IVOA Provenance Data Model
Version 1.0

IVOA Working Draft 2017-09-21

Working group
DM
This version
http://www.ivoa.net/documents/ProvenanceDM/20170921
Latest version
http://www.ivoa.net/documents/ProvenanceDM
Previous versions
WD-ProvenanceDM-1.0-20161121.pdf
ProvDM-0.2-20160428.pdf
ProvDM-0.1-20141008.pdf
Author(s)
Kristin Riebe, Mathieu Servillat, François Bonnarel, Mireille Louys, Markus Nullmeier, 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 Astronomy ESFRI and Research Infrastructure Cluster – ASTERICS project, funded by the European Commission under the Horizon 2020 Programme (GA 653477).

Thanks for fruitful discussions to: Catherine Boisson and Karl Kosak for the binding to the Cerenkov Telescope Array (CTA) project, Gerard Lemson and Laurent Michel for the VODML expression of the data model, Markus Demleitner, Harry Enke, Jochen Klar, Ole Streicher, Anastasia Galkin and Adrian Partl for fruitful discussions, remarks and comments during the different stages of this specification.

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, such entities are generally datasets composed of VOTables, FITS files, database tables or files containing values (spectra, light curves), logs, parameters, etc. The activities correspond to processes like 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.

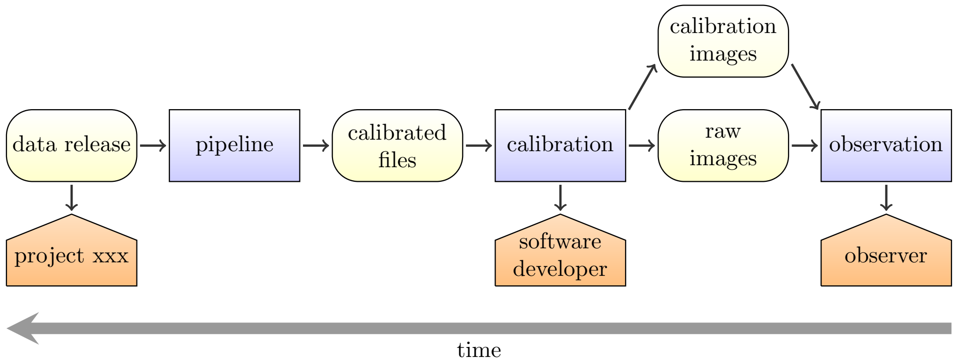

Figure 1: An example graph of provenance discovery. Starting with a released dataset (left), the involved activities (blue boxes), progenitor entities (yellow rounded boxes) and responsible agents (orange pentagons) are discovered.

The currently discussed Provenance Data Model 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. However, here we focus on astronomical data. The links between the Provenance Data Model and other IVOA data models will be discussed in Section 3. We note here that the provenance of simulated data is already covered by the Simulation Data Model (SimDM, Lemson and Wozniak et al., 2012). Therefore we also give a mapping between SimDM and the Provenance Data Model 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 observation activities such as ambient conditions and instrument characteristics can be associated to provenance information. Experimental configuration or contextual information during the execution of processing activities (computer structure, nodes, operating system used, . . .) can also be connected to provenance information. However, they will not be modeled here explicitly. This additional information can be included in the form of data or entities linked to those activities, or as attributes of activities (see also Section 2.2.7 for parameters of activities).

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 were 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.
- 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 for providing astronomical data. Even if it borrows the main general concepts from 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,

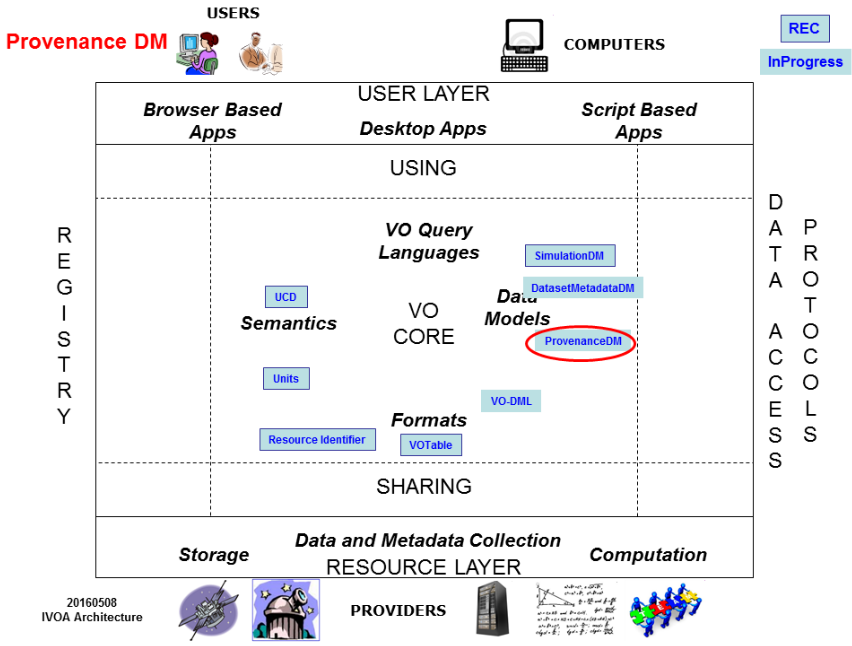

Figure 2: Architecture diagram for the Provenance Data Model. It is based on existing concepts defined in existing IVOA data models, and existing formats and semantics and fully integrated in the IVOA framework

VO protocols and service design; and it 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 specifically dedicated to 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 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 their 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 descriptions. Despite the attempts at an UML description of the model and writing XML serialization examples, the IVOA efforts 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, the 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 (OPM) (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 are 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 for each entity, agent or relation using the attributes `prov:type` and `prov:role`. By specifying well-defined values for the IVOA model, we can adjust the model to our needs while still being compliant with 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, in the following sections, we give more details for each class and relation. An auto-generated documentation of all classes (VO-DML compliant) is available in the Volute repository at `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/vo-dml/ProvenanceDM.html`.

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

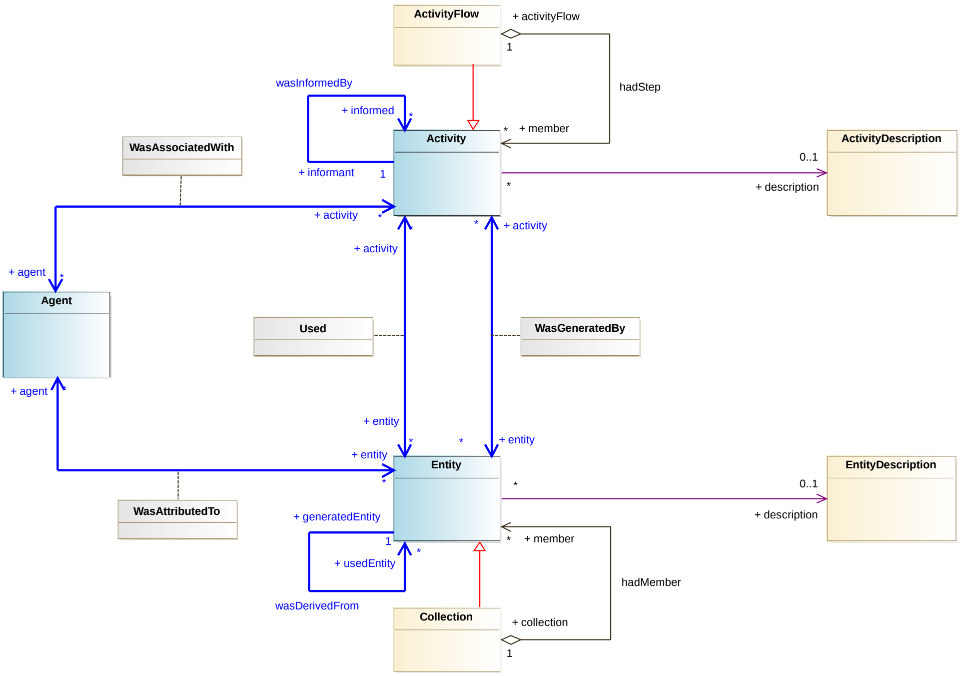

Figure 3: Overview of the classes for the Provenance Data Model in a conceptual class diagram. The blue classes are core elements. There are a number of many-to-many relationships with attached association classes (grey) that may contain additional attributes.

Figure 3 shows the conceptional UML diagram for an IVOA Provenance Data Model. The core elements of the Provenance Data Model are *Entity*, *Activity* and *Agent*. For these elements we chose the same names that 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 short descriptions and some examples:

- *Entity:* a thing in a certain state
  examples: data products such as images, catalogs, parameter files, calibration data, instrument characteristics
- *Activity:* an action/process or a series of actions, occurring over a period of time, performed on or caused by entities, usually resulting 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

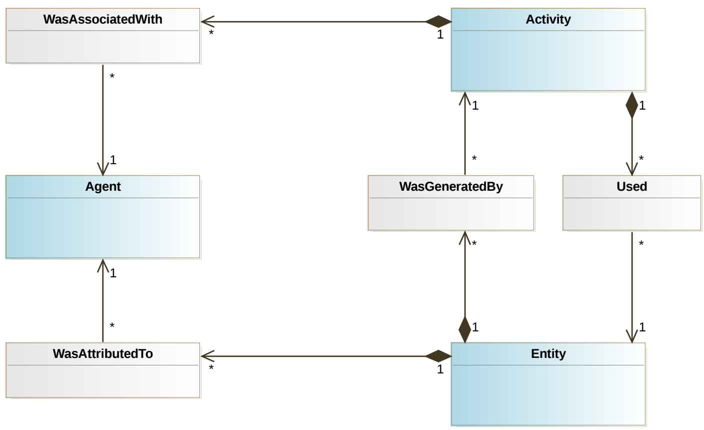

Figure 4: The main core classes and relations of the Provenance Data Model, which also occur in the W3C model.

These core classes, along with their relations to each other, are displayed in Figure 4. We use the following relation classes to specify the mapping between the three core classes. The relation 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")

Note that the relations appear as extra classes (and thus boxes in the diagrams, instead of just having annotated relations), because they can have additional attributes – when mapping the model to a relational database, these relations would appear as mapping tables.

In the domain of astronomy, certain processes and steps are repeated over and over again, using 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 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 can 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 Class diagram and VO-DML compatibility

Figure 5 shows the full class diagram, with the association classes for the many-to-many relations modeled more directly as mapping classes. When implementing the model in a relational database, these classes can be represented as individual tables for mapping the relation. We model one of each of the associations of the many-to-many relationships as a composition (full diamond) if the mapping class belongs more strongly to one of its linked classes; e.g. the *Used* relations are strongly dependent on the corresponding *Activities*. The documentation of all classes and an automatically generated figure based on the underlying xmi-description behind this UML diagram is available in the Volute repository at `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/vo-dml/ProvenanceDM.html`.

This version of the UML diagram is fully VO-DML compliant, i.e. we just used the restricted subset of UML to model Provenance and reused the IVOA datatypes.

2.2.2 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, files containing system information, environment variables, names and versions of packages, ambient conditions, or, in a wider sense, also observation proposals, scientific articles, or manuals and other documents.

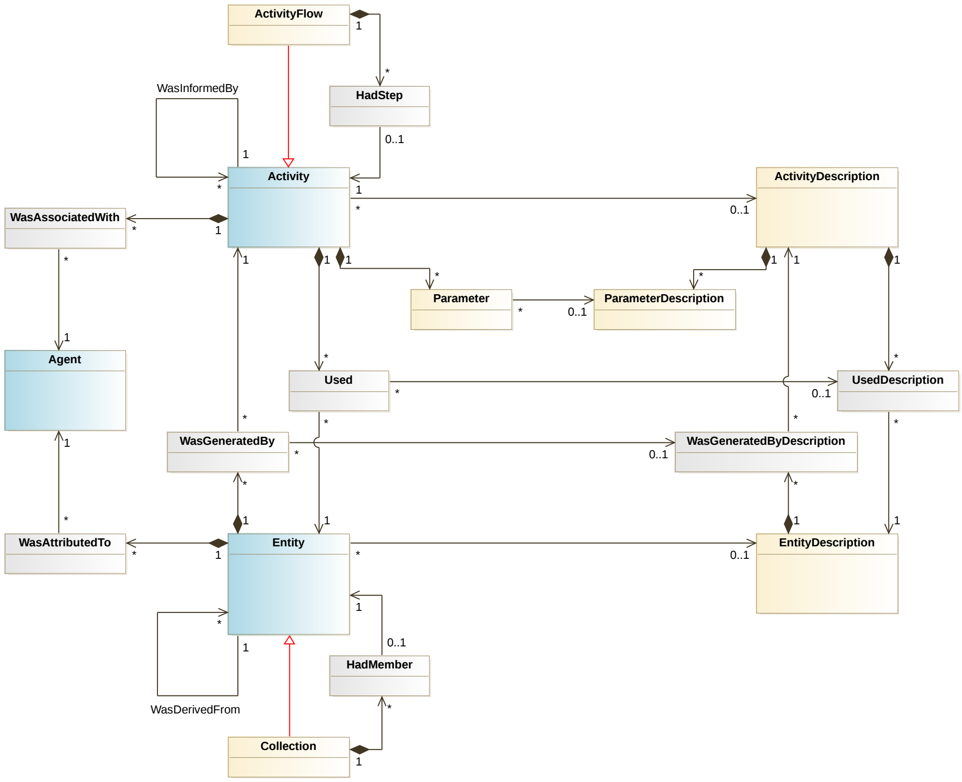

Figure 5: More detailed overview of the classes for the Provenance Data Model. Note that this UML class diagram is compatible with VO-DML.

