International
Virtual
Observatory
Alliance

IVOA Provenance Data Model
Version 1.0

IVOA Working Draft 2016-11-21

Working group
DM
This version
http://www.ivoa.net/documents/ProvenanceDM/20161121
Latest version
http://www.ivoa.net/documents/ProvenanceDM
Previous versions
ProvDM-0.2-20160428.pdf
ProvDM-0.1-20141008.pdf
Author(s)
Kristin Riebe, Mathieu Servillat, François Bonnarel, Mireille Louys, Florian Rothmaier, Michèle Sanguillon, IVOA Data Model Working Group
Editor(s)
Kristin Riebe, Mathieu Servillat

Abstract

This document describes how provenance information for astronomical datasets can be modeled, stored and exchanged within the astronomical community in a standardized way. We follow the definition of provenance as proposed by the W3C[1], i.e. that provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness. Such provenance information in astronomy is important to enable any scientist to trace back the origin of a dataset (e.g. an image, spectrum, catalog or single points in a spectral energy distribution diagram or a light curve), learn about the people and organizations involved in a project and assess the quality of the dataset as well as the usefulness of the dataset for her own scientific work.

Status of This Document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Contents

Acknowledgments

This document has been developed in part with support from the German Astrophysical Virtual Observatory, funded by BMBF Bewilligungsnummer 05A14BAD and 05A08VHA. The Provenance Working Group acknowledges support from the ASTERICS Project, funded by the European Commission (project 653477).

Thanks for fruitful discussions to (in alphabetical order): Markus Demleitner, Harry Enke, Jochen Klar, Gerard Lemson, Markus Nullmeier and Adrian Partl.

Conformance-related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard, Bradner (1997).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education,

and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

In this document, we discuss a draft of an IVOA standard data model for describing the provenance of astronomical data. We follow the definition of provenance as proposed by the W3C (Belhajjame and B'Far et al., 2013), i.e. that provenance is "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness".

In astronomy, entities are generally datasets composed of VOTables, FITS files or database tables, or files containing logs, values (spectra, lightcurves), parameters, etc. The activities correspond to an observation, a simulation, or processing steps (image stacking, object extraction, etc.). The people involved can be individual persons (observer, publisher...), groups or organisations. An example for activities, entities and agents as they can be discovered backwards in time is given in Figure 1.

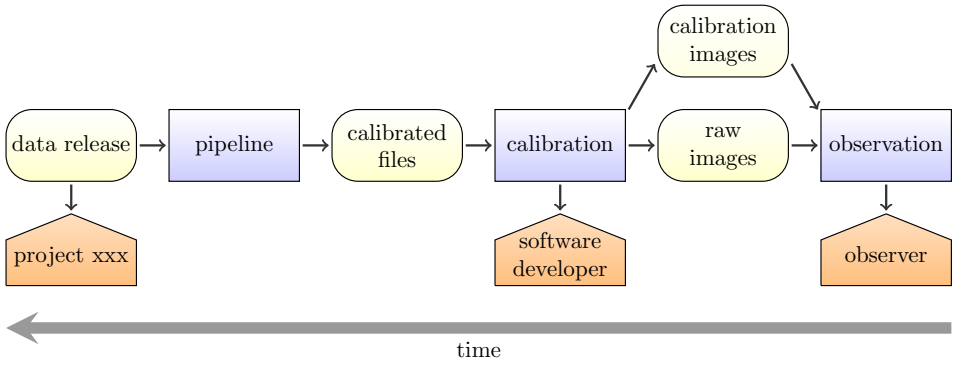

Figure 1: An example graph of provenance discovery. Starting with a released dataset (left), the involved activities (blue boxes), progenitor entities (yellow rounded boxes) and responsible agents (orange pentagons) are discovered.

We note that the provenance of simulated data is already described inside the Simulation Data Model (SimDM, Lemson and Wozniak et al., 2012). However, the Provenance Data Model currently discussed is sufficiently abstract that its core pattern could be applied to any kind of process using either observation or simulation data. It could also be used to describe the workflow for observation proposals or the publication of scientific articles based on (astronomical) data. The links between the Provenance Data Model and other IVOA data models will be discussed in Section 3.

1.1 Goal of the provenance model

The goal of this Provenance Data Model is to describe how provenance information can be modeled, stored and exchanged. Its scope is mainly modeling of the flow of data, of the relations between data, and of processing steps.

Characteristics of observations such as ambient conditions and instrument characteristics can be associated to provenance information. However, they will not be modeled here explicitly. They can be included in the form of additional data linked to observations, or as attributes of observation processes (see Section 2.2.6).

In general, the model shall capture information in a machine-readable way that would enable a scientist who has no prior knowledge about a dataset to get more background information. This will help the scientist to decide if the dataset is adequate for her research goal, assess its quality and get enough information to be able to trace back its history as far as required or possible.

Provenance information may be recorded in minute detail or by using coarser elements, depending on the intended usage and the desired level of detail for a specific project that records provenance. This granularity depends on the needs of the project and the intended usage when implementing a system to track provenance information.

The following list is a collection of tasks which the Provenance Data Model should help to solve. They are flagged with [S] for problems which are more interesting for the end user of datasets (usually a scientist) and with [P] for tasks that are probably more important for data producers and publishers. More specific use cases in the astronomy domain for different types of datasets and workflows along with example implementations are given in Section 6.

A: Tracking the production history [S]

Find out which steps were taken to produce a dataset and list the methods/tools/software that was involved. Track the history back to the raw data files/raw images, show the workflow (backwards search) or return a list of progenitor datasets.
Examples:

- Is an image from catalogue xxx already calibrated? What about dark field subtraction? Were foreground stars removed? Which technique was used?
- Is the background noise of atmospheric muons still present in my neutrino data sample?

We do not go as far as to consider easy reproducibility as a use case – this would be too ambitious. But at least the major steps undertaken to create a piece of data should be recoverable.

B: Attribution and contact information [S]

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

- I want to use an image for my own work – who was involved in creating it? Who do I need to cite or who can I contact to get this information? Is a license attached to the data?
- I have a question about column xxx in a data table. Who can I ask about that?
- Who should be cited or acknowledged if I use this data in my work?

C: Locate error sources [S, P]

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

- I found something strange in an image. Where does the image come from? Which instrument was used, with which characteristics etc.? Was there anything strange noted when the image was taken?
- Which pipeline version was used – the old one with a known bug for treating bright objects or a newer version?
- This light curve doesn't look quite right. How was the photometry determined for each data point?

D: Quality assessment [P]

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

- Since wrong calibration images may increase the number of artifacts on an image rather than removing them, knowledge about the calibration image set will help to assess the quality of the calibrated image.

E: Search in structured provenance metadata [P, S]

This would allow one to also do a "forward search", i.e. locate derived datasets or outputs, e.g. finding all images produced by a certain processing step or derived from data which were taken by a given facility.
Examples:

- Give me more images that were produced using the same pipeline.
- Give me an overview on all images reduced with the same calibration dataset.
- Are there any more images attributed to this observer?
- Which images of the Crab Nebula are of good quality and were produced within the last 10 years by someone not from ESO or NASA?
- Find all datasets generated using this given algorithm for this given step of the data processing

This task is probably the most challenging. It also includes tracking the history of data items as in A, but we still have listed this task separately, since we may decide that we can't keep this one, but we definitely want A.

1.2 Minimum requirements for provenance

We derived from our goals and use cases the following minimum requirements for the Provenance Data Model:

- Provenance information must be stored in a standard model, with standard serialization formats.
- Provenance information must be machine readable.
- Provenance data model classes and attributes should be linked to other IVOA concepts when relevant (DatasetDM, ObsCoreDM, SimDM, VOTable, UCDs...).
- Provenance information should be serializable into the W3C provenance standard formats (PROV-N, PROV-XML, PROV-JSON) with minimum information loss.
- Provenance metadata must contain information to find immediate progenitor(s) (if existing) for a given entity, i.e. a dataset.
- An entity must point to the activity that generated it (if the activity is recorded).
- Activities must point to input entities (if applicable).
- Activities may point to output entities.
- Provenance information should make it possible to derive the chronological sequence of activities.

