Towards a Provenance Data Model

Thoughts from GAVO

Kristin Riebe, AIP Potsdam

Florian Rothmaier, ZAH Heidelberg

IVOA Interop, ESAC Madrid, 18-23 May 2014

Outline

- Use cases for a Provenance Data Model
- Existing Provenance Data Models
- ProvDM's class diagram and description
- Usage of the VO-DML infrastructure
- Open issues

Use cases for a Provenance Data Model

- for a given data set, such a model should help to …
  - track the production history
    - **"Is the image from Catalogue A already calibrated?"**
    - **"Which pipeline version was used to produce this data set?"**
  - find the person(s) involved in the production
    - **"Who can provide details on the observation or the individual processing steps?"**

Use cases for a Provenance Data Model

- for a given data set, such a model should help to …
  - track the production history
    - **"Is the image from Catalogue A already calibrated?"**
    - **"Which pipeline version was used to produce this data set?"**
  - find the person(s) involved in the production
    - **"Who can provide details on the observation or the individual processing steps?"**
  - get aid in debugging
    - **"Where in the pipeline is the bug leading to an erroneous result?**
  - assess the "quality" of an observation/processing
    - **"Which ambient conditions were present during the observation?"**

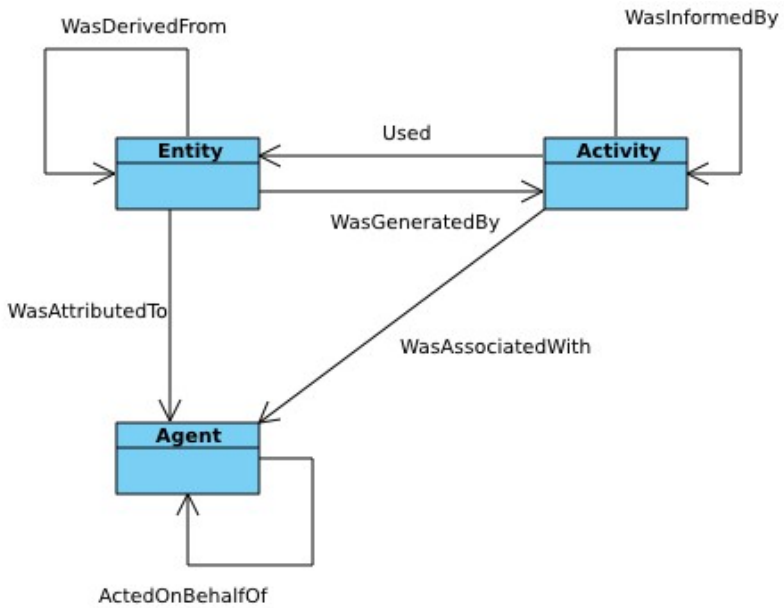
Existing Provenance Data Models

- existing models:
  - Open Provenance Model ("OPM"): L. Moreau (2010)
  - W3C Provenance Data Model: L. Moreau and P. Missier (2013)
- OPM can be considered the ancestor of the W3C model

➔ our starting point: studying the concepts of the W3C Provenance Data Model

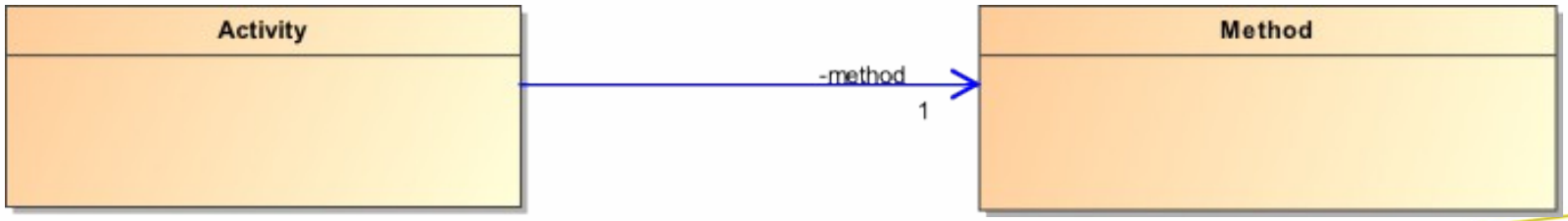
W3C Provenance Data Model

- W3C ProvDM:
  - defines three data types:
    - **"entity": a "thing", e.g. a physical or digital object**
    - **"activity": action upos or between entities**
    - **"agent": a role, e.g. taking over responsibility for an activity taking place**
  - specifies seven relations between the data types, e.g.
    - **Entity 1 "is generated by" Activity 1**
    - **Entity 2 "is attributed" to Agent 1**