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.

The VO concept closest to *Entity* is the notion of *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 a *Dataset* as "a file or files which are considered to be a single deliverable". Most attributes of the *Dataset* class can be mapped directly to attributes of the *Entity* and *EntityDescription* class, see the mappings of Table 17 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 second column.

The difference between entities that are used as input data and those used as output data becomes clear by specifying the relations between the data and the activities producing or using these data. More details on this will follow in Section 2.2.6.

EntityDescription. The types of entities, or datasets in astronomy, can be predefined using a description class *EntityDescription*. This class is meant

Entity

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	a unique id for this entity (unique in its realm)
name	prov:label	string	a human-readable name for the entity (to be displayed by clients)
type	prov:type	string	a provenance type, i.e. one of: prov:collection, prov:bundle, prov:plan, prov:entity; not needed for a simple entity
annotation	prov:description	string	text describing the entity in more detail
rights	–	string	access rights for the data, values: public, restricted or internal; can be linked to Curation.Rights from ObsCore/Dataset Metadata Model
creationTime	–	datetime	date and time at which the entity was created (e.g. timestamp of a file)
→ description		link	link to *EntityDescription*
Additional attributes:			
Further project-specific attributes (e.g. size, path, url, . . .) can be added (see Section 3.1)			

Table 1: Attributes of *Entities.* Mandatory attributes are marked in **bold**, references are indicated with an arrow (→). Attributes from *EntityDescription* (see next section) may appear here as well.

to store descriptive information about an entity that is known before an *Entity* instance is created. For example, if we run an activity to create an RGB image from three greyscale images, we may know a mandatory `format` for the input and output images before the activity's execution (JPG, PNG, FITS, . . .), but we probably cannot know the final `size` of the image that will be created. Therefore, `format` would be an *EntityDescription* attribute, while `size` would be an attribute of the *Entity* instance.

Additional attributes that describe the content of the data could be derived from the Dataset Metadata Model (see Section 3.1)

The *EntityDescription* class does NOT contain any information about the usage of the data, in particular, it tells nothing about them being used as input or as output. This kind of information is exclusively provided by

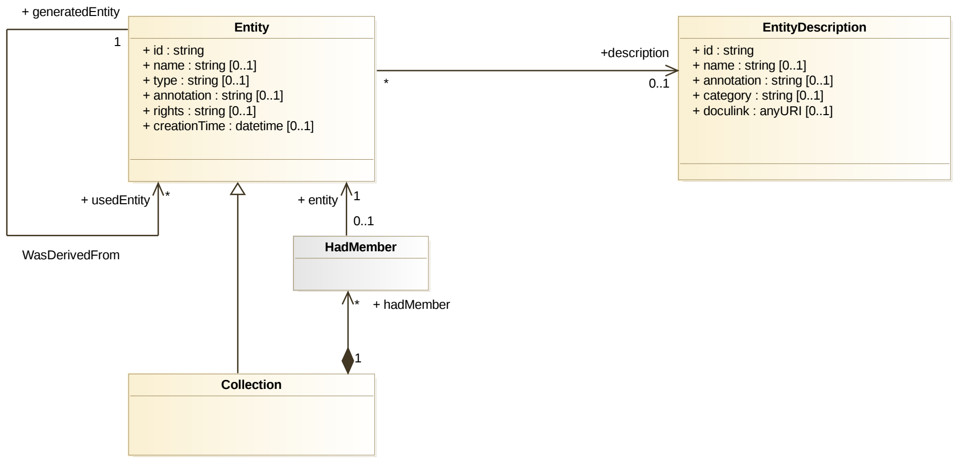

Figure 6: The relations between Entity, EntityDescription and Collection (see Section 2.2.3). Links to the Dataset class from the Dataset Metadata Model are described in Section 3.

EntityDescription

Attribute	**Data type**	**Description**
id	(qualified) string	a unique identifier for this description
name	string	a human-readable name for the entity description
annotation	string	a decriptive text for this kind of entity
category	string	specifies if the entity contains information on logging, system (environment), calibration, simulation, observation, configuration, ...
doculink	url	link to more documentation

Additional attributes:

Further project-specific attributes (e.g. format, content_type) and/or attributes from other data models (e.g. dataproduct_type and -_subtype, version, calibLevel from DatasetDM) can be added (Section 3.1).

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

the relations (and their relation descriptions) between activities and entities (see Section 2.2.6).

The *EntityDescription* general attributes are summarized in Table 2.

WasDerivedFrom

Attribute	Data type	Description
→ **generatedEntity**	link	link to the *Entity*
→ **usedEntity**	link	link to the progenitor *Entity*, from which the generatedEntity was derived

Table 3: Attributes of the *WasDerivedFrom* relation. These are the same as those used in W3C's ProvDM. **Mandatory** attributes are marked in bold, references in the data model are indicated with an arrow (→). The W3C model contains additional optional links to the related *Activity*, *WasGeneratedBy* and *Used* relations, which we do not include here for simplicity.

WasDerivedFrom. In Figure 6 there is one more relation that we have not mentioned yet: the *WasDerivedFrom* relation, linking two entities together, which is borrowed from the W3C model. It 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 look for the activities in between. It can therefore serve as a shortcut.

The information this relation 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).

Note that the *WasDerivedFrom* relation cannot always automatically be inferred from following *WasGeneratedBy* and *Used* relations alone: If there is more than one input and more than one output of an activity, it is not clear (without consulting the activityDescription and entity roles in the relation descriptions) which entity was derived from which. Only by specifying the descriptions and roles accordingly, or by adding a *WasDerivedFrom* relation, this direct derivation becomes known.

2.2.3 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 compliant to VO-DML, we model the membership relation explicitly 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).

2.2.4 Activity and ActivityDescription

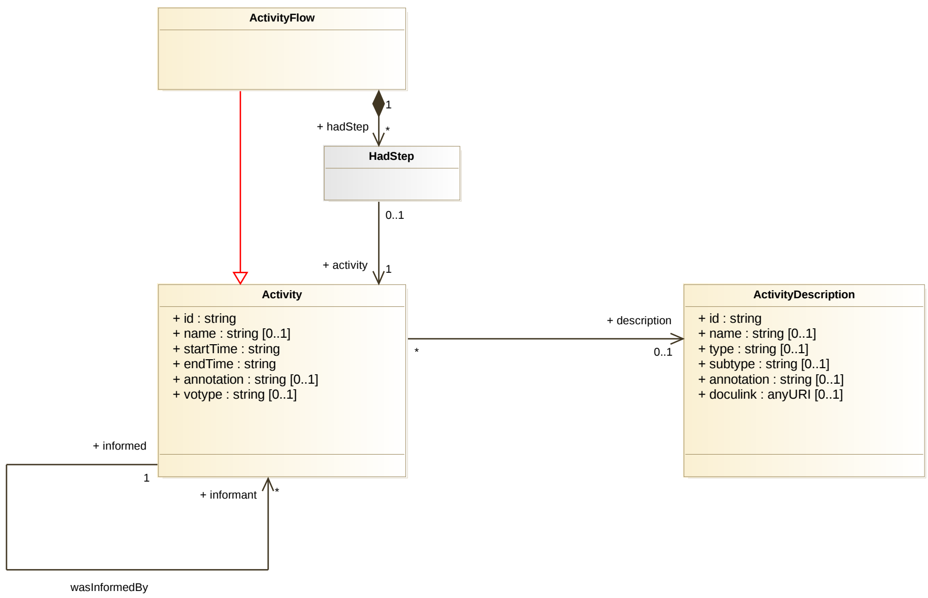

Figure 7: Details for Activity, ActivityDescription and ActivityFlow (Sections 2.2.4 and 2.2.5).

Activities in astronomy include all steps from obtaining data to the reduction of images and production of new datasets, such as image calibration,

Activity

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	a unique id for this activity (unique in its realm)
name	prov:label	string	a human-readable name (to be displayed by clients)
startTime	prov:startTime	datetime	start of an activity
endTime	prov:endTime	datetime	end of an activity
annotation	prov:description	string	additional explanations for the specific activity instance
votype		string	can be either "activity" or "activityFlow"
→ description		link	link to *ActivityDescription*

Table 4: Attributes of *Activity*, their data types and equivalents in the W3C Provenance Data Model, if existing. Attributes in bold are **mandatory**, references are indicated with an arrow (→). If no *ActivityDescription* class is used, those attributes can be used here as well.

bias subtraction, image stacking, light curve generation from a number of observations, radial velocity determination from spectra, post-processing steps of 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 an 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 then be referred 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, which is the same procedure 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 shortcut if the

ActivityDescription

Attribute	Data type	Description
id	string	a unique id for this activity description (unique in its realm)
name	string	a human-readable name (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
annotation	string	additional free text description for the activity
code	string	the code (software) used for this process, if applicable
version	string	a version number, if applicable (e.g. for the code)
doculink	url	link to further documentation on this process, e.g. a paper, the source code in a version control system etc.

Table 5: Attributes of *ActivityDescription*.

WasInformedBy

Attribute	Data type	Description
→ **informed**	link	link to the *Activity* being informed by another ("second" activity)
→ **informant**	link	link to the informing *Activity* ("first" activity)

Table 6: Attributes of the *WasInformedBy* relation. We just use this class to link chained activities together. The attribute names correspond to the W3C PROV-DM names.

skipped datasets are deemed to be not important enough to be recorded. For grouping activities, also see the next Section 2.2.5.

2.2.5 ActivityFlow

For facilitating grouping of activities (and their related entities, etc.) we introduce the class *ActivityFlow*. It can be used for hiding and grouping a part of the workflow/pipeline or provenance description, if different levels of granularity are required. Such pipelines and workflows are very common in astronomical data production and processing. Figure 8 illustrates an example provenance graph at a detailed level (left side), and also using an *ActivityFlow* (right side).

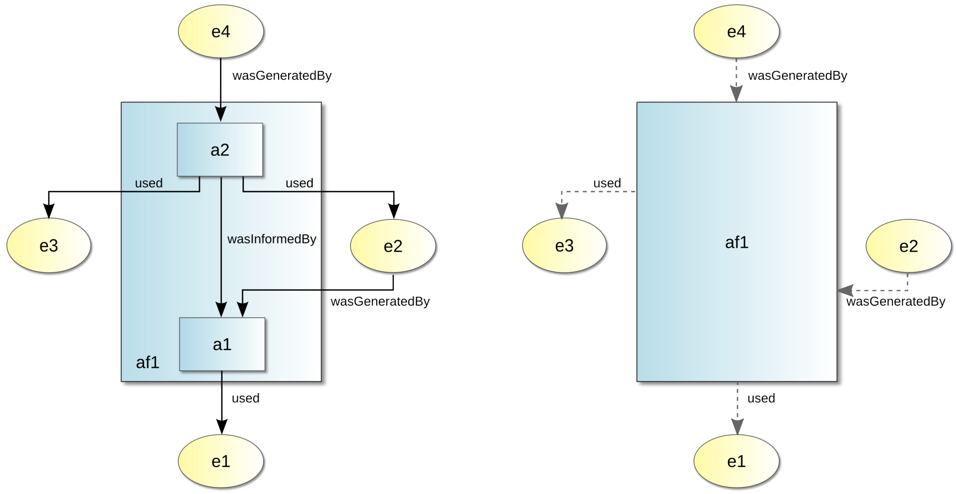

Figure 8: An example provenance graph. The detailed version is shown on the left side. It also shows the shortcut *WasInformedBy* to connect two activities, which could be used if the entity e2 would not be needed anywhere else. An ActivityFlow can be used to "hide" a part of the provenance graph as is shown on the right side. Activities are marked by blue rectangles, entities by yellow ellipses.