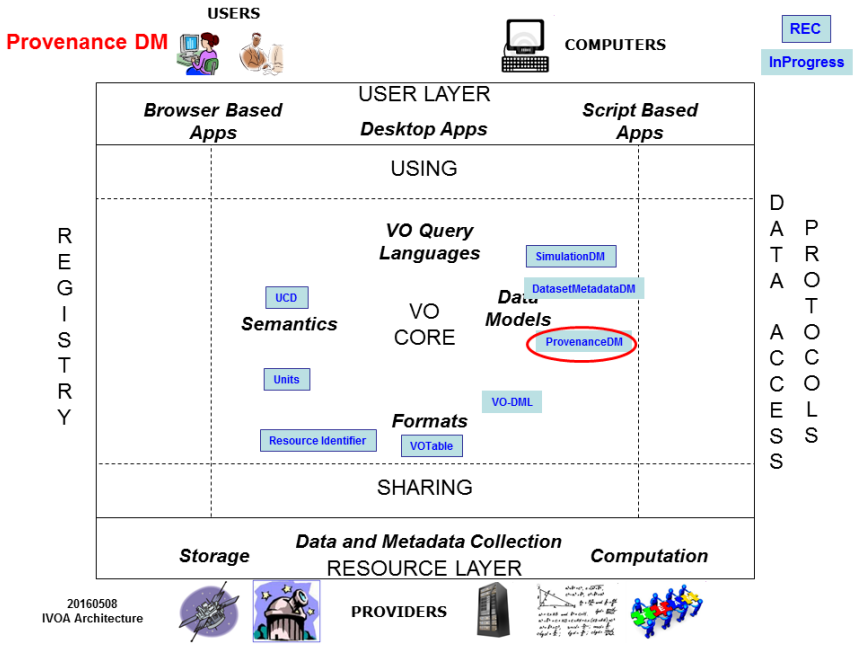

Figure 2: Architecture diagram for the Provenance Data Model. It is based on existing concepts defined in existing IVOA data models, and existing formats and semantics and fully integrated in the IVOA framework

- Provenance information can only be given for uniquely identifiable entities, at least inside their domain.
- Released entities should have a main contact.
- It is recommended that all activities and entities have contact information and contain a (short) description or link to a description.

1.3 Role within the VO architecture

The IVOA Provenance Data Model is structuring and adding metadata to trace the original process followed during the data production to provide astronomical data. Even if it borrows the main general concepts defined from the data management science, it binds to the specific context of astronomical metadata description and re-uses or interacts with existing IVOA models. It takes benefits from existing IVOA notations and standards like UCD, VOUnits, VO protocols and service design and is planned for a full integration into the VO landscape.

Fig. 2 shows the dependencies of this document with respect to other existing standards.

1.4 Previous efforts

The provenance concept was early introduced by the IVOA within the scope of the Observation Data Model (see IVOA note by IVOA Data Model Working Group, 2005) as a class describing where the data is coming from. A full observation data model dedicated to the specific spectral data was then designed (Spectral Data Model, McDowell and Salgado et al., 2016) as well as a fully generic characterisation data model of the measurement axes of the data (Characterisation Data Model, IVOA Data Model Working Group, 2008) while the progress on the provenance data model was slowing down.

The IVOA Data Model Working Group first gathered various use cases coming from different communities of observational astronomy (optical, radio, X-ray, interferometry). Common motivations for a provenance tracing of the history included: quality assessment, discovery of dataset progenitors and access to metadata necessary for reprocessing. The provenance data model was then designed as the combination of *Data processing*, *Observing configuration* and *Observation ambient conditions* data model classes. The *Processing class* was embedding a sequence of processing stages which were hooking specific ad hoc details and links to input and output datasets, as well as processing step description. Despite the attempts of UML description of the model and writing of xml serialization examples the IVOA effort failed to provide a workable solution: the scope was probably too ambitious and the technical background too unstable. A compilation of these early developments can be found on the IVOA site (Bonnarel and the IVOA Data Model Working Group, 2016). From 2013 onwards IVOA concentrated on use cases related to processing description and decided to design the model by extending the basic W3C provenance structure, as described in the current specification.

Outside of the astronomical community, the Provenance Challenge series (2006 – 2010), a community effort to achieve inter-operability between different representations of provenance in scientific workflows, resulted in the Open Provenance Model (Moreau and Clifford et al., 2010). Later, the W3C Provenance Working Group was founded and released the W3C Provenance Data Model as Recommendation in 2013 (Belhajjame and B'Far et al., 2013). OPM was designed to be applicable to anything, scientific data as well as cars or immaterial things like decisions. With the W3C model, this becomes more focused on the web. Nevertheless, the core concepts are still in principle the same in both models and very general, so they can be applied to astronomical datasets and workflows as well. The W3C model was taken up by a larger number of applications and tools than OPM, we are therefore basing our modeling efforts on the W3C Provenance data model, making it less abstract and more specific, or extending it where necessary.

The W3C model even already specifies PROV-DM Extensibility points (section 6 in Belhajjame and B'Far et al. 2013) for extending the core model.

This allows one to specify additional roles and types to each entity, agent or relation using the attributes `prov:type` and `prov:role`. By specifying the allowed values for the IVOA model, we can adjust the model to our needs while still being compliant to W3C.

2 The provenance data model

In this section, we describe the currently discussed Provenance Data Model. We start with an UML class diagram, explain the core elements and then give in the following sections more details for each class and relation.

2.1 Overview: UML class diagram and introduction to core classes

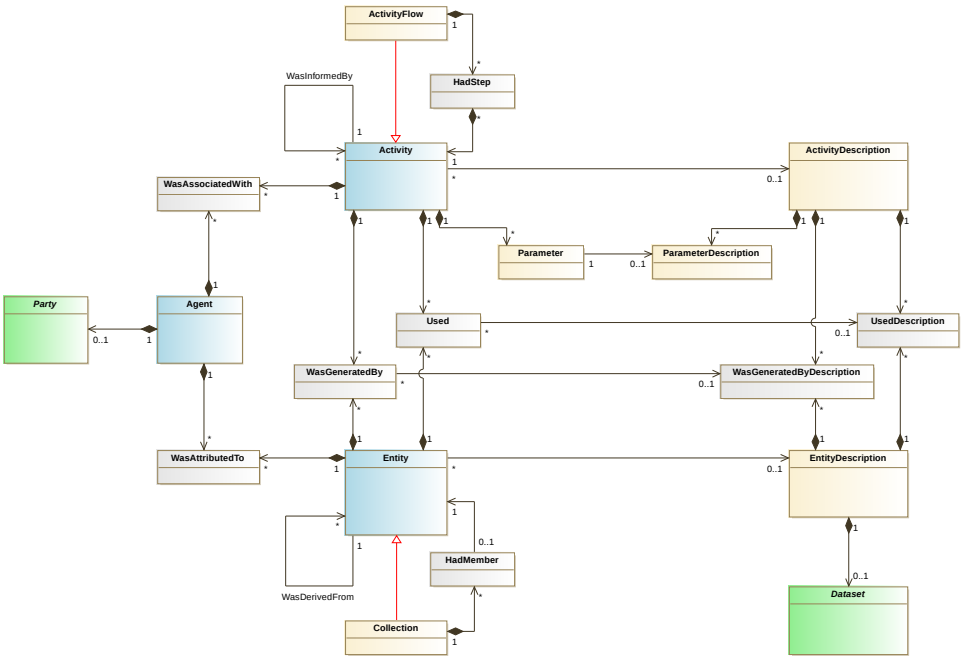

Figure 3: Overview of the classes for the Provenance Data Model in a class diagram. The blue classes are core elements. Green classes are links to the IVOA Dataset Metadata Model.

