International
Virtual
Observatory
Alliance

IVOA Provenance Data Model
Version 0.1

IVOA Working Draft 2015-05-18

Working group
DM
This version
http://www.ivoa.net/documents/ProvDM/20150518
Latest version
http://www.ivoa.net/documents/ProvDM
Previous versions
This is still an internal draft
Author(s)
Kristin Riebe, Florian Rothmaier, Markus Demleitner, Mireille Louys, Francois Bonnarell
Editor(s)

Abstract

[TODO:]

This version is still an internal working draft, any comments welcome!

NOTE: The urls above won't work yet, because this draft is only circulated via the volute-repository for now.

Status of This Document

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

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

Contents

Acknowledgments

This document has been developed in part with support from the German Astrophysical Virtual Observatory (BMBF Bewilligungsnummer 05A08VHA).

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 for an IVOA standard data model for describing the provenance of data. We focus here on observational data, since provenance for simulated data is already covered by SimDM Lemson et al. (2012). However, the currently discussed version is abstract enough so that it could be applied to any kind of processes, including extraction of data from databases or the flow of scientific proposals from application to acceptance and scheduling of the proposed observations.

This draft is at a very preliminary state and may change at any time.

1.1 Role within the VO Architecture

1.2 Requirements for Provenance and Use Cases

An IVOA provenance data model should provide solutions to the following tasks:

A: Tracking the production history

Find out which steps were taken to produce a dataset and list the methods/tools/software that was involved. Track the history back to the raw data files/raw images, show the workflow.
Examples:

- Is the image from catalogue xxx already calibrated? What about dark field subtraction? Were foreground stars removed? Which technique was used?

- Is the background noise of atmospheric muons still present in my neutrino data sample?

We don't go so 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 further information

Find the people involved in the production, 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? Whom do I need to cite or who can I contact to get this information?
- I have a question about column xxx in the data table. Who can I ask about that?

C: Aid in debugging

Find possible error sources.
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

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

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

E: Search in structured provenance metadata

Find all images produced by a certain processing step and similar tasks. Examples:

- Give me more images that were produced using the same pipeline.
- Give me an overview on all images reduced with the same calibration data set.
- 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?

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.

1.3 Further possible applications

The previously listed use cases were collected having an external scientist in mind who retrieves data from the virtual observatory and needs to get more background information. When looking at internal processes, one can also use provenance for checking workflows, e.g. if reduced images from a pipeline don't look quite right, the pipeline can be re-run with different parameters. Tracking the workflow in a common provenance description can help to quickly identify the problematic parameters and to keep a better track of changes. It can also be used to exchange pipeline recipes between different projects in a standardized way.

The provenance of the flow of scientific proposals at observatories could also be tracked. The information could be used to check internal processes, e.g. if the proposal was approved by a person from a certain committee, if the time span between application and acceptance/refusal does not extend a certain period etc.

1.4 Goal of the provenance model

The goal of the provenance data model is to describe how provenance information can be modeled and stored/exchanged within the virtual observatory. Its scope is mainly to allow modelling of the flow of data, the relations between data and processing steps. Characteristics of observations like ambient conditions and instrument characteristics won't be modeled here explicitely. They are to be included in the form of data sets (entities) only.

1.5 Previous efforts

Provenance was already discussed since the early days of the IVOA, but previous efforts from the IVOA community unfortunately are mostly not well documented or we were not able to find them. There exists a vocabulary list by Arnold Rots, we got a provenance diagram from a talk by Francois Bonnarell, and there exist workflows to track provenance for specific projects (e.g. CTA), but this is all very specific and too detailed for a general provenance model. We therefore went a step back to start with a more abstract core model as discussed in this document.
[**TODO:** Give more specific references, check other data models and where they touch provenance]

Outside of the astronomical community, the Provenance Challenge series (2006 - 2010), a community effort to achieve inter-operability between different representations of provenance in scientific workflows, resulted in the Open Provenance Model (Moreau et al. (2010)). Later, the W3C Provenance Working Group was founded and released the W3C Provenance Data Model as Recommendation in 2013 (Belhajjame 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 fairly general, so they can be applied to astronomical data sets 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, more specific or extend it where necessary.

The W3C model even specifies already PROV-DM Extensibility points (section 6 in Belhajjame et al. (2013)) for extending the core model. This allows to specify additional roles and types to each entity, agent or relation using the attributes `prov:type` and `prov:role`. By specifying the allowed values for the IVOA model, we could adjust the model to our needs while still being compliant to W3C.