We also explored the different ways to describe a set of activities within 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*, 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 characterize activities. Therefore, we do not reuse W3C's classes for describing workflows and plans, but add the class *ActivityFlow* as an activity composed of other activities. The composition is represented by the *HadStep* relation, as is shown in Figure 7. In implementations, *ActivityFlow* can either be represented as an extra class or as *Activity* with attribute `votype="ActivityFlow"`.

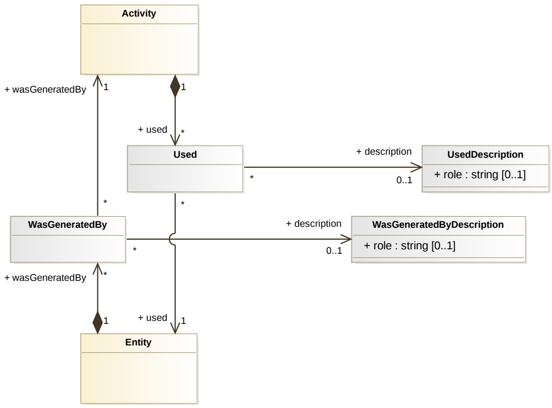

Figure 9: *Entity* and *Activity* are linked via the *Used* and *WasGeneratedBy* relations. In the left image, the `role` that an entity played when being used or generated by an activity is recorded with the corresponding *UsedDescription* and *WasGeneratedByDescription*, also see Section 2.2.6. If these description classes are not used, the *role* can be used directly as an attribute within the *Used* and *WasGeneratedBy* classes (right image).

Used

Attribute	**W3C ProvDM**	**Data type**	**Description**
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the entity, defines as what it is being used
time	prov:time	datetime	time at which the usage of an entity started
→ **activity**	prov:activity	link	link to an *Activity*
→ entity	prov:entity	link	link to an *Entity*
→ description		link	link to the corresponding *UsedDescription*, if existing

Table 7: Attributes and references of *Used* relation class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→). The `role` attribute can also be defined in the *UsedDescription* class instead.

2.2.6 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* rela-

WasGeneratedBy

Attribute	W3C ProvDM	Data type	Description
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the entity that is generated by an activity, defines which output type it is
time	prov:time	datetime	time at which the generation of an entity is finished
→ **entity**	prov:entity	link	link to an *Entity*
→ activity	prov:activity	link	link to an *Activity*
→ description		link	link to the corresponding *WasGeneratedByDescription*, if existing

Table 8: Attributes and references of *WasGeneratedBy* relation class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→). The `role` attribute can also be defined in the *WasGeneratedByDescription* class instead.

tion, 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* (see Tables 7 and 8), instead of just one mapping class with a flag for input/output, because their descriptions and role-attributes can be different.

The *Used* and *WasGeneratedBy*-relation can have the optional attribute `time` – this is the time when usage of an entity started or the generation of an entity is finished.

Compositions and multiplicities In principle, an entity is produced by just one activity. However, by introducing the *ActivityFlow* class for grouping activities together, one entity can now have many wasGeneratedBy-links to activities. One of them must be the actual generation activity, the other activities can only be activityFlows containing this generation-activity. This restriction of having only one "true" generation activity is not explicitly expressed in the current model[2].

The *Used* relation is closely coupled to the *Activity*, so we use a com-

[2]The reason for this is that we want to keep the model simple and avoid introducing even more classes.

position here, indicated in Figure 5 by a filled diamond: if an activity is deleted, then the corresponding used relations need to be removed as well. The entities that were used still remain, since they may have been used for other activities as well. We need a multiplicity * between *Used* and *Entity*, because an entity can be used more than once (by different activities).

Similarly, the *WasGeneratedBy* relation is closely coupled with the *Entity* via a composition, since a wasGeneratedBy relation makes no sense without its entity. So if an entity is deleted, then its wasGeneratedBy relation must be deleted as well. There is a multiplicity * between *Activity* and *WasGeneratedBy*, because an activity can generate many entities.

UsedDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entityDescription, as what it is used for the linked type of activityDescription
→ **activityDescription**	link	link to *ActivityDescription*
→ entityDescription	link	link to *EntityDescription*

Table 9: Attributes and references of *UsedDescription* class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→).

WasGeneratedByDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entityDescription, which kind of output it is
→ **entityDescription**	link	link to *EntityDescription*
→ activityDescription	link	link to an *ActivityDescription*

Table 10: Attributes and references of *WasGeneratedByDescription* class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→).

Entity roles Each activity requires specific roles for each input or output entity, thus we store this information with description classes, in the role-attributes for the *UsedDescription* and *WasGeneratedByDescription* relation (see Tables 9 and 10). 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 fulfils 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 11. 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.

Role	**Example entities**
configuration	configuration file
auxiliary input	calibration image, dark frame, etc.
main input	raw image, science-ready images
main result	image, cube or spectrum
log	logging output file
red	image used for red channel of a composite activity

Table 11: Examples for 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.

2.2.7 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
value	(value dependent)	the value of the parameter
→ description	link	link to *ParameterDescription*

Table 12: Attributes of *Parameter*. Attributes in bold are **mandatory**, references to other classes are indicated with an arrow (→). Attributes of *ParameterDescription* can be added here as well, if that class is not used.

ParameterDescription

Attribute	**Data type**	**Description**
id	string	parameter unique identifier
name	string	parameter name
annotation	string	additional free text description
datatype	string	datatype
unit	string	physical unit
ucd	string	Unified Content Descriptor, supplying a standardized classification of the physical quantity
utype	string	UType, meant to express the role of the parameter in the context of an external data model
min	number	minimum value
max	number	maximum value
options	list	list of accepted values

Table 13: Attributes of *ParameterDescription*.

For example, observations generally require information on *ambient conditions* as well as *instrument characteristics*. This contextual data associated with an observation is 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 entities 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.

2.2.8 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 (CTA) or an institute. This is also reflected in the IVOA Dataset Metadata Model, where *Party* represents an agent, and it has two types: *Individual* and *Organization*, which are explained in more detail in Table 14 (also see Section 3 for comparison between *Agent* and *Party*). Both agent types are also used in the W3C Provenance Data Model, though *Individual* is called *Person* there. We decided to not include the type *SoftwareAgent* from W3C (yet), since it is not required for our current use cases. This may change in the future.

AgentType

Class or type	**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 14: Agent class and types of agents/subclasses in this data model, compared to W3C ProvDM and DatasetDM.

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 referred 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

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, agency name...
type	prov:type	string	type of the agent: either Individual (Person) or Organization
email		string	Contact email of the agent
address		string	Address of the agent

Table 15: Agent attributes

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, a summary of the attributes for *Agent* is given in Table 15. We added the optional attributes `address` and `email`, since they appeared in our use cases and are quite commonly used. Not every project will need them; e.g. an advanced system may use permanent identifiers (e.g. ORCIDs) to identify agents and retrieve their properties from an external system instead. 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.

It is desired to have at least one agent given for each activity (and entity), but it is not enforced. There can also 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 that was used in the 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, ...).

AgentRoles

role	**type or sub class**	**Comment**
author	Individual	someone who wrote an article, software, proposal
contributor	Individual	someone who contributed to something (but not enough to gain authorship)
editor	Individual	editor of e.g. an article, before publishing
creator	Individual	someone who created a dataset, creators of articles or software are rather called "author"
curator	Individual	someone who checked and corrected a dataset before publishing
publisher	Organization	organization (publishing house, institute) that published something
observer	Individual	observer at the telescope
operator	Individual	someone performing a given task
coordinator/PI	Individual	someone coordinating/leading a project
funder	Organization	agency or sponsor for a project as in Prov-N
provider	Organization	"an organization that makes data and/or services available to users over the network" (definition from RM)

Table 16: Examples for roles of agents and the typical type of that agent

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 16. For comparison, SimDM contains following roles for their contacts: owner, creator, publisher and contributor. Note that the *Party* class in Dataset and SimDM are very similar to the *Agent* class, which is explained in more detail in Section 3.

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

The Provenance Data Model can be applied without making any links to other IVOA data model classes. For example, when the data is not yet published, provenance information can be stored already, but a DatasetDM-description for the data may not yet exist. However, if there are data models implemented for the datasets, then it is very useful to connect the classes and attributes of the other data models with provenance classes and attributes (if applicable), which we are going to discuss in this Section. These links help to avoid unnecessary repetitions in the metadata of datasets, and also offer the possibility to derive some basic provenance information from existing data model classes automatically, e.g. for creating provenance serializations from DatasetDM or SimDM metadata.

3.1 Links with Dataset/ObsCore Model

Entity and *EntityDescription* in ProvenanceDM are tightly linked to the *DataSet*-class in DatasetDM/ObsCoreDM, as well as to *InputDataset* and *OutputDataSet* in the SimulationDM (Lemson and Wozniak et al., 2012). Tables 17 and 18 map classes and attributes from DatasetDM to concepts in ProvenanceDM.

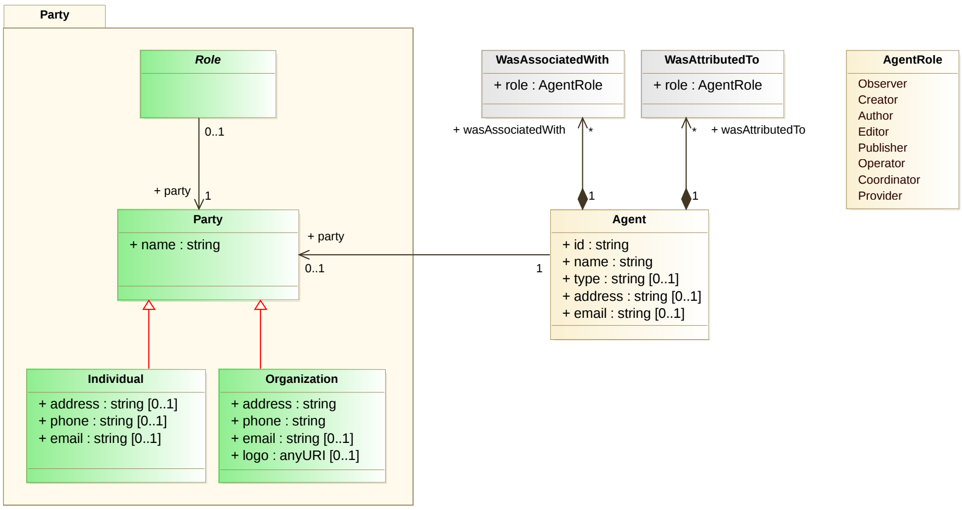

Figure 10: The relations between the *Agent* class within ProvenanceDM (grey and yellow classes) with classes from the DatasetDM, party package (green).

The *Agent* class, which is used for defining responsible persons and organizations in ProvenanceDM, is very similar to the *Party* class in DatasetDM (and in SimDM). Its details are depicted in Figure 10. The main difference between *Agent* and *Party* is that *Individual* and *Person* are subclasses in

ProvenanceDM attribute	DatasetDM attribute	Comment
Entity.id	Curation.PublisherDID	unique identifier for the dataset assigned by the publisher
Entity.id	DataID.creatorDID	alternative id for the dataset given by the creator, could be used as Entity.id if no PublisherDID exists (yet)
Entity.name	DataID.title	title of the dataset
Entity.rights	Curation.Rights	access rights to the dataset; one of [...]
Entity.creationTime	DataID.date	date and time when the dataset was completely created
HadMember.collection	DataID.collection	link to the collection to which the dataset belongs
WasGeneratedBy.activityId	DataID.ObservationID	identifier to everything describing the observation
Agent	Curation.Contact	link to Agent with role contact
Agent.id	Curation.PublisherID	link to the publisher, i.e. to an Agent with role="publisher"
Agent.name, wasAttributedTo.role= Creator	DataID.creator	name of agent creating the dataset
Agent.name, wasAttributedTo.role= Publisher	Curation.Publisher	name of the publisher

Table 17: Mapping attributes from DatasetDM classes to (optional) attributes in ProvenanceDM. This list is not complete.

DatasetDM, whereas we just use the same class *Agent* for both and distinguish between them using the *Agent.type* attribute (which can have the value "Individual" or "Organization"), which is closer to W3C's provenance data model.