Figure 3 shows the UML diagram for an IVOA Provenance Data Model. The core elements of the Provenance Data Model are *Entity*, *Activity* and *Agent*. We chose for these elements the same names as were used in the Provenance Data Model of the World Wide Web Consortium (W3C, Belhajjame and B'Far et al. 2013), which defines a very abstract pattern that can

be reused here. Here are the core classes with a short description and some examples:

- *Entity:* a thing at a certain state
  examples: data products like images, catalogs, parameter files, calibration data, instrument characteristics

- *Activity:* an action/process or a series of actions, occurs over a period of time, performed on or caused by entities, usually results in new entities
  examples: data acquisition like observation, simulation; regridding, fusion, calibration steps, reconstruction

- *Agent:* executes/controls an activity, is responsible for an activity or an entity
  examples: telescope astronomer, pipeline operator, principal investigator, software engineer, project helpdesk

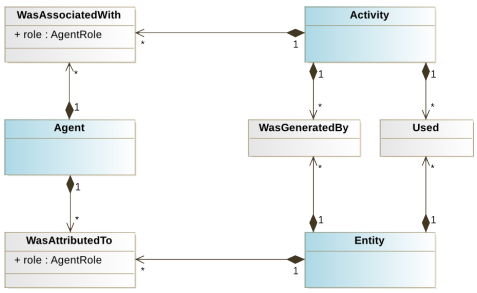

Figure 4: The main core classes and relations of the Provenance Data Model, which also occur in the W3C model.

These core classes along with their relations to each other are provided in Figure 4. We use the following relation classes to specify the mapping between the three core classes. The names were again chosen to match the W3C model names:

- *WasGeneratedBy:* a new entity is generated by an activity
  (entity "image m31.fits" wasGeneratedBy activity "observation")

- *Used:* an entity is used by an activity
  (activity "calibration" used entities "calibration data", "raw images")

- *WasAssociatedWith:* agents have responsibility for an activity
  (agent "observer Max Smith" wasAssociatedWith activity "observation")

- *WasAttributedTo:* an entity can be attributed to an agent
  (entity "image m31.fits" wasAttributedTo "M31 observation campaign")

In the domain of astronomy, certain processes and steps are repeated again and again with different parameters. We therefore separate the descriptions of activities from the actual processes and introduce an additional *ActivityDescription* class (see Figure 3). Likewise, we also apply the same pattern for *Entity* and add an *EntityDescription* class. Defining such descriptions allows them to be reused, which is very useful when performing a series of tasks of the same type, as is typically done in astronomy.

A similar normalization of descriptions of the actual processes and datasets can also be found in the IVOA Simulation Data Model (SimDM, Lemson and Wozniak et al., 2012)), which describes simulation metadata. The SimDM classes *Experiment* and *Protocol* correspond to the Provenance terms *Activity* and *ActivityDescription*.

This separation into two classes may not be needed for each and every project, and everyone is free to choose which classes make sense for his/her use case. When serializing provenance, one could integrate the description side into the other classes, thus producing a W3C compliant provenance description. More details about all these classes and relations are given in the following section.

2.2 Model description

2.2.1 Entity and EntityDescription

Entities in astronomy are usually astronomical or astrophysical datasets in the form of images, tables, numbers, etc. But they can also be observation or simulation log files or, in a wider sense, also observation proposals, scientific articles, or manuals and other documents. An entity is not restricted to being a file. It can even be just a number in a table, depending on how fine-grained the provenance shall be described.

Entities in the VO are often called "dataset", which could mean a single table, an image or a collection of them. The Dataset Metadata Model (Bonnarel and Laurino et al., 2015) specifies an "IVOA Dataset" as "a file or files which are considered to be a single deliverable". Most parts of the *Dataset* class can be mapped directly to the *Entity* class, as indicated in Figure 5. If no *EntityDescription* is used, then most parts of the *Dataset* class can be mapped directly to the *Entity* class or its description class *EntityDescription*, as indicated in Figure 5. The detailed mapping of classes and attributes from the Dataset Metadata Model to *Entity*/*EntityDescription* are given in Section 3.

For entities, we suggest the attributes given in Table 1. If the attribute also exists in the W3C Provenance Data Model, we list its name in the

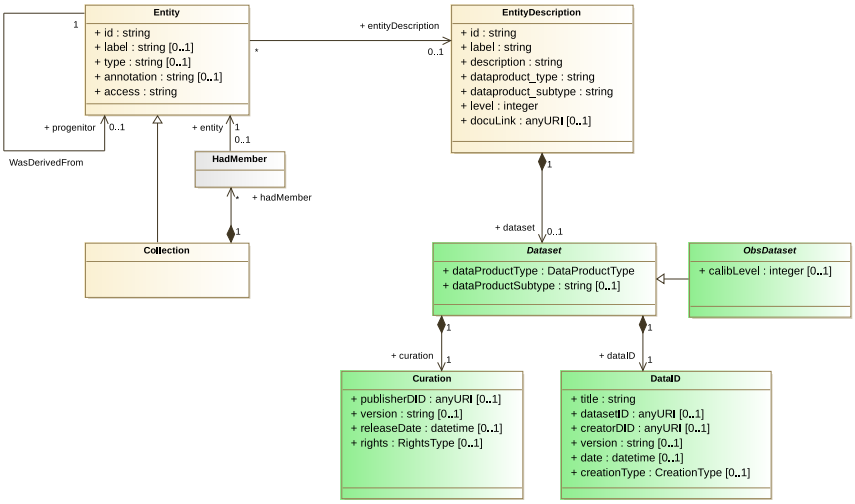

Figure 5: The relation between Entity, Dataset and Collection (see Section 2.2.2). The Dataset class as well as the classes with green boxes belong to the IVOA Dataset Metadata Model. Some attributes of Dataset actually link to Entity-attributes, see Section 3 for more details.

second column. We discussed further attributes like *size* and *format*, but we decided to treat an entity of the same content but different format (and thus size) as the same entity, unless they do not have the same provenance (e.g. when the "transformation" activity for converting one format into another is included in the provenance description).

The difference between entities that are used as input data or output data becomes clear by specifying the relations between the data and activities producing or using these data. More details on this will follow in Section 2.2.5.

EntityDescription. As already mentioned before, the types of entities or datasets in astronomy can be predefined using a description class *EntityDescription*. This class stores entity-related attributes, describing the content of the data, which can mainly be derived from the Dataset Metadata Model, the general model for observational data. The description attributes are summarized in Table 2.

The *EntityDescription* does NOT contain any information about the usage of the data, it tells nothing about them being used as input or output. This is defined only by the relations (and the relation descriptions) between activities and entities (see Section 2.2.5).

WasDerivedFrom. In Figure 5 there is one more relation that we have not mentioned yet: the *WasDerivedFrom*-relation which links two entities to-

Entity

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	a unique id for this entity (unique in its realm)
label	prov:label	string	a label (to be displayed by clients)
type	prov:type	string	a provenance type, i.e. one of: prov:collection, prov:bundle, prov:plan, not needed for a simple entity
[description]	[prov:description]	string	link to text describing the entity in more detail or link (foreign key) to *EntityDescription*
access	–	string	access rights for the data, values: public, restricted or internal; can be linked to Curation.Rights from ObsCore/DatasetDM

Table 1: Attributes of entities. Mandatory attributes are marked in bold.

gether, borrowed from the W3C model. Is is used to express that one entity was derived from another, i.e. it can be used to find one (or more) progenitor(s) of a dataset, without having to mention the activities in between. It can therefore serve as a shortcut. The information this link provides is somewhat redundant, since progenitors for entities can be found through the links to activity and the corresponding descriptions. Nevertheless, we include *WasDerivedFrom* for those cases where an explicit link between an entity and its progenitor is useful (e.g. for speeding up searches for progenitors or if the activity in between is not important).

2.2.2 Collection

Collections are entities that are grouped together and can be treated as one single entity. From the provenance point of view, they have to have the *same origin*, i.e., they were produced by the same activity (which could also be the activity of collecting data for a publication or similar). The term "collection" is also used in the Dataset Metadata Model for grouping datasets. As an example, a collection with the name 'RAVE survey' could consist of a number of database tables and spectra files.

