IVOA Provenance Data Model Version 1.0

IVOA Proposed Recommendation 2018-10-11

Working group
DM
This version
http://www.ivoa.net/documents/ProvenanceDM/20181011
Latest version
http://www.ivoa.net/documents/ProvenanceDM
Previous versions
WD-ProvenanceDM-1.0-20180530.pdf
WD-ProvenanceDM-1.0-20170921.pdf
WD-ProvenanceDM-1.0-20161121.pdf
ProvDM-0.2-20160428.pdf
ProvDM-0.1-20141008.pdf
Author(s)
Mathieu Servillat, Kristin Riebe, François Bonnarel, Anastasia Galkin, Mireille Louys, Markus Nullmeier, Michèle Sanguillon, Ole Streicher, and the IVOA Data Model Working Group
Editor(s)
Mathieu Servillat, Kristin Riebe

Abstract

This document describes how provenance information 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), a document (e.g. an article, a technical note) or a device (e.g. a camera, a telescope), learn about the people and organizations involved in a project and assess the quality as well as the usefulness of the dataset, document or device for her own scientific work.

Status of this document

This is an IVOA Proposed Recommendation made available for public review. It is appropriate to reference this document only as a recommended standard that is under review and which may be changed before it is accepted as a full Recommendation.

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

Contents

Acknowledgments

The Provenance Working Group acknowledges support from the Astronomy ESFRI and Research Infrastructure Cluster – ASTERICS project[2],

[2]http://www.asterics2020.eu/

funded by the European Commission under the Horizon 2020 Programme (GA 653477). This document has been developed in part with support from the German Astrophysical Virtual Observatory, funded by BMBF Bewilligungsnummer 05A14BAD and 05A08VHA. Additional funding was provided by the INSU (Action Spécifique Observatoire Virtuel, ASOV), the Action Fédératrice CTA at the Observatoire de Paris and the Paris Astronomical Data Centre (PADC).

Thanks for fruitful discussions to: Catherine Boisson, Karl Kosak, Johann Bregeon and Jose Enrique Ruiz 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, Florian Rothmaier, Jochen Klar and Adrian Partl for fruitful discussions, remarks and comments during the different stages of this specification.

We thank the participants of the Provenance Week 2018 in London, in particular Michael Johnson for joining one of our meeting.

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. The activities correspond to processes like an observation, a simulation, processing steps (image stacking, object extraction, etc.), execution of data analysis code, publication, etc. The people involved can be for example individual persons (observer, publisher, etc.), groups or organisations. An example for activities, entities and agents as they can be discovered backwards in time is given in Figure 1.

The links between the Provenance DM and other IVOA data models will be discussed in Section B. We note here that the provenance of simulated data is included in the Simulation DM (SimDM, Lemson and Wozniak et al., 2012). Therefore we also give a mapping between SimDM and the Provenance DM in Section B.

We also introduce a few constructs from the ProvONE data model (Cuevas-Vicenttín and Ludäscher et al., 2016), that proposes extensions to W3C PROV for scientific workflow provenance. We discuss the mapping between ProvONE and the Provenance DM in Appendix B.

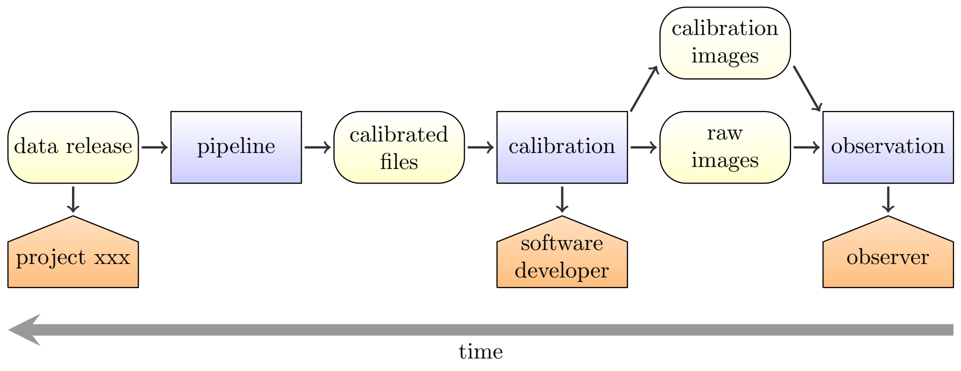

Figure 1: An example graph of provenance discovery. Starting with a released dataset (left), the involved activities (blue boxes), progenitor entities (yellow rounded boxes) and responsible agents (orange pentagons) are discovered.

1.1 Goal of the provenance model

The goal of this Provenance DM is to describe how provenance information can be modelled, stored and exchanged. Its scope is mainly modelling of the flow of data, of the relations between data, and of processing steps. The currently discussed Provenance DM is sufficiently abstract that its core pattern (see Section 2.1) 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.

Characteristics of observation activities such as ambient conditions and instrument characteristics provide useful information to assess the quality and reliability of the generated entities. Experimental configuration or contextual information during the execution of processing activities (computer structure, nodes, operating system used, etc) can also be relevant for the description of the main entities generated. This complementary information should be included in the form of metadata or additional entities and connected to an activity (see Section 2.2). However, the precise structure and modelling of those characteristics is out of the scope of this document.

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 DM 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 the Implementation Note (Servillat and Riebe et al., 2017).

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 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, when relevant, to IVOA semantics, data models and formats (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 be linked to the activity that generated it (if the activity is recorded).
- Activities must be linked to input entities (if applicable).
- Activities may point to output entities.
- Provenance information should make it possible to derive the chronological sequence of activities.
- Entities, Activities and Agents must be uniquely identifiable within a domain and should have persistent identifiers.
- Released entities should have a main contact.
- All activities and entities should have contact information and contain a (short) description or link to a description.

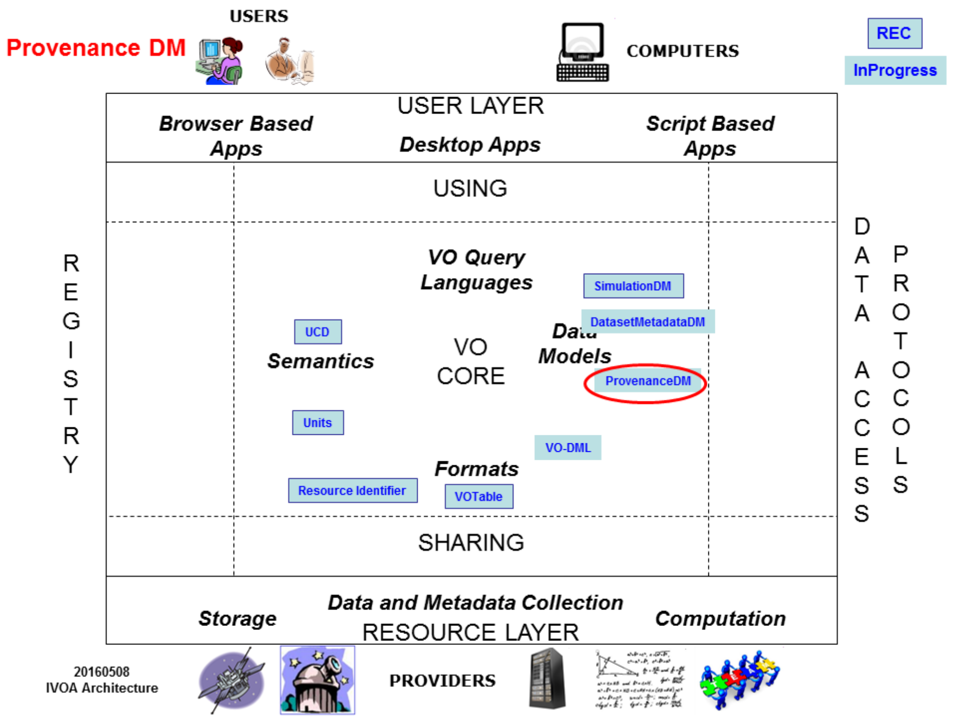

Figure 2: Architecture diagram for the Provenance Data Model. It is based on existing concepts defined in existing IVOA data models, and existing formats and semantics and fully integrated in the IVOA framework

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, 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 IVOA Provenance data model

In this section, we describe the proposed Provenance DM. We first explain the core elements with details for each class and relation (Section 2.1). We then define the extended model, introducing and defining specialized concepts and relation that brings useful provenance information for astronomy (Section 2.2).

