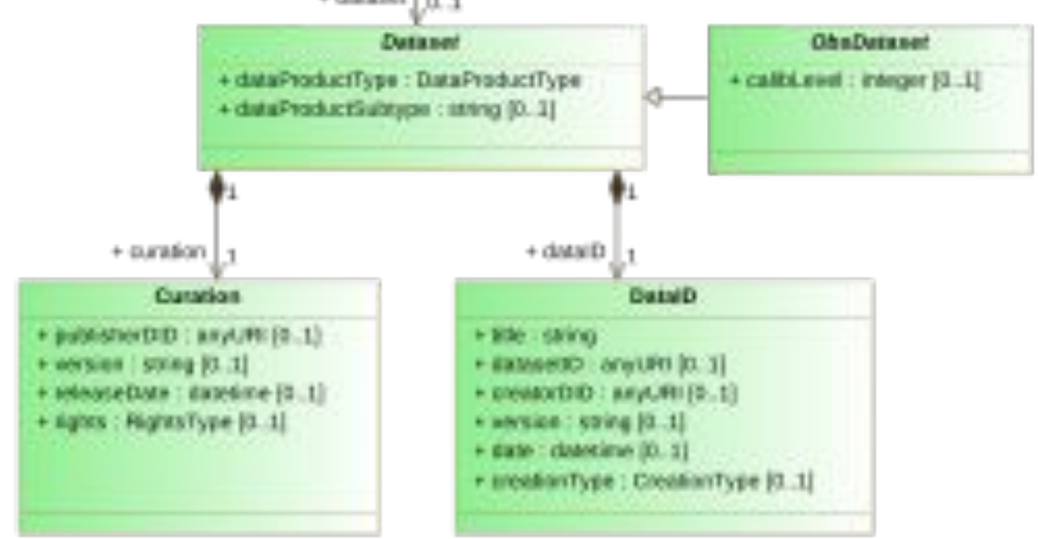
Conclusions

- Many sections in the draft are stable
- ProvSAP and ProvTAP moved to DAL drafts
- Implementation note based on many use cases
- Still some open questions:
  - modeling of Parameter
  - relations with Description classes
  - mapping for valid W3C serialization without loss
- Next steps
  - move to RFC track before next IVOA meeting

