Leibniz-Institut für
Astrophysik Potsdam

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

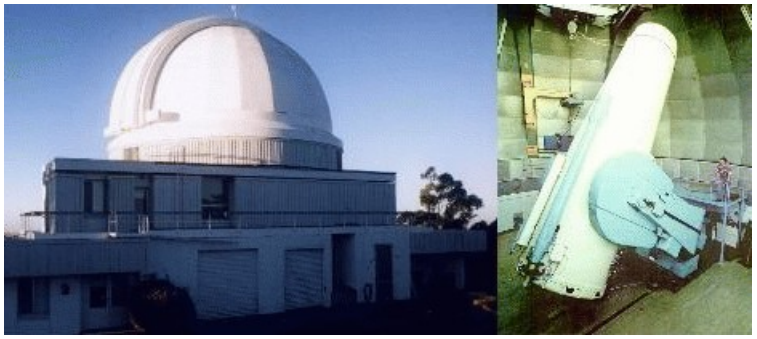
Provenance Webapp for RAVE - Recent updates

Provenance Day, Paris, 27th + 28th July 2017

Kristin Riebe

RAVE Provenance

- Using RAVE pipeline (workflow) as example
  - 1/2 million sources observed, spectra
  - different calibration steps, combining and splitting files, generating radial velocities, stellar properties, cross-matching with other catalogues
  - data release: mainly tables with stellar properties



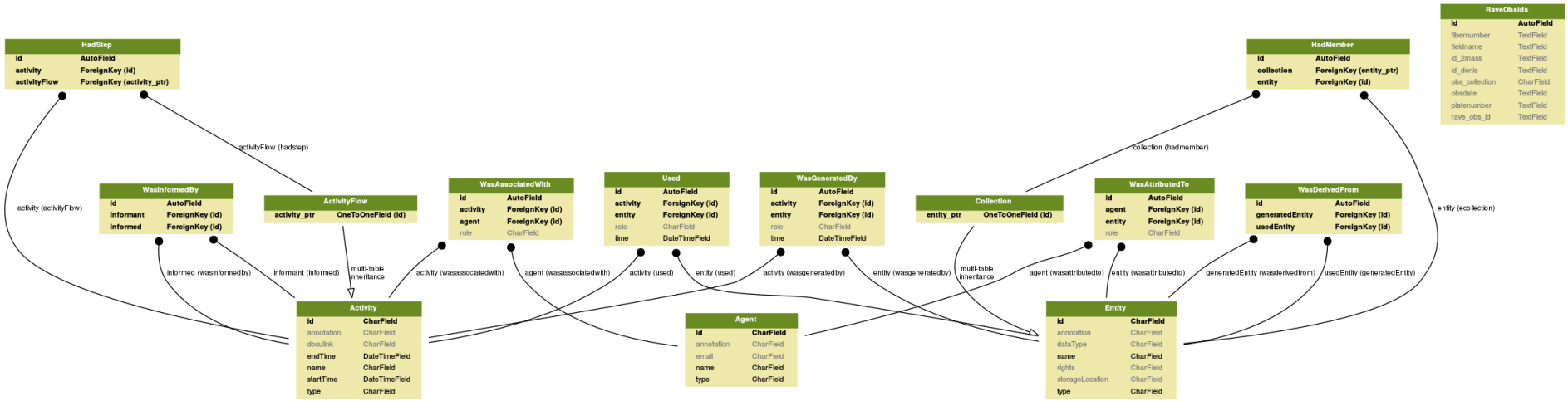
RAVE Provenance webapp

- Django web application (Python)
- Prototype for implementing IVOA ProvenanceDM
- Features:
  - implementation of main classes as Django models
  - list all instances of a class (Rest API)
  - show details for a single object (Rest API)
  - ProvDAL access for retrieving provenance for given id
  - serialisation of provenance information, IVOA and W3C versions
  - visualisation of provenance using javascript

https://github.com/kristinriebe/provenance-rave
https://escience.aip.de/provenance-rave