2.1 Core Model

2.1.1 Class diagram

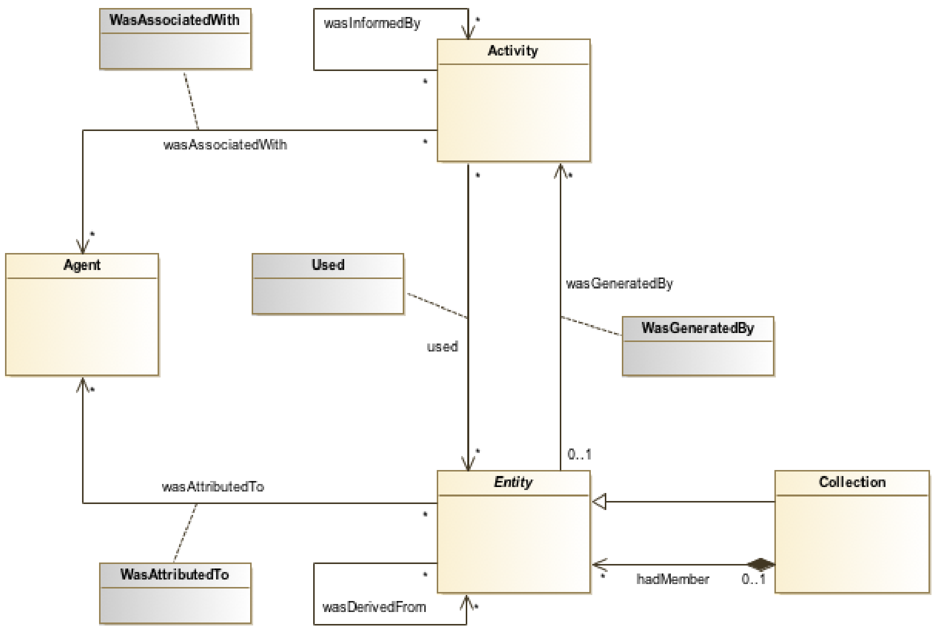

Figure 3: The main core classes and relations of the Provenance DM, which are taken from the W3C PROV-DM. The different relations can be further described with associated classes.

The core elements of the Provenance DM are *Entity*, *Activity* and *Agent*. For these elements, we chose the same names that were used in the PROV recommendation of the World Wide Web Consortium (W3C PROV, Belhajjame and B'Far et al. 2013), which defines a very abstract pattern that can be reused here. The provenance information can be discovered via the relations between those core elements, also taken from the W3C PROV-DM and the related ontology PROV-O Belhajjame and Cheney et al. 2013).

We present the core classes, along with their relations to each other, in Figure 3 and give short descriptions with some examples:

- *Entity:* a physical, digital, conceptual, or other kind of thing with some fixed aspects. For example: data products such as images, catalogs, parameter files, calibration data, instrument characteristics, articles, web pages.

- *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. For example: data acquisition like observation, simulation; regridding, fusion, calibration steps, reconstruction; edition, publication, release of data.

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

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 PROV-DM names:

- *wasGeneratedBy:* a new entity is generated by an activity
(entity "image.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.fits" wasAttributedTo "observatory")

The W3C PROV-DM and its related ontology PROV-O (Belhajjame and Cheney et al., 2013) contain several components that are not described in this document. The IVOA Provenance DM having the same core concepts, it is possible to extend it with any W3C PROV component or extension. In addition, the ProvONE proposed extension to W3C PROV can be connected to this model in the context of scientific workflows.

Entity

Attribute	**W3C PROV**	**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)
location	prov:location	string	a path or a geographical location, e.g. a URL
type	prov:type	string	a provenance type, i.e. one of: prov:collection, prov:bundle, prov:plan; or any of the specialized entities defined in Section 2.2.2
annotation	prov:description	string	text describing the entity in more detail
value	prov:value		provides a value that is a direct representation of an entity
creationTime	prov:generatedAtTime	datetime	date and time at which the entity was created (e.g. timestamp of a file)
destructionTime	prov:invalidatedAtTime	datetime	date and time at which the entity was erased or invalidated
rights	–	string	access rights for the entity, values: public, secure or proprietary; see Curation.Rights, RightsType in DatasetDM

Table 1: Attributes of the *Entity* class. Attributes in **bold** must not be null. Further project-specific attributes (e.g. size, local path, url, ...) could be added when relevant for the project (see also Section B.1).

2.1.2 Entity

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, observation proposals, scientific articles, or manuals and other documents.

An entity is not restricted to being a file. It can even be just a number in a table, depending on how fine-grained the provenance shall be described. An entity can also carry its value directly in its `value` attribute.

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 (Cresitello-Dittmar and Bonnarel et al., 2015) specifies that 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* class, see the mappings of Table 18 in Section B.

Entities have the attributes given in Table 1. If an attribute also exists in the W3C PROV-DM, we list its name in the second column.

The difference between entities that are used as inputs and those used as outputs becomes clear by specifying the relations between the entities and the activities producing or using these entities. More details on this will follow in Section 2.1.5.

WasDerivedFrom. We include the *wasDerivedFrom* relation for those cases where an explicit link between an entity and its progenitor(s) is useful. This relation infers the existence of an activity, but this activity may not be explicitly defined. Sub-classes of this relation in W3C PROV include the concepts of revisions and specializations (Belhajjame and B'Far et al., 2013).

Note that the *WasDerivedFrom* relation cannot always automatically be inferred from following existing *WasGeneratedBy* and *Used* relations alone. If there is more than one input and more than one output to an activity, it is not clear 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.

WasDerivedFrom

Attribute	**Data type**	**Description**
id	string	an identifier for this relation
→ **generatedEntity**	link	link to the *Entity*
→ **usedEntity**	link	link to the progenitor *Entity*, from which the generatedEntity was derived

Table 2: Attributes of the *WasDerivedFrom* relation class. These are the same as those used in W3C's PROV-DM. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null. The W3C model contains additional optional links to the related *Activity*, *WasGeneratedBy* and *Used* relations, which we do not include here for simplicity.

2.1.3 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. It is also included in the W3C PROV-DM.

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

The *Entity-Collection* relation can be modelled 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.

2.1.4 Activity

Activity

Attribute	**W3C PROV**	**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
status		string	can be used to describe the terminal status of the activity (e.g. completed, aborted, error...)

Table 3: Attributes of the *Activity* class. Attributes in **bold** must not be null.

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

An Activity can be seen as the node that helps to discover the progenitors of a dataset and any additional information on its generation.

WasInformedBy. Two activities can be chained together directly, without mentioning the intermediate entity exchanged, using the *WasInformedBy* relation. This relation can be used as a shortcut if the skipped datasets are deemed to be not important enough to be recorded in a pipeline. It can also be used to state that an activity communicates with another through an output-input relation, and thereby triggers its execution.

WasInformedBy

Attribute	Data type	Description
id	string	an identifier for this relation
→ **informed**	link	link to the *Activity* being informed by another (“second” activity)
→ **informant**	link	link to the informing *Activity* (“first” activity)

Table 4: Attributes of the *WasInformedBy* relation class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

2.1.5 Entity-Activity relations

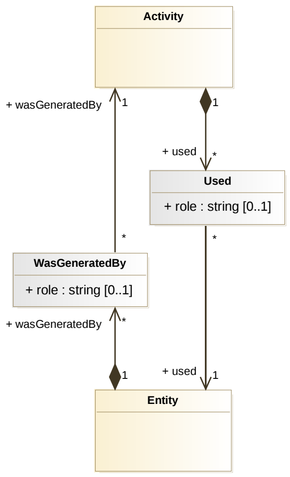

Figure 4: Entity and *Activity* are linked via the *Used* and *WasGeneratedBy* relation classes. The `role` that an entity played when being used or generated by an activity is recorded within the *Used* and *WasGeneratedBy* classes.

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

We use two relations, *Used* and *WasGeneratedBy* (see Tables 5 and 6), instead of just one mapping class with a flag for input/output, because their descriptions and `role` attributes can be different. The `role` should always be provided if it is not obvious.

Used

Attribute	W3C PROV	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*

Table 5: Attributes and references of the *Used* relation class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

WasGeneratedBy

Attribute	W3C PROV	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*

Table 6: Attributes and references of the *WasGeneratedBy* relation class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

The *Used* and *WasGeneratedBy* relations can have the attribute `time`.

