Leibniz-Institut für Astrophysik Potsdam

AIP

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

German Astrophysical GAVO Virtual Observatory

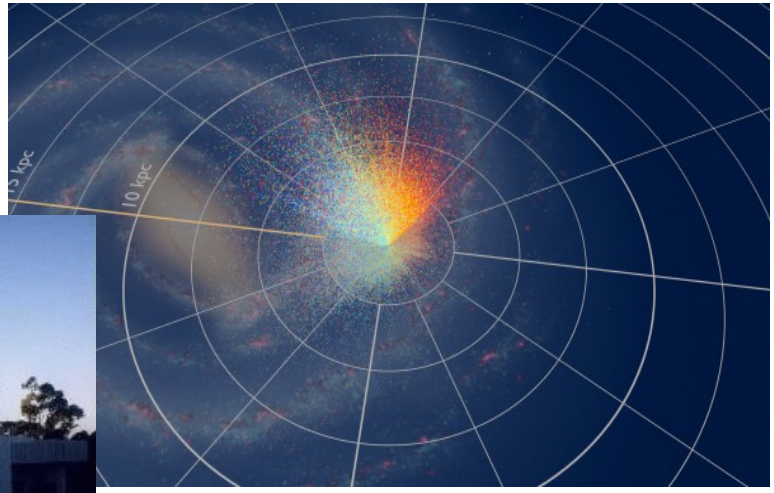
Provenance Webapp for RAVE

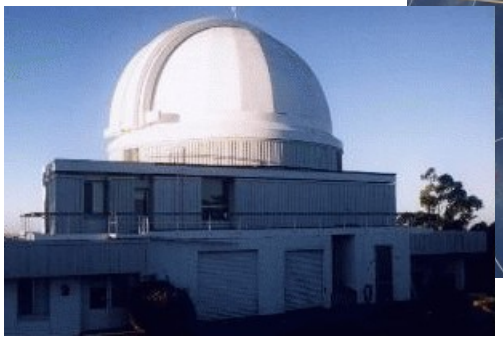
Provenance Days, Paris, 29th August 2018

Kristin Riebe

RAVE Survey - example use case

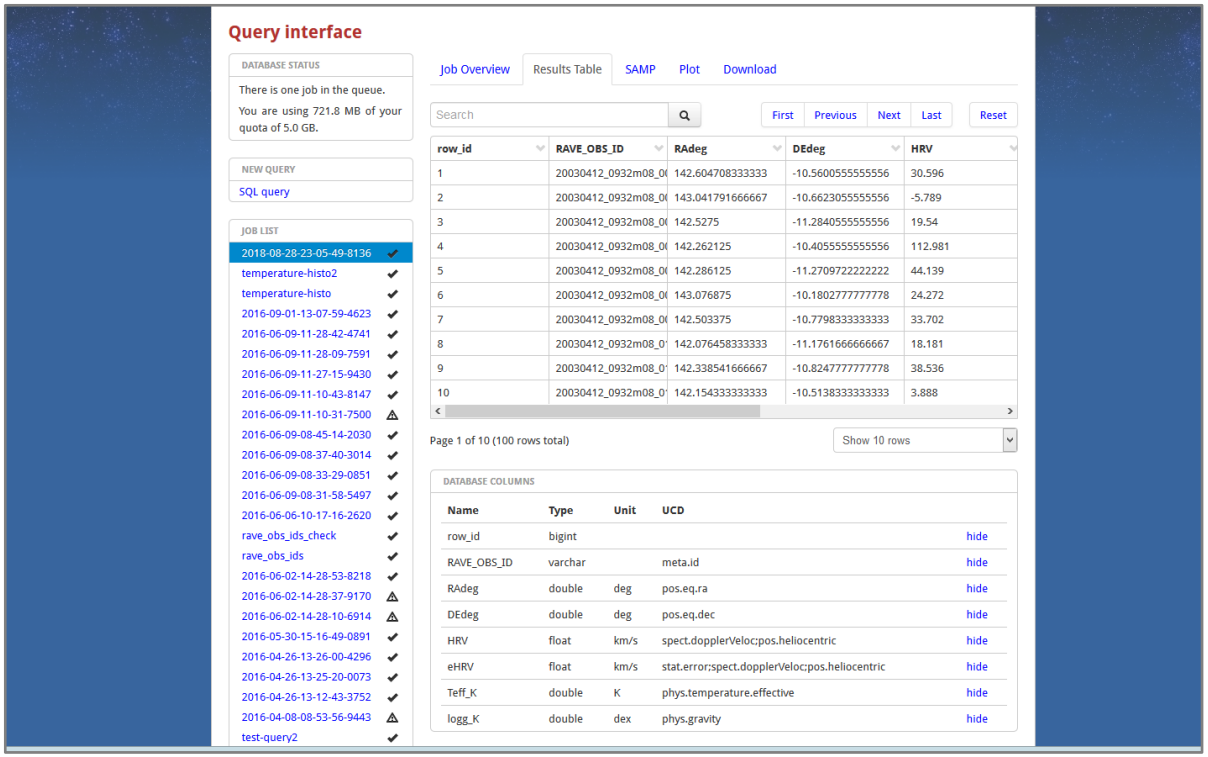
- Radial Velocity Experiment, ~ 500,000 spectra of stars observed at Anglo-Australian Observatory
- spectra processed with RAVE pipeline (workflow):
  - different calibration steps, combining and splitting files, generating radial velocities, stellar properties, cross-matching with other catalogues
- data release: mainly tables with stellar properties