We imagine that services implementing both data models, *Dataset* and *ProvenanceDM* may just use *one* class: either *Agent* or *Party*, enriched with all the necessary (project-specific) attributes. When delivering the data on request, the serialised versions can be adjusted to the corresponding notation. Note that for Provenance queries using a ProvTAP service or for W3C compatible serializations, the name *Agent* for the responsible individuals/organizations is required.

ProvenanceDM class	DatasetDM attribute	Comment
Entity	Curation.Date	release date of the dataset
Entity	Curation.Version	version of the dataset
Entity	Curation.Reference	link to publication
EntityDescription	DataProductType	the type of a dataproduct from DatasetDM can be used as attribute to EntityDescription
EntityDescription	DataProductSubType	subtype of a dataproduct/entity
EntityDescription	ObsDataset.calibLevel	(output) calibration level, integer between 0 and 3

Table 18: Mapping attributes from DatasetDM classes to the ProvenanceDM classes to which they could be added. Attributes like `EntityDescription.calibLevel` are very specific to entities described with DatasetDM and thus are not included in this ProvenanceDM directly. This list is not complete.

3.2 Links with Simulation Data Model

In SimDM one also encounters a normalization similar to our separation 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 direct mappings between classes and attributes of both models are given in Table 19.

If simulations are already described with SimDM, this table can be used to map from SimDM properties to ProvenanceDM, e.g. when serving serialized provenance metadata via an additional ProvDAL interface or for storing provenance metadata together with each released simulation dataset (e.g. in a VOTable).

ProvenanceDM	**SimDM**	**Comment**
Activity	Experiment	
Activity.name	Experiment.name	human readable name; name attribute in SimDM is inherited from Resource-class
Activity.endTime	Experiment.executionTime	end time of the execution of an experiment/activity
Activity.description	Experiment.protocol	reference to the protocol or ActivityDescription class
ActivityDescription	Protocol	
ActivityDescription.name	Protocol.name	human readable name
ActivityDescription.doculink	Protocol.referenceURL	reference to a webpage describing it
Parameter	ParameterSetting	value of an (input) parameter
ParameterDescription	InputParameter	description of an (input) parameter
Agent	Party	responsible person or organization
Agent.name	Party.name	name of the agent
WasAssociatedWith	Contact	classes for linking to agent/party
WasAssociatedWith.role	Contact.role	role which the agent/party had for a certain experiment (activity); SimDM roles contain: `owner`, `creator`, `publisher`, `contributor`
WasAssociatedWith.agent	Contact.party	reference to the agent/party
Entity	DataObject	a dataset, which can be/refer to a collection

Table 19: Mapping between classes and attributes from SimDM to classes/attributes in ProvenanceDM. This list is not complete.

4 Serialization of the provenance data model

4.1 Introduction

Serialization files constitute the building blocks of the client/server and client/client dialogs. The provenance information as represented in the data model is split in three main concepts that can be searched following many different relations between the main 3 classes, *Activity*, *Entity* and *Agent*. The selection of the relations to expose when distributing the provenance information depends on the usage and will be described more extensively in the use cases in Section 6, the Implementation Note (Riebe and Servillat et al., 2017) and the links therein.

To give a very simple example, suppose a client asks for the context of execution for one specified activity, which computes a simple RGB color composition. On the server side, exposing the provenance information for this activity or for an entity, corresponding to a monocolor or RGB image, means to just expose the structure of the classes and relation tables and feed them with the related tuples in the database. On the client side, the content of a VO-Provenance serialization document can then be explored and represented using graphical interfaces, as inspired by the Provenance Southampton suite or by customized visualization tools.

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

The serialization formats PROV-N, PROV-JSON and PROV-XML are proposed in the W3C Provenance framework for storing and exchanging the provenance metadata: PROV-N, PROV-JSON and PROV-XML, defined in Moreau and Missier (2013), Huynh and Jewell et al. (2013) and Hua and Tilmes et al. (2013), respectively. They can be reused here as well for serializations of our data model. For producing fully W3C compatible serializations, see Section 4.5.

Here is an example serialization instance document for an entity being processed by an activity, in PROV-N notation:

```
document
  prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
  prefix ex <http://www.example.com/provenance/>
  prefix voprov <http://www.ivoa.net/documents/dm/provdm/voprov/>
  entity(ivo://example#Public_NGC6946, [voprov:name="Processed image of NGC 6946"])
  entity(ivo://example#DSS2.143, [voprov:name="Unprocessed image of NGC 6946"])
  activity(ex:Process1, 2017-04-18T17:28:00, 2017-04-19T17:29:00, [voprov:name="Process 1"])
  used(ex:Process1, ivo://example#DSS2.143, -)
  wasGeneratedBy(ivo://example#Public_NGC6946, ex:Process1, 2017-05-05T00:00:00)
endDocument
```

Here is the same example in PROV-JSON format:

```
{
  "prefix": {
    "ivo": "http://www.ivoa.net/documents/rer/ivo/",
    "voprov": "http://www.ivoa.net/documents/dm/provdm/voprov/",
    "ex": "http://www.example.com/provenance/"
  },
  "activity": {
    "ex:Process1": {
      "voprov:startTime": "2017-04-18T17:28:00",
      "voprov:endTime": "2017-04-19T17:29:00",
      "voprov:name": "Process 1"
    }
  },
  "wasGeneratedBy": {
    "_:id4": {
      "voprov:time": "2017-05-05T00:00:00",
      "voprov:entity": "ivo://example#Public_NGC6946",
      "voprov:activity": "ex:Process1"
    }
  },
  "used": {
    "_:id1": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143",
      "voprov:activity": "hips:AlaRGB1"
    }
  }
  "entity": {
    "ivo://example#DSS2.143": {
      "voprov:name": "Unprocessed image of NGC6946"
    },
    "ivo://example#Public_NGC6946": {
      "voprov:name": "Processed image of NGC 6946"
    }
  }
}
```


PROV-JSON, PROV-N and PROV-XML can be converted into each other, e.g. using the prov or voprov python package (see Section 6.2).

4.3 PROV-VOTable format

To emphasize the compatibility to the IVOA framework, where the VOTable-XML format is a reference to circulate metadata, we define a PROV-VOTable mapping specification. All classes' declarations and relations described for this data model are translated into separated tables, one for each class of the model, see Appendix B. All attributes of these classes are translated to columns, i.e. VOTable FIELDS. In addition, the specification defines the VOTable values of the FIELD and PARAM attributes `ucd`, `datatype`, `utype`, `unit`, `description`, etc.

This can be appropriately used for two goals:

- Publishing full provenance metadata for data collections in VOTable format. This can be produced by data processing workflows or as output of databases containing provenance metadata.

- Providing the backbone for the TAP schema describing IVOA provenance metadata which isused for ProvTAP

These VOTable serializations can be produced using the voprov python module, available to the community, see also Section 6.2 and the IVOA ProvenanceDM Implementation Note (Riebe and Servillat et al., 2017).

Here is a VOTable document transcription of the serialization example given above in PROV-N and PROV-JSON:


```
<?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>
            <TD>Process 1</TD>
            <TD>2017-04-18 17:28:00</TD>
            <TD>2017-04-19 17:29:00</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Entity" utype="voprov:Entity">
```

```
      <FIELD arraysize="*" datatype="char" name="id" ucd="meta.id" utype="voprov:Entity.id"/>
      <FIELD arraysize="*" datatype="char" name="name" ucd="meta.title" utype="voprov:Entity.name"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://example#DSS2.143</TD>
            <TD>Unprocessed image of NGC6946</TD>
          </TR>
          <TR>
            <TD>ivo://example#Public_NGC6946</TD>
            <TD>Processed image of NGC 6946</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <INFO name="QUERY_STATUS" value="OK"/>
  </RESOURCE>
</VOTABLE>
```

This VOTable serialization can be considered as a flat view on the various tables stored in a database implementing the datamodel structure explained in Section 2. More examples of serialization documents are provided in Appendix A.

Such serializations can be retrieved through access protocols (see 5.1) or directly integrated in dataset headers or "associated metadata" in order to provide provenance metadata for these datasets. E.g. for FITS files, a provenance extension called "PROVENANCE" could be added which contains provenance information of the workflow that generated the FITS file. This information could be stored directly using one of the serialization formats, for example as a unique cell in an ASCII TABLE extension.

4.4 Serialization of description classes in the data processing context

Description classes in ProvenanceDM gather information on the data processing which can preexist to the processing itself. First, the *ActivityDescription* class gives generic information on the activity (`name`, `description`, `doculink`...) and the parameters expected as an input. In addition, *UsedDescription* and *WasGeneratedByDescription* classes indicate the expected roles of the input and output entities respectively. Finally, The activity may expect specific kinds of entities as inputs or outputs, for which there may be detailed descriptions stored as *EntityDescription* records.

The serialization of an ActivityDescription, that includes all those description classes, is based on the IVOA DataLink Service Descriptors for service resources (Dowler and Bonnarel et al., 2015), and can thus be stored as a VOTable (Ochsenbein and Williams et al., 2013). Indeed, a service descriptor points to a service that may execute an activity using the given input parameters, some of which probably point to entities. One may thus easily translate an ActivityDescription VOTable to a DataLink service descriptor VOTable block, and vice-versa.

The VOTable contains one resource with attributes type="meta" and utype="voprov:ActivityDescription". This resource contains PARAM elements to describe the activity and then GROUP elements gathering additional PARAM elements to describe:

- the input parameters (group name="InputParams"),
- the input entities (group name="Used"),
- the output entities (group name="Generated").

The standard PARAM elements for an activity resource correspond to the attributes of the *ActivityDescription* class (see Section 2.2.4) and may include an Agent name and email. Only one agent can be given, corresponding to the main contact for this activity ("contact_name" and "contact_email"). The `utype` attribute is used to connect the PARAM element to its location in the ProvenanceDM, so that other optional elements can be added.

For the input parameters, each attribute located in the *ParameterDescription* class in the model (e.g. `units`, `ucd`, `utype`, `min`, ...) is mapped to an attribute of a PARAM element in the VOTable (both have the same structure, see Section 2.2.7). The `type` attribute of the PARAM element can be set to "no_query" to indicate that the parameter is optional for the activity (i.e. a default value will be used).

For the input and output entity groups, each entity is described with a GROUP block that contains PARAM elements with e.g. the following names (all attributes describing the entity are optional):

- name="default" for the default identifier of the entity (e.g. a file name) or its default value (if the entity is a value),
- name="role" that gives the role of the entity with respect to the Used or WasGeneratedBy relation (e.g. "red", "green" or "blue" channel image, or the output "RGB" file),
- name="content_type" for the MIME type expected by the activity for the input or output entity,
- other additional PARAM elements corresponding to attributes of *EntityDescription*, *UsedDescription*, *WasGeneratedByDescription* or possibly *Agent*.

It is possible to reference an input parameter using the `ref` attribute of PARAM, if an input or output entity is referenced as an input parameter to the activity (e.g. the name of an input file). In the following example the output file name "RGB.jpg" is expected as a parameter to the activity and is thus referenced by the id PARAM element located in the GROUP of generated entities (with ref="RGB").