2 A provenance data model

2.1 Overview - W3C core

We describe here the core concepts for modelling provenance. The resulting model can then be reused as a pattern everywhere where provenance is needed in the VO. Some examples for different use cases are given in Section 4.

The elements of a provenance model can be expressed as a directed graph to capture the causal dependencies. Based on the W3C PROV-DM, we identified the need for the same three core elements, which represent the

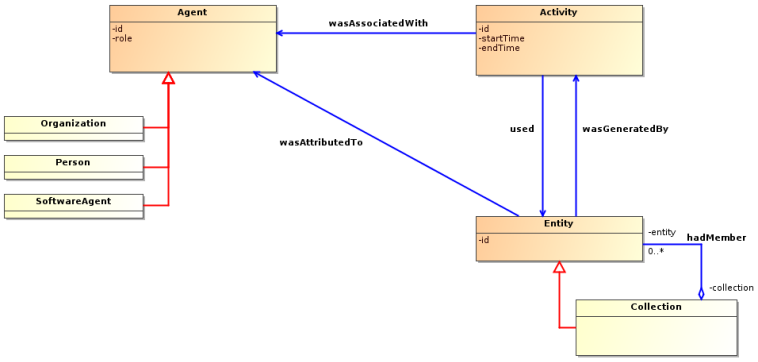

Figure 1: The main core classes and relations of the W3C Provenance Data Model.

nodes of a provenance graph (see Section 1+2 and Fig. 1 in Belhajjame et al. (2013)). They will be discussed in detail later on.

- *Entity:* a thing at a certain state (car, paper, data set, idea).
- *Activity:* an action/process or a series of actions, occurs over a period of time, performed on or caused by entities, usually results in new entities.
- *Agent:* executes/controls an activity, is responsible for an activity or an entity

There are also different possible relations between these components. The main important core relations are:

- *Generation:* a new entitiy is generated by an activity (output_data wasGeneratedFrom activity)
- *Usage:* an entity is used by an activity (activity used input_data)
- *Association:* agents have responsibility for an activity (agent wasAssociatedWith activity)
- *Attribution:* an entity can be attributed to an agent (entity wasAttributedTo agent)

Figure 1 summarizes these (core) classes and relations of the W3C provenance model that are interesting for us. Its components are now discussed in more detail with respect to the role they play in provenance for the virtual observatory.
[**TODO:** Decide, which elements of W3C-PROV are needed, what else to include here.]

2.2 Detailed discussion

2.2.1 Entity

Entities in the VO are usually (astronomical/astrophysical) data sets in the form of images, tables, numbers, etc.. But they can also be observation/simulation log files, observation proposals, scientific articles or manuals and other documents. The Dataset Data ModelBonnarel et al. (2015) specifies an IVOA Dataset as "a file or files which are considered to be a single deliverable".

According to the W3C model, provenance information can also be an entity itself (in the form of a "Bundle").

For each entity, we should store a number of attributes:

- id: a unique id for this entity (unique in its realm)
- prov:label: a label (to be displayed by clients)
- prov:type: a provenance type, i.e. one of: prov:collection, prov:bundle, prov:plan,
- prov:description: text describing the entity in more detail
- datatype: type of the physical representation of the entity, e.g. binary file, fits file, database, database table, ASCII file, tar-file, directory, ...
- prov:location: (or storage or accessReference?) where the entity can be found

Possible further attributes could include "access" ("public" or "restricted" or "internal") or how to access data on a certain machine etc.

2.2.2 Activity

Activities in the VO include all steps in the reduction of images and production of new data sets, like image calibration, bias subtraction, image stacking; light curve generation from a number of observations, radial velocity determination from spectra, etc.

- For each data flow it should be possible to clearly identify entities and activities. If the activities shall not be recorded explicitely, one can also use the *Derivation*-relation (see below) to link derived entities to their originals.
- Data entities are results from activities (*wasGeneratedBy*-relation), and can be used as input for other activities (*used*-relation).

The W3C provenance model requires for activities an id, a startTime and endTime. Here's a list of useful attributes for activities:

- id
- prov:label
- prov:type: one of the processes from a vocabulary, e.g. observation, reduction, calibration, ...
- prov:description
- prov:startTime
- prov:endTime
- docuLink: link to further documentation on this process, e.g. a paper, the source code in a version control system etc.

A "version" attribute could also be useful.