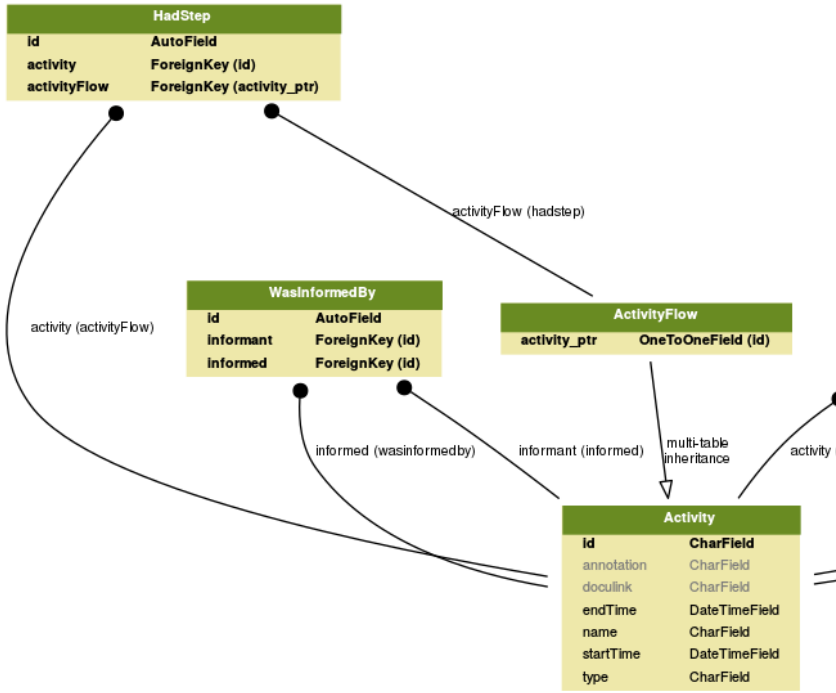
RAVE Provenance webapp

- Each provenance class implemented as Django model (Python class), database table generated automatically
- Contains main classes from draft, no description classes (yet)
- Overview of implemented classes (auto-generated):



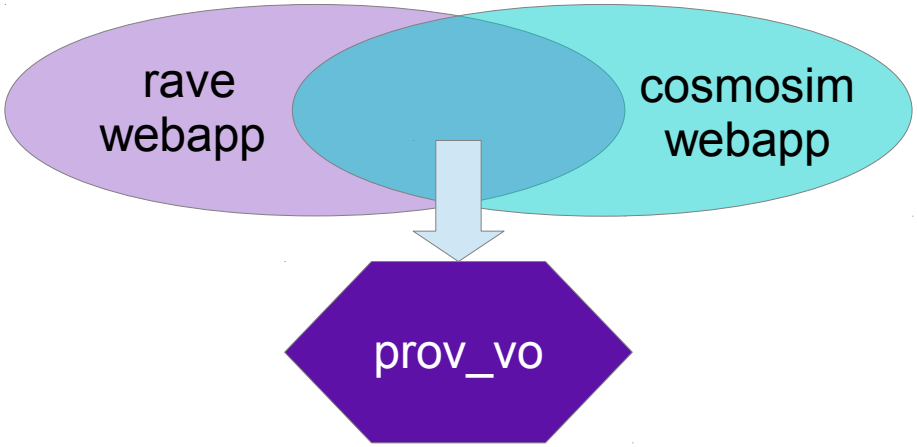
RAVE Provenance webapp

- ActivityFlow + HadStep:
  - could just add flag/attribute to Activity to mark ActivityFlow
  - decided here for extra class, inherit from Activity
  - advantage: explicit link to class ActivityFlow in HadStep
  - no flow-specific attributes needed