Here is an example of an *ActivityDescription* VOTable that describes an activity to create an RGB image from three input images mapped to the red, green, blue image planes in the composition.

```
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.ivoa.net/xml/VOTable/v1.3" version="1.3"
    xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.3
    http://www.ivoa.net/xml/VOTable/v1.3">
  <RESOURCE ID="make_RGB_image" name="make_RGB_image"
      type="meta" utype="voprov:ActivityDescription">
    <DESCRIPTION>Create an RGB image from 3 images</DESCRIPTION>
    <LINK content-role="doc" href="..." />
    <PARAM name="name" datatype="char" arraysize="*"
        value="make_RGB_image" utype="voprov:ActivityDescription.label" />
    <PARAM name="type" datatype="char" arraysize="*"
        value="..." utype="voprov:ActivityDescription.type"/>
    <PARAM name="subtype" datatype="char" arraysize="*"
        value="..." utype="voprov:ActivityDescription.subtype" />
    <PARAM name="version" datatype="float"
        value="..." utype="voprov:ActivityDescription.version" />
    <PARAM name="contact_name" datatype="char" arraysize="*"
        value="..." utype="voprov:Agent.name" />
    <PARAM name="contact_email" datatype="char" arraysize="*"
        value="...@..." utype="voprov:Agent.email" />
    <GROUP name="InputParams" utype="voprov:ParameterDescription">
      <PARAM ID="RGB" arraysize="*" datatype="char" name="RGB"
          type="no_query" value="RGB.jpg">
        <DESCRIPTION>RGB image name</DESCRIPTION>
      </PARAM>
    </GROUP>
    <GROUP name="Used" utype="voprov:UsedDescription">
      <GROUP name="R" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for red channel</DESCRIPTION>
        <PARAM name="default" value="R.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="red" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
      </GROUP>
      <GROUP name="G" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for green channel</DESCRIPTION>
        <PARAM name="default" value="G.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="green" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
           utype="voprov:EntityDescription.content_type" />
      </GROUP>
      <GROUP name="B" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for blue channel</DESCRIPTION>
        <PARAM name="default" value="B.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="blue" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
```

```
      </GROUP>
    </GROUP>
    <GROUP name="Generated" utype="voprov:WasGeneratedByDescription">
      <GROUP name="RGB" utype="voprov:EntityDescription">
        <DESCRIPTION>RGB image generated</DESCRIPTION>
        <PARAM name="role" value="RGB" arraysize="*" datatype="char"
            utype="voprov:WasGenereratedByDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
      </GROUP>
      </PARAM>
    </GROUP>
  </RESOURCE>
</VOTABLE>
```

4.5 W3C PROV-DM compatible serializations

According to our minimum requirements (see Section 1.2), it must be possible to serialize the provenance metadata into a format compatatible with the W3C Provenance Data Model (W3C PROV-DM), so that it can be exchanged within a wider context and can be processed by already existing tools, e.g. for visualizing provenance. W3C PROV-DM is a larger set of classes and relations compared to this model, but sharing the same core structure. It allows the possibility to add IVOA or *ad hoc* attributes to the basic ones in each class. Thus we can add our additional attributes without problems and still be W3C conform. However, we also definved a few additional classes and relations that are not W3C conform and thus need to be restructured. Using "voprov" as the namespace prefix for our model and "prov" for W3C PROV-DM, the necessary changes for mapping from ProvenanceDM to W3C PROV-DM are:

- namespace `voprov` → `prov` for those attributes that are the same in W3C (e.g. ID, role, startTime, endTime)
- attribute `voprov:name` → `prov:label`
- attribute `voprov:annotation` → `prov:description`
- attribute `prov:role` is not allowed in W3C's *WasAttributedTo*, thus use `voprov:role`
- *hadMember* has no ID and no optional attributes in W3C
- *Collection* → *Entity* with `prov:type = prov:collection`
- restructure *ActivityFlow*:
  - *ActivityFlow* → *Activity* with additional attribute `voprov:votype = 'voprov:activityFlow'`

    – replace *HadStep* relation by W3C's general *WasInfluencedBy* relation with additional attribute `voprov:votype = 'voprov:hadStep'` or just use `voprov:hadStep` as attribute in activities of type activityFlow

- restructure **Description* classes: add their attributes to the linked core classes, using a "desc_" prefix, e.g. Entity.desc_category.
- restructure *Parameter* and *ParameterDescription*: merge them into one parameter class, model it as an entity

This way, it is possible to produce W3C compatible serializations of our model with minimum information loss. W3C tools would ignore the voprov-attributes, whereas VO clients could make sense of this additional information and could even uncover the original structure or convert it to a VO serialization.

5 Accessing provenance information

5.1 Access protocols

We envision two possible access protocols:

- ProvDAL: retrieve provenance information based on given ID of a data entity or activity.
- ProvTAP: allows detailed queries for provenance information, discovery of datasets based on e.g. code version.

5.2 ProvDAL

ProvDAL is a simple data access layer interface (see DALI specification of the VO, Dowler and Demleitner et al., 2013) that can be implemented by a web service to serve provenance information to a client. The client sends GET request to the basic URL endpoint (`{provdal-base-url}`) of a ProvDAL service, providing at least the main parameter **ID**, the (unique, qualified) identifier of an entity (obs_publisher_did of an ObsDataSet for example), activity or an agent. This parameter can occur more than once in a request in order to retrieve provenance details for several activities, datasets or agents at the same time. Here are two simple example requests:

```
{provdal-base-url}?ID=rave:dr4
{provdal-base-url}?ID=rave:dr4&ID=rave:act_irafReduction
```

Additional parameters can complete the request to refine the query. They are described in the next paragraphs and summarized in Table 20.

Parameter	Values	Description
ID	qualified ID	a valid qualified identifier for an entity, activity or agent (can occur multiple times)
DEPTH	0,1,2,..., ALL	number of relations to be followed or `ALL` for everything, independent of the relation type
RESPONSEFORMAT	PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE	serialisation format of the response
DIRECTION	BACK, FORTH	BACK = track the provenance history, FORTH = explore the results of activities and where entities have been used
MEMBERS	true (1) or false (0)	if true/1, retrieve and track members of collections
STEPS	true (1) or false (0)	if true/1, retrieve and track steps of activityFlows
AGENT	true (1) or false (0)	if true/1, explore all relations for agents, i.e. find out what an agent is responsible for
MODEL	IVOA or W3C	compatibility of the serialization to the IVOA or W3C provenance data model

Table 20: ProvDAL request parameters. Options that are **required** to be implemented by ProvDAL services are marked with bold face. Default values are underlined. The parameter names are case-insensitive, but the parameter values are not.

RESPONSEFORMAT The format of the response can be defined using the RESPONSEFORMAT parameter. Its value is one of the provenance serialization formats: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE.

DEPTH The DEPTH parameter gives the number of relations that shall be tracked along the provenance history – independent of the type of relation. Its value is either 0, a positive integer or ALL. If this parameter is omitted, the default is 1, which returns all relations and nodes that can be reached by following 1 relation. If `DEPTH=ALL` is requested, the server should return the complete provenance history that the service has stored for the given entity, activity or agent. Services may restrict the returned data by redirecting `DEPTH=ALL` to `DEPTH={maxdepth}`, where `{maxdepth}` is an integer defining the maximum depth number that the server allows.

If description classes (EntityDescription etc.) are used, we expect that the descriptions are retrieved together with each corresponding main class. For example, for each entity node also return the entityDescription – either by adding the description attributes to the entity in the serialized response, by adding a link to a URL with entityDescription information or by returning a direct serialization of the description class along with a link from the entity to the entityDescription.

Note that the relations *wasDerivedFrom* and *wasInformedBy* are "short-cuts" in a provenance graph. Thus for e.g. `DEPTH=2` more progenitors of an entity may be reached via *wasDerivedFrom* relations than via the "long path" along the corresponding *used* and *wasGeneratedBy* relations (see e.g. progenitor entity E1 in Figure 11). (A better solution for the future may be to use 1/2*DEPTH for walking along these short-cut relations, but we don't want to make ProvDAL more complex for now.)

DIRECTION For services which allow tracking the provenance information forward, e.g. in order to check for which activities an entity was used, the optional parameter DIRECTION can be set to FORTH. Its default value is BACK. This only influences the direction in which the used, wasGeneratedBy, wasDerivedFrom and wasInfluencedBy relations are followed. Any other relations are tracked according to the behaviour specified below, independent of the DIRECTION value.

Figure 11 shows an example provenance graph with different relations and nodes. Only the relations marked by solid lines are influenced by the DIRECTION parameter. A ProvDAL GET request with `ID=E6` and `DEPTH=2` returns only the highlighted nodes and relations (thick lines) by default.

MEMBERS, STEPS The provenance data model defines the hierarchical relations *hadMember* for entity collections and *hadStep* for activityFlows. If a node belongs to a collection or activityFlow, these relations shall be returned as well, independent of the specified tracking direction. If someone is interested in more details and wants to follow the *members* of an entity collection or the *steps* of an activityFlow, these can be included by setting the optional parameter MEMBERS or STEPS to true, respectively. The default is false. As detailed in DALI (Dowler and Demleitner et al., 2013), the values 1 and 0 are equivalent to true and false.

AGENT By default, it is recommended to stop any further tracking at an agent node, unless an additional optional parameter AGENT is set to true. Note that this means that the request for any agent will always return just the agent node itself and nothing else, unless AGENT=true is used. An example request if one wants to know which entities and activities an agent has influenced could look like this:

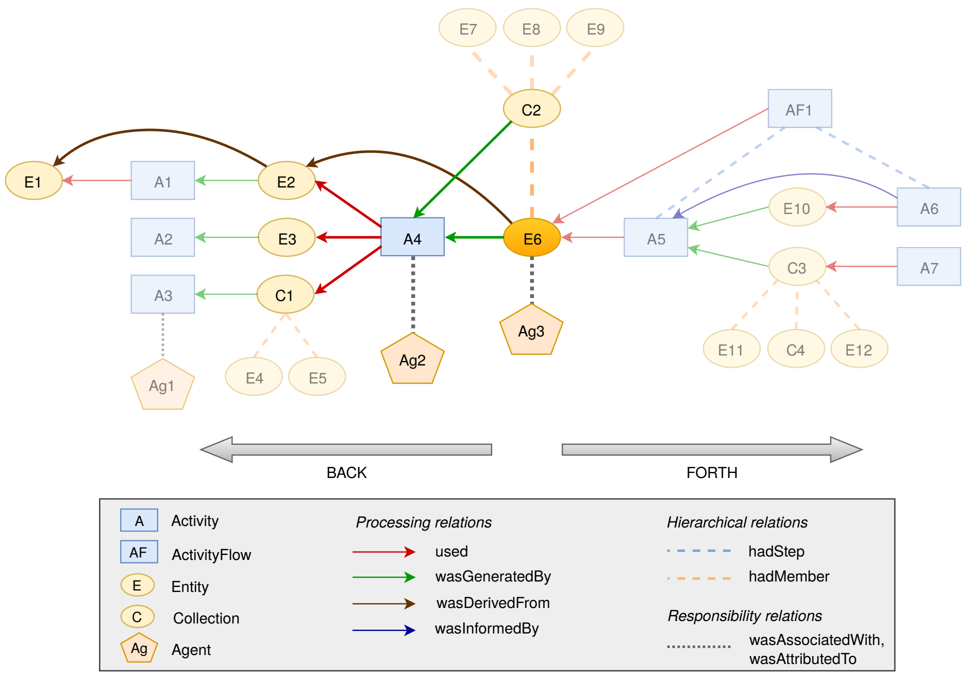

Figure 11: An example provenance graph, highlighting the objects and relations returned from a ProvDAL service with `ID=E6` and `DEPTH=2`. The BACK and FORTH values for DIRECTION are only important for the processing relations (solid lines). Hierarchial (dashed) and responsibility (dotted) relations are only followed "upwards" (to collection/activityFlow) and towards agents by default (unless the optional parameters MEMBERS, STEPS and/or AGENT are set to true.

`{provdal-base-url}?ID=org:rave&AGENT=true&DEPTH=1`.
`DEPTH=1` is used here in order to avoid following the found entities and activities any further (can be omitted, since this is the default for DEPTH).

MODEL If a provenance service supports it, the MODEL parameter is used to distinguish between serializations compatible with the IVOA or the W3C provenance data model. The default value is IVOA.

A ProvDAL service MUST implement the parameters ID, DEPTH and RESPONSEFORMAT; the remaining parameters are optional. If a service does not implement the optional parameters, but they appear in the request, then the service should return with an error. Please note that according to the DALI specification (Dowler and Demleitner et al., 2013), the parameter names are case-insensitive, but the parameter values are not. E.g. `direcion=FORTH` is allowed, but `DIRECTION=forth` may not work.

5.2.1 ProvDAL example use cases

Here are a few example use cases for ProvDAL in order to show its usefulness for astronomical datasets.

- The RAVE DR4 release contains a main table with stellar properties for each observation of a star. Given the RAVE observation ID, retrieve all the processing steps for this specific observation result:

  ```
  {provdal-base-url}?ID=rave:20121220_0752m38_089&DEPTH=ALL
  ```

  The result will not only contain the processing steps (activities), but also entities and agents. The information that the user is actually interested in can be filtered out by a client application (e.g. using the voprov python package). If a W3C tool shall be used, e.g. when the result shall be loaded to ProvStore[3] for further processing, one needs to retrieve the information in a W3C compatible way like this:

  ```
  {provdal-base-url}?ID=rave:20121220_0752m38_089&DEPTH=ALL&MODEL=W3C
  ```

- Get the direct progenitor of an entity:

  ```
  {provdal-base-url}?ID=rave:20121220_0752m38_089&DEPTH=1
  ```

  If this request only returns a collection and no "backwards" information about progenitors, then one needs to track the collection further, i.e. repeat the request for the collection entity.

- Get all datasets that were derived from a specific data file in the CTA pipeline:

  ```
  {provdal-base-url}?ID=cta:df1&DEPTH=ALL&DIRECTION=FORTH
  ```

ProvDAL is meant to be used to just retrieve parts of a provenance graph from a provenance web service. It cannot be used to explore the provenance graphs and filter the information based on specific properties, e.g. the creationTime of an entity or a parameter value for an activity. For such cases, a ProvTAP service can be used – which is described in the next section.