2.2.3 Agent

An agent can be an organization or a person, and can take different roles for an activity or an entity. The possible types of agents are:

- prov:person: a person, specified by first and last name, email address, possibly also by affiliation (though all these parts may change in time)
- prov:organization: a publishing house, institute, scientific project
- prov:SoftwareAgent (still needs to be discussed)

A definition of organizations in the sense of the VO is given in the IVOA Recommendation on Resource Metadata (et al., 2007), hereafter refered to as RM. This also specifies that scientific projects can be considered as organizations on a finer lever.

There can be more than one agent for each activity/entity (with different roles) and one agent can be responsible for more than one activity/entity. This many-to-many relationship could be made more explicite by adding role-maps for agents explicitely in the UML-class diagram, as "qualifiers" for the relations. The W3C PROV-DM document specifies two relations instead, which we can extend with a "role" attribute as well. These are:

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

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.

In order to make it clearer what an agent is useful for, we suggest here possible roles an agent can have (along with descriptions partially taken from RM)

AgentRoles

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

This list is not complete. Such roles are domain-specific and thus not fixed in W3C's PROV-DM. However, since we are considering here the Astronomy-domain, we could consider fixing these roles, most likely by keeping them in a vocabulary list.

[**TODO:** Do we have a specific use case for fixing the agent-roles? Is anyone going to search for specific roles in the Provenance meta-data? Or shall we leave it open, which roles can be defined and just give examples here?]

[**TODO:** We still need to clarify precisely, in which way a *software agent* is distinct from an activity.]

2.2.4 Roles of entities

[**TODO:** Probably these roles are best stored with the used/wasGeneratedBy relation directly.] The W3C PROV-DM specifies a **Derivation** relation,

which directly links a derived entity to its predecessor. This was introduced to make the link between entities more explicite. If activity a1 used entities e1 and e2 as input and produced e3 as output, then it is not clear, if e3 is a derivative (or, in W3C terms, "wasDerivedFrom") of e1 or if it was derived from e2 or from both or neither. e1 and e2 could have been only parameter files for an observation.

However, for the IVOA provenance model we currently favour adding a *role* to each data item or entity, which can explain in more detail in which way the entity was used.

We will discuss later an approach using the additional Method-class as prototype or template for each activity. The types of input/output data and their roles are described using additional classes for the method, so that any kind of relation between input/output data can be covered.

One of the important details here is that e.g. many data sets used by one activity may have different roles for that process (one file is a parameter file, another one is the raw image, and the third one is the dark field that should be subtracted). Since these roles are very important for an activity, they have to be included in the provenance model.

These roles don't have to be unique (see Moreau et al. (2010), after Definition 10), many data sets may have the same role for a process (e.g. raw image input).

Each activity requires specific roles for each entity (provided in the used- or wasGeneratedFrom-relation).

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

2.2.5 Mapping classes

We could adjust the W3C model by replacing the used- and wasGeneratedBy-relations by a mapping class between activity and entity in the class diagram in figure 2, for making the possible *roles* of input and output data more specific.

The different methods and data types are here hidden in pre-defined vocabulary lists. An example for writing down the provenance for a stacked image in W3C and with the protoype-model (which is discussed in a later section) is given in the accompanying files `prov-example-incl-prototypes.txt` and `prov-example-w3c.txt` at volute.

2.2.6 Bundle/Collection

W3C describes a collection as an aggregation of entities. This is a useful concept to adapt here, since it allows to treat many data (e.g. a RAVE data

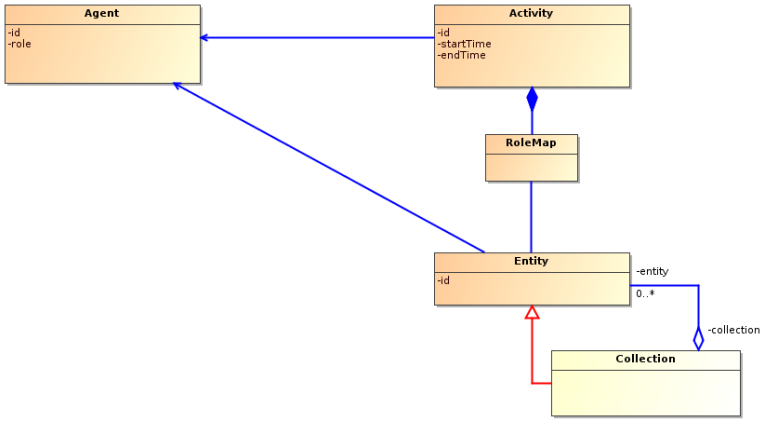

Figure 2: An adjusted W3C model for provenance of astronomical data.

set containing data tables and spectra) and then define provenance for this complete bundle. On the other hand, this means that we need to define for each element of the bundle an "isChildOf" or "isMemberOf"-relation with the bundle. This can become quite cumbersome and will not look much more concise than writing the used- or wasGeneratedBy-relationship down for each of them.