This is the time when usage of an entity started or the generation of an entity is finished, which may not be the activity start or end time.

The *Used* relation is closely coupled to the *Activity*, so we use a composition here, indicated in Figure 4 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. However, an entity can be generated by only one activity, so the multiplicity is 1 or 0 between *Entity* and *WasGeneratedBy*.

Entity roles Each activity generally requires specific roles for each input or output entity. For example, an activity for dark-frame 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 cannot be an attribute for *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.

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. 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.

2.1.6 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, a project 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 8 (also see Section B for comparison between *Agent* and *Party*). Both agent types are also used in the W3C PROV-DM, though *Individual* is called *Person* there.

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 organization is [a]

Agent

Attribute	W3C PROV	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 as given in Table 8
email		string	Contact email of the agent
affiliation		string	Affiliation of the agent
address		string	Address of the agent
phone		string	Phone number

Table 7: Attributes of the *Agent* class. Attributes in **bold** must not be null.

AgentType

Class or type	W3C PROV	DatasetDM	Comment
Agent	prov:Agent	Party	
Individual	prov:Person	Individual	a person, specified by name, email, address, (though all these parts may change in time)
Organization	prov:Organization	Organization	a publishing house, institute or scientific project
SoftwareAgent	prov:SoftwareAgent		A software agent is running software, e.g. a cron job or a trigger

Table 8: Agent class and types of subclasses in this data model, compared to W3C PROV-DM and Dataset DM.

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 organizations on a finer level: "At a high level, an organization could be a university, observatory, or government agency. At a finer level, it could be a specific scientific project, space mission, or individual researcher. A provider is an organization that makes data and/or services available to users over the network."

For each agent a `name` must be specified. A summary of the attributes for *Agent* is given in Table 7. We added the optional attributes `address` and `email`, since they appeared in our use cases and are commonly used. Not every project will need them; e.g. an advanced system may use permanent identifiers (e.g. ORCIDs, identities in federations, etc) 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.

Association and Attribution. 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 fulfil 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*, that relates an *Activity* to an agent, and *wasAttributedTo*, that relates an *Entity* to an agent.

We adopted here the same naming scheme that was used in the W3C PROV-DM. 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, ...).

WasAssociatedWith

Attribute	W3C PROV	Data type	Description
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the agent, see e.g. Table 11
→ **agent**	prov:agent	link	link to an *Agent*
→ **activity**	prov:activity	link	link to an *Activity*

Table 9: Attributes and references of *WasAssociatedWith* relation class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

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 11. For comparison, SimDM, the IVOA Simulation

WasAttributedTo

Attribute	W3C PROV	Data type	Description
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the agent, see e.g. Table 11
→ **agent**	prov:agent	link	link to an *Agent*
→ **entity**	prov:entity	link	link to an *Entity*

Table 10: Attributes and references of *WasAttributedTo* relation class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

DM (Lemson and Wozniak et al., 2012) 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 the appendix, Section B.

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.

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

2.2 Extended Model

2.2.1 Class diagram and VO-DML compatibility

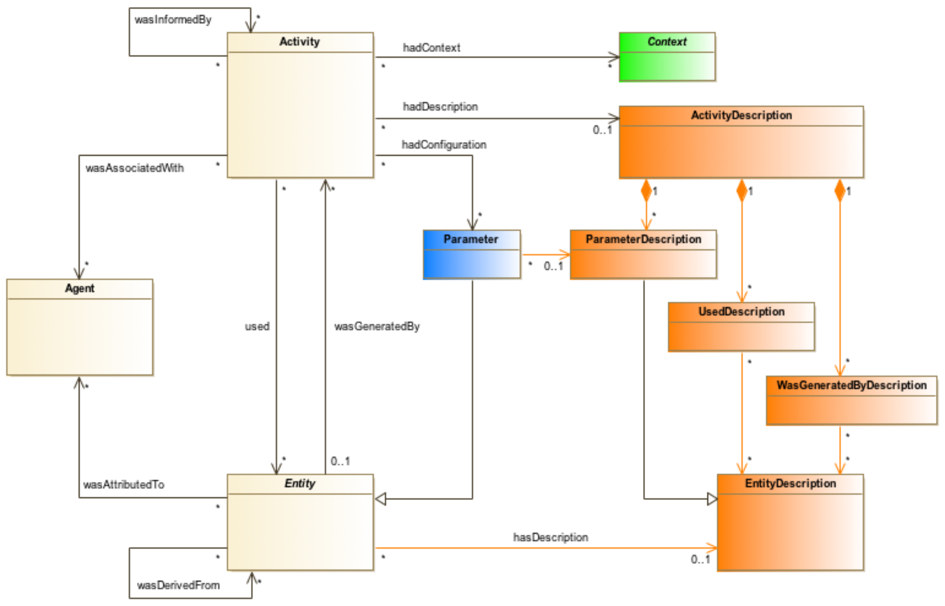

Figure 5: Class diagram showing the main functional features of the Provenance DM. This diagram includes a subset of all the specialized entities and relations defined in this section (2.2) which are further presented in diagrams 7 and 2.2.3. The various *hasDescription* relations are shown with red arrows.

In the domain of astronomy, certain processes and steps are repeated over and over again, maybe using a different configuration and within a different context. We therefore separate the descriptions of activities from the actual processes and introduce an additional *ActivityDescription* class. Likewise, we also apply the same pattern for *Entity* and add an *EntityDescription* class. Defining such descriptions allows them to be reused, which is less redundant when performing a series of tasks of the same type. A similar normalization of descriptions of the actual processes and datasets can be found in the IVOA Simulation DM (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*.

Figure 5 shows the global class diagram. In addition to the core model (Section 2.1), we define specialized entities (Section 2.2.2) and relations (Section 2.2.3) that were identified as useful in one or several use cases. For some

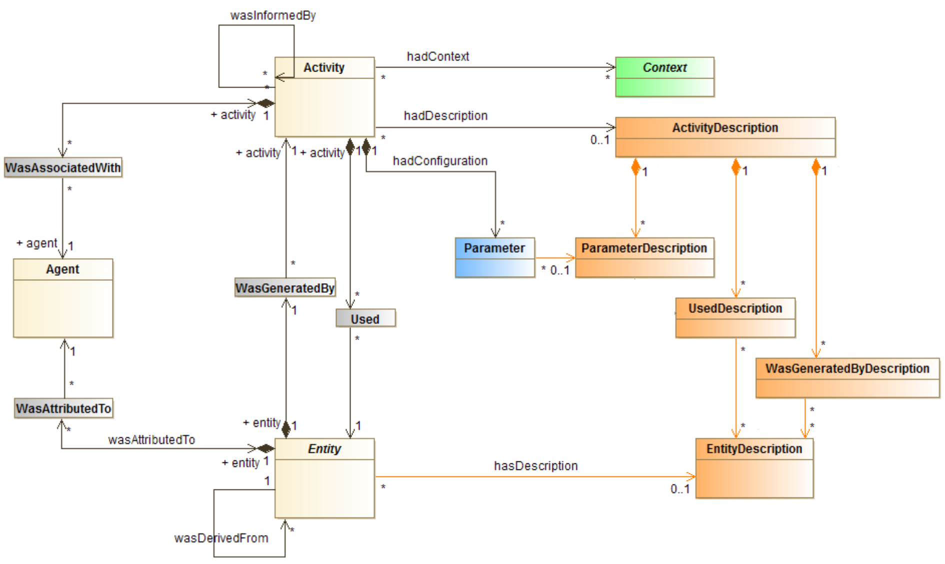

Figure 6: VO-DML compatible version of the class diagram in Figure 5.

of the concepts that are well known in the IVOA ecosystem, we propose a detailed structure in order to facilitate the access to such resources and foster interoperability (Sections 2.2.4, 2.2.5 and 2.2.6).

Figure 6 shows a version of the UML diagram applying the VO-DML designing rules and reusing VO-DML IVOA datatypes package.

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.

2.2.2 Specialized Entities

The abstraction level of the W3C PROV-DM being high, one of the objective of this IVOA recommendation is to guide the usage of this model in the astronomy context by providing specialized entities that are connected to concepts known in astronomy and relevant to assess the quality and reliability of the exchanged entities.

We first remind the W3C PROV definition of an *Entity*: “An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.”