RAVE Survey - example use case

- Example data table (from querying the RAVE database)



RAVE: Basic use cases

- Get overview of all processing steps
  - workflow/ActivityDescription
  - list of activities/activityDescriptions
- Graph representation (visualisation) of these steps
  - possible with ProvStore, if provided in W3C serialization format
- Provide detailed information for individual observations
  - given a unique identifier for an entry in the published RAVE DR5 table, return file names and locations of intermediate and raw files, how they are linked with each other

RAVE: Advanced use cases

- Who created the stellar_parameters-table?
  - i.e.: get the agent associated with this entity, thus: retrieve details for this entity
- Where do the values in column Teff_K come from? In which paper are the methods described? The uncertainties?
  - errors are in additional columns "e..."-something
- Are intermediate files (spectrum png/ascii) for a given obsId available? How could I get them?
  - Or: who do I need to ask for them?
  - Need: permission/accessibility flag, contact details
  -

RAVE: Advanced use cases (continued)

- How are values (for a given star) changing for each data release? What's the difference in processing?
  - First part can be answered with published data alone, provenance only needed for second question.
- Are there multiple observations of the same star?
  - If the derived heliocentric radial velocity differs more than the error bars suggest: what was causing this difference? (Which processing step(s)?)
- What is the coverage of this survey? Compare intended/actual coverage for studies of completeness/selection effects.
  - Needs additional information on failed fibers per field