The Entity-Collection relation can be modeled using the *Composite* design pattern: Collection is a subclass of Entity, but also an aggregation of 1 to many entities, which could be collections themselves. In order to be compli-

EntityDescription

Attribute	Data type	Description
id	(qualified) string	a unique identifier for this description
label	string	a name or label for the entity description
description	string	a decription for this kind of entity
docuLink	url	link to more documentation
dataproduct_type	string	from ObsCore data model (Louys and Bonnarel et al., 2011), if applicable; describes, what kind of product it is (e.g. image, table)
dataproduct_subtype	string	from ObsCore data model, more specific subtype
level	enum integer	the level of processing or calibration; for ObsCore's calib_level it is an integer between 0 and 3

Table 2: Attributes of *EntityDescription*. For simple use cases, the description classes may be ignored and its attributes may be used for *Entity* instead.

ant to VODML, we model the membership-relation explicitely by including a *HadMember* class in our model, which is connected to the *Collection* class via a composition. It may contain an additional role attribute.

Collections are also known in the W3C model, in the same sense as used here. The relation between entity and collection is also called "HadMember" in the W3C model.

An additional class *CollectionDescription* is only needed if it has different attributes than the *EntityDescription*. This class should therefore only be introduced if a use case requires it.

Advantages of collections: Collections can be used to collect entities with the same provenance information together, in order to hide complexity where necessary. They can be used for defining different levels of detail (granularity).

2.2.3 Activity and ActivityDescription

Activities in astronomy include all steps from obtaining data to the reduction of images and production of new datasets, like image calibration, bias subtraction, image stacking; light curve generation from a number of observations, radial velocity determination from spectra, post-processing steps of

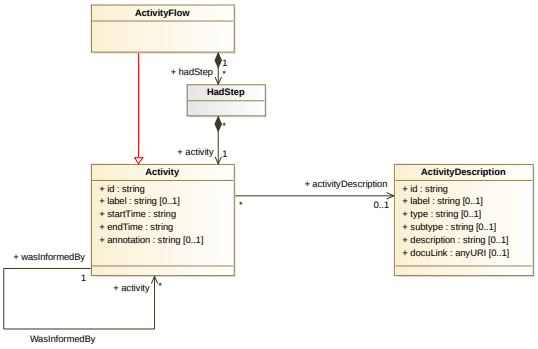

Figure 6: Details for Activity, ActivityDescription and ActivityFlow (see Section 2.2.4).

Activity

Attribute	**W3C ProvDM**	**Data type**	**Description**
id	prov:id	(qualified) string	a unique id for this activity (unique in its realm)
label	prov:label	string	a label (to be displayed by clients)
startTime	prov:startTime	datetime	start of an activity
endTime	prov:endTime	datetime	end of an activity
annotation	–	string	additional explanations for the specific activity instance
[description]	[prov:description]	string/url/ foreign key	a description for the activity, link to documentation or link to *ActivityDescription*

Table 3: Attributes of *Activity*, their data types and equivalents in the W3C Provenance Data Model, if existing. Attributes in bold are **mandatory**.

simulations etc.

ActivityDescription. The method underlying an activity can be specified by a corresponding *ActivityDescription* class (previously named *Method*, corresponds to the *Protocol* class in SimDM). This could be, for instance, the name of the code used to perform an activity or a more general description of the underlying algorithm or process. An activity is then a concrete case (instance) of using such a method, with a startTime and endTime, and it refers to a corresponding description for further information.

There MUST be exactly zero or one *ActivityDescription* per *Activity*. If steps from a pipeline shall be grouped together, one needs to create a proper *ActivityDescription* for describing all the steps at once. This method can

ActivityDescription

Attribute	Data type	Description
id	string	a unique id for this activity description (unique in its realm)
label	string	a label (to be displayed by clients)
type	string	type of the activity, from a vocabulary or list, e.g. data acquisition (observation or simulation), reduction, calibration, publication
subtype	string	more specific subtype of the activity
description	string	additional free text description for the activity
docuLink	url	link to further documentation on this process, e.g. a paper, the source code in a version control system etc.

Table 4: Attributes of *ActivityDescription*.

then be refered to by the pipeline-activity.

When serializing the data model, the attributes of the description class may be assigned to the activity in order to produce a W3C compliant serialization (same as with Entity/EntityDescription).

WasInformedBy. The individual steps of a pipeline can be chained together directly, without mentioning the intermediate datasets, using the *WasInformedBy*-relation. This relation can be used as a short-cut, if the exchanged datasets are deemed to be not important enough to be recorded. For grouping activities, also see the next section 2.2.4.

2.2.4 ActivityFlow

For facilitating grouping of activities (and their related entities etc.) we introduce the class *ActivityFlow*. It can be used for hiding a part of the workflow or provenance description, if different levels of granularity are needed. Figure 7 illustrates an example provenance graph in a detailed level (left side) and using the ActivityFlow (right side).

We explored the different ways to describe a set of activities in the W3C provenance model. This model uses *Bundle*, i.e. an entity with type "Bundle", for wrapping a provenance description. Each part of a provenance description can be put into a bundle, and the bundle can then be reused in other provenance descriptions. W3C's *Plan* is an entity with type "Plan" and is used for describing a set of actions or steps. Both, *Bundle* and *Plan*,

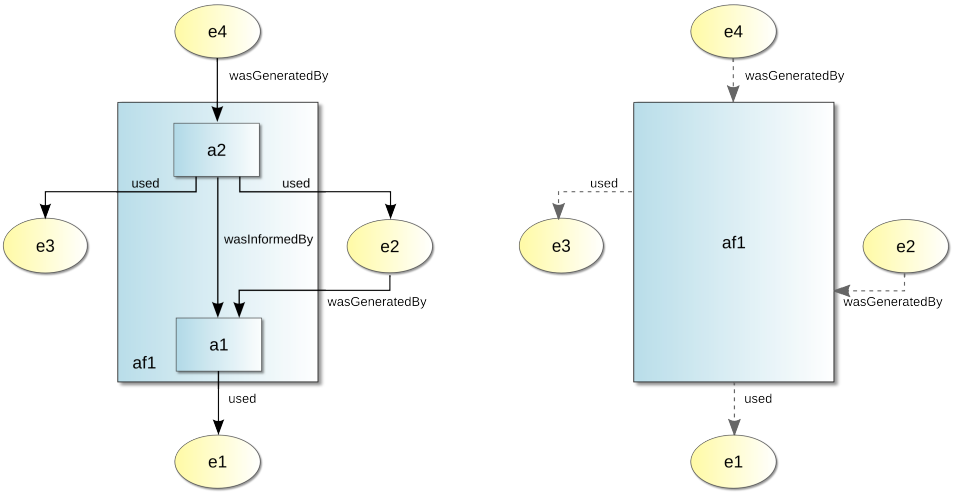

Figure 7: An example provenance graph. The detailed version is shown on the left side. It also shows the shortcut *WasInformedBy* to connect two activities, which could be used if the entity e2 would not be needed anywhere else. An ActivityFlow can be used to "hide" a part of the provenance graph as is shown on the right side. Activities are marked by blue rectangles, entities by yellow ellipses.

are entities and have the attributes and relations of this class (and thus one can define provenance of bundles and plans as well).

But we would like to consider a set of activities as being an *Activity* itself, with all the relations and properties that an activity also has. Therefore we do not reuse W3C's classes for describing workflows and plans, but added the class *ActivityFlow* as an activity composed of activities. The composition is represented by the "hadStep" relation, as is shown in Figure 6.

2.2.5 Entity-Activity relations

For each data flow it should be possible to clearly identify entities and activities. Each entity is usually a result from an activity, expressed by a link from the entity to its generating activity using the *WasGeneratedBy* relation, and can be used as input for (many) other activities, expressed by the *Used* relation. Thus the information on whether data is used as input or was produced as output of some activity is given by the *relation-types* between activities and entities.

We use two relations, *Used* and *WasGeneratedBy*, instead of just one mapping class with a flag for input/output, because their descriptions and role-attributes can be different. In previous versions of this model we only allowed one wasGeneratedBy-activity per entity. However, we introduced in Section 2.2.4 the additional *ActivityFlow* as a subclass to *Activity* for

grouping activities together. Thus we need to weaken the constraint and allow more than one wasGeneratedBy-activity per entity.

Each activity requires specific roles for each input or output entity, thus we store this information on the description side, in the role-attributes for the *UsedDescription* and *WasGeneratedByDescription* relation. For example, an activity for darkframe-subtraction requires two input images. But it is very important to know which of the images is the raw image and which one fulfills the role of dark frame.