We already expressed the necessity of adding **Descriptions** to the concepts of Activity and Entity, in order to avoid redundancy and give detail explanations on the method or algorithms that compose the core of an activity. In addition, from our use cases in Astronomy, activities require specific metadata that is related to the **Configuration** of an activity, or to the **Context**. We therefore define specialized entities hierarchically following those main categories. The structuring links of specialized entities is presented in Figure 7. The list given below is not intended to be exhaustive, additional project-dependant entities may be defined when relevant.

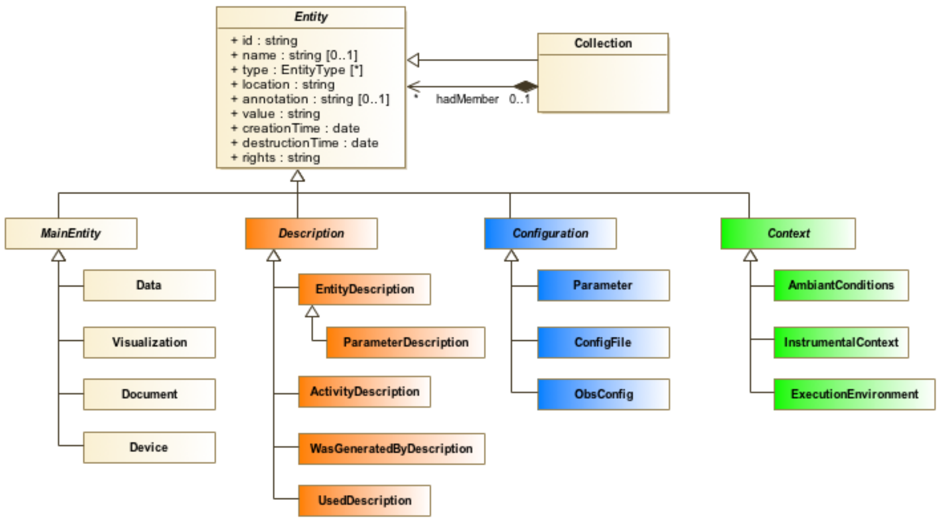

Figure 7: Class diagram showing the structuring links for specialized entities

It is important to note that specialized entities inherit from *Entity*, all those classes thus share the identifier attribute `id`, as well as the relations of the core model. The `id` attribute must be unique for each different entity so that it can be connected to other concepts using core or specialized relations (see Section 2.2.3). It is possible to attach an Agent to any of those subclasses of *Entity* in order to provide specific contacts for the content of a specialized entity.

MainEntity: Provenance information are expected to be recorded primarily for those main entities, to which we may attach descriptions, detail the configuration that led to their generation and the context in which they were generated.

Data : digital, machine-readable information in some content that will be used/transformed/analysed. It could be a cell or a column in a table, a file, an image, a cube of data... This *Data Entity* might be bounded to a IVOA *Dataset* record or an IVOA ObsCore record (see also Section B.1 in the Appendix).

Visualization : a digital visual representation.

Document : information presented in a human readable form, e.g. a book, an article, or an observation proposal.

Device : a physical object, such as a tool, an instrument, a telescope, etc.

We note that *Data*, *Visualization* and *Document* are classes that are also defined in the ProvONE ontology (Cuevas-Vicenttín and Ludäscher et al., 2016), with the following text: ”The *Data* class is defined to be generic and represents data items of various types (e.g. XML, JSON, CSV files, etc.). *Visualizations* are a differentiated class intended to represent various visualization items often output from workflows (JPG, PNG, SVG, MP4, etc.). The *Document* class is a generic representation of a published or unpublished article or report.“

Description: This subclass of entities carries detailed information on the expected behaviour of an *Activity* and on the expected structure and use of an *Entity*. It describes in a structured way the plan prepared for and followed during an activity, and as such, influences directly the activity.

EntityDescription : describes a category of entities, and contains descriptive information about an entity that is known before an entity instance is created (file format, MIME or content type, etc), for example: all files that follow the FITS-LDAC structure and format in a project. This class is further described in Sections 2.2.4 and 2.2.6.

ParameterDescription : contains the attributes of a *Parameter* that are similar to the attributes of the PARAM block in a VOTable (unit, UCD, UType...). This class is further described in Section 2.2.5.

ActivityDescription : explanations on the activity, such as the method used, the algorithm, the source code. This class is further described in Section 2.2.6.

UsedDescription : describes the roles of expected inputs in an Activity, e.g. a red channel in the creation of an RGB image.

WasGeneratedByDescription : describes the roles of expected outputs in an Activity, e.g. a master bias in the stacking of a set of bias images.

Configuration: This subclass of entities corresponds to the information passed to an activity in order to configure its execution, and it thus directly influences the development of an activity.

Parameter : configuration of the activity before execution as a key=value input parameter in the IVOA framework, e.g. the number of bins desired in the sampling of a signal. This class is further described in Section 2.2.5.

ConfigFile : file containing configuration information for the activity, e.g. a list of parameters (key=value) for a script, generally in a given format (txt, json, yaml, xml...)

ObsConfig : structured list of parameters that gives the configuration of an observation. For example this can point to specific observing modes of an instrument, the filters used, the trigger mode, etc.

Context: This subclass of entities gives more information on the context that influences the development of an activity, but for which there are no or little control at the moment of its execution:

AmbientConditions : common, prevailing, and uncontrolled atmospheric and weather conditions in a room or place that influence the activity.

InstrumentalContext : precisions on how the instrument is structured and how the hardware is set up during an activity.

ExecutionEnvironment : describes a particular execution platform, such as an operating system or a database management system. Execution environments are used to describe the context in which the execution of an activity takes place. Execution environments could also describe the computing hardware of a system.

2.2.3 Specialized Relations

In order to distinguish the different specialized entities in their usage or influence, we define the corresponding specialized relations between *Activity* and *Entity*: *hadDescription*, *hadConfiguration* and *hadContext*, that should be connected to the entity categories *Description*, *Configuration* and *Context*, respectively. An *Activity* must have one or zero *Description* element of type *ActivityDescription*. In addition, we introduce the *hasDescription* relation that connects an *Entity* class to a *Description* class (hence between two *Entity* classes). Those relations are illustrated in Figure 8.

We thus provide a way to qualify the usage or influence of a specialized entity on an activity, or another entity. The `role` attribute in those relations is not expected to be defined as they are already qualified and should point to specialized entities. Those relations can thus be seen as simple association tables.

2.2.4 EntityDescription

The category of entities can be predefined using a description class *EntityDescription*. This class is meant to store descriptive information about an entity that is known before an *Entity* instance is created. For example,

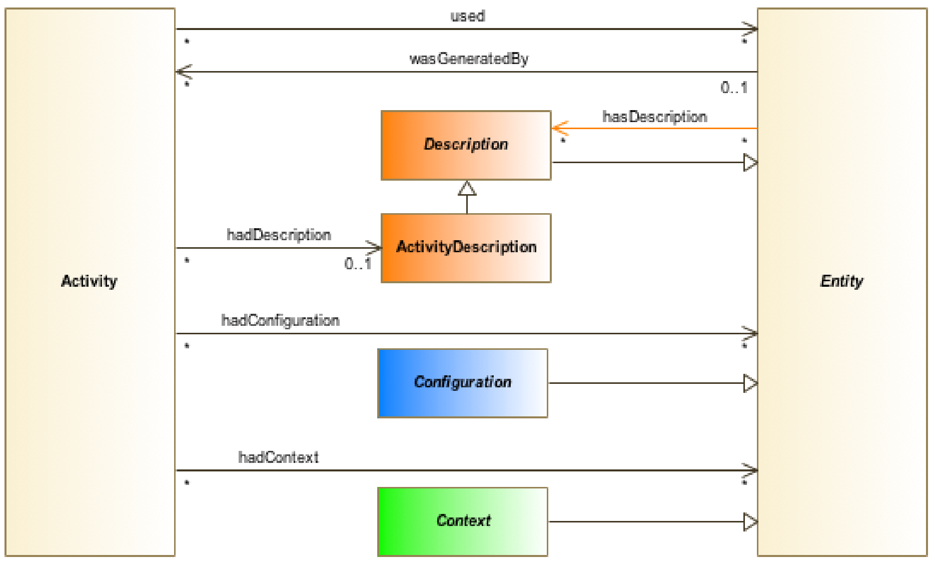

Figure 8: Class diagram showing the relations between specialized entities and *Activity* or *Entity*

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
doculink	url	link to more documentation
Optional attributes:		
content_type	string	MIME type for the content of the entity
format	string	type of container for the entity

Table 12: Attributes of the *EntityDescription* class. Attributes in **bold** must not be null.

a `format` (e.g. JPG images, FITS, FITS-LDAC, . . .) can be common to several entities. However, the size of this image or file cannot be known before it is created. In this example, `format` would be an *EntityDescription* attribute, while size would be a property attached to the *Entity* instance. The *EntityDescription* general attributes are summarized in Table 12. Additional attributes that describe the content of the data could be derived from the Dataset Metadata Model (see Section B.1)

The *EntityDescription* class should *not* contain 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 should be provided by the relations (and their relation descriptions) between activities and entities (see Section 2.1.5).