Webapp for RAVE provenance

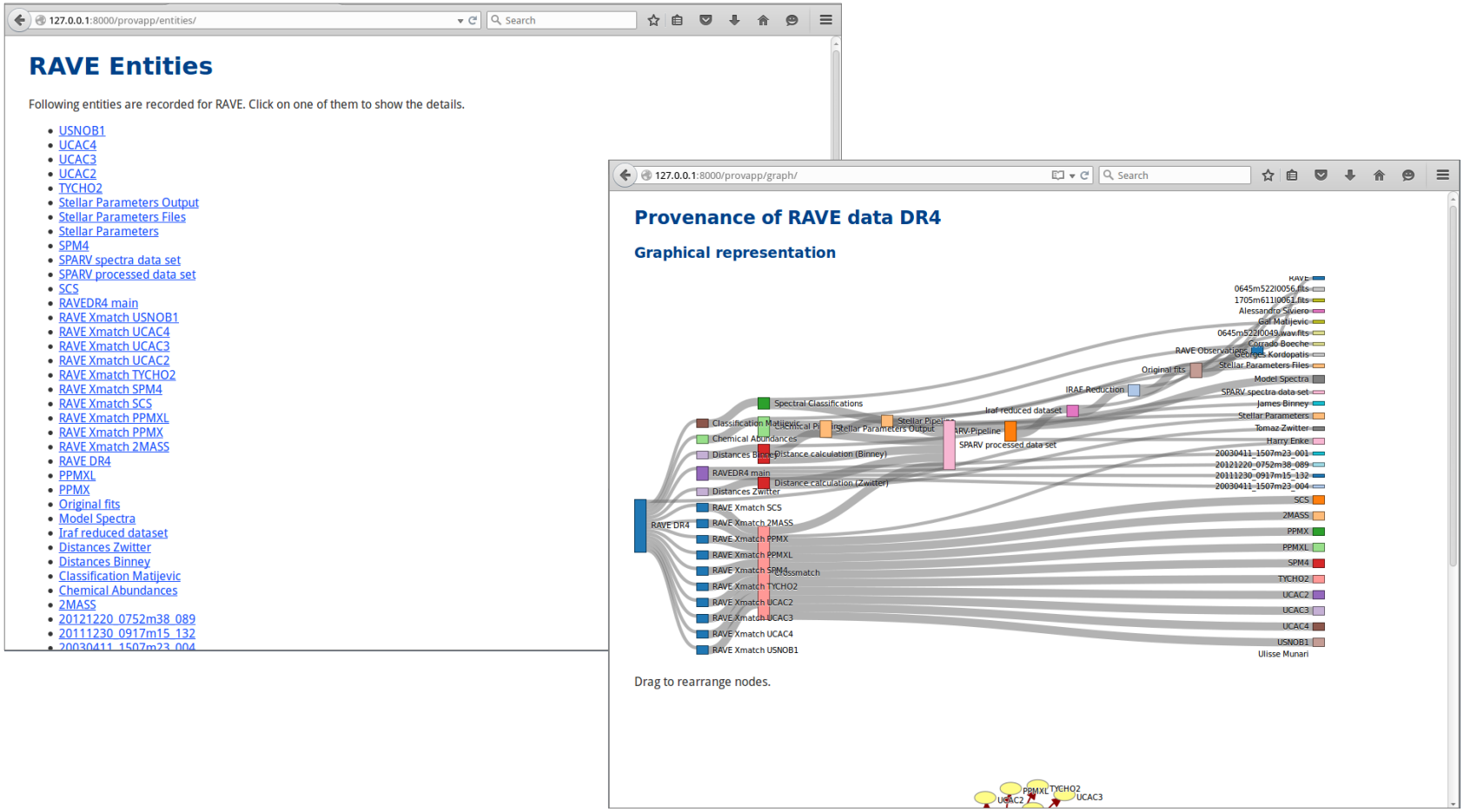
- Testing how to implement the data model
- Simple setup using Django Framework with SQlite3 database
- Define classes “as is”, main provenance classes, one DB table for each:
  - **entity**
  - **activity**
  - **agent**
  - used -- foreign keys to activity, entity
  - wasGeneratedBy -- foreign keys to entity, activity
  - wasAssociatedWith -- foreign keys to entity, agent
  - hadMember -- foreign keys to entities (one with type collection)
  - wasDerivedFrom -- foreign keys to entities

RAVE Provenance webapp

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

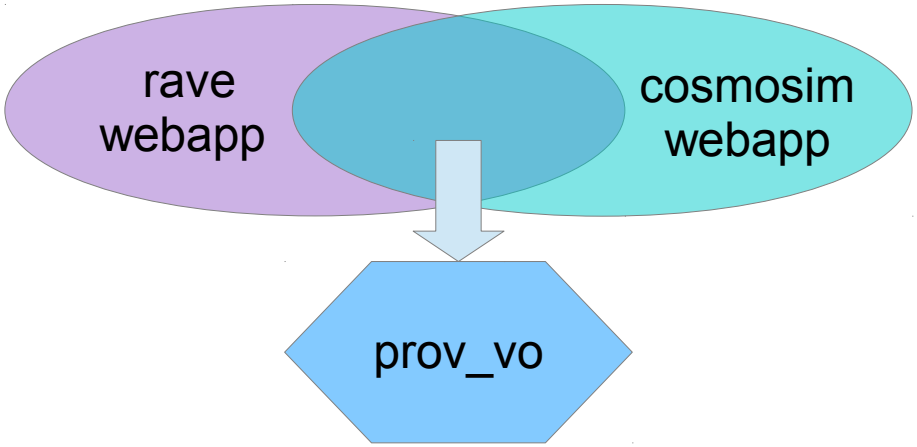
https://github.com/kristinriebe/provenance-rave
https://escience.aip.de/provenance-rave