django-prov_vo

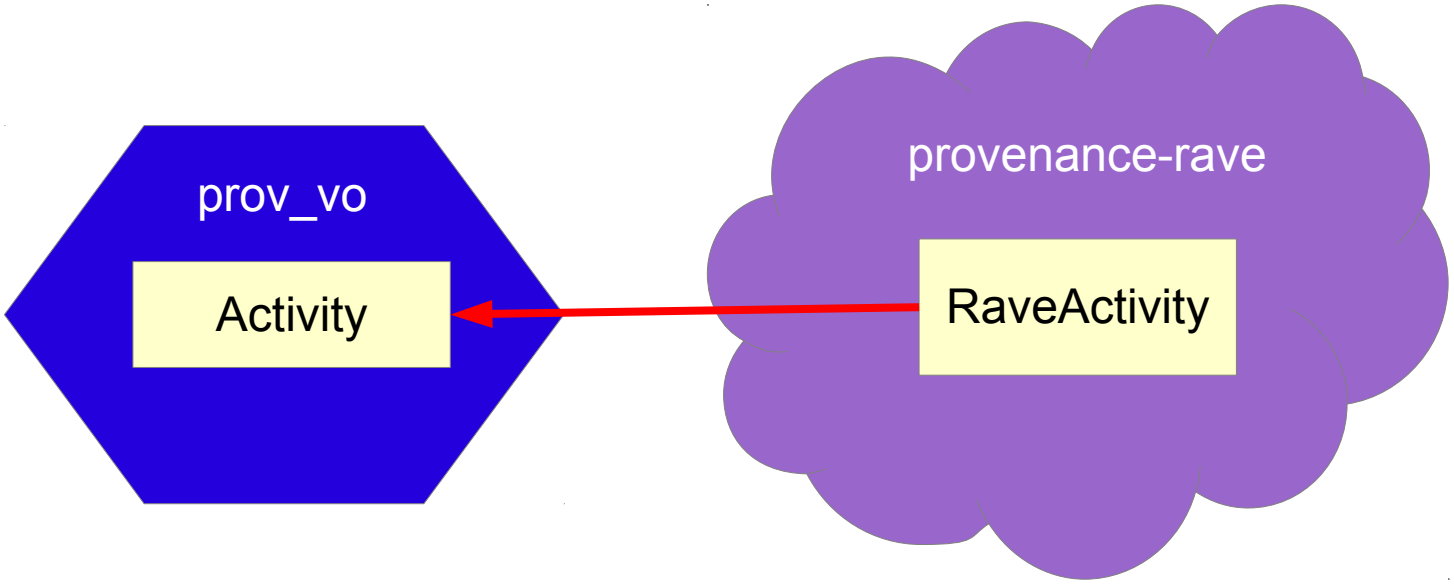
- Basic provenance implementation now (mostly) separated from RAVE-specific attributes etc.
- => reusable package „django-prov_vo“ (~ abstract classes)



- => all project specific attributes, extensions can be stored in the main app,
  => common provenance implementation can be the same for each webapp