The role is in general NOT an attribute for *EntityDescription* or *Entity*, since the same entity (e.g. a specific fits file containing an image) may play different roles with different activities. If this is not the case, if the image can only play the same role everywhere, only then it is an intrinsic property of the entity and should be stored in the *EntityDescription*.

Some example roles are given in Table 5. Note that these roles don't have to be unique, many datasets may play the same role for a process. For example, many image entities may be used as science-ready-images for an image stacking process.

Name
parameter
dark frame
calibration image
raw image
science-ready image

Table 5: Example values for the entity roles as attributes in the *UsedDescription* and *WasGeneratedByDescription*.

In order to facilitate interoperability, the possible entity-roles could be defined and described for each activity by the IVOA community, in a vocabulary list or thesaurus.

2.2.6 Parameters

The concept of activity configuration, generally a set of parameters that can be configured, is different to the concept of provenance information. However, it is tightly connected. We identify three different ways to link configuration information to an activity:

- Declare a parameter set (or each parameter) as an input entity that is used by the activity.
  This also allows tracking the provenance of the parameter further.

- Define families of activities, each one with fixed attributes.
  I.e. use different subclasses for activities with different fixed attributes.
- Add activity attributes in the form of key-value parameters.

To enable the latter solution, we add a *Parameter* class along with a *ParameterDescription* for describing additional properties of activities. In this solution, Parameters are directly connected to an Activity without complex Entity-Activity relations. Moreover, we can then describe each parameter in the same way as in FIELD and PARAM elements in VOTable (Ochsenbein and Williams et al., 2013).

Parameter

Attribute	**Data type**	**Description**
id	string	parameter unique identifier
[label]	string	parameter name or link to *ParameterDescription*
value	(value dependent)	the value of the parameter

Table 6: Attributes of *Parameter*. Attributes in bold are **mandatory**.

ParameterDescription

Attribute	**Data type**	**Description**
id	string	parameter unique identifier
label	string	parameter name
description	string	additional free text description for the parameter
datatype	string	datatype of the parameter
unit	string	physical unit of the parameter
ucd	string	Unified Content Descriptor for the parameter, supplying a standardized classification of the physical quantity
utype	string	UType of the parameter, meant to express the role of the parameter in the context of an external data model

Table 7: Attributes of *ParameterDescription*.

For example, observations generally require information on *ambient conditions* as well as *instrument characteristics*. This contextual data associated with an observation are not directly modelled in the ProvenanceDM. However, this information can be stored as different entities. Alternatively, one

could list the instrument characteristics as a set of key-value parameters using the *Parameter* class, so that this information is structured and stored with the provenance information (and can thus be queried simultaneously). In the case of a processing activity that cleans an image with a sigma-clipping method, the input and output images would be an entity and the value of the number of sigma for sigma-clipping could be a parameter instead of an entity. We may also want to define a 3-sigma-clipping activity where this parameter is fixed to 3.

2.2.7 Agent

An *Agent* describes someone who is responsible for a certain task or entity, e.g. who pressed a button, ran a script, performed the observation or published a dataset. The agent can be a single person, a group of persons (e.g. MUSE WISE Team), a project (RAVE) or an institute. This is also reflected in the IVOA Dataset Metadata Model, where *Party* represents an agent, and it has two subtypes: *Individual* and *Organization*, which are explained in more detail in Table 8. Both types are also used for agent types in the W3C Provenance Data Model, though *Individual* is called *Person* there. We do not include the type *SoftwareAgent* from W3C, since it is not required for our use cases.

AgentType			
Class	**W3C ProvDM**	**DatasetDM**	**Comment**
Agent	Agent	Party	
Individual	Person	Individual	a person, specified by name, email, address, (though all these parts may change in time)
Organization	Organization	Organization	a publishing house, institute or scientific project

Table 8: Types of agents

A definition of organizations is given in the IVOA Recommendation on Resource Metadata (Hanisch and the IVOA Resource Registry Working Group et al., 2007), hereafter refered to as RM: "An organisation is [a] specific type of resource that brings people together to pursue participation in VO applications." It also specifies further that scientific projects can be considered as organisations on a finer level: "At a high level, an organisation could be a university, observatory, or government agency. At a finer level, it could be a specific scientific project, space mission, or individual researcher.

A provider is an organisation that makes data and/or services available to users over the network."

For each agent a *name* should be specified. It would also increase the value of the given information if the (current) affiliation of the agent (and a project leader/group leader) were specified in order to maximize the chance of finding any contact person later on. The contact information is needed in case more information about a certain step in the past of a dataset is required, but also in order to know who was involved and to fulfill our "Attribution" requirement (Section 1.2), so that proper credits are given to the right people/projects.

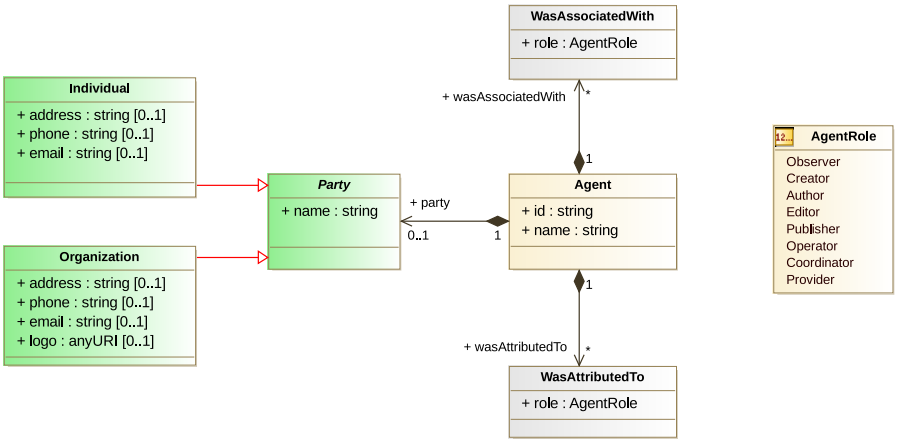

Figure 8: The relations between the *Agent* class within the Provenance Data Model (grey and yellow classes) with classes from the Dataset Metadata Model (green).

The relations between *Agent* and other classes from the Provenance Data Model and the IVOA Dataset Metadata Model are detailed in Figure 8.

It is desired to have at least one agent given for each activity (and entity), but it is not enforced. There also can be more than one agent for each activity/entity with different *roles* and one agent can be responsible for more than one activity or entity. This many-to-many relationship is made explicit in our model by adding the two following relation classes:

- wasAssociatedWith: relates an *activity* to an agent
- wasAttributedTo: relates an *entity* to an agent

We adopted here the same naming scheme as was used in W3C ProvDM. Note that the attributed-to-agent for a dataset may be different from the agent that is associated with the activity that created an entity. Someone who is performing a task is not necessarily given full attribution, especially if he acts on behalf of someone else (the project, university, ...).

In order to make it clearer what an agent is useful for, we suggest the possible roles an agent can have (along with descriptions partially taken from RM) in Table 9. For comparison, SimDM contains following roles for their *Contact* class: owner, creator, publisher and contributor.

AgentRoles

prov:role	**prov:type**	**Comment**
author	prov:person	someone who wrote an article, software, proposal
contributor	prov:person	someone who contributed to something (but not enough to gain authorship)
editor	prov:person	editor of e.g. an article, before publishing
creator	prov:person	someone who created a dataset, creators of articles or software are rather called "author"
curator	prov:person	someone who checked and corrected a dataset before publishing
publisher	prov:organization (maybe also person?)	organization (publishing house, institute) that published something
observer	prov:person	observer at the telescope
operator	prov:person	someone performing a given task (executor?)
coordinator/PI	prov:person	someone coordinating/leading a project
provider	prov:organization	"an organization that makes data and/or services available to users over the network" (definition from RM)

Table 9: Examples for roles of agents and the typical type of that agent

This list is *not* complete. We consider providing a vocabulary list for this in a future version of this model, collected from (future) implementations of this model.