Webapp for RAVE provenance



django-prov_vo

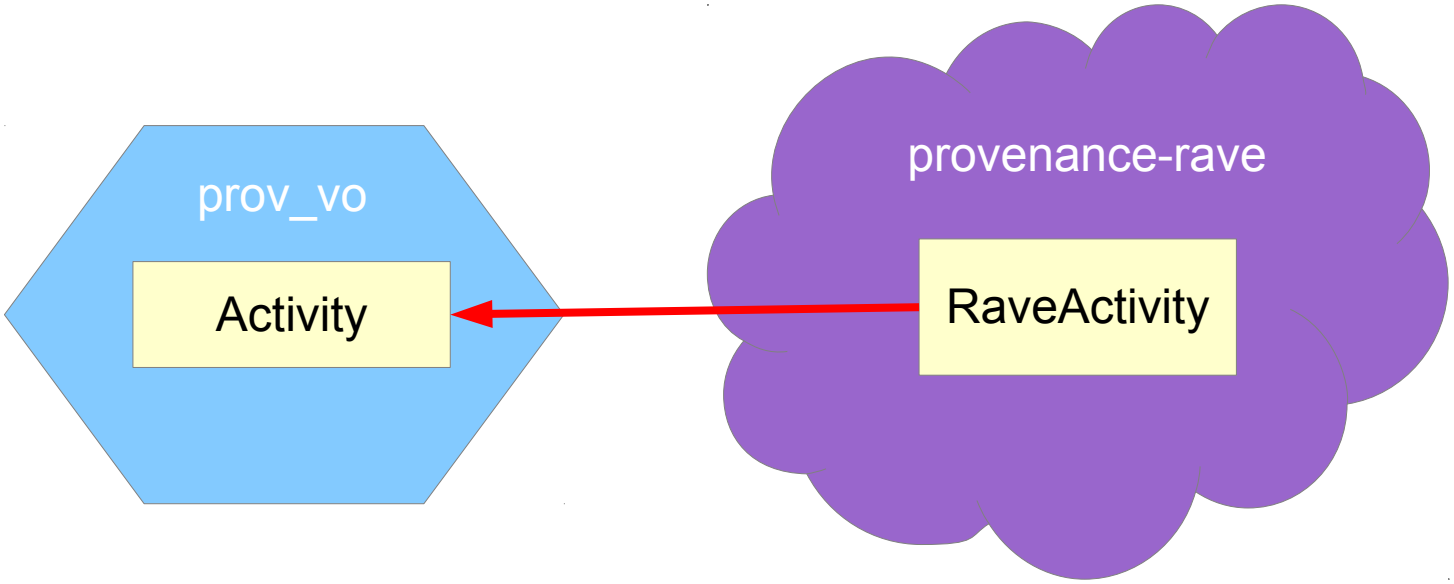
- Basic provenance implementation now (mostly) separated from RAVE-specific attributes etc.
- => reusable package „django-prov_vo“ (~ abstract classes)



- => 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



ProvSAP - definition

- Interface for retrieving serialized provenance description for a given entity/activity/agent ID
- GET request with main parameter “ID”
- **Parameters:**

required

- **ID** *(of entity, activity or agent, can occur multiple times)*
- **DEPTH** *(= 1,2,... or ALL)*
- **RESPONSEFORMAT** *(PROV-N, PROV-JSON, PROV-XML, PROV-VOTable)*

optional

- DIRECTION *(= BACK or FORTH)*
- MEMBERS *(include members of collections)*
- ~~STEPS *(include steps of activityFlows)*~~
- AGENT *(explore relations beyond agent)*
- MODEL *(= IVOA or W3C)*

DIRECTION affects only:

- Used
- WasGeneratedBy
- WasDerivedFrom
- WasInformedBy

AGENT – rename?