[3]https://provenance.ecs.soton.ac.uk/store/

5.3 ProvTAP

ProvTAP is a TAP service specialization for delivering IVOA Provenance metadata. As for any TAP service (Dowler and Rixon et al., 2010), it is providing ADQL query responses consisting of a single table each, gathering columns selected from various tables defined in the service's TAP schema. By default, this table is formatted as a VOTable. Other formats (JSON, tsv, etc.) could be added optionally.

The table and column definitions available in the TAP schema, including their metadata (`unit`, `ucd`, `utype`, `datatype`, `description`), are exactly the ones defined in the PROV-VOTable mapping of the IVOA Provenance data model (see table and column definitions and metadata in Appendix B). This corresponds to what has been done in the RegTAP specification (Demleitner and Harrison et al., 2013) for mapping the VOResource data model.

Here are a few examples of ADQL queries for ProvTAP:

- To retrieve all activity metadata for activies sharing the same activityDescription:
```
SELECT * FROM Activity WHERE
Activity.a_description = 'hipsgend_mean'
```
- To retrieve all activities associated with agent `obspm`:
```
SELECT WasAssociatedWith.waw_activity_id, Activity.a_name,
       Activity.a_annotation FROM WasAssociatedWith
INNER JOIN Activity ON
WasAssociatedWith.waw_activity_id = Activity.a_id
WHERE WasAssociatedWith.waw_agent_id = 'obspm'
```
- To retrieve all entities attributed to `curator` agents:
```
SELECT WasAttributedTo.wat_entity_id FROM WasAttributedTo
WHERE WasAttributedTo.wat_role = 'curator'
```

5.4 VOSI availability and capabilities

According to the DALI specification for VO services (Dowler and Demleitner et al., 2013), a provenance service implementing ProvDAL and/or ProvTAP must provide a VOSI availability interface as well as a capabilities interface with entries for ProvDAL and/or ProvTAP. The `standardId`s for these provenance interfaces are:

```
ivo://ivoa.net/std/ProvenanceDM#ProvDAL-1.0
ivo://ivoa.net/std/ProvenanceDM#ProvTAP-1.0
```

The capability for a TAP service to support the Provenance DM is expressed by the dataModel element as:

```
<dataModel ivoid="ivo://ivoa.net/std/ProvenanceDM-1.0">
  ProvenanceDM-1.0
</dataModel>
```

For ProvTAP, the VOSI tables interface must also be provided.

6 Use cases – applying the data model

This section presents some general guidelines for applying the data model and specific use cases for which the provenance data model helps to solve certain tasks. Details on specific implementions of the provenance data model are provided in a separate document, the ProvenanceDM Implementation Note (Riebe and Servillat et al., 2017).

6.1 How to use the data model

The IVOA Provenance Data Model has been developed along with its implementations from different projects. We gather here some tips to implement and use the model for a specific project.

Before using the model We noticed that the simple knowledge of what is provenance information is important for the conception of all projects. Before using or not ProvenanceDM and associated services, it is good practice to locate and collect information on the activities, entities and agents that will be manipulated, and be sure that this information is not lost along the way. For example, a script may use intermediate files (such as calibration files for observations) that may not be tracked by the system.

Define unique identifiers 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. IVOIDs, DOI's or ORCIDs are potentially good candidates for unique identifiers.

Use of the description classes One may only use the core data model without the description classes if they are not needed for the project. In that case, it is recommended to merge the attributes of the description classes into the main classes (*Entity*/*Activity*/*Agent*, and if needed *Parameter*). In the same way, when serializing the provenance information, the description classes can be merged to the main classes, which is needed to produce W3C compliant provenance files.

Add project specific attributes to your entities, activities and agents We proposed generic attributes for the different classes, but there are probably project specific attributes that need to be added. It is also required at this level to disentangle *Entity* properties and *EntityDescription* properties: everything that you may know about an entity before its creation, belongs to the *EntityDescription* (e.g. file `format`, `content_type`, `dataproduct_type`, `category`, ...).

Group common features for entities and/or activities If inside the project, the different entities that will be manipulated are already defined precisely, as well as the activities producing them, then the description classes are probably of interest and will help reducing the redundancy in the provenance information stored.

Create ActivityDescription files A model using description classes to define 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. If description classes are relevant for a project, it may be sufficient to store the descriptions directly as *ActivityDescription* files (see Section 4.4). Those can be created and centralized before implementing a provenance database or service.

Link to an Authentication System We proposed to add optional attributes to the *Agent* class (email, phone, address). Some projects may include an authentication system with a user directory. In that case a link should be kept between the *Agent* identifiers and the ones used in the authentication system when it is relevant, and information should be synchronized. However, there may be some agents that are not defined in the authentication system, so the information may not be easy to merge.

Adding additional metadata to an existing Entity It could happen that after creating entities and storing their provenance, additional metadata is needed for those entities. For example, an ObsCoreDM description may have to be added to some entities when they are made available to the public. In that case, the entity will receive an external identifier (e.g. a `publisher_did`), which should thus be associated to the entity provenance identifier (Entity.id) in a relation table. This allows to match existing Ids between different data models metadata descriptions as highlighted in the mapping table 17 in Section 3.

6.2 voprov Python package

The voprov[4] package is an open source Python library derived from the prov Python library (MIT license) developed by Trung Dong Huynh (University of Southampton).

The prov package implements the W3C Provenance Data Model. It offers to describe the provenance and provides different output formats of the serialized data: PROV-N, PROV-JSON, PROV-XML and to build diagrams in the following graphic formats: PDF, PNG, SVG.

The voprov library allows users to describe the provenance of their data according to the IVOA Provenance Data Model. It allows the description of flows of activities (pipelines) with eventually the composition of the different steps. It provides the VOTable serialization in PROV-VOTable format.

This library is currently used in the context of the POLLUX database which hosts synthetic stellar spectra. The provenance files are created from the non normalized information found in the Pollux header files. The serialization is proposed in three levels of detail and in different formats: on one hand in PROV-N, PROV-JSON, PROV-XML and PROV-VOTable formats and on the other hand in PDF, PNG and SVG graphic formats. The VO user or the VO tool is informed of the existence of the provenance in a DataLink entry of the Simple Spectral Access protocol (SSAP) response which gives information on how to retrieve a given provenance file.

Code, serializations and graphic format conversions examples are given in the Implementation Note (Riebe and Servillat et al., 2017).

6.3 Provenance of RAVE database tables

The RAVE[5] (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 RAVE data releases. Providing provenance information, 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.

We put together some major steps of the RAVE pipeline in W3C-compatible PROV-N notation and uploaded the file to the Provenance Store[6]. This allows to view graphs of the workflow by visualizing only the main entities or activities and restricting the displayed relations. It shows that the provenance concepts explained in this draft can be applied directly to data obtained from astronomical observations.

[4] https://github.com/sanguillon/voprov
[5] http://www.rave-survey.orgsurvey
[6] https://provenance.ecs.soton.ac.uk/store/

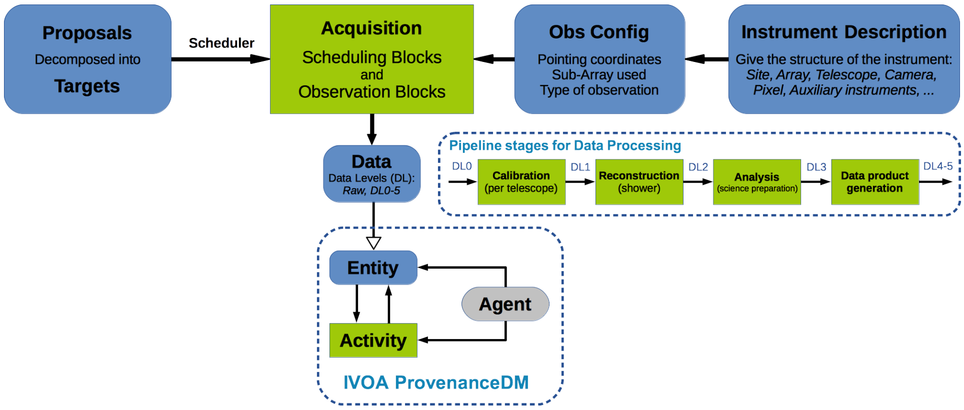

Figure 12: CTA high level data model structure with Pipeline stages and connection to IVOA ProvenanceDM.

We also tested a Django implementation of the classes in this document along with provenance metadata from the RAVE project stored in an SQLite database. This allows to quickly setup a provenance web service with interfaces to view all instances of a class or details for a single object (REST API), extract provenance information for single entities and visualizing the provenance information using Javascript. More details about this web application are available in the Implementation Note (Riebe and Servillat et al., 2017).

6.4 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 12.

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 12. 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 during the data analysis using the Python `prov` package inside OPUS[7] (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 serializations, as well as a graph visualization (see Figure 13).

The CTA Pipeline contains a specific Provenance class dedicated to the collection of provenance information after each processing step. This information is returned as an output file for now.

More details about the related implementations are available in the implementation notes (Riebe and Servillat et al., 2017).

6.5 Provenance for the 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 line lists from VALD database and specific molecular line lists 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, VOTable, ...) 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 to another format the description of the provenance of these spectra and on a second hand to select data depending on provenance criteria.

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

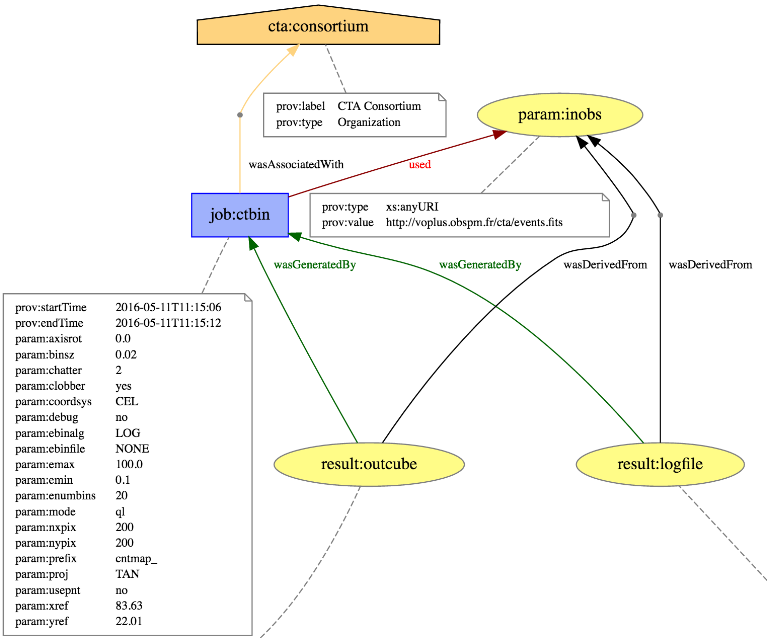

Figure 13: Provenance description of a CTA analysis step.

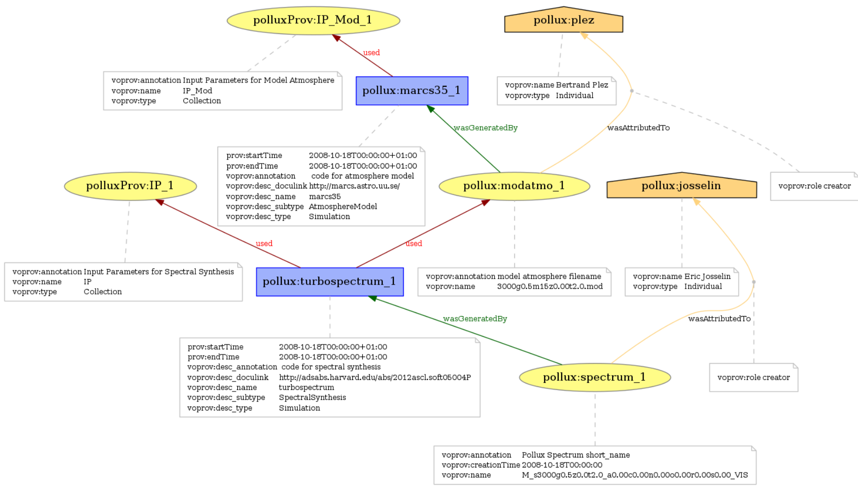

Figure 14: Example for provenance metadata for a POLLUX spectrum, produced from a W3C compatible format (e.g. using `prov:startTime` instead of `voprov:startTime`).