3 Links to other data models

In this section we discuss how the Provenance Data Model interacts with other VO data models (especially DatasetDM). Entities and their descriptions in the Provenance Data Model are tightly linked to the *DataSet*-class in the DatasetDM/ObsCore Data Model, as well as to InputDataset and OutputDataSet in the Simulation Data Model (SimDM, Lemson and Woz-

Agent			
Attribute	**W3C ProvDM**	**Data type**	**Description**
id	prov:id	(qualified) string	unique identifier for an agent
name	prov:name	string	a common name for this agent; e.g. first name and last name; project name, ...
type	prov:type	string	type of the agent: either Individual (Person) or Organization

Table 10: Agent attributes

niak et al., 2012). Table 11 maps classes and attributes from the Dataset Data Model to concepts in the Provenance Data Model.

The *Agent* class, which is used for defining responsible persons and organizations, is similar to the *Party* class in the Dataset Metadata Model and SimDM. In SimDM one also encounters a normalization similar to our split-up of descriptions from actual data instances and executions of processes: the SimDM class "experiment" is a type of *Activity* and its general, reusable description is called a "protocol", which can be considered as a type of this model's *ActivityDescription*.

More similarities and links to other data models will be detailed in future versions of this working draft.

4 Accessing provenance information

4.1 Provenance Data Model serialization

There are three possible families of ProvenanceDM metadata serializations.

- W3C serializations: PROVN, PROVJSON, PROVXML. These are serialization of the W3C provenance data model. They allow the possibility to add additional IVOA or ad hoc attributes to the basic ones in each class. This way the IVOA models can produce W3C compliant serializations.
- Mapping of ProvenanceDM classes onto tables with appropriate relationships. This can allow management by a TAP service (the model mapping is then described with the TAP schema). The serialization will result in a single table according to the query.

Dataset DM	Provenance DM	Comment
DataID.title	Entity.label	title of the dataset
DataID.collection	HadMember.collectionId	link to the collection to which the dataset belongs
DataID.creator	Agent.name	name of agent
DataID.creatorDID	AlternateOf.entityId	id for the dataset given by the creator
DataID.ObservationID	WasGeneratedBy.activityId	identifier to everything describing the observation; maybe it belongs to entity?
Curation.PublisherDID	Entity.id	unique identifier for the dataset assigned by the publisher
Curation.PublisherID	Agent.id	link to the publisher; role: publisher, type: organization/astronomer private collection)
Curation.Publisher	Agent.name	name of the publisher
Curation.Date	Entity.releaseDate	release date of the dataset
Curation.Version	Entity.version	version of the dataset
Curation.Rights	Entity.access	access rights to the dataset; one of [...]
Curation.Reference	Entity.link	link to publication
Curation.Contact	Agent.Id or name?	link to Agent with role contact
DataProductType	EntityDescription.type	subclass to EntityDescription
DataProductSubType	EntityDescription.subtype	subclass to EntityDescription
ObsDataset.calibLevel	EntityDescription.level	subclass to EntityDescription, calibration level

Table 11: Mapping between attributes from *Dataset*-classes from DatasetDM to classes in ProvenanceDM.

- Direct VOTABLE mapping by using some ad hoc mapping based on transcription of PROV-N format: this is called PROV-VOTABLE. Moreover in the future we could also define a VO-DML (Lemson and Laurino et al., 2016) version of the mapping. The following is an example of provenance metadata in this PROV-VOTABLE format. Objects become tables, their classes are rendered by a utype. Attributes and re-

lationships become FIELDS or PARAMS. The model attribute names also become VOTABLE utypes.

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
  xsi:schemaLocation=
  "http://www.ivoa.net/xml/VOTable/v1.2 http://www.ivoa.net/xml/VOTable/v1.2">

<RESOURCE name="Stage1">

<TABLE name="activities" utype="prov:activity">
  <FIELD name="name" utype="prov:activity.name" datatype="char"
         arraysize="*"/>
  <FIELD name="start" utype="prov:startTime" datatype="char"
         arraysize="*" xtype="ISO8601"/>
  <FIELD name ="stop" utype="prov:endTime"  datatype="char"
         arraysize="*" xtype="ISO8601"/>
  <FIELD name="methodname" utype="voprov:method_name"
         dataype="char" arraysize="*"/>
  <FIELD name="version" utype="voprov:method_version"
         datatype="char" arraysize="*"/>
  <DATA>
    <TABLEDATA>
      <TR><TD>cta:telescope_stage_520</TD>
          <TD>2015-07-30T09:45:00</TD><TD>2015-07-30T10:00:00</TD>
          <TD>Telescope_stage</TD><TD>1.0</TD></TR>
    </TABLEDATA>
  </DATA>
</TABLE>

<TABLE name="entities" utype="prov:entity">
  <FIELD name="name" utype="prov:entitity.name" datatype="char"
         arraysize="*"/>
  <FIELD name="label" utype="prov:label" datatype="char"
         arraysize="*"/>
  <FIELD name="type" utype="prov:type" datatype="char"
         arraysize="*"/>
  <FIELD name="run" utype="cta:runNumber"
         datatype="int"/>
  <FIELD name="tel" utype="cta:telescope" datatype="char"
         arraysize="*"/>
  <DATA>
    <TABLEDATA>
      <TR><TD>cta:Stage1Config_520</TD><TD></TD><TD>file</TD>
          <TD></TD><TD></TD></TR>
      <TR><TD>cta:run1000_EVT1</TD><TD>EVT1 file</TD><TD>file</TD>
```

```
            <TD>1000</TD><TD>MST21</TD></TR>
          <TR><TD>cta:run13000_EVT0</TD><TD>EVT0 file</TD><TD>file</TD>
            <TD>13000</TD><TD>MST21</TD></TR>
        </TABLEDATA>
      </DATA>
    </TABLE>

    <TABLE name="usedRelationship" utype="voprov:used" >
      <FIELD name="head" datatype="char" arraysize="*" />
      <FIELD name="tail" datatype="char" arraysize="*" />
      <DATA>
        <TABLEDATA>
          <TR><TD>cta:telescope_stage_520</TD>
            <TD>cta:run13000_EVT0</TD></TR>
          <TR><TD>cta:telescope_stage_520</TD>
            <TD>cta:Stage1Config_5250</TD></TR>
        </TABLEDATA>
      </DATA>
    </TABLE>

    <TABLE name="wasGeneratedByRelationship" utype="voprov:wasGeneratedBy" >
      <FIELD name="head" datatype="char" arraysize="*" />
      <FIELD name="tail" datatype="char" arraysize="*" />
      <DATA>
        <TABLEDATA>
          <TR><TD>cta:run1000_EVT1</TD><TD>cta:telescope_stage_520</TD></TR>
        </TABLEDATA>
      </DATA>
    </TABLE>