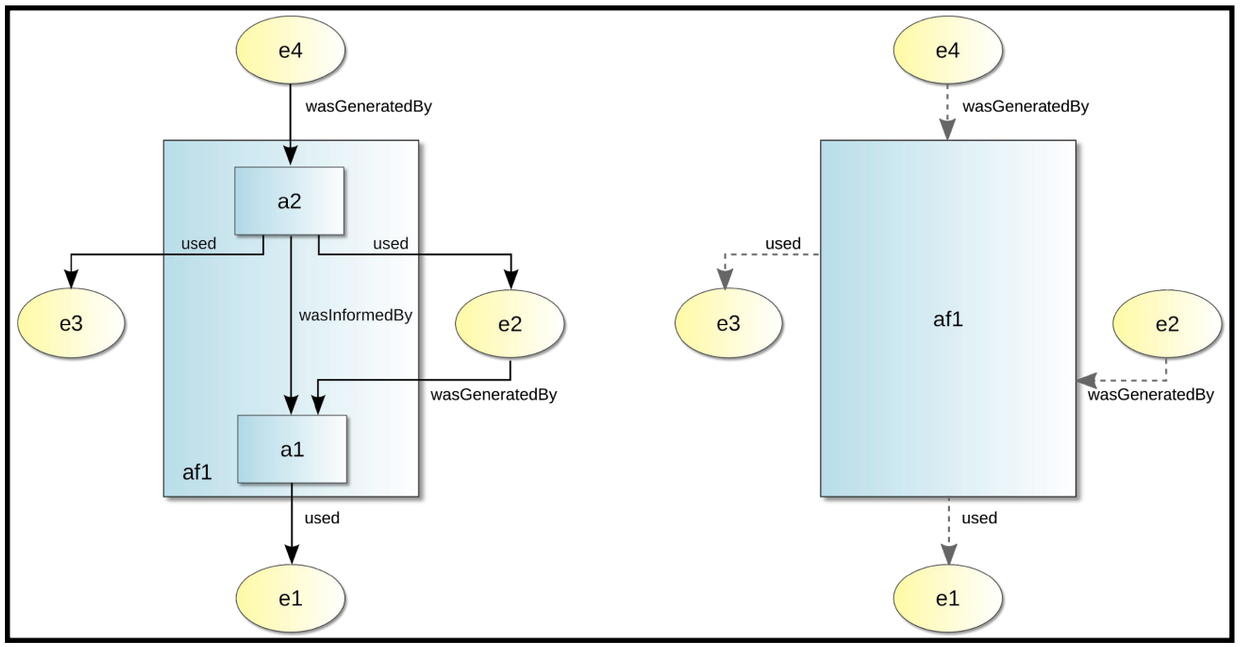
Entity

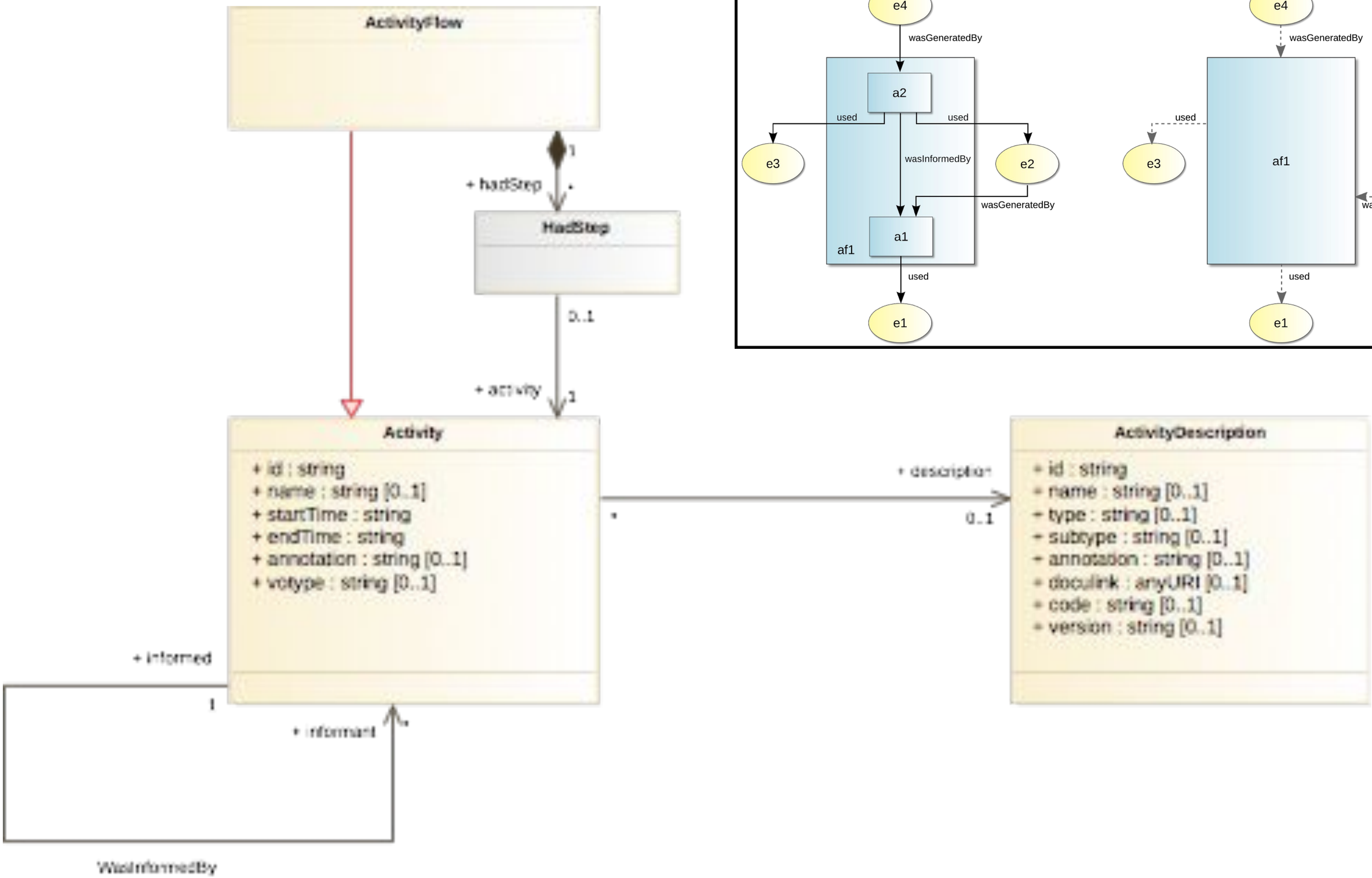

+ generatedEntity
1
Entity
+ id : string
+ name : string [0..1]
+ type : string [0..1]
+ annotation : string [0..1]
+ rights : string [0..1]
+ creationTime : datetime [0..1]
+ usedEntity
WasDerivedFrom
+ entity
1
0..1
HadMember
+ hadMember
Collection


Entity
+ id : string
+ label : string [0..1]
+ type : string [0..1]
+ annotation : string [0..1]
+ access : string
+ entityDescription
0..1
EntityDescription
+ id : string
+ label : string
+ description : string
+ dataproduct_type : string
+ dataproduct_subtype : string
+ level : integer
+ doculink : anyURI [0..1]
+ progenitor
WasDerivedFrom
+ entity
HadMember
+ hadMember
Collection