django-prov_vo

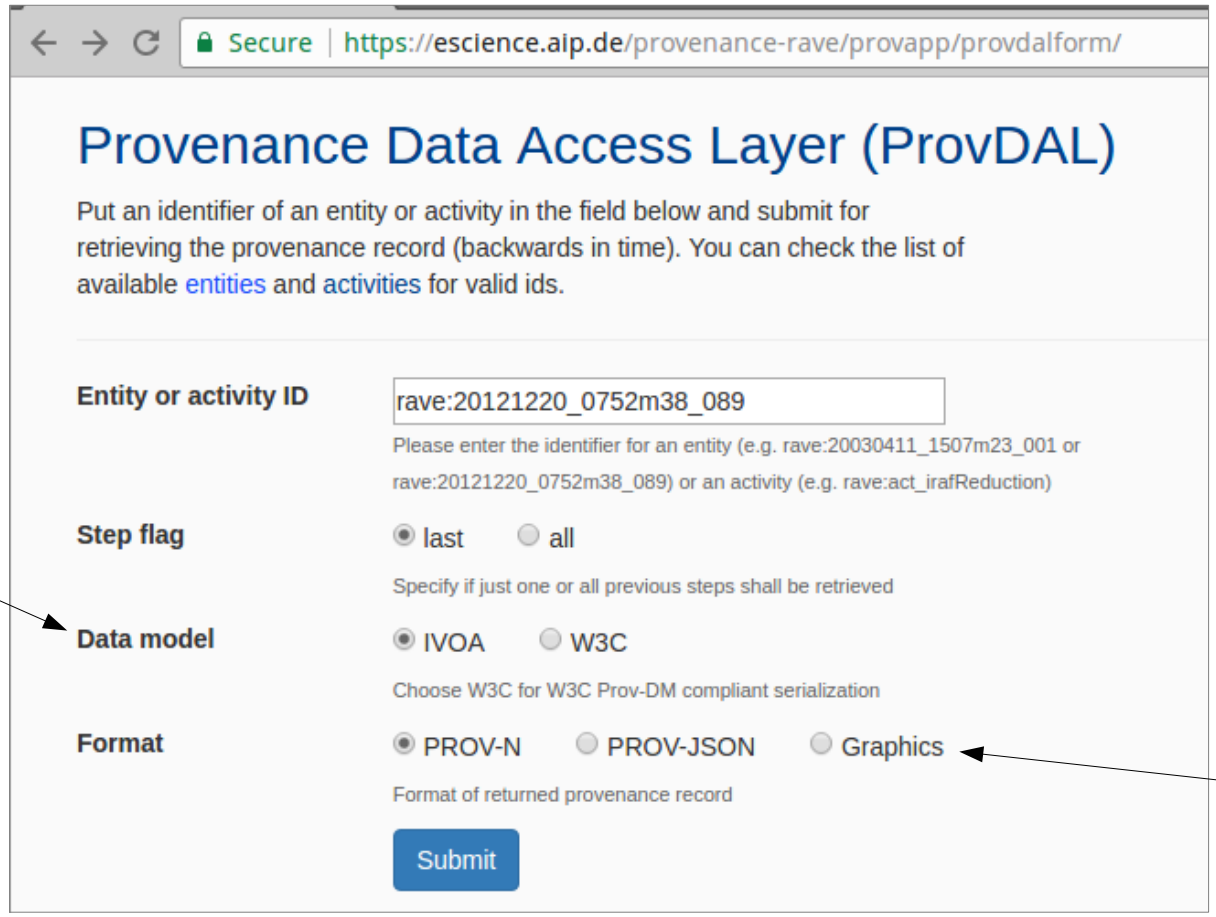
- classes in RAVE webapp inherit from basic classes
- e.g.: class RaveActivity(prov_vo.models.Activity)
- still work in progress



ProvDAL

- Implemented ProvDAL interface for retrieving serialized provenance description for a given entity/activity, included in django-prov_vo package
- **Parameters** (from draft):
  - ID *(of entity or activity, can occur multiple times)*
  - STEP *(=LAST or ALL)*
  - FORMAT *(=PROV-N or PROV-JSON)*
- **Additionally:**
  - option FORMAT=GRAPH
  - parameter MODEL *(=IVOA or W3C)*
  - Web form for nice user interface