    </RESOURCE>
    </VOTABLE>
```

4.2 Access protocols

We envision two possible access protocols:

- ProvDAL: retrieve provenance information based on given id of a data entity or activity

  ProvDAL is a service the interface of which is organized around one main PARAMETER, the "ID" of the entity (obs_publisher_did of an ObSDataSet for example). The response is given in one of the following formats: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE. Additional parameters can complete ID to refine the query. FORMAT allows to choose the output format. STEP allows to discriminate between STEP=LAST which gives the last step in the provenance chain and STEP=ALL which gives the whole chain. Multiple ID PARAME-

TER is allowed in order to retrieve several data set provenance details at the same time.

- ProvTAP: allows detailed queries for provenance information, discovery of datasets based on e.g. code version.

  ProvTAP is a TAP service implementing the ProvenanceDM data model. The data model mapping is included in the TAP schema (see above). The result of any query is a single table joining information coming from one or several "provenance" tables available in the database.

  A special case is considered where ProvenanceDM and ObsCore are both implemented in the same TAP service and queried together. The TAP response is then providing an Obscore table with a ProvenanceDM extension. We can imagine that in the future this could be hard-coded and registered as an ObsTapProv service.

- Do we need combined query possibilities, i.e. ask for ObsCore-fields and Provenance fields in one query? Or rather use a 2-step-process, decoupling them from each other?

5 Discussion

5.1 Links, ids

It would be convenient, if each data object or even each file gets a unique id that can be referenced. The W3C provenance model requires ids for entities, activities and agents, and they have to be qualified strings, i.e. containing a namespace. For example, an activity in the RAVE-pipeline could have the id '`rave:radialvelocity_pipeline_20160901`'. Using a namespace for each project for these ids will help to make them unique.

If several copies of a dataset exist, and one of them is corrupted, it would even be useful to know exactly which copy was used by a given activity. This can be modeled already with the existing tools (using a copy-activity), but we doubt that many people would actually need this level of detail.

IVOIDs and DOI's are potentially good candidates for unique identifiers.

5.2 Description classes

This model was established mainly having a database implementation in mind. However, it may be better in the long run to store provenance with the entities themselves, e.g. as an additional extension in fits-headers.

A model using description classes for defining templates for activities and entities has an advantage for normalization: the common processes could be described once and for all at some place and then be reused when recording

provenance information for certain entities and activities. This *some place* is actually the crucial point here. In an ideal world, "some place" could collect all the descriptions from all the possible datasets and methods in astronomy, but building such a look-up place is a quite challenging task – it will probably never be complete. There's also the issue of persistent identifiers/broken links to consider. Normalisation is useful for closed systems, e.g. for describing the provenance for data produced by a certain pipeline (e.g. MuseWise system) or with workflow tools or when a task needs to be repeated many times. However, the VO is quite the contrary of a closed system and we need to keep an eye on what is actually achievable.

When writing down a simple serialisation of e.g. the provenance for a stacked image using the current model including the description classes, it soon becomes quite cumbersome to define everything twice: first the descriptions, then the instances. This basically doubles the number of entries to describe provenance (unless there is already some place with all the descriptions to which we can refer).

Expressing provenance for a stacked image with this smaller set of classes may be simpler, but on the other hand constructing a database schema becomes much harder. We could leave it to the implementors to choose what is more useful for them. When extracting a serialisation of the provenance information from a provenance service, the attributes of the description classes could be combined with the corresponding activity/entity classes. This will produce some repetition (e.g. many entities may have the same descriptive attributes), but avoid having too many classes and links between them.

5.3 ActivityFlow and implications for multiplicities

By introducing the *ActivityFlow* class, one entity can now have many wasGeneratedBy-links to activities. One of them would be the actual generation-activity, the other activities can only be activityflows containing this generation-activity. This is not expressed explicitly in the current model.

We could introduce an additional abstract class, e.g. *AbstractActivity*, with *Activity* and *ActivityFlow* being subclasses to this one. But this adds another layer of complexity that we may not want in this data model.

Since we introduced *ActivityFlow* mainly for having different view levels, we may want to add an attribute *viewLevel* to descriptions of activityflows.

We are planning to test how it all works in implementations, which classes and attributes are needed or not and will then adjust the model accordingly.

5.4 VO-DML representation

We do not yet have a VO-DML compliant representation of the model. This is one of the issues to be clarified for the next version.

5.5 Links to other data models

Section 3 still needs to be expanded further, especially making detailed links with the Simulation Data Model will be very useful.

6 Use cases and implementations

6.1 Provenance of RAVE database tables (DR4)

The RAVE survey (Radial Velocity Experiment) recorded spectra for about half a million stars. These spectra are processed in a number of steps until the derived properties are published in the RAVE data releases at http://www.rave-survey.org. Providing provenance information for the data, from which spectrum and fibre the data was coming from and which steps were involved in processing the data, can help scientists to understand the data and their restrictions and judge their quality. It would also be useful to be able to compare if, how and why the derived data for some stars have changed between different releases. Provenance information for the major steps of RAVE DR4 was recorded in W3C-compatible PROV-N notation and uploaded to the provenance store at https://provenance.ecs.soton.ac.uk/store/documents/84064/. This allows to view graphs of the workflow by visualising only the main entities, activities and agents with their relations. It shows that the provenance concepts explained in this draft can be applied directly to data obtained from astronomical observations. We also tested a Django implementation of the classes in this document along with provenance data stored in an SQLite database. This allows to quickly setup a provenance web service which gives the possibility to view all instances of a class or details for a single object, extract provenance information for single entities (backwards in time) and visualise the provenance information using for example d3.js (see https://escience.aip.de/prov/graphs/example.html for an example). A preliminary version of the Django webapp is available at https://github.com/kristinriebe/provenance-website.

6.2 Provenance for CTA

The Cherenkov Telescope Array (CTA) is the next generation ground-based very high energy gamma-ray instrument. It will provide a deep insight into the non-thermal high-energy universe. Contrary to previous Cherenkov experiments, 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 that may be unaware of the Cherenkov astronomy specificities. The proposed structure of the metadata is presented in Figure 9.