2.2.7 Work flows

W3C suggests to use the term *Plan* for entities that represent workflows and recipes. This could be interesting to include for workflow systems such as AstroTaverna.

2.2.8 Hierarchical descriptions

OPM also suggests to allow hierarchical descriptions, i.e. allow to include different ways of getting from dataset A to dataset B, with different levels of detail. This needs to be discussed further.

2.3 A model using prototypes

Inspired by Lemson et al. (2012), a data model for simulation data published in May 2013, we also discussed a provenance data model for the IVOA using prototypes. Currently, we are still debating about this, but favour to follow the example of the W3C-model and postpone a decision for or against prototypes to a later version of this model. The W3C-model has the advantage of being already an approved standard, and it contains all the necessary main features needed for a Provenance model for Astronomy. Nevertheless, we

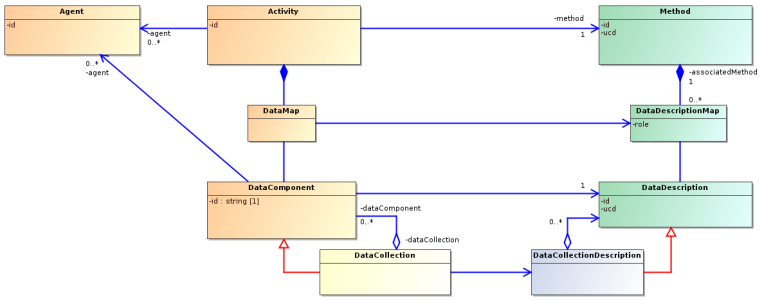

Figure 3: The (current) Provenance Data Model class diagram. It will certainly change again, so use it with care. In this version, we stripped it now from nearly all derived classes, so we can concentrate on the core elements. The green and blue boxes belong to 'prototype' classes.

still include here some sections on what we had in mind for the prototype model.

[**TODO:** The following sections need to be rewritten.]

From SimDM we adopt 2 central classes: Protocol and Experiment. A Protocol is the plan or method which describes how to do something whereas Experiment is the actual execution of this plan. Defining such protocols allows them to be reused, which is very useful when performing series of experiments of the same type, as is typically done in astronomy. For our model here, we use the more generic term `Activity` instead of Experiment, as it us used by the W3C Provenance Data Model (Belhajjame et al. (2013)). Instead of `Protocol`, we use the term `Method`, for not confusing it with e.g. IVOA protocols.

The class diagram generated with MagicDraw Community Edition is given in the figure3; the details are described in the next sections.

2.3.1 Activity and method

Activity corresponds to Experiment and Method to Protocol from Lemson et al. (2012).

Examples for astronomical activities are observations and processing steps like flat-fielding, correcting bias, astrometric calibration etc. The type or method underlying an activity is specified by the corresponding *Method* class. This could be, for instance, the name of the code used to perform an activity or a more general description. An activity is a concrete case of using such a method, thus it has to refer to a corresponding method.

There MUST be exactly one method per activity. If steps from a pipeline shall be grouped together, one needs to create a proper method for describing all the steps at once. This method can then be refered by the pipeline-activity.

2.3.2 Data and DataDescription

We define a general `Data` class (actually: DataComponent, see next section), which can represent either an individual data item or a collection of data. The W3C equivalent to this class is called `Entity`, but we use `Data` instead, since we do not aim to model cars and abstract ideas here, but more concretely astronomical data of any kind.

Following the scheme that each `activity` is described by the more general `method`, we associate each input or output data set with a corresponding description of the data type. Each method usually makes assumptions on the structure of input data (FITS file, data table, binary file of a certain structure ...) and produces a certain form of output data, which should be described in the general `DataDescription` class, so that it can be reused for multiple instances of this class.

There has been some dispute (before and at the InterOp in Madrid, May 2014) about the different roles of input and output data. Since the results of one activity can be input data for another activity, the structure of input and output data is necessarily the same. The link from an `Activity` to the corresponding `Data` object could tell then if the piece of data is used as input or output, see Figure 4 in section 2.3.6.