ProvDAL webform



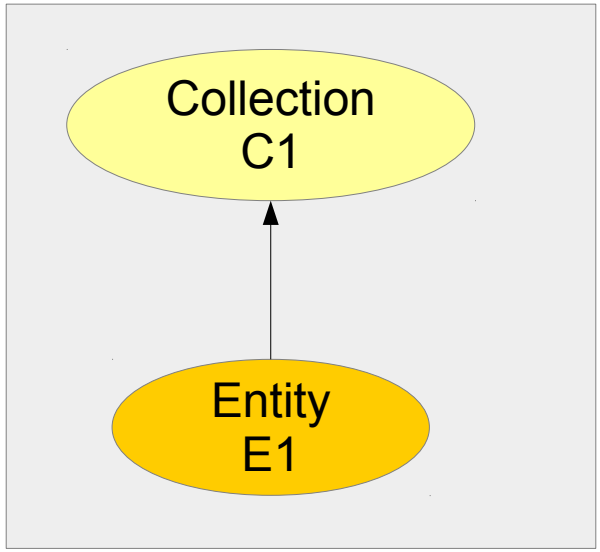
new parameter

additional option

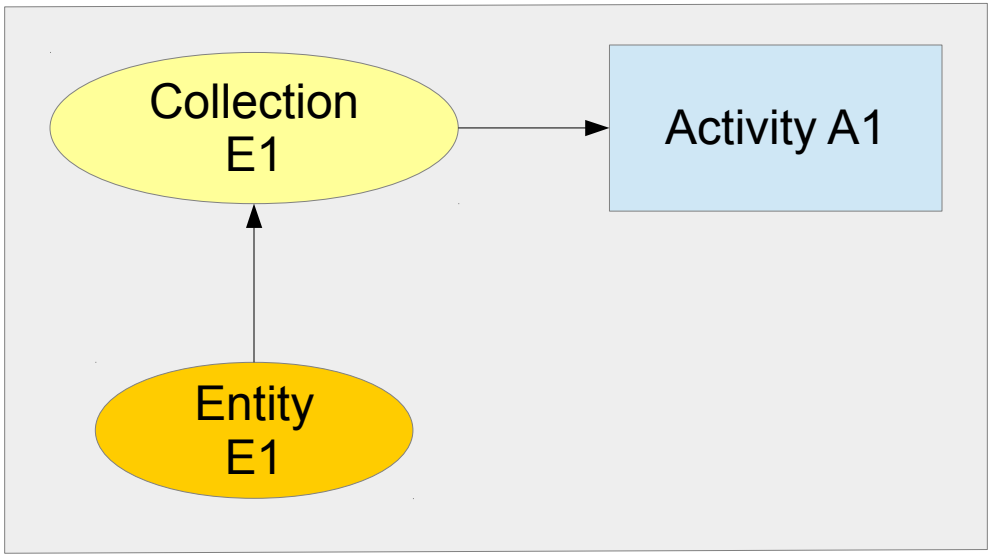
Automatically generates the ProvDAL GET request URL: https://escience.aip.de/provenance-rave/provapp/provdal/?ID=rave:20121220_0752m38_089&STEP=LAST&FORMAT=PROV-N&MODEL=IVOA

ProvDAL questions

- STEP=LAST:
  - Interprete as 1 step *backwards in time*?
  - Or just go exactly 1 *relation* further (in each direction)?



1 step only



1 step backwards in time

ProvDAL questions

- STEP=LAST:
  - Maybe rename (STEP=ONE)?
  - Maybe use integer instead and allow to specify depth?
    - STEP=1 (follow 1 relation (=LAST))
    - STEP=3 (follow 3 relations)
    - STEP=-1 (follow all)

Serialisations

- General remarks:
  - everything needs to be qualified!
- Need W3C **and** IVOA serialisations:
  - W3C: as defined in W3C ProvDM standards, for compatibility with the world
  - IVOA: same formats as W3C, + VOTable
    - more direct representation of the data model classes
    - use „voprov“ as prefix everywhere
- ActivityFlow in W3C
  - Several options tried:

Serialisations: ActivityFlow as W3C

- Use Bundle?
  - But bundles are entities, not activities
  - Content of bundle cannot be accessed/linked to directly
- D-PROV: use Plan & wasAssoc.With for workflows, no agents specified
  - But Plan is an *entity*, not activityFlow
- First approach:
  - Create *plan* for each *activityFlow*
  - Link *activities* to their *activityFlow* by linking with its *plan* via a *wasAssociatedWith*-relation without *agent*

Serialisations: ActivityFlow as W3C

- Final solution: wasInfluencedBy
  - = general relation between entities, activities or agents
  - Used, WasInformedBy, etc. are special cases of this
  - W3C does not define the special case „HadStep“, so we just use wasInfluencedBy instead

Tracking provenance

- Recursive functions:
  - find_entity
  - find_activity
- Recursive, because: do not know, how many steps to go backwards
- There may be loops (it's a graph, not a tree), so nodes may be visited more than once
- only backwards in time
- only „upwards“, i.e. follow to parents of hadStep/hadMember, but do not follow children

Open for discussions