2.2.5 Parameter and ParameterDescription

The concept of activity configuration, generally a set of parameters that can be configured before the execution of an activity, is tightly linked to the concept of provenance. Configuration information can indeed be relevant to assess the quality and reliability of an activity or an entity.

Activity configuration typically implies a set of input parameters attached to the activity before execution. The concept of parameter is very general, and already exists in the VO context, for example as PARAM elements in VOTable (Ochsenbein and Williams et al., 2013), as uws:param elements in the UWS pattern (Harrison and Rixon, 2010), or as POST/GET parameters for web services (see for example Bonnarel and Demleitner et al., 2017, for SODA services). A DataLink Service Descriptor (Dowler and Bonnarel et al., 2015) contains a group of InputParams with PARAM elements to define those input parameters. The Simulation DM also contains a *ParameterSetting* class (Lemson and Wozniak et al., 2012).

In order to provide a link between configuration and provenance information, we add a *Parameter* class along with a *ParameterDescription*, so that configuration information is structured and stored with the provenance information (and can thus be queried simultaneously). A *Parameter* is an *Entity* with a `value` attribute that must be set. The *ParameterDescription* class can be used to describe this `value` attribute The optional attributes

Parameter

Attribute	Data type	Description
id	string	parameter unique identifier
value	(value dependent)	the value of the parameter, type depends on `ParameterDescription.datatype` and `xtype`; follows same rules as VOTable TABLEDATA and DALI
→ description	link	link to *ParameterDescription*

Table 13: Attributes of the *Parameter* class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

ParameterDescription

Attribute	Data type	Description
id	string	parameter unique identifier
name	string	parameter name
annotation	string	additional free text description
datatype	string	datatype as in VOTable 1.2 and above
arraysize	number	number of values of specified `datatype`, if there is more than one
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
xtype	string	extended datatype as in VOTable 1.2 and above. A list of proposed
Optional attributes:		
min	number	minimum value
max	number	maximum value
options	list	list of accepted values
value	(value dependent)	the default value of the parameter, type depends on `datatype` and `xtype`; follows same rules as VOTable TABLEDATA and DALI
→ activity-Description	link	link to *ActivityDescription*

Table 14: Attributes of the *ParameterDescription* class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

of *ParameterDescription* are taken from the FIELD and PARAM elements in VOTable (Ochsenbein and Williams et al., 2013).

We note that a parameter can be replaced by an existing entity, or can be derived from another entity (the content or part of the content of an entity), thus having itself an origin, with provenance information.

For example, in the case of a processing activity that cleans an image with a sigma-clipping method, the input and output images would be the main entities and the value of the number of sigma for sigma-clipping would be carried by a *Parameter* entity set before running the activity. The corresponding *ParameterDescription* defines the type and range of the expected value for this parameter (using the attributes `datatype`, `min`, `max`, `options`...).

Parameter value vs. ParameterDescription value. The *ParameterDescription* class being a child of the *Entity* class, it contains a `value` attribute. As this class contains information known before a *Parameter* instance is created (as for EntityDescription, see Section 2.2.4), its value can be considered to be the default value for the parameter.

2.2.6 Activity Description classes

ActivityDescription

Attribute	Data type	Description
id	string	a unique id for this activity description
name	string	a human-readable name (to be displayed by clients)
annotation	string	additional free text description for the activity
doculink	url	link to further documentation on this activity, e.g. a paper, the source code in a version control system etc.
Optional attributes:		
activity_type	string	type of the activity, from a vocabulary or list, e.g. data acquisition (observation or simulation), reduction, calibration, publication
activity_subtype	string	more specific subtype of 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)

Table 15: Attributes of the *ActivityDescription* class. Attributes in **bold** must not be null.

The inner working of an activity can be explained by a corresponding *ActivityDescription* class. This could be, for instance, the name of the `code` and its `version` used to perform an activity or a more general description of the underlying algorithm or process. An activity is then a concrete case (instance) that follows the described inner working, with a `startTime` and an `endTime`, and it refers to a corresponding description for further information.

A close concept in the W3C PROV-DM is the *Plan* (defined as a subclass of *Entity*). However a plan in PROV-DM is primarily attached to the *Agent* class with *hadPlan* through the *wasAssociatedWith* relation. It is accepted to omit the agent, but it is always supposed that an agent exist. *Activity-Description* is directly attached to *Activity* and can thus be seen as a list of attributes that can be known before an *Activity* instance is created.

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.

Descriptions of the Used and WasGeneratedBy relations. In order to describe more largely an activity, it is common to define the expected inputs and outputs of this activity, i.e. what we expect to store in the Used and WasGeneratedBy relations.

In the case of workflow description models, such as ProvONE (but also Kepler or Taverna for example), input and output **ports** are defined, and can be connected to build a workflow of activities. In ProvONE (Cuevas-Vicenttín and Ludäscher et al., 2016), an *ActivityDescription* is restricted to a *Program*, and an *Activity* is an *Execution* associated to a *Program*, with further entities and relations dedicated to workflow descriptions (see SectionB). However, the description of workflows is out of the scope of this document, and the more general concepts we introduce here are the Used-Description and the WasGeneratedByDescription classes. Those classes are meant to store descriptive information about the usage or generation of an entity that is known before the activity is executed.

In particular, if the `role` attribute is given in those description classes, the corresponding *Used* and *WasGeneratedBy* relations must contain the same `role` value.

A `multiplicity` attribute can indicate that more than one entity may have the same role, e.g. in the case of the stacking of several images, an undefined number of input images is expected and will share the same role.

EntityDescription in the context of an Activity. When related to the *UsedDescription* or *WasGeneratedByDescription*, the attributes of Entity-Description (see Section 2.2.4) help to describe the category of entities expected as an input or an output in an activity. For example: the input bias files must be in FITS format, or the red, green and blue channel images must be in PNG or JPEG format.

UsedDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entity, as what it is used for the linked type of activityDescription
Optional attributes:		
multiplicity	string	Number of expected input entities to be used with the given role. The multiplicity is 1 by default and a * indicates an undefined or unlimited number of input entities.
→ **activityDescription**	link	link to *ActivityDescription*
→ entityDescription	link	link to *EntityDescription*

Table 16: Attributes and references of the *UsedDescription* class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

Input parameters. An activity may expect a list of input parameters, each one directly described with the attributes of *ParameterDescription* as defined in Section 2.2.5. Such a list of input parameters already exist in the IVOA ecosystem for service resources. In this context, they are written in the form of an IVOA DataLink Service Descriptor (Dowler and Bonnarel et al., 2015). We find also a list of parameters in UWS services (Harrison and Rixon, 2010), some of which may carry identifiers referencing another entity. The link to the UWS pattern is further developed in Section B.3 of the Appendix.

WasGeneratedByDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entity, which kind of output it is
Optional attributes:		
multiplicity	string	Number of expected output entities that will be generated with the given role. The multiplicity is 1 by default and a * indicates an undefined or unlimited number of input entities.
→ **activityDescription**	link	link to an *ActivityDescription*
→ entityDescription	link	link to *EntityDescription*

Table 17: Attributes and references of the *WasGeneratedByDescription* class. References in the data model are indicated with an arrow (→). Attributes in **bold** must not be null.

3 Serialization of the provenance data model

3.1 Introduction

Serialization files constitute the building blocks of the client/server 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 Implementation Note (Servillat and Riebe 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 expose only 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 visualisation tools.

Such serializations can be retrieved through IVOA access protocols (see Section 4), or directly integrated in dataset headers or "associated metadata" in order to provide provenance metadata for these datasets.