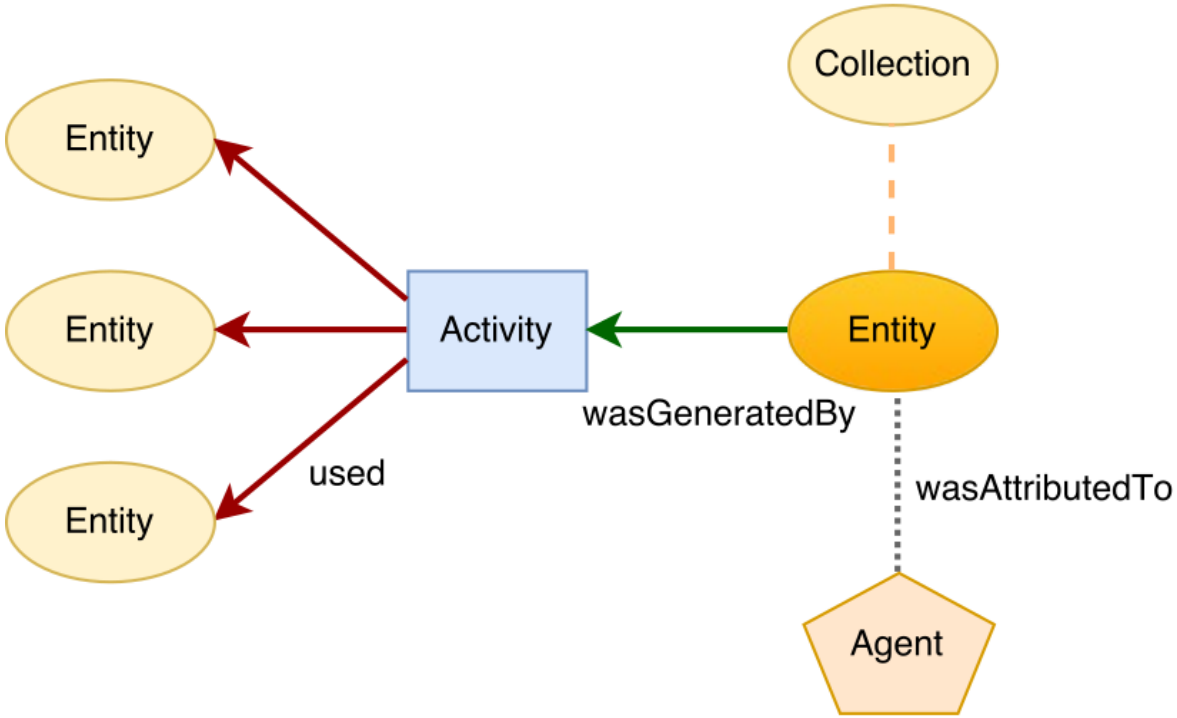
- EXPLORE_AGENT
- TRACK_AGENT
- AGENT=STOP, AGENT=EXPLORE

ProvSAP - Parameters

- **ID**
  - Identifier for an activity, entity or agent
- **RESPONSEFORMAT**
  - = format of the response
  - one of the W3C serialization formats (PROV-JSON, PROV-N, PROV-XML) or PROV-VOTable
- **DEPTH**
  - How much of the provenance graph shall be retrieved?
  - Everything (DEPTH=ALL) or just the most recent processing steps?
  - DEPTH=1: go exactly 1 “relation” backwards
  - DEPTH=ALL: services may also restrict to a max. depth instead (HTTP 302 redirect to DEPTH=<MAXDEPTH>)

