Provenance Meeting

April 14
Observatoire de Paris, Salle du Conseil
https://indico.in2p3.fr/event/13027/

Catherine Boisson; François Bonnarel; Johan Bregeon; Pierre Le Sidaner; Julien Lefaucheur; Mireille Louys; Markus Nullmeier; Ana Palacios; Kristin Riebe; Michèle Sanguillon; Mathieu Servillat

---

Présentations:

- **Kristin: RAVE**
  - W3C PROV presentation
  - Test with ProvStore
    - https://provenance.ecs.soton.ac.uk/store/documents/84064/
  - Web App for RAVE Provenance, using Django
  - ProvPy: serialization
  - ProvJS: visualization
  - Structured DB (SQlite3) with entity, activity, agent, relations...
  - Users: project management, scientist, pipeline writer
    - require different levels of description for the same model
  - Use cases per user
    - project manager (not a lot of details),
    - pipeline (processes involves, files, …),
    - scientists (details for an entity, are intermediate files for a given obs available? how I could get them? What's the difference in processing? how are values changing for each data release?...)
  - Agent = contact person here, if question, the user can contact someone
    - type = « project » : webpage

Simulation use case?
re-use of various data sets for various purposes (ingredients, various recipes)

- **Mathieu: CTA**
  - CTA project, complex metadata, pipeline
  - CTA Data Model includes PROV inside
  - DM field categories: ObsCore, ObsCore options, ObsConfig, Provenance
  - Use case families: Cone Search, ObsCore queries, ObsConfig queries, Provenance Queries
  - CTA Data Distiller: implementation of VO access to CTA data
    - https://voparis-cta-test.obspm.fr
  - Test case with UWS job server
    - https://voparis-uws-test.obspm.fr/
    - 1 UWS job = 1 activity (ex: CTA DL3 to DL4 using ctools ctbin)
    - generate a ProvDocument (ProvPy)

        - adds xml, json, svg to job results
    - Need job description (= ActivityDescription)
        - UWS JDL, WADL too limited, need more structure
        - includes a parameter list (which one are queryable?)
    - Storing Provenance: files, header or frames, DB
    - Retrieving Provenance: from files, from DB
        - request Provenance path (or progenitor) for a data product
        - Search data products based on Provenance
            - filter by activities
            - filter by input parameters to an activity
    - Access: ProvTAP?

Prov in data product header (e.g. FITS):
Johan: Standard File Catalog : provenance is a extension of it ?
Johan: Scientists don't want to have a lot of provenance details
Catherine : often goes to see some details when analyzing the data
Mathieu:
- ESO provides long headers (structured with HIERARCH keywords) that are relevant for the final user
- new format in defintion at STSci for JWST (ASDF format) with structured header
Catherine: efforts to get a common dictionary for Cherenkov data
Mireille: what level of details? short, intermediate, full according to use case.

- **Michèle: Pollux**
    - Pollux context
    - Workflow with 2 activities
    - Access: web, or SSA, via CASSIS
    - Provenance = FITS header keywords (including simu parameters)
    - Provenance info not directly visible in CASSIS
        - add DataLink to header?
    - Data Models: SpectrumDM, SimDM, ProvDM (mix of those)
    - Use cases (examples of queries)
    - Implementation using IVOA Prov DM
        - store PROV files generated from headers
        - JSON, XML, PNG, SVG, PDF...
        - Should add VOTable
    - namespaces and names (too long to be readable)
    - groups of parameters (too many)
    - SimDM and SimDAL should be further explored

- **François: DAL landscape in IVOA**
    - DAL in IVOA
    - base: VOTable, TAP, ADQL
    - data cubes access: ObsTAP, SIAV2, SODA
        - ObsTAP or SIAV2 --> request VOTable
        - DataLink or service descriptor --> available service
        - SODA (Server-side operation for DataAccess)

- Spectra, 2D images:
  - idem via SSA or SIAV1,
  - but no SODA (cutout/mosaic inside SSA/SIAV1)
- ObsCore (additions)
- Several implementations exist
- DataLink: use and examples
  - 2 mechanisms: Service descriptor, Link resource
- Prov access:
  - add provenance attributes to main discovery response table
  - ProvDAL: HTTP interface with ID=… (e.g. pub_did), FORMAT=...
  - ProvTAP: ProvDM mapping in TAP Schema

Discussions:

Mireille: discussion on the current version of the data model

Objective: we reconsider and explore further attributes in the classes of the data model. We also checked what should be stored in the xxxmap classes , the classes describing relationships between the main classes.