For FITS files, a provenance extension called "PROVENANCE" could be added which contains provenance information of the activities 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.

3.2 W3C serialization formats: PROV-N, PROV-JSON and PROV-XML

Serialization formats are proposed in the W3C PROV framework for storing and exchanging the provenance metadata: PROV-N, PROV-JSON,PROV-XML and PROV-RDF, that are defined in Moreau and Missier (2013), Huynh and Jewell et al. (2013), Hua and Tilmes et al. (2013),and Belhajjame and Cheney et al. (2013) respectively. They can be reused here as well for serializations of our data model.

In order to produce W3C compatible serializations, the classes and attributes defined in the IVOA Provenance DM model must be qualified names with the namespace `voprov`, except when they exist in the W3C PROV namespace `prov` (e.g. `prov:id`, `prov:type` and `prov:startTime`). A mapping is given in e.g. Tables 1, 2, 3, 4, 5 and 6.

The specialized entities defined in Section 2.2.2 must be written as Entity instances and the attribute `prov:type` must be set to the category of the specialized entity e.g. `voprov:Data`, `voprov:Parameter`, `voprov:ActivityDescription`. This is also the rule for *Collection*, as done in W3C PROV-DM. We note here that several `prov:type` values can be provided.

The specialized relations defined in Section 2.2.3 must be written as a W3C relation (e.g. used relation between an Activity and and Entity, or the more general wasInfluencedBy relation between all classes). The attribute `prov:type` must be set to the name of the specialized relation, e.g. `voprov:hadDescription` or `voprov:hasConfiguration`.

To further export some concepts of the IVOA model to equivalent concepts in the W3C model, the following rules may be respected:

- attribute `voprov:name` → `prov:label`
- attribute `voprov:annotation` → `prov:description`
- *ActivityDescription*: also add `prov:type = 'prov:Plan'`
- *hadDescription*: replace by *prov:wasAssociatedWith* with the activityDescription given as the plan through the *prov:hadPlan* relation

PROV-JSON, PROV-N and PROV-XML can be converted into each other, e.g. using the `prov` or `voprov` python package (see Section "voprov" in Implementation Note (Servillat and Riebe et al., 2017)).

Here is an example of a 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 voprov <http://www.ivoa.net/documents/dm/provdm/voprov/>
  prefix prov <http://www.w3.org/ns/prov#>
  prefix ex <http://www.example.com/provenance/>

  entity(ivo://example#Public_NGC6946, [prov:label="Processed image of
   ↪   NGC 6946", prov:type="voprov:Data"])
  entity(ivo://example#DSS2.143, [prov:label="Unprocessed image of NGC
   ↪   6946", prov:type="voprov:Data"])
  activity(ex:Process1, 2017-04-18T17:28:00, 2017-04-19T17:29:00,
   ↪   [prov:label="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/",
    "prov": "http://www.w3.org/ns/prov#",
    "ex": "http://www.example.com/provenance/"
  },
  "activity": {
    "ex:Process1": {
      "prov:startTime": "2017-04-18T17:28:00",
      "prov:endTime": "2017-04-19T17:29:00",
      "prov:label": "Process 1"
    }
  },
  "wasGeneratedBy": {
    "_:id4": {
      "prov:time": "2017-05-05T00:00:00",
      "prov:entity": "ivo://example#Public_NGC6946",
      "prov:activity": "ex:Process1"
    }
  },
  "used": {
    "_:id1": {
      "prov:entity": "ivo://example#DSS2.143",
      "prov:activity": "hips:AlaRGB1"
    }
  },
  "entity": {
    "ivo://example#DSS2.143": {
      "prov:label": "Unprocessed image of NGC6946",
      "prov:type": "voprov:Data"
    },
    "ivo://example#Public_NGC6946": {
      "prov:label": "Processed image of NGC 6946",
      "prov:type": "voprov:Data"
    }
  }
}
```

3.3 VOTable format for Provenance metadata

To emphasize the compatibility to the IVOA framework, where the XML-based VOTable format is a reference to circulate metadata, we define a 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. 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 is used for ProvTAP

The VOTable serialization can be considered as a flat view on the various tables stored in a database implementing the data model structure explained in Section 2. More examples of serialization documents are provided in Appendix A. It is possible to create separate tables for each specialized entity, where the TABLE tag must have the name attribute set to the name of this specialized entity, and `utype=voprov:Entity`.

A VOTable serialization can be produced using the `voprov` python module, available to the community, as mentioned in see also in Implementation Note (Servillat and Riebe 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="Used" utype="voprov:Used">
      <FIELD arraysize="*" datatype="char" name="u_activity_id"
       ↪  ucd="meta.id" utype="voprov:Used.activity"/>
      <FIELD arraysize="*" datatype="char" name="u_entity_id"
       ↪  ucd="meta.id" utype="voprov:Used.entity"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ex:Process1</TD>
            <TD>ivo://example#DSS2.143</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="WasGeneratedBy" utype="voprov:wasGeneratedBy">
      <FIELD arraysize="*" datatype="char" name="wgb_entity_id"
       ↪  ucd="meta.id" utype="voprov:WasGeneratedBy.entity"/>
      <FIELD arraysize="*" datatype="char" name="wgb_activity_id"
       ↪  ucd="meta.id" utype="voprov:WasGeneratedBy.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="a_id" ucd="meta.id"
      ↪  utype="voprov:Activity.id"/>
      <FIELD arraysize="*" datatype="char" name="a_name"
      ↪  ucd="meta.title" utype="voprov:Activity.name"/>
      <FIELD arraysize="*" datatype="char" name="a_startTime"
      ↪  ucd="time.start" utype="voprov:Activity.startTime"/>
      <FIELD arraysize="*" datatype="char" name="a_endTime"
      ↪  ucd="time.end" 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="e_id" ucd="meta.id"
      ↪  utype="voprov:Entity.id"/>
      <FIELD arraysize="*" datatype="char" name="e_name"
      ↪  ucd="meta.title" utype="voprov:Entity.name"/>
      <FIELD arraysize="*" datatype="char" name="e_type" ucd="meta.main"
      ↪  utype="voprov:Entity.type"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://example#DSS2.143</TD>
            <TD>Unprocessed image of NGC6946</TD>
            <TD>Data</TD>
          </TR>
          <TR>
            <TD>ivo://example#Public_NGC6946</TD>
            <TD>Processed image of NGC 6946</TD>
            <TD>Data</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <INFO name="QUERY_STATUS" value="OK"/>
  </RESOURCE>
```

```
</VOTABLE>
```

3.4 Serialization of Description classes for web services

The serialization of an ActivityDescription, that includes all the description classes presented in Section 2.2.6, 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 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"), which is similar to the group defining input parameters in a DataLink service descriptor,
- the input entities (group name="InputEntities"),
- the output entities (group name="OutputEntities").

The PARAM elements of the resource correspond to the attributes of the *ActivityDescription* class (see Section 2.2.6) and may include a main contact with a name and email ("contact_name" and "contact_email"), that corresponds to an agent associated with the *ActivityDescription*. The `utype` attribute is used to connect the PARAM element to its corresponding element in the Provenance DM. Other optional elements can be added with their corresponding `utype` set, if relevant.

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 2.2.5).

The input group can be used to extend some of the parameters that point in fact to entities (e.g. if the parameter is a file name, a URL, an entity identifier). Otherwise, it can indicate the other entities that may be used internally.

The output group indicates the expected entities that may be generated by the activity.