However, input data can take different roles in an activity (auxiliary data like a parameter file, two images, with one that needs to be subtracted from the other) and it must be made explicit which data component that is used by or generated from an activity fulfills which role. In W3C, this is partially solved by adding a derivation relation between the entities (data). Here, we add a mapping-class between activity and data as well as between method and dataDescription. The mapping-class at the description side, i.e. between the Method and its DataComponentDescriptions, contains additionally a role for each relation, e.g. parameter, dark frame, raw image, etc. If a dataComponent is used as input to an activity or if it results from it, will become clear with these roles.

Note: Suggestion by Jochen: What about defining a Role-class for methods? This way each method should define the roles that are allowed for the method (probably taken from a specified list of vocabularies), enhancing interoperability.

Without the mapping tables, the relation between activity/method and data/datadescription would be an aggregation relation, or in other words: an association with the aggregationKind "shared". That would be required to ensure that all DataComponents linked to an activity (either as input or output) will survive if the activity is destroyed, since they are almost always shared with other activities.

By using the mapping tables we make the role of a dataComponent in an activity more explicit and thus can replace the aggregation by a composition

relation to the activity/method and simple associations to the individual data components and their descriptions.

2.3.3 Data Collections - Composite pattern

There are two major data classes:

- data collections
  e.g. RAVE-DR4 with its databases and database tables, SDSS DR9 with all its tables and files, all files from one observing slot

- individual data components
  e.g. a file, table in a database, parameter value, an image, a fits-file containing a table and image

Data can only be grouped to a data collection, if they have the same origin (i.e. they were produced by the same activity, i.e. they have the same provenance). We would leave it to the person/tool recording provenance to decide, how detailed a data set will be separated into individual items. It is also possible to just record provenance for e.g. RAVE DR4 as a whole, without listing everything that belongs to DR4. The level of detail could then be specified via the DataDescription.

For our data model, it would be desirable to be able to treat data collections and individual data items in the same way, with the same interface. This demand led us to the **Composite design pattern**:

We introduce an abstract class called `DataComponent`, which includes the basic properties and functionality for both, data items like files or parameter values or a complete data set. Such common properties are e.g. the link to the activity which created the data/dataCollection, a creation date/time and a link to a storage-object (which is not further specified in this data model).

Each DataComponent can either be a single item (like a file, database table or a parameter) or a DataSet. A DataCollection contains additionally a (non-empty) list of child-DataComponents, which could again contain a list of further children etc. This way, one could represent complete data trees, if necessary.

Each DataComponent also has a link to its parent, which would be emtpy (or point to itself) for the root-DataComponent.

We hope that such a representation would make it easier for clients to handle data objects, since they don't have to make the distinction between data items and data collections.

2.3.4 Agent

In SimDM, someone performing a certain experiment is called `Contact`, the W3C provenance data model suggests the term `Agent`, so we adopted it here. We want to describe someone who is responsible for an activity, e.g. who pressed a button, ran a script or performed the observation. The agent could be a single person (specified by name), a group of persons (e.g. MUSE WISE Team), project/organization (RAVE) or an institute. For each of them a name should be specified.

It is desired to have an agent given for each activity, but it is not enforced (hence 0..*). 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 someone later on. The agent should not only be used for getting contact information, but also to fulfill our "Attribution" requirement, so that proper credit goes to the right people/projects. To this end, it could also make sense to add a relation between a DataComponent and Agent, similar to W3C's wasAttributedTo-relation.

[**TODO:** Specify pre-defined contact or agent roles to choose from, in order to maximize interoperability (see SimDM: owner, creator, publisher, contributer; any more?)]

2.3.5 Storage

The modelling of storage is not included in this model. It would be useful to be able to make a reference from the DataComponent to a storage object that contains the link to where the data is stored. But we haven't fixed the details here.

2.3.6 Links Between Data

It would be convenient if each data object or even each file (`Storage object`) gets a unique `dataId` that can be referenced. If several copies have been made out of a data set and one of them is corrupted, it would even be useful to know exactly which copy was used by a given activity.

Maybe DataLink can help here? Or DOIs?

We need to be able to refer from an activity to the ResultData of another activity. A possible data flow is shown in the figure below:

2.3.7 Calibration data

The calibration data set consists of images that can be used to calibrate the raw data. It is not necessary to mention them explicitly in the model, they are just another dataSet that is used by activities with a calibration-method.

Figure 4: The black arrows indicate the data flow, gray lines just show which activity would be attached to the step. It is defined by the role in the corresponding Activity whether data is used as input or output.