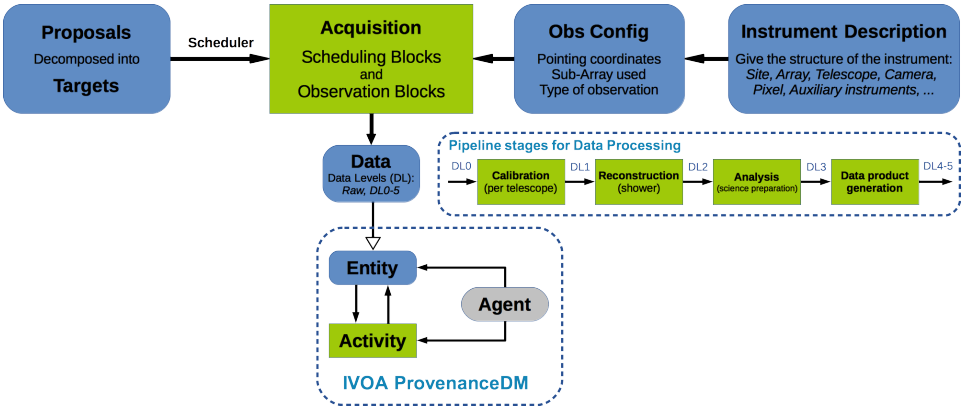

Figure 9: CTA high level data model structure with Pipeline stages and connection to IVOA ProvenanceDM.

Cherenkov telescopes indirectly detect gamma-rays by observing the flashes of Cherenkov light emitted by particle cascades initiated when the gamma-rays interact with nuclei in the atmosphere. The main difficulty is that charged cosmic rays also produce such cascades in the atmosphere, which represent an enormous background compared to genuine gamma-ray-induced cascades. Monte Carlo simulations of the shower development and Cherenkov light emission and detection, corresponding to many different observing conditions, are used to model the response of the detectors. With an array of such detectors the shower is observed from several points and, working backwards, one can figure out the origin, energy and time of the incident particle. The main stages of the CTA Pipeline are presented inside Figure 9. Because of this complexity in the detection process, provenance information of data products is necessary to the user to perform a correct scientific analysis.

Provenance concepts are relevant for different aspects of CTA :

- Data diffusion: the diffused data products have to contain all the relevant context information with the assumptions made as well as a description of the methods and algorithms used during the data processing.
- Pipeline: the CTA Observatory must ensure that data processing is traceable and reproducible.
- Instrument Configuration: the characteristics of the instrument at a given time have to be available and traceable (hardware changes, measurements of e.g. a reflectivity curve of a mirror, ...)

We tested the tracking of Provenance information using the Python prov

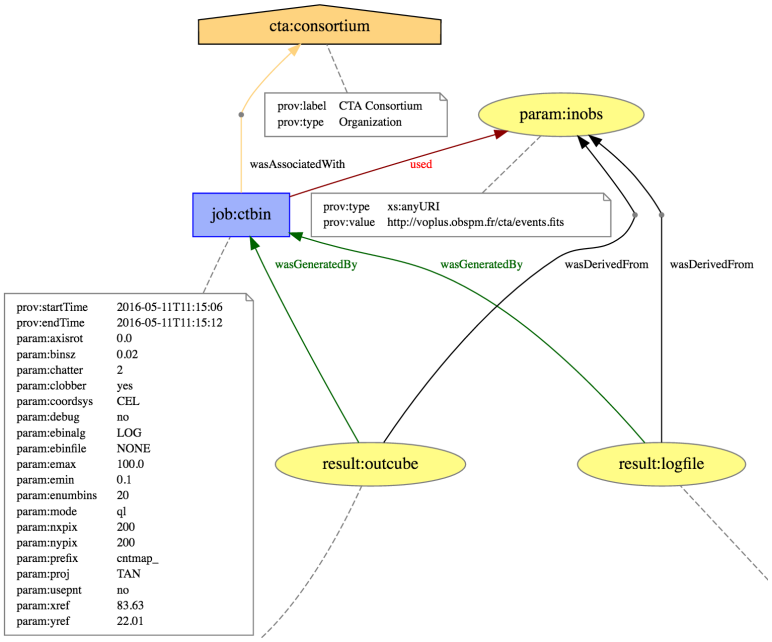

Figure 10: Provenance description of a CTA analysis step.

package inside OPUS[2] (Observatoire de Paris UWS System), a job control system developed at PADC (Paris Astronomical Data Centre). This system has been used to run CTA analysis tools and provides a description of the Provenance in the PROV-XML or PROV-JSON serialisations, as well as a graph visualization (see Figure 10).

6.3 POLLUX database

POLLUX is a stellar spectra database proposing access to high resolution synthetic spectra computed using the best available models of atmosphere (CMFGEN, ATLAS and MARCS), performant spectral synthesis codes (CMF_FLUX,SYNSPEC and TURBOSPECTRUM) and atomic linelists from VALD database and specific molecular linelists for cool stars.

Currently the provenance information is given to the astronomer in the header of the spectra files (depending on the format: FITS, ascii, xml, votables, ...) but in a non normalized description format.

The implementation of the provenance concepts in a standardized format allows users on one hand to benefit from tools to create, visualize and transform in another format the description of the provenance of these spectra and on a second hand to select data depending on provenance criteria.

[2] https://github.com/ParisAstronomicalDataCentre/OPUS

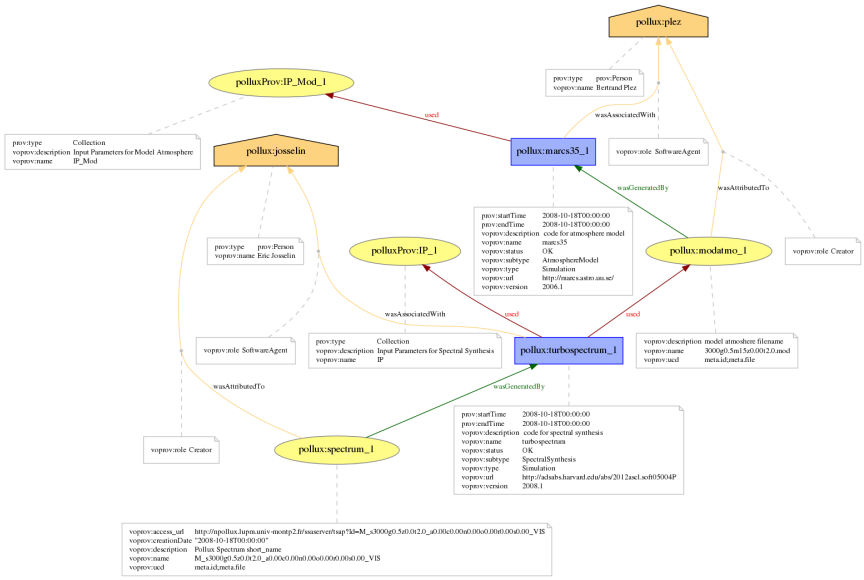

Figure 11: Pollux Example 1

6.4 HiPS use case

HiPS is a new all sky organization of pixel data. It is based on HealPix tesselation of the sky on equal area cells (pixels) for a given HealPix order gathered in tiles. Adaptative resolution is achieved by a hierarchy of tiles at increasing order. Sorting and organization is based on a tree of including directories each of those associated with a tile. HiPS specification has entered the IVOA recommendation process and is becoming an interoperability standard. In the processing chain, HiPS can be seen as a kind of "legacy level" for observational data.

An HiPS dataset can be generated either by Aladin in "hipsgen" mode or by other softwares. The processing distinguishes 3 main different methods for estimating cell values: FIRST or NEAREST neighbour, Mean or Median of the neigbouring pixels. Up to 50 parameters can help to tune the processing, among which can be found the higher resolution HealPix order, sky background value to be substracted, border width or mask to apply to original images to avoid including bad area in the computing, etc.

We will give below an example of provenance metadata for an HiPS collection generated from a collection of SERC Schmidt plates scanned by CAI (Observatoire de Paris) with the MAMA facility and serialized in PROV-N format.

TBD: HiPS in PROV-N.

6.5 Lightcurves use case

TBD. See Provenance webpage in IVOA Twiki for now.

A Changes from Previous Versions

No official previous versions yet.

B Implementation details

In this section we will give more details on the classes and attributes which were used in implementations for each use case.

TBD.

References

Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S. and Tilmes, C. (2013), 'PROV-DM: The prov data model', W3C Recommendation.
URL: *http://www.w3.org/TR/prov-dm/*

Bonnarel, F., Laurino, O., Lemson, G., Louys, M., Rots, A., Tody, D. and the IVOA Data Model Working Group (2015), 'IVOA dataset metadata model', IVOA Working draft.
URL: *http://www.ivoa.net/documents/DatasetDM/*

Bonnarel, F. and the IVOA Data Model Working Group (2016), 'Provenance data model legacy', Webpage.
URL: *http://wiki.ivoa.net/twiki/bin/view/IVOA/ProvenanceDataModelLegacy*

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.
URL: *http://www.ietf.org/rfc/rfc2119.txt*

Hanisch, R., the IVOA Resource Registry Working Group and the NVO Metadata Working Group (2007), 'Resource metadata for the virtual observatory, version 1.12', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/latest/RM.html*

IVOA Data Model Working Group (2005), 'Data model for observation, version 1.00', IVOA Note.
URL: *http://www.ivoa.net/documents/latest/DMObs.html*

IVOA Data Model Working Group (2008), 'Data model for astronomical dataset characterisation, version 1.13', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/latest/CharacterisationDM.html*

Lemson, G., Laurino, O., Bourges, L., Demleitner, M., Dowler, P., Graham, M., Gray, N. and Salgado, J. (2016), 'Vo - dml: a consistent modeling language for ivoa data models, version 1.0', IVOA Draft.
URL: *http://www.ivoa.net/documents/VODML/*

Lemson, G., Wozniak, H., Bourges, L., Cervino, M., Gheller, C., Gray, N., LePetit, F., Louys, M., Ooghe, B. and Wagner, R. (2012), 'Simulation Data Model Version 1.0', IVOA Recommendation 03 May 2012, arXiv:1402.4744.
URL: *http://adsabs.harvard.edu/abs/2012ivoa.spec.0503L*

Louys, M., Bonnarel, F., Schade, D., Dowler, P., Micol, A., Durand, D., Tody, D., Michel, L., Salgado, J., Chilingarian, I., Rino, B., de Dios Santander, J. and Skoda, P. (2011), 'Observation data model core components and its implementation in the Table Access Protocol, version 1.0', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/ObsCore/20111028/REC-ObsCore-v1.0-20111028.pdf*

McDowell, J., Salgado, J., Blanco, C. R., Osuna, P., Tody, D., Solano, E., Mazzarella, J., D'Abrusco, R., Louys, M., Budavari, T., Dolensky, M., Kamp, I., McCusker, K., Protopapas, P., Rots, A., Thompson, R., Valdes, F., Skoda, P., Rino, B., Cant, J., Laurino, O., the IVOA Data Access Layer and Groups, D. M. W. (2016), 'Ivoa spectral data model, version 2.0', IVOA Draft.
URL: *http://www.ivoa.net/documents/SpectralDM/*

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. and den Bussche, J. V. (2010), 'The open provenance model core specification (v1.1)', Future Generation Computer Systems, 27, (6), 743-756. (doi:10.1016/j.future.2010.07.005), University of Southampton.
URL: *http://openprovenance.org/; http://eprints.soton.ac.uk/271449/*

Ochsenbein, F., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A., Taylor, M. and Wicenec, A. (2013), 'Votable format definition, version 1.3', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/VOTable/*