Each input or output entity is then described within a GROUP block where the name attribute is set to the role or the name of the reference parameter. This GROUP block contains PARAM elements with names corresponding to attributes of *UsedDescription*, *WasGeneratedByDescription* or *EntityDescription*. For example, the following names can be found :

- 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,

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 name="make_RGB_image" type="meta"
   ↪  utype="voprov:ActivityDescription">

    <DESCRIPTION>Create an RGB image from 3 images</DESCRIPTION>
    <PARAM name="type" value="..." datatype="char" arraysize="*"
     ↪  utype="voprov:ActivityDescription.type"/>
    <PARAM name="subtype" value="..." datatype="char" arraysize="*"
     ↪  utype="voprov:ActivityDescription.subtype" />
    <PARAM name="version" value="..." datatype="float"
     ↪  utype="voprov:ActivityDescription.version" />
    <PARAM name="doculink" value="..." arraysize="*" datatype="char"
     ↪  utype="voprov:ActivityDescription.doculink"/>
    <PARAM name="contact_name" value="..." datatype="char" arraysize="*"
     ↪  utype="voprov:Agent.name" />
    <PARAM name="contact_email" value="..." datatype="char"
     ↪  arraysize="*" utype="voprov:Agent.email" />

    <GROUP name="InputParams">
      <PARAM name="R" value="R.jpg" arraysize="*" datatype="char"
       ↪  ucd="meta.id">
        <DESCRIPTION>Name of the image for red channel</DESCRIPTION>
      </PARAM>
      <PARAM name="G" value="G.jpg" arraysize="*" datatype="char"
       ↪  ucd="meta.id">
        <DESCRIPTION>Name of the image for green channel</DESCRIPTION>
      </PARAM>
      <PARAM name="B" value="B.jpg" arraysize="*" datatype="char"
       ↪  ucd="meta.id">
        <DESCRIPTION>Name of the image for blue channel</DESCRIPTION>
      </PARAM>
      <PARAM name="normalize" value="true" datatype="boolean">
        <DESCRIPTION>Apply normalization</DESCRIPTION>
      </PARAM>
      <PARAM name="RGB" value="RGB.jpg" arraysize="*" datatype="char"
       ↪  type="no_query">
        <DESCRIPTION>Name of the generated RGB image</DESCRIPTION>
      </PARAM>
    </GROUP>

    <GROUP name="InputEntities">
      <GROUP name="R" utype="voprov:UsedDescription">
        <DESCRIPTION>Image for red channel</DESCRIPTION>
        <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"
         ↪  ucd="meta.code.mime" />
      </GROUP>
      <GROUP name="G" utype="voprov:UsedDescription">
        <DESCRIPTION>Image for green channel</DESCRIPTION>
        <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"
         ↪  ucd="meta.code.mime" />
      </GROUP>
      <GROUP name="B" utype="voprov:UsedDescription">
        <DESCRIPTION>Image for blue channel</DESCRIPTION>
        <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"
         ↪  ucd="meta.code.mime" />
      </GROUP>
    </GROUP>

    <GROUP name="OutputEntities">
      <GROUP name="RGB" utype="voprov:WasGeneratedByDescription">
        <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>
    </GROUP>

  </RESOURCE>
</VOTABLE>
```

4 Accessing provenance information

We envision two possible access protocols that will be specified in separate documents:

- ProvSAP (for Simple Access Protocol): retrieve provenance information based on given ID of a data entity or activity.
- ProvTAP (for Table Access Protocol): allows detailed queries for provenance information, discovery of datasets based on e.g. code version, parameter values of a specified kind of Activity, etc.

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.

```
document
  prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
  prefix cds <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(cds:AlaRGB1, 2017-04-18T17:28:00, 2017-04-19T17:29:00, [
        voprov:name="Aladin RGB 1",
        voprov:annotation="Aladin RGB image generation for NGC 6946",
        voprov:description="cds:AlaRGB"])

  entity(cds:AlaRGB, [
        prov:type="voprov:ActivityDescription"
        voprov:annotation="Aladin RGB image generation",
        voprov:group="RGBencoding",
        voprov:name="Aladin RGB image generation algorithm",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gml"])

  used(cds:AlaRGB1, 'cds:AlaRGB', - , prov:type="hadDescription")
  used(cds:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143', -)
  used(cds:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143', -)
  used(cds:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143', -)

  wasGeneratedBy('ivo://CDS/P/DSS2color#RGB_NGC6946', cds: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.

```
{
  "prefix": {
    "ivo": "http://www.ivoa.net/documents/rer/ivo/",
    "voprov": "http://www.ivoa.net/documents/dm/provdm/voprov/",
    "cds": "http://cds.u-strasbg.fr/data/"
  },
  "activity": {
    "cds: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:description": "cds:AlaRGB"
    }
  },
  "wasGeneratedBy": {
    "_:id4": {
      "voprov:time": "2017-05-05T00:00:00",
      "voprov:entity": "ivo://CDS/P/DSS2color#RGB_NGC6946",
      "voprov:activity": "cds:AlaRGB1"
    }
  },
  "used": {
    "_:id1": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143",
      "voprov:activity": "cds:AlaRGB1"
    },
    "_:id3": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143",
      "voprov:activity": "cds:AlaRGB1"
    },
    "_:id2": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143",
      "voprov:activity": "cds:AlaRGB1"
    },
    "_:id5": {
    "voprov:entity": "cds:AlaRGB",
    "voprov:activity": "cds:AlaRGB1",
    "prov:type" "voprov:ActivityDescription"
    }
  },
  "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"
    }
     "cds:AlaRGB": {
      "prov:type": "voprov:ActivityDescription",
      "voprov:group": "RGBencoding",
      "voprov:name": "Aladin RGB image generation algorithm",
      "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gml"
    }
  }
}
```

Here is the mapping obtained for the same data description with a serialization in VOTable format.

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2" xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
xmlns:cds="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="results">
    <DESCRIPTION>Provenance VOTable</DESCRIPTION>
    <TABLE name="Used" utype="voprov:Used">
      <FIELD name="u_activity_id" utype="voprov:Used.activity"
       ↪  arraysize="*" ucd="meta.id" datatype="char" />
      <FIELD name="u_activity_id" utype="voprov:Used.entity"
       ↪  arraysize="*" ucd="meta.id" datatype="char" />
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>cds:AlaRGB1</TD>
            <TD>ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="WasGeneratedBy" utype="voprov:WasGeneratedBy">
      <FIELD name="wgb_entity_id" utype="voprov:wasGeneratedBy.entity"
       ↪  arraysize="*" datatype="char"  ucd="meta.id"/>
      <FIELD name="wgb_activity_id"
       ↪  utype="voprov:wasGeneratedBy.activity" arraysize="*"
       ↪  datatype="char" ucd="meta.id"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://CDS/P/DSS2color#RGB_NGC6946</TD>
            <TD>cds:AlaRGB1</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Activity" utype="voprov:Activity">
      <FIELD name="a_id"  utype="voprov:Activity.id" ucd="meta.id"
       ↪  arraysize="*" datatype="char" />
      <FIELD name="a_name" utype="voprov:Activity.name" ucd="meta.title"
       ↪  arraysize="*" datatype="char" />
      <FIELD name="a_starttime" utype="voprov:Activity.startTime"
       ↪  ucd="time.start" arraysize="*" datatype="char" />
      <FIELD name="a_endtime" utype="voprov:Activity.endTime"
       ↪  ucd=""time.end" arraysize="*" datatype="char"  "/>
      <FIELD name="a_annotation" utype="voprov:Activity.annotation"
       ↪  ucd="meta.description" arraysize="*" datatype="char" />
      <FIELD name="a_description" utype="voprov:Activity.description"
       ↪  ucd="meta.id" arraysize="*" datatype="char" />
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>cds: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>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="ActivityDescription" utype="voprov:ActivityDescription">
      <FIELD name="ad_id" utype="voprov:ActivityDescription.id"
       ↪  ucd="meta.id" arraysize="*" datatype="char" />
      <FIELD name="ad_name" utype="voprov:ActivityDescription.name"
       ↪  ucd="meta.title" arraysize="*" datatype="char" />
      <FIELD name="ad_group" utype="voprov:ActivityDescription.group"
       ↪  ucd="meta.code.class" arraysize="*" datatype="char" />
      <FIELD name="ad_doculink"
       ↪  utype="voprov:ActivityDescription.doculink"
       ↪  ucd="meta.ref.url" arraysize="*" datatype="char" />
      <DATA>
        <TABLEDATA>
          <TR>
            <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="e_id" utype="voprov:Entity.id" ucd="meta.id"
       ↪  arraysize="*" datatype="char" />
      <FIELD name="e_name" utype="voprov:Entity.name" ucd="meta.title"
       ↪  arraysize="*" datatype="char" />
      <FIELD name="e_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   Links to other data models

The Provenance Data Model can be applied without making any links to other IVOA data model classes. However, other IVOA standards and external models contain concepts that can be mapped with some Provenance DM concepts. This section provides links between those concepts, and gives some elements to extend the Provenance DM.

For example, when the data is not yet published, provenance information can be stored, 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.