An example visualization of (W3C compatible) provenance metadata for a POLLUX spectrum is given in Figure 14.

6.6 Provenance of HiPS datasets

HiPS (Fernique and Allen et al., 2015) is an IVOA recommendation for a new hierarchical organization of pixel data, allowing smooth browsing of all-sky data collections in astronomy. It is based on the HEALPix tesselation of the sky into equal-area cells for a given HEALPix order, where each cell is given an image tile. Adaptative resolution is achieved by a hierarchy of tiles at increasing order. Sorting and organization is based on a tree of nested directories, each of which is associated with a tile. In the processing chain, HiPS can be seen as a kind of "legacy level" for observational data.

A HiPS dataset can be generated either by Aladin in "hipsgen" mode or by other software. The processing distinguishes 3 main different methods for estimating cell values with parameters: FIRST (nearest neighbour), MEAN, and MEDIAN of the neighbouring pixels. Up to 50 parameters can help to tune the processing, among them the highest resolution's HEALPix order, the sky background value to be substracted, the border width or the mask to apply to original images in order to avoid including bad areas, etc.

An example of provenance metadata for a 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 is given at https://volute.g-vo.org/svn/trunk/projects/dm/provenance/example/HiPS-prov-provn.txt, the corresponding VOTable-format is available at https://volute.g-vo.org/svn/trunk/projects/dm/provenance/example/HiPS-prov-vot.xml.

Here is an excerpt of the corresponding PROV-N serialization:

```
prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
prefix ex <http://www.example.com/provenance/>
prefix voprov <http://www.ivoa.net/documents/dm/provdm/voprov/>
prefix ds <http://www.ivoa.net/documents/dm/datasetdm/>
prefix hips <http://www.ivoa.net/documents/apps/hips/>
Entity
( ivo://CDS/P/MAMA/ESO-R,
[ voprov:name = "ESO-R MAMA HIPS at CDS",
voprov:annotation = "HiPS version of ESO Schmidt survey digitized by Mama and processed by CDS",
voprov:type= "voprov:entity",
voprov:doculink = "http://cds.u-strasbg.fr/hips/documentation.html#structure",
ds:access_reference = "http://CDS/P/MAMA/ESO-R", // as defined in ObsCore and Dataset DM
ds:calibLlevel = 3,
ds:dataproduct_type = "voprov:hips_pixels",
hips:HiPS_properties = "http://cds.u-strasbg.fr/hips/p/mama/eso-r/properties.txt"] )
// Relationship
WasAttributedTo(ivo://CDS/P/MAMA/ESO-R, ivo://cds, voprov:role= "voprov:creator")
Agent
(ivo://cds,
[ voprov:name= "CDS",
voprov:email = "question@astro.unistra.fr",
voprov:type = "Organisation" ])
WasGeneratedBy  (ivo://CDS/P/MAMA/ESO-R, EHG1, -)
Activity
(EHG1,
[ voprov:name = "ESO HiPS generation 1",
```

```
voprov:startTime = "2016-07-18",
voprov:endTime = "2016-07-20",
voprov:annotation = "Final generation activity of HiPS for ESO Mama survey",
voprov:activityDescription = "HipsgenM"] )
ActivityDescription
(HipsgenM,
[ voprov:name = "HiPSgen_Mean",
voprov:type = "data encoding",
voprov:subtype= "HiPSgen",
voprov:doculink = "http://cds.u-strasbg.fr/HiPSGEN-Documentation"])
WasAssociatedWith( EHG1, Buga, voprov:role="voprov:operator")
WasAssociatedWith(EHG1, ivo://CDS, voprov:role="voprov:creator")
WasAttributedTo(( ivo://CDS/P/MAMA/ESO-R, buga, voprov:role= "voprov:operator")
```

Appendix A Serialization Examples

Here is a a simple example of serialization of ProvenanceDM metadata for describing an activity of color composition and the entity used as input as well as the resulting RGB image.

The PROV-N format Moreau and Missier (2013) as proposed by the W3C is a text format which allows the description of instances of the 3 main classes, as well as the various relations between each instance involved.

Listing 1: PROV-N serialisation example for a color composition activity

```
document
  prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
  prefix hips <http://cds.u-strasbg.fr/data/>
  prefix voprov <http://www.ivoa.net/documents/dm/provdm/voprov/
     >

  entity(ivo://CDS/P/DSS2color#RGB_NGC6946,
        [voprov:annotation="PNG RGB image built from DSS2 with
            Aladin for galaxy NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gml",
        voprov:name="RGB DSS2 image for NGC 6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143,
        [voprov:annotation="DSS2 digitization of the Blue POSSII
             Schmidt survey around  NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gm",
        voprov:name="POSSII Blue Survey DSS2 NGC6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143,
        [voprov:annotation="DSS2 digitization of the Red POSSII
            Schmidt survey around NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gml",
        voprov:name="POSSII Red Survey DSS2 NGC6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143,
        [voprov:annotation="DSS2 digitization of the Infra red
            POSSII Schmidt survey around NGC 6946",
```

```
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gm",
        voprov:name="POSSII Infra Red Survey DSS2 NGC6946"])

  activity(hips:AlaRGB1, 2017-04-18T17:28:00, 2017-04-19
      T17:29:00, [
        voprov:name="Aladin RGB 1",
        voprov:annotation="Aladin RGB image generation for NGC
            6946",
        voprov:desc_id="AlaRGB",
        voprov:desc_type="RGBencoding",
        voprov:desc_name="Aladin RGB image generation algorithm"
            ,
        voprov:desc_doculink="http://cds.u-strasbg.fr/aladin.gml
            "])

  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143'
      , -)
  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143'
      , -)
  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143'
      , -)

  wasGeneratedBy('ivo://CDS/P/DSS2color#RGB_NGC6946',
      hips:AlaRGB1, 2017-05-05T00:00:00)
endDocument
```

Here is the transcription of the same metadata in the PROV-JSON format (Huynh and Jewell et al., 2013). Each class and relation of the provenance model lists its corresponding database table tuples grouped by the name of the table.

Listing 2: JSON serialization example for a color composition activity

```
{
  "prefix": {
    "ivo": "http://www.ivoa.net/documents/rer/ivo/",
    "voprov": "http://www.ivoa.net/documents/dm/provdm/voprov/",
    "hips": "http://cds.u-strasbg.fr/data/"
  },
  "activity": {
    "hips:AlaRGB1": {
      "voprov:name": "Aladin RGB 1",
      "voprov:startTime": "2017-04-18T17:28:00",
                         "voprov:endTime": "2017-04-19T17:29:00",
      "voprov:annotation": "Aladin RGB image generation for NGC
          6946",
      "voprov:desc_type": "RGBencoding",
      "voprov:desc_name": "Aladin RGB image generation algorithm
          ",
      "voprov:desc_doculink": "http://cds.u-strasbg.fr/aladin.
          gml",
      "voprov:desc_id": "AlaRGB"
    }
```

```
    },
    "wasGeneratedBy": {
      "_:id4": {
        "voprov:time": "2017-05-05T00:00:00",
        "voprov:entity": "ivo://CDS/P/DSS2color#RGB_NGC6946",
        "voprov:activity": "hips:AlaRGB1"
      }
    },
    "used": {
      "_:id1": {
        "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2
            .143",
        "voprov:activity": "hips:AlaRGB1"
      },
      "_:id3": {
        "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2
            .143",
        "voprov:activity": "hips:AlaRGB1"
      },
      "_:id2": {
        "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2
            .143",
        "voprov:activity": "hips:AlaRGB1"
      }
    },
    "entity": {
      "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143": {
        "voprov:name": "POSSII Blue Survey DSS2 NGC6946",
        "voprov:annotation": "DSS2 digitization of the Blue POSSII
             Schmidt survey around  NGC 6946",
        "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gm"
      },
      "ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143": {
        "voprov:name": "POSSII Red Survey DSS2 NGC6946",
        "voprov:annotation": "DSS2 digitization of the Red POSSII
            Schmidt survey around NGC 6946",
        "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gml"
      },
      "ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143": {
        "voprov:name": "POSSII Infra Red Survey DSS2 NGC6946",
        "voprov:annotation": "DSS2 digitization of the Infra Red
            POSSII Schmidt survey around NGC 6946",
        "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gm"
      },
      "ivo://CDS/P/DSS2color#RGB_NGC6946": {
        "voprov:name": "RGB DSS2 image for NGC 6946",
        "voprov:annotation": "PNG RGB image built from DSS2 with
            Aladin for galaxy NGC 6946",
        "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gml"
      }
    }
}
```


Here is the mapping obtained for the same data description with the

PROV-VOTABLE serialisation format.

Listing 3: PROV-VOTABLE serialisation example for a color composition activity

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2" xmlns="http://www.ivoa.net/xml/VOTable/v1
    .2"
xmlns:hips="http://cds.u-strasbg.fr/data/"
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 name="activity" ucd="meta.id" utype="voprov:Using.
          Activity"arraysize="*" datatype="char" />
      <FIELD name="entity" utype="voprov:used.Entity arraysize="
          *" datatype="char" ucd="meta.id" "/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>hips:AlaRGB1</TD>
            <TD>ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Generation" utype="voprov:wasGeneratedBy">
      <FIELD name="entity" utype="voprov:wasGeneratedBy.Entity"
          arraysize="*" datatype="char" ucd="meta.id"/>
      <FIELD name="activity" utype="voprov:wasGeneratedBy.
          activity" arraysize="*" datatype="char" ucd="meta.id"
          />
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://CDS/P/DSS2color#RGB_NGC6946</TD>
            <TD>hips:AlaRGB1</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Activity" utype="voprov:Activity">
      <FIELD name="id" utype="voprov:Activity.id" ucd="meta.id"
           arraysize="*" datatype="char" />
      <FIELD name="name" utype="voprov:Activity.name" ucd="meta.
          title" arraysize="*" datatype="char" />
      <FIELD name="start" utype="voprov:Activity.startTime" ucd=
          "time.start" arraysize="*" datatype="char" />
      <FIELD name="stop" utype="voprov:Activity.endTime" ucd=""
          time.end" arraysize="*" datatype="char" "/>
```

```
    <FIELD name="annotation" utype="voprov:Activity.annotation
        " ucd="meta.description" arraysize="*" datatype="char"
        />
    <FIELD name="desc_id" utype="voprov:ActivityDescription.id
        " ucd="meta.id" arraysize="*" datatype="char" />
    <FIELD name="desc_name" utype="voprov:ActivityDescription.
        name" ucd="meta.title" arraysize="*" datatype="char" />
    <FIELD name="desc_type" utype="voprov:ActivityDescription.
        type" ucd="meta.code.class" arraysize="*" datatype="
        char" />
    <FIELD name="desc_doculink" utype="
        voprov:ActivityDescription.doculink" ucd="meta.ref.url"
         arraysize="*" datatype="char" />
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>hips:AlaRGB1</TD>
          <TD>Aladin RGB 1</TD>
          <TD>2017-04-18 17:28:00</TD>
          <TD>2017-04-19 17:29:00</TD>
          <TD>Aladin RGB image generation for NGC 6946</TD>
          <TD>AlaRGB</TD>
          <TD>Aladin RGB image generation algorithm</TD>
          <TD>RGB encoding</TD>
          <TD>http://cds.u-strasbg.fr/aladin.gml</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <TABLE name="Entity" utype="voprov:Entity">
    <FIELD name="id" utype="voprov:Entity.id" ucd="meta.id"
        arraysize="*" datatype="char" />
    <FIELD name="name" utype="voprov:Entity.name" ucd="meta.
        title" arraysize="*" datatype="char" />
    <FIELD name="annotation" utype="voprov:Entity.annotation"
        ucd="meta.description" arraysize="*" datatype="char" />
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143</TD>
          <TD>POSSII Blue Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Blue POSSII Schmidt
              survey around NGC 6946</TD>
        </TR>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143</TD>
          <TD>POSSII Red Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Red POSSII Schmidt
              survey around NGC 6946</TD>
        </TR>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143</TD>
          <TD>POSSII Infra Red Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Infra red POSSII
```