2.3.8 Ambient Conditions

Ambient conditions are environmental properties, which are special in a way: they represent the part of an activity, usually an observation, which cannot be (fully) controlled by an observer, in contrast to other data that can be fully reproduced. Nevertheless we decided that they can be fully described by our data class already and don't need a separate class in our data model.

Our model can then also take into account that a certain observation method requires special ambient conditions, already defined via the method (e.g. radio observations rely on different ambient conditions than observations of gamma rays), just following our data - data description scheme.

Ambient conditions are recorded for a certain time (executionDate) and

are usually only valid for a certain time interval. This time interval should be recorded with a validity-attribute for DataComponents.

We wondered if ambient conditions could also play a role for some processing steps or any other activity besides observations? Is there any additional step performed in which room temperature etc. may play a role and thus should be recorded? The only example that came to our minds was the storage of photographic plates, where the humidity and temperature variations can affect the quality of the photographic plates.

2.3.9 Instrument Characteristics

In contrast to ambient conditions, instrument characteristics do (usually) not change from one observation to the other, so they are static, strictly related to the instrument. All the characteristics could be described either as key-value pairs directly with the observation or just as data, using the DataComponent class. One would then link the instrument characteristics as a type of input data set to a certain observation activity. Thus we don't need a separate Instrument or Device class.

One should also keep in mind that some instrument related parameters can change within time, e.g. the CCD temperature. The instruments can also change within time because of aging.

2.3.10 Quality

We could simply define additional attributes for each `Activity` or `Data` object, i.e. zero, one, or more properties in the form of key-value pairs. We could use a `Quality` namespace to mark a keyword as quality-related:

- quality.comment: [some text]
- quality.seeing: [some value]

The values could range from a float number to free text.

2.3.11 Discussion

This model was established with having a database implementation in mind. However, the W3C model may offer simpler possibilities to store provenance with the dataComponents themselves, e.g. as an additional structure in fits-headers.

A model using prototypes has the advantage of normalisation: methods could be described once and for all at some place (this *some place* is actually the crucial point here!) and then be reused when describing the actual provenance of certain dataComponents and activities. However, building such a look-up place for all the possible methods and dataDescriptions is

a quite challenging task – it will probably never be complete. There's also the issue of persistent identifiers/broken links to consider. Normalisation is useful for closed systems, e.g. for describing the provenance for data produced by a certain pipeline (e.g. MuseWise system) or with workflow tools or when a task needs to be repeated many times. However, the VO is quite the contrary of a closed system and we need to keep an eye on what is actually achievable.

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

Expressing provenance for a stacked image with this smaller set of classes may be simpler, but on the other hand constructing a database schema becomes much harder. The question is: could we just derive a simple exchange format from the prototype-model? Should we be instance-driven here? Which model would be more useful? Though we currently favour the simpler model, we could still come back to the prototype model in a later version, if we consider it more useful.

3 Applications/Interactions with other Data models

In this section we discuss how the provenance data model interacts with other VO data models and how provenance information can be stored.

4 Examples

4.1 One processing step in PROV-N notation

[**TODO:** Put the very simple example here (see talk)] See `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/description/prov-example-incl-prototypes.txt` and `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/description/prov-example-w3c.txt`

4.2 Provenance of RAVE database tables (DR4)

This example shows how the workflow of RAVE data, from images to the final database tables, can be expressed using Provenance. The workflow is not included completely, only some major steps are taken into account. It shows that the provenance concepts explained in this draft can be applied directly to data obtained from astronomical observations.

[**TODO:** Include here figure from InterOp talk. See also https://provenance.ecs.soton.ac.uk/store/doc

5 Implementations

[**TODO:** Maybe combine with examples?]

A Changes from Previous Versions

[No official previous versions yet, so not keeping track of changes here.]

References

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

Bonnarel, F., Laurino, O., Lemson, G., Louys, M., Rots, A., Tody, D. and the IVOA Data Model Working Group (2015), 'IVOA dataset metadata model', IVOA Working draft.
URL: *http://www.ivoa.net/documents/DatasetDM/20151021/*

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

et al., R. H. (2007), 'Resource metadata for the virtual observatory', IVOA Recommendation.

Lemson, G., Bourgès, L., Cerviño, M., Gheller, C., Gray, N., LePetit, F., Louys, M., Ooghe, B., Wagner, R. and Wozniak, H. (2012), 'Simulation data model, version 1.0', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/SimDM/*

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