NB : proposal to simplify Class names :
DataEntity --> Entity
DataCollection --> EntityCollection ? or simply Collection ?

- **Agent**
  - Several cases :
    - contact **person** : name email tel affiliation
    - **organisation** : same parameters apply.
    - --> add type to Agent with possible values Person, Organisation...
  - **role**: what the agent was responsible for. Role examples:
    - data producer for operation quality check
    - data publisher for credit
    - for error report and corrections
    - data scientist for credit, collaboration
    - funding organisation
  - NB: the role is an attribute of the relationship between an agent and a data entity should be stored as an attribute of ActivityAgentMap, for any couple.

- **Entity** (instead of DataEntity)
  - for a data set: It should be identified by a **unique identifier** in the VO (IVOID). Such an Id can be ObsCore obs_publisher_did which is unique in the VO frame.
  - Entities may be associated files (mosaics, multi-segment spectra eventlists, etc) and form a **Complex Entity**.
  - the term **DataCollection** was not discussed in details : is it a survey, a full

one band or one mode set of observations within a survey? need more discussion... (*Mireille*)

- **EntityDescription** (instead of DataEntityDescription)
  - Related to **ObsCore** for datasets
  - **Dataset DM**: in evolution, already includes attributes corresponding to provenance (+ curation, contact)

  - Possible attributes:
    - identifier --> dataset persistent id
    - name
    - dataproduct type
    - dataproduct sub-type
    - format (TSV, VOTABLE, JSON, XML, ...)
    - url : etc... linkToObscoreSerialisation document

- **Behaviour**/properties of an Entity class:
  - An Entity has several relations into the W3C pattern :
    - was generated by --> activity
    - was derived from ---> entity
    - was attributed to --> agent
    - used by ---> activity
  - an Activity class has these relations:
    - was informed by --> activity
    - was associated to --> agent

- **Granularity**
  - Entity = Data Product or Entity = parameter of activity
  - add "ParameterList" with parameter description?
  - add "Parameter" with parameter description, e.g. name, unit, UCD, description, datatype
  - too many parameters = not human readable (but is it necessary to make the files human readable?)
  - maybe 1st level: list of parameters, 2nd level: parameter
  - The Entity **relationships** do not apply for individual parameters in a simulation or for a reduction algorithm for instance.
  - The **proper granularit**y might by the set (list) of parameters that a task takes as input to a processing step (i.e. an Activity) in the WorkFlow (an activity).

Michèle: Pollux param collections (as an input entity) that include parameters as entities (2 levels of details)
Kristin : we could support several end-users profiles: VO-User, Pipeline manager, etc. allowing to extract serialisation documents with different levels of details.
François: what about "quantities" as defined in the VO? looks like parameters here
Kristin: Parameters could also be attributes to the activity

Michèle: param list are entities, but parameters also
Mathieu: parameter is connected to activity the same way as entity is connected to activity, but is parameter an entity? with all its attributes and relations?

Mireille: Should we define an ActivityParameterList Class? SImDM might propose guidelines for that (to check). Need to work on practical examples not only simulation worflows.

    - **Parameter** class
        - Possible attributes:
            - name
            - description
            - datatype
            - unit
            - ucd
            - default value
        - also min/max, choices (?)

    - **ActivityParameterList**
        - Possible attributes:
            - numberOfParam
            - description
            - format (TSV, VOTABLE, JSON, XML, ...)
            - url
        - for URL: linkToParameterList document (suggestion ML: cf VOSpace storage for jobs ?)

- **Activity** class
    - identifier
    - startTime
    - endTime
    - status
    - next: link to the next activity in a sequence (NULL if end of list?)