B.1   Links with the IVOA Dataset/ObsCore models

Entity and *EntityDescription* in ProvenanceDM are tightly linked to the *DataSet*-class in DatasetDM/ObsCoreDM (Cresitello-Dittmar and Bonnarel et al., 2015; Louys and Bonnarel et al., 2011), as well as to *InputDataset* and *OutputDataSet* in the SimulationDM (Lemson and Wozniak et al., 2012). Tables 18 and 19 map classes and attributes from DatasetDM to concepts in ProvenanceDM.

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

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 9. The main difference between *Agent* and *Party* is that *Individual* and *Person* are subclasses in DatasetDM, whereas we just use the same class *Agent* for both and dis-

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 19: 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.

tinguish between them using the *Agent.type* attribute (which can have the value "Individual" or "Organization"), which is closer to W3C's provenance data model.

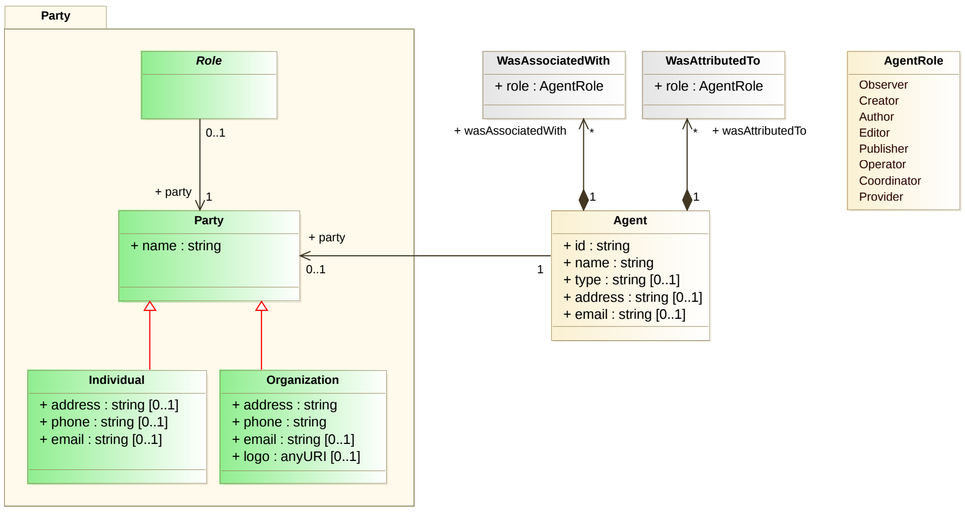

Figure 9: The relations between the *Agent* class within ProvenanceDM (grey and yellow classes) with classes from the DatasetDM, party package (green).

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 serialized 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.

B.2 Links with the IVOA Simulation Data Model

In SimDM (Lemson and Wozniak et al., 2012), 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 20.

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 ProvSAP interface or for storing provenance metadata together with each released simulation dataset (e.g. in a VOTable).

B.3 Links with the IVOA UWS pattern

The IVOA Universal Worker Service Pattern (Harrison and Rixon, 2010) defines how to manage asynchronous execution of jobs on a service. Within this pattern, a *Job* can be seen as an *Activity*. A job uses input parameters and generates results.

Job Description Language. The UWS pattern states that "the rules for setting and arranging the parameters for a job are called the Job-Description Language (JDL). The combination of the UWS pattern, a JDL and details of the job state visible to the client defines a service contract". It is interesting to map this JDL with the Activity Description classes defined in the Provenance DM (section 2.2.6). When using the Activity Description classes as a JDL, a service following the UWS pattern is turned into a "universal worker service". This service is in addition *Provenance enabled*, in that the provenance information can be automatically generated from the JDL. The serialization proposed in section 3.4 completes the definition to make the JDL exchangeable.

Value of UWS parameters. The definition of UWS parameters corresponds to the definition of the *Parameter* class (see Section 2.2.5. The UWS pattern states that "each parameter value may be expressed either

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

directly as the content of the parameter element, or the value expressed *by reference*, where there returned parameter value is a URL that points to the location where the actual parameter value is stored". A parameter value may also be an identifier, a serial number, a file name, that points to an external entity. It is generally not clear if the parameter value points to an input or output entity, or if it is just some value. This is resolved in the UWS pattern (Harrison and Rixon, 2010) by adding an attribute `byReference=true` to a parameter if it is an URL to be resolved. In the same way, this can be resolved in the Provenance DM by adding to a *ParameterDescription* instance a link to a *UsedDescription* instance that explains in details the referenced external entity. In that case, the *ParameterDescription* instance should indicate that the parameter is an identifier (e.g. `ucd="meta.id"` or `ucd="meta.file"`) or a URL (`xtype="xs:anyURI"`).

B.4 Links with the ProvONE data model

The IVOA Provenance DM includes several concepts that can be the starting point to define and execute scientific workflow-based computational experiments. In this context, the provenance information could be translated and further developed following the ProvONE ontology and data model (Cuevas-Vicenttín and Ludäscher et al., 2016), that was developped as an extension to W3C PROV-DM (Belhajjame and B'Far et al., 2013). A general mapping is presented in Table 21.

ProvONE extends W3C PROV with classes and relation that could also be used to extend the IVOA Provenance DM. For example, the relation *wasPartOf* between a parent and a child entity can be used to describe the structure of *Execution* instances, in that a parent *Execution* (associated with a *Workflow*) has child *Executions* (associated with *Programs* and subworkflows). In the same way, the *hasSubProgram* relation specifies the recursive composition of *Programs*, i.e. a parent *Program* includes a child *Program* as part of its specification.

Provenance DM	**ProvONE DM**
Activity	Execution
ActivityDescription	Program
ActivityDescription	Workflow
UsedDescription	Port
WasGeneratedByDescription	Port

Table 21: Mapping between classes from ProvONE DM to classes in ProvenanceDM. This list is not complete.

Appendix C Changes from Previous Versions

C.1 Changes from WD-ProvenanceDM-1.0-20180530

- Separate core model (W3C only) and extended model (IVOA Provenance DM).
- Add definitions for specialised entities, including all the description classes and parameters.
- Add definitions for specialised relations.
- Updated serialization of the description classes for web services.

C.2 Changes from WD-ProvenanceDM-1.0-20170921

- Moved ProvDAL to separate document.
- Moved ProvTAP section and full definition of VOTable serialisation to separate ProvTAP document.
- Moved chapter 6 with use cases and “How to use the data model” to separate Implementation Note (Servillat and Riebe et al., 2017).
- Moved section on links to other data model into appendix.
- ParameterDescription: Added attributes `xtype` and `arraysize`.
- Agent.roles: Removed “PI” alternative to “coordinator”.
- Use values of RightsType of DatasetDM, public, secure, proprietary, for `Entity.rights`.
- Minor corrections in HiPS use case, Appendix A and tables in TAP schema.
- Minor correction in role names for hadStep/hadMember relationship.
- Modified text on Parameters
- Rename 3.4 section to Serialization of description classes for web services
- Modified text on W3C serialization
- add `location` and `value` attributes to *Entity*

C.3 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 3 for explaining the different serialization formats and differences to W3C serializations, Section "Accessing provenance information"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 (Servillat and Riebe et al., 2017).
- Complete rewrite of the ProvDAL section in Section "Accessing provenance information";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 B.
- Extended the entity role examples in table `tab:entity-roles`.
- 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 B, 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 11 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 18.

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/

Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S. and Zhao, J. (2013), 'Prov-o: The prov ontology', W3C Recommendation.
http://www.w3.org/TR/prov-o/

Bonnarel, F., Demleitner, M., Dowler, P., Tody, D. and Dempsey, J. (2017), 'Ivoa server-side operations for data access, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/SODA/

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

Cresitello-Dittmar, M., 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/

Cuevas-Vicenttín, V., Ludäscher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., Dey, S., Kianmajd, P., Koop, D., Bowers, S., Altintas, I., Jones, C., Jones, M. B., Walker, L., Slaughter, P., Leinfelder, B. and Cao, Y. (2016), 'Provone: A prov extension data model for scientific workflow provenance', DataONE Cyberinfrastructure Working Group proposed specification.
https://purl.dataone.org/provone-v1-dev

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

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

Harrison, P. and Rixon, G. (2010), 'Universal worker service pattern, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/UWS/20101010/

Hua, H., Tilmes, C., Zednik, S. and Moreau, L. (2013), Prov-xml: The prov xml schema, Project report, World Wide Web Consortium.
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

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

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

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