```
            Schmidt survey around  NGC 6946</TD>
        </TR>
        <TR>
          <TD>ivo://CDS/P/DSS2color#RGB_NGC6946</TD>
          <TD>RGB DSS2 image for NGC 6946</TD>
          <TD>PNG RGB image built from DSS2 with Aladin for
            galaxy NGC 6946</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <INFO name="QUERY_STATUS" value="OK"/>
</RESOURCE>
</VOTABLE>
```

Appendix B PROV-VOTable serialisation and ProvTAP TAP schema

PROV-VOTable is a serialization format mapping the Provenance data model into VOTable. This format is useful to provide full serialization of Provenance metadata for an archive or data reduction workflow in VOTable. It can also be used to build the TAP schema of a ProvTAP service.

B.1 Tables in the PROV-VOTable serialization

The following table gives the full list of tables necessary to map the IVOA provenance data model. The contents of the 'name' and 'utype' columns of this table are normative.

B.2 Detailed table description

Tables 22 to 34 provide full descriptions of the FIELDS of each of the mapping tables. The contents of all these tables' columns are normative. For convenience, in all these tables an arrow (→) indicates columns that contain references (foreign keys) to other tables. The column names are prefixed by a short-cut for the table name in order to ensure unique column names in queries with table joins for ProvTAP.

Name	utype	Description
Entity	voprov:Entity	instances of *Entity* class
EntityDescription	voprov:EntityDescription	instances of *EntityDescription* class
Activity	voprov:Activity	instances of actvity class
ActivityDescription	voprov:ActivityDescription	instances of *ActivityDescriptions*
Agent	voprov:Agent	instances of *Agent* class
Parameter	voprov:Parameter	instances of *Parameter* class
ParameterDescription	voprov:ParameterDescription	instances of *ParameterDescription*
Used	voprov:Used	instances of *Used* relationships
UsedDescription	voprov:UsedDescription	instances of *UsedDescription* relationships
WasGeneratedBy	voprov:WasGeneratedBy	instances of *WasGeneratedBy relationships*
WasGeneratedByDescription	voprov:WasGeneratedByDescription	instances of *WasGeneratedByDescription* relationships
WasAssociatedWith	voprov:WasAssociatedWith	instances of *WasAssociatedWith* relationships
WasAttributedTo	voprov:WasAttributedTo	instances of *WasAttributedTo* relationships

Table 21: List of tables in the PROV-VOTable serialization

Name	ucd	utype	datatype
e_id	meta.id	voprov:Entity.id	char
e_name	meta.title	voprov:Entity.name	char
e_type	meta.code.class	voprov:Entity.type	char
e_rights	meta.code.class	voprov:Entity.rights	char
e_annotation	meta.description	voprov:Entity.annotation	char
→ e_hadMember	meta.code.member	voprov:Entity.hadMember	char
→ e_description	meta.id	voprov:entity.description	char
→ e_usedEntity	meta.id	voprov:entity.wasDerivedFrom.usedEntity	char

Table 22: Column description for entity table

Name	ucd	utype	datatype
ed_id	meta.id	voprov:EntityDescription.id	char
ed_name	meta.title	voprov:EntityDescription.name	char
ed_doculink	meta.ref.url	voprov:EntityDescription.doculink	char
ed_category	meta.code.class	voprov:EntityDescription.category	char
ed_annotation	meta.description	voprov:EntityDescription.annotation	char

Table 23: Column description for entityDescription table

Name	**ucd**	**utype**	**datatype**
a__id	meta.id	voprov:Activity.id	char
a__name	meta.title	voprov:Activity.name	char
a__startTime	time.start	voprov:Activity.startTime	char
a__stopTime	time.stop	voprov:Activity.stopTime	char
a__annotation	meta.description	voprov:Activity.annotation	char
a__votype	meta.code.class	voprov:Activity.votype	char
→ a__hadStep	meta.code.member	voprov:Activity.hadStep	char
→ a__description	meta.id	voprov:Activity.description	char
→ a__parameter	meta.id	voprov:Activity.parameter	char
→ a__informant	meta.id	voprov:Activity.wasInformedBy.informant	char

Table 24: Column description for activity table

Name	**ucd**	**utype**	**datatype**
ad__id	meta.id	voprov:ActivityDescription.id	char
ad__name	meta.title	voprov:ActivityDescription.name	char
ad__type	meta.code.class	voprov:ActivityDescription.type	char
ad__subtype	meta.code.class	voprov:ActivityDescription.subtype	char
ad__annotation	meta.description	voprov:ActivityDescription.annotation	char
ad__doculink	meta.ref.url	voprov:ActivityDescription.doculink	char
→ ad__param	meta.id	voprov:ActivityDescription.parameter	char

Table 25: Column description for activityDescription table

Name	**ucd**	**utype**	**datatype**
ag__id	meta.id	voprov:Agent.id	char
ag__name	meta.title	voprov:Agent.name	char
ag__type	meta.code.class	voprov:Agent.type	char OPTIONS
ag__address	meta.address	voprov:Agent.address	char
ag__email	meta.email	voprov:Agent.email	char

Table 26: Column description for agent table

Name	**ucd**	**utype**	**datatype**
p__id	meta.id	voprov:Parameter.id	char
p__value	stat.value	voprov:Parameter.value	param dependent
→ p__description	mata.id	voprov:Parameter.descripton	reference to parameter description

Table 27: Column description for parameter table

Name	ucd	utype	datatype
pd_id	meta.id	voprov:ParameterDescription.id	char
pd_name	meta.title	voprov:ParameterDescription.name	param dependent
pd_annotation	meta.description	voprov:ParameterDescription.annotation	char
pd_datatype	meta	voprov:ParameterDescription.datatype	char
pd_unit	meta.unit	voprov:ParameterDescription.unit	char
pd_ucd	meta.ucd	voprov:ParameterDescription.ucd	char
pd_utype	meta	voprov:ParameterDescription.utype	char
pd_min	stat.min	voprov:ParameterDescription.min	param dependent
pd_max	stat.max	voprov:ParameterDescription.max	param dependent
pd_options	meta	voprov:ParameterDescription.options	param dependent

Table 28: Column description for parameterDescription table

Name	ucd	utype	datatype
→ u_entity_id	meta.id	voprov:Used.entity	char
→ u_activity_id	meta.id	voprov:Used.activity	char
→ u_usedDescription_id	meta.id	voprov:Used.usedDescription	char
u_role	meta.code.class	voprov:Used.role	char OPTION
u_time	time.start	voprov:Used.time	char

Table 29: Column description for used relationship table

Name	ucd	utype	datatype
→ ud_entityDescription_id	meta.id	voprov:UsedDescription. entityDescription	char
→ ud_activityDescription_id	meta.id	voprov:UsedDescription. activityDescription	char
ud_role	meta.code.class	voprov:UsedDescription.role	char OPTION

Table 30: Column description for usedDescription relationship table. The value of a role string belongs to a set of possible literal values given for instance using the VOTable OPTION element.

Name	ucd	utype	datatype
→ wgb_entity_id	meta.id	voprov:WasGeneratedBy.entity	char
→ wgb_activity_id	meta.id	voprov:WasGeneratedBy.activity	char
wgb_role	meta.code.class	voprov:WasGeneratedBy.role	char OPTION
wgb_time	time.stop	voprov:WasGeneratedBy.time	char

Table 31: Column description for wasGeneratedBy relationship table

Name	ucd	utype	datatype
→ wgbd__entityDescription__id	meta.id	voprov:WasGeneratedByDescription. entityDescription	char
→ wgbd__activityDescription__id	meta.id	voprov:WasGeneratedByDescription. activityDescription	char
wgbd__role	meta.code.class	voprov:WasGeneratedByDescription.role	char OPTION

Table 32: Column description for wasGeneratedByDescription relationship table

Name	ucd	utype	datatype
→ waw__agent__id	meta.id	voprov:WasAssociatedWith.Agent	char
→ waw__activity__id	meta.id	voprov:WasAssoatciatedWith.Activity	char
waw__role	meta.code.class	voprov:WasAssociatedWith.agentRole	char OPTION

Table 33: Column description for wasAssociatedWith relationship table

Name	ucd	utype	datatype
→ wat__entity__id	meta.id	voprov:WasAttributedTo.Entity	char
→ wat__agent__id	meta.id	voprov:WasAttributedTo.Agent	char
wat__role	meta.code.class	voprov:WasAttributedTo.agentRole	char OPTION

Table 34: Column description for wasAttributedTo relationship table

Appendix C Changes from Previous Versions

C.1 Changes from WD-ProvenanceDM-1.0-20161121

- New appendix for PROV-VOTable/TAP SCHEMA tables added
- Corrected and extended attribute tables and mapping tables for links with DatasetDM and SimDM.
- Restructured Accessing provenance section by splitting it in two: Section 4 for explaining the different serialization formats and differences to W3C serializations, Section 5 for describing the access protocols ProvDAL and ProvTAP.
- Removed discussion section, since now all the topics are addressed in the main text.
- Added paragraph on how to use the model in Section 6.
- Shortened serialization examples, partially moved them to appendix.
- Added paragraph on VOSI interface.
- Added a proposed serialization of description classes.
- Modified text on the content of EntityDescription, now seen as Entity attributes known before the Entity instance exists.
- Renamed Section 6 to stress that it explains applications of the model (use cases); implementation details and code examples can be found in Implementation Note (Riebe and Servillat et al., 2017).
- Complete rewrite of the ProvDAL section in Section 5.1; new parameters, new figure and examples added.
- Added additional figure for entity-activity relations.
- Moved the figure showing relations between Provenance.Agent and Dataset.Party into Section 3.
- Extended the entity role examples in table 11.
- Added links to provn and votable-serialization for HiPS-use case, added first part of provn as example in the HiPS-use case section.
- More explanations on links to data models in Section 3, introduced subsections, added table with SimDM-mapping.
- Moved detailed implementation section from appendix to a separate document (implementation note), shortened the use cases & implementation section.

- Attribute/class updates:
  - Added attribute *votype* to *Activity*, can be used for ActivityFlows
  - Added attribute *time* to *Used* and *WasGeneratedBy*
  - Added optional attributes *Entity.creationTime* and *EntityDescription.category*
  - Added optional attributes *Parameter.min*, *Parameter.max*, *Parameter.options*
  - Removed the obscore/dataset attributes from EntityDescription, since they are specific for observations only and are not applicable to configuration entities etc.
  - Use voprov:type and voprov:role in Table 16 with example agent roles, i.e. replaced prov:person by Individual and prov:organization by Organization.
  - Renamed *label* attribute to *name* everywhere, for more consistency with SimDM naming scheme (*label* is reserved there for SKOS labels).
  - Renamed attribute *Entity.access* to *Entity.rights* for more consistency with DatasetDM etc.
  - Avoid double-meaning of *description* (as reference and free-text description) by renaming the free-text description to *annotation*. Mark description-references with arrows in attribute tables.
  - Applied similar naming scheme to *Parameter* and *ParameterDescription*-classes
  - Renamed *docuLink* to *doculink*
  - Corrected attribute names in Table 17.

List of Figures

List of Tables

Bibliography

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

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

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

Demleitner, M., Harrison, P., Molinaro, M., Greene, G., Dower, T. and Perdikeas, M. (2013), 'IVOA registry relational schema', IVOA Working Draft.
http://www.ivoa.net/documents/RegTAP/

Dowler, P., Bonnarel, F., Michel, L. and Demleitner, M. (2015), 'IVOA datalink', IVOA Recommendation 17 June 2015.
http://www.ivoa.net/documents/DataLink/

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2013), 'Data access layer interface, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/DALI/20131129/

Dowler, P., Rixon, G. and Tody, D. (2010), 'Table access protocol version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/TAP

Fernique, P., Allen, M., Boch, T., Donaldson, T., Durand, D., Ebisawa, K., Michel, L., Salgado, J. and Stoehr, F. (2015), 'Hips - hierarchical

progressive survey version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/HiPS/20170519/

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.
http://www.ivoa.net/documents/latest/RM.html

Hua, H., Tilmes, C., Zednik, S. and Moreau, L. (2013), Prov-xml: The prov xml schema, Project report.
https://eprints.soton.ac.uk/356855/

Huynh, T. D., Jewell, M., Sezavar Keshavarz, A., T. Michaelides, D., Yang, H. and Moreau, L. (2013), 'The prov-json serialization'.
https://www.w3.org/Submission/2013/SUBM-prov-json-20130424/

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

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.
http://adsabs.harvard.edu/abs/2012ivoa.spec.0503L

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.
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.
http://openprovenance.org/;http://eprints.soton.ac.uk/271449/

Moreau, L. and Missier, P. (2013), 'PROV-N: The provenance notation', W3C Recommendation.
http://www.w3.org/TR/prov-n/

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

Riebe, K., Servillat, M., Bonnarel, F., Louys, M., Sanguillon, M. and the IVOA Data Model Working Group (2017), 'Provenance implementation note', IVOA Note.
http://volute.g-vo.org/svn/trunk/projects/dm/provenance/implementation-note/