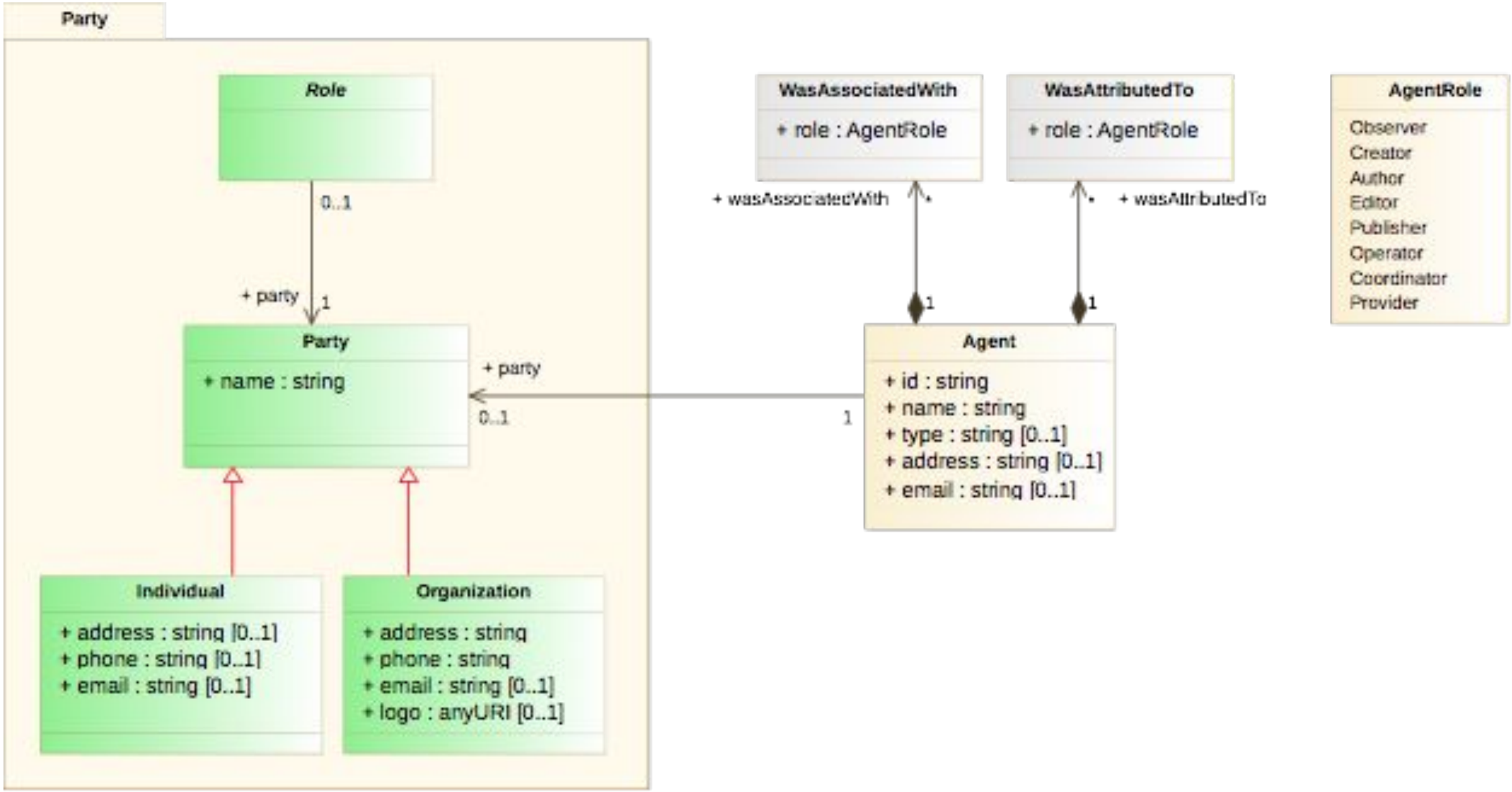

+ dataset
0..1
Dataset
+ dataProductType : DataProductType
+ dataProductSubtype : string [0..1]
ObsDataset
+ calibLevel : integer [0..1]
+ curation
+ dataID
Curation
DataID
+ title : string
+ datasetID : anyURI [0..1]
+ creatorDID : anyURI [0..1]
+ version : string [0..1]
+ date : datetime [0..1]
+ creationType : CreationType [0..1]

Activity


e4
wasGeneratedBy
a2
used
used
e3
wasInformedBy
e2
wasGeneratedBy
a1
af1
used
e1
e4
wasGeneratedBy
used
e3
af1
e2
wasGeneratedBy
used
e1


ActivityFlow
1
+ hadStep
*
HadStep
0..1
+ activity
1
Activity
+ id : string
+ name : string [0..1]
+ startTime : string
+ endTime : string
+ annotation : string [0..1]
+ votype : string [0..1]
+ description
*
0..1
ActivityDescription
+ id : string
+ name : string [0..1]
+ type : string [0..1]
+ subtype : string [0..1]
+ annotation : string [0..1]
+ doculink : anyURI [0..1]
+ code : string [0..1]
+ version : string [0..1]
+ informed
1
+ informant
WasInformedBy

Agent


Party
Role
0..1
+ party
1
Party
+ name : string
+ party
0..1
Individual
+ address : string [0..1]
+ phone : string [0..1]
+ email : string [0..1]
Organization
+ address : string
+ phone : string
+ email : string [0..1]
+ logo : anyURI [0..1]
WasAssociatedWith
+ role : AgentRole
+ wasAssociatedWith
*
1
WasAttributedTo
+ role : AgentRole
*
+ wasAttributedTo
1
1
Agent
+ id : string
+ name : string
+ type : string [0..1]
+ address : string [0..1]
+ email : string [0..1]
AgentRole
Observer
Creator
Author
Editor
Publisher
Operator
Coordinator
Provider