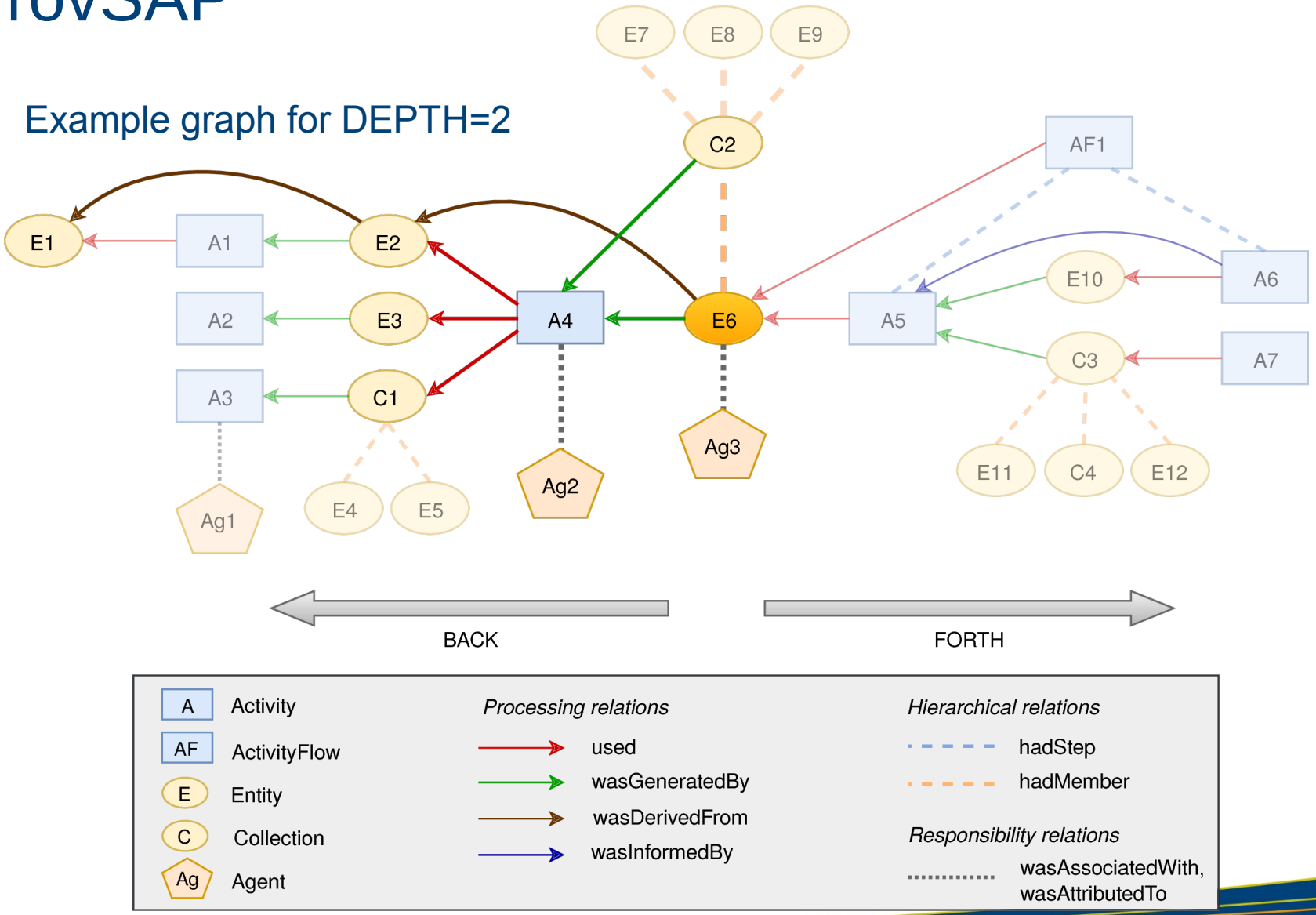
ProvSAP – Parameter DEPTH

- Example graph with DEPTH=2



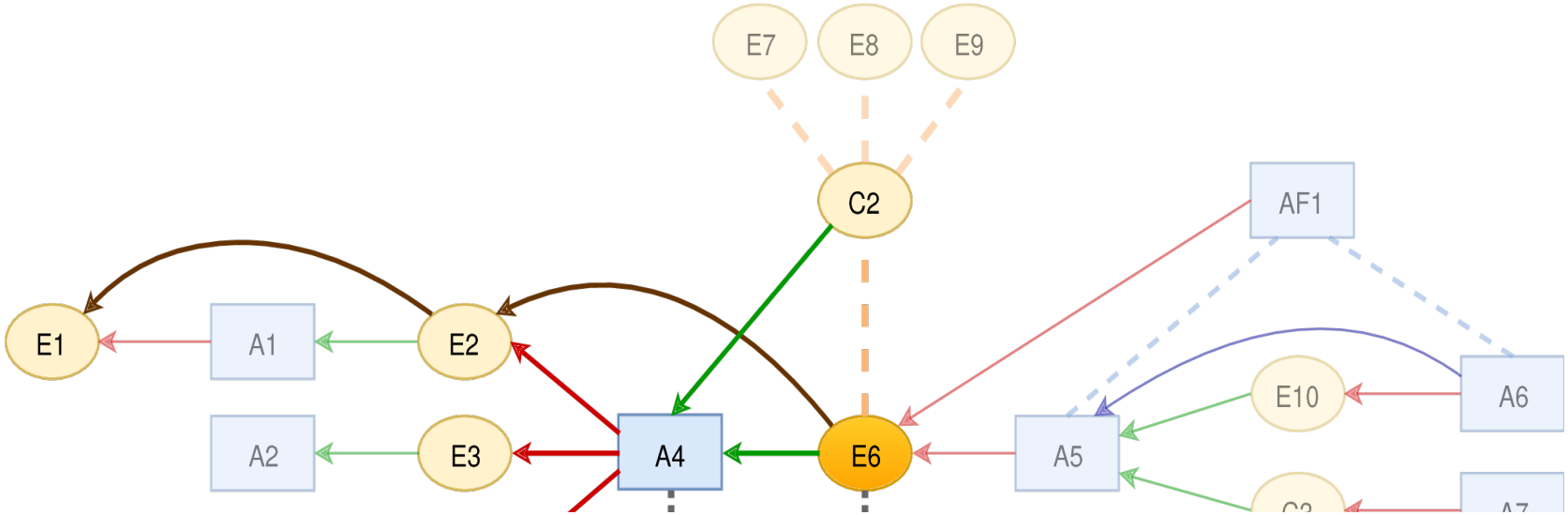
ProvSAP

- Example graph for DEPTH=2



ProvSAP

- Derivation, Information = short-cut relations
- Proposal:
  - redefine DEPTH=1: always go the next entity in the graph
    - (stop at agent and collections, if AGENT=false, MEMBERS=false)



ProvSAP parameters

- DIRECTION = BACK or FORTH
  - Allow to track provenance forward, i.e. find out which processes used an image, which outputs were produced etc.
  - Use cases
    - pipeline development
    - bug tracking
  - Only affects the processing relations:
    - Used
    - WasGeneratedBy
    - WasDerivedFrom
    - WasInformedBy
    - Because FORTH/BACK makes not much sense for e.g. hadMember or wasAttributedTo relations; thus the hierachical/responsibility relations are always tracked, independent of DIRECTION

ProvSAP parameters

- MEMBERS, STEPS = true/<u>false</u>
  - Collection groups entities together
    => hadMember relationship
- If tracking members of collections by default, a lot of data is returned
- => always follow the relations “up” (to the “container”), but only follow the “children”, if MEMBERS=true

ProvSAP parameters

- AGENT = true/<u>false</u>
  - Usually stop tracking when an agent is reached,
    but maybe want to know which other activities/entities an agent was involved with?