- **ActivityDescription** (= Method in IVOA document v0.1)
    - Possible attributes:
        - name
        - type
        - description
        - url webpage
        - bibreference
        - rank : the execution number in the sequence //discussed for ordering the sequence
    - Similar to **SimDM**? "protocol", "simulator" and "post-processor"
    - **type** = qualified name pointing to the project with its namespace (direct link

to description)
    - Should we add **subtypes**? e.g. defined at the project level
    - From *Kristin* presentation:
        - type: observation, reduction, classification, crossmatch, chemical pipeline, distances, other
        - docuLink: link to documentation, e.g. paper, webpage, ...
    - Possible list of Activity types: [added while writing the minutes]
        - preparation (subtypes: proposition, approval, scheduled, ...)
        - observation
        - calibration
        - reduction
        - reconstruction
        - analysis (classification, crossmatch, ...)
        - simulation (simulator, post-processor as in SimDM)
        - ...
        - publication
    - *François*: discussion for the semantics group, related to DataLink vocabulary

- **ActivitySequence** (ordered set of activities) proposed in place of ActivitiesCollection
    - nb of steps
    - name
    - version
    - description
    - url webpage
    - startTime
    - endTime
    - status

- **ActivityEntityMap** (instead of ActivityDataMap)
    - activityId: internalID // execution ID for the system
    - EntityId : IVOID
    - EntityRole: string

Mireille: suggestion to remove **ActivityDataMapDescription**: seems redundant with other role item in ActivityEntityMap and description in ActivityDescription .

- **Access to Provenance data**
    - How to navigate Obs <--> Prov ?
    - 2 cases:
        - A- 2-step discovery: Obscore, then Prov
        - B- enrich Obscore with Prov in TAP Schema

Mathieu: CTA Data Distiller example:
2 steps in the discovery of Provenance, but not Obs --> Prov. When dataproduct_type is

selected in the search form, a Prov profile is read for the given data product type and the form is modified accordingly (adding Prov fields specific to the data product type). Or one could list all possible Provenance fields for all data product types as a flat list appended to ObsCore fields, but some fields not relevant for some data products.

François provides more details on the 2 cases:

- A- 2 steps:
  - ObsTAP service
  - ADQL: select...
  - VOTable with pub_dids
  - DataLink to "ProvDAL"
    - url?id=pub_did&FORMAT=PROV-XML&STEP=LAST
    - url?id=pub_did&FORMAT=PROV-XML&STEP=ALL
  - +/-:
    - - not relevant for simulations (ObsCore based solution)
- B- enriched ObsCore
  - 1st option:
    - ObsCore fields + Activities (id, start_time, ...) + ActDescription (id, name, ...) + ...
    - "ProvTAP"
    - +/-:
      - + diff type of queries
      - + maintenance is easier
      - - not for simulation data (ObsCore based solution)
  - 2nd option:
    - Reverse solution, Prov --> Obs
    - table of provenance: Activities, Act Desc, DataEntity...
    - somewhere there is data_id to get ObsCore view

Michèle: A- solution is used for Pollux (but using SSA, not ObsTAP)

Kristin: maybe add details_level to A- (provides activities only, + entities, then + parameters)

Mathieu: all those solutions provide the data product progenitor and history, but difficult to build a query based on ObsCore + Provenance fields filters. 1st step could be: get the provenance fields for a given data product type (DL3, DL4 spectrum or image, ...), then 2nd step: perform ObsCore+Prov query. Closer to B- 2nd option.

- **Format for Provenance serialization**
  - FITS
    - CTA (and probably most projects) will use FITS
    - FITS words as in HISTORY fields
    - could use a specific FITS extension for metadata
  - PROV-N machine and human readable
  - JSON easy when dealing with javascript code or Python
  - PROV-XML: more descriptive that JSON (?)
  - VOTable : used for ObsCore and possibly ObsCore+Prov serialization

        - Need a conversion program that could translate from VOTable to the other formats

- **How to express the model?**
    - Use the VO-DML definition of a VO model PROV-Vodml.xml
    - Compatible UML modeler tool could be:
    - Modelio community edition (to check)
    - This document will describe the clases and attributes definition.
    - How to check that a serialisation document is compliant to the model ?
    - Should we write our own validator?

Questions:

- Where do we draw the line provenance / metadata?
- How to use namespaces? (links to doc description, to data, to webpage...)
- Should we include prov namespace in voprov? (redefine or redirect to prov)
- Add loop on Activity (wasInformedBy)?
    - information is already accessible in DM through several queries
    - add wasInformedBy table: no multiple queries, but redundancy
    - same for other W3C PROV relations, add quick access tables
- Add Parameter or ParameterList classes?

Actions:

- All: try to update use cases to the DM changes
- Next meetings:
    - Mireille: IVOA Cape Town, ask for 5 Provenance talks... session?
    - Prov follow-up meeting June 14, before ASTERICS meeting (15-16, +17)
    - Markus: check feasibility Heidelberg meeting
- Michèle: update UML diagram during the meeting
- Mathieu: reshape minutes