  - => allow tracking the agent further, using AGENT=true
- Discussion:
  - AGENT = false may be misleading
  - Better ideas?
    - EXPLORE_AGENT = true/false
    - TRACK_AGENT = true/false
    - AGENT = STOP/EXPLORE

ProvSAP parameters

- MODEL:
  - Allow to choose between IVOA and W3C serialization
  - IVOA:
    - directly map the classes to JSON, VOTable, …
    - => more direct representation of the data model classes
    - For exchange in the VO
    - To be used with VO tools, e.g. for loading into a ProvTAP service for further querying
  - W3C:
    - rename and restructure classes and attributes to produce W3C compatible serialization
    - For exchange with the world outside of the VO
    - For usage with W3C tools (e.g. ProvStore)

ProvSAP implementation

- Live version for RAVE:
  - https://escience.aip.de/provenance-rave
- Decoupled django-prov_vo package as reusable web app:
  - https://github.com/kristinriebe/django-prov_vo

    and an extra package for the VOSI resources (availability/capabilities):
  - https://github.com/kristinriebe/django-vosi

ProvSAP implementation

- Implemented all parameters from the draft
- Recursive tracking of the relations
- Each visited node of the provenance graph is returned only once (It's a graph, not a tree → loops possible!)
- Allows W3C compatible serialization (model=W3C)
- Formats: PROV-N or PROV-JSON

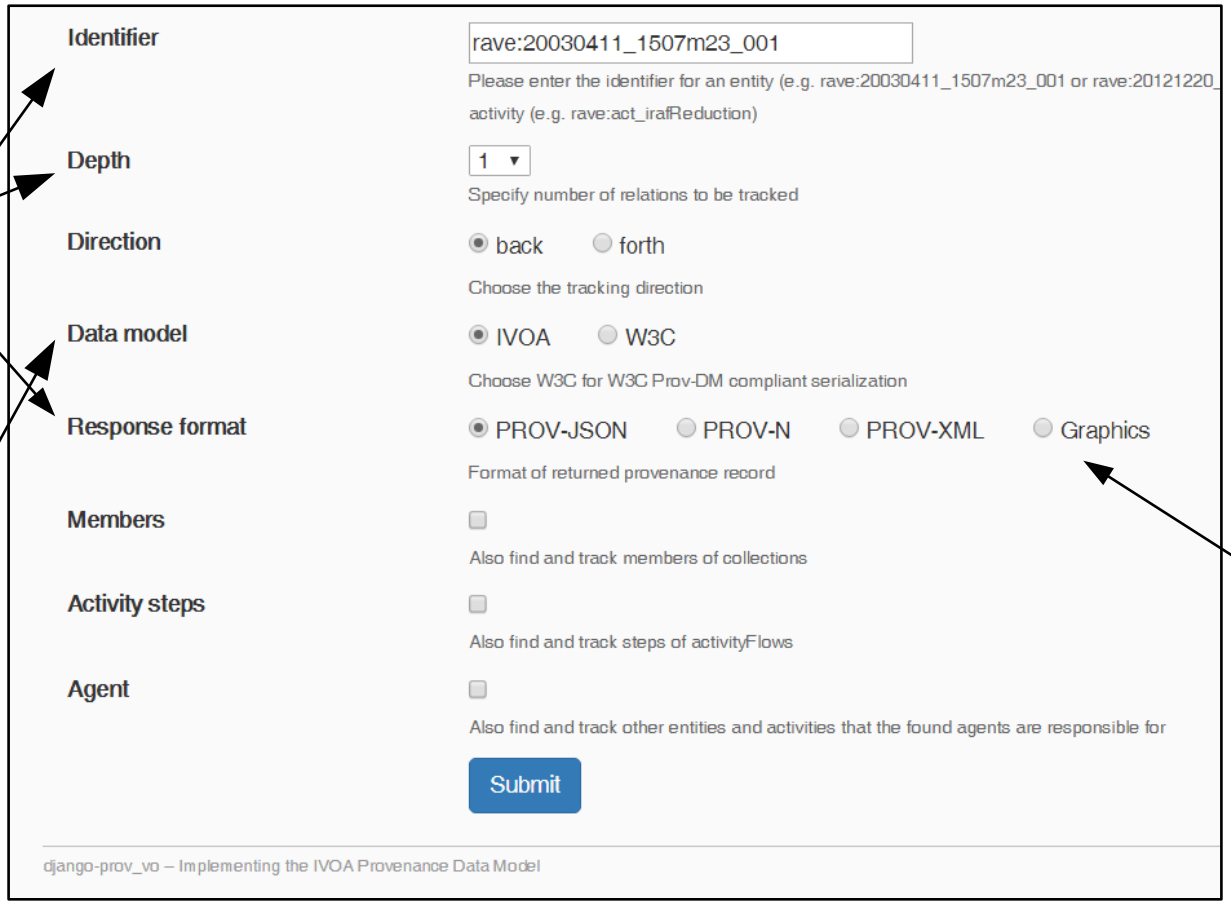
- Additionally:
  - Visualization of provenance (Javascript)
    - option RESPONSEFORMAT=GRAPH
  - Web form for nice user interface

ProvSAP webform (prev. called: ProvDAL)

mandatory parameters

new parameter

additional option



Automatically generates the ProvSAP GET request URL: https://escience.aip.de/provenance-rave/provapp/provdal/?ID=rave:20121220_0752m38_089&DEPTH=1&RESPONSEFORMAT=PROV-JSON&DIRECTION=BACK&MODEL=IVOA&MEMBERS=false&STEPS=false&AGENT=false

Summary

- RAVE survey of stellar spectra as use case for provenance
- Many example questions to be answered by provenance
- Simple prototype implementation of the W3C/VO data model is possible (but didn't try with a significant fraction of the data)
- ProvSAP implementation works in principle, some details need to be decided:
  - really allow querying for agent or activity as well?
  - always walk the graph until an end node or the next entity is reached?
  - keep both, IVOA and W3C serialisation?

Questions?