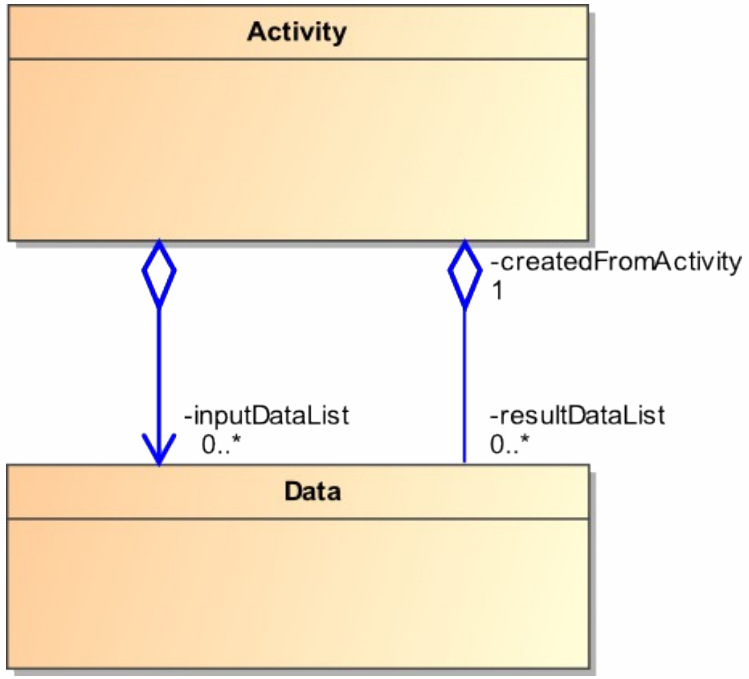
Towards a data model ...

- looking at **SimDM**:
  - includes provenance for simulation data
  - two part concept for main part:
    - **Experiment:** processing, simulation etc., execution of an experiment
    - **Protocol:** design of experiment, description, reusable prototype
- adopt same structure here, but replace terms
    - Experiment => "**Activity**" (W3C)
    - Protocol => "**Method**"
- each Activity has exactly one Method



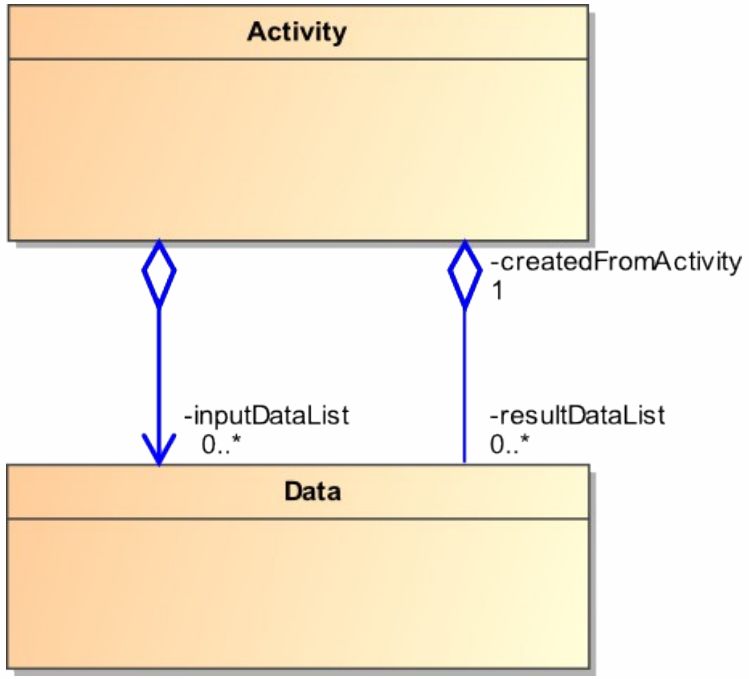
Relation Activity – Data

- links between data sets involve an activity in between
- activities have input/output data; provide links to them
- provide 'backward' links, from result to previous activity
- otherwise treat input/result data in exactly the same way



Relation Activity – Data

- links between data sets involve an activity in between
- activities have input/output data; provide links to them
- provide 'backward' links, from result to previous activity
- otherwise treat input/result data in exactly the same way



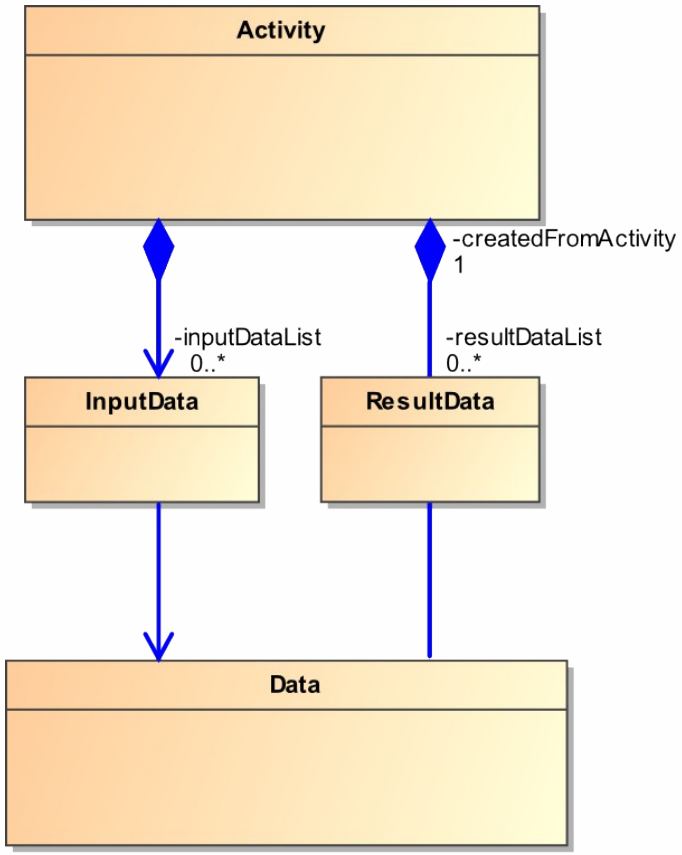
Problem:

need to assign roles to data,
e.g. sky subtraction: img1 – img2, need to identify raw image/parameters, bias frame, ...

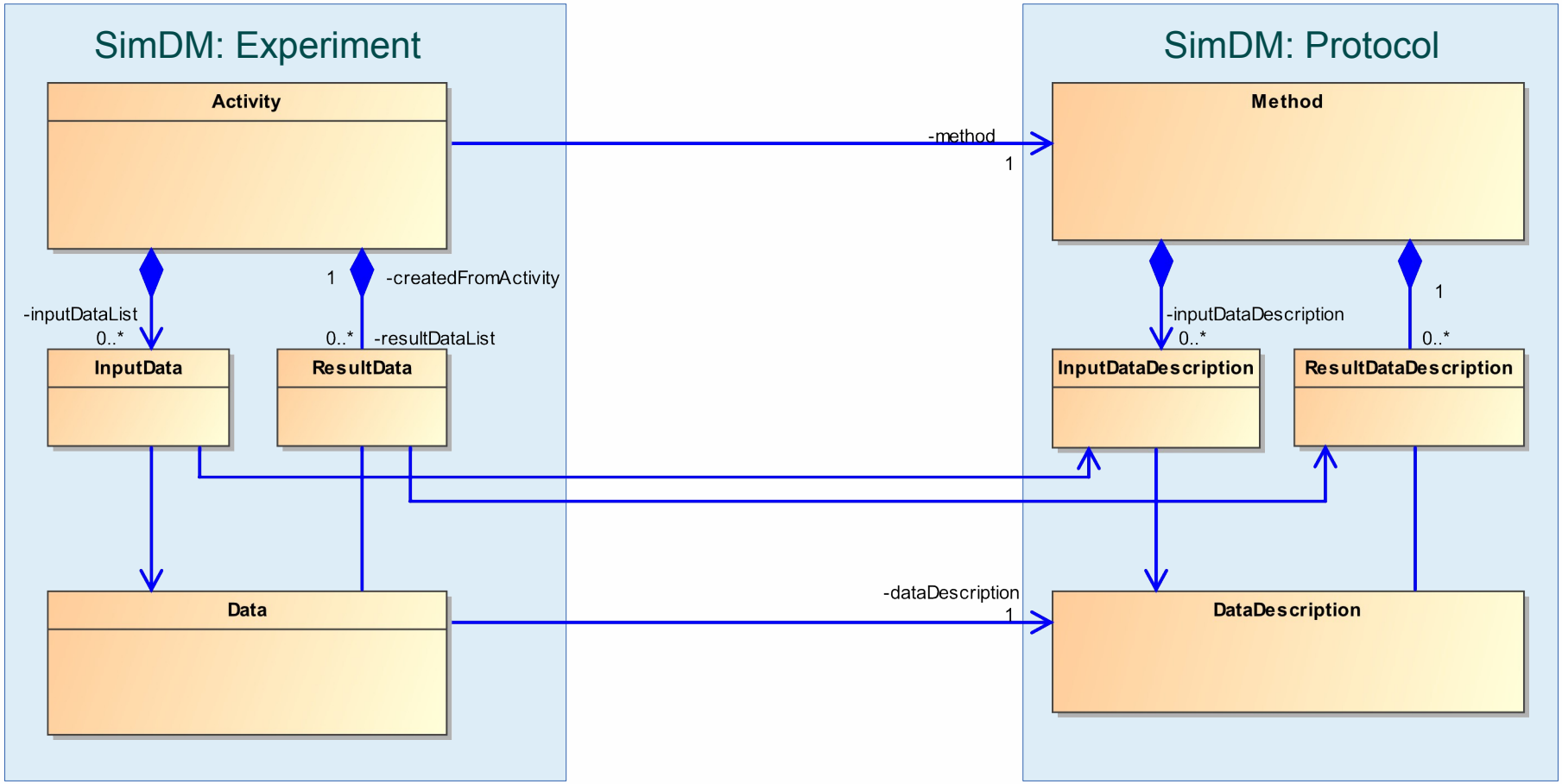
Relation Activity – Data

- links between data sets involve an activity in between
- activities have input/output data; provide links to them
- provide 'backward' links, from result to previous activity
- otherwise treat input/result data in exactly the same way

Insert additional classes to define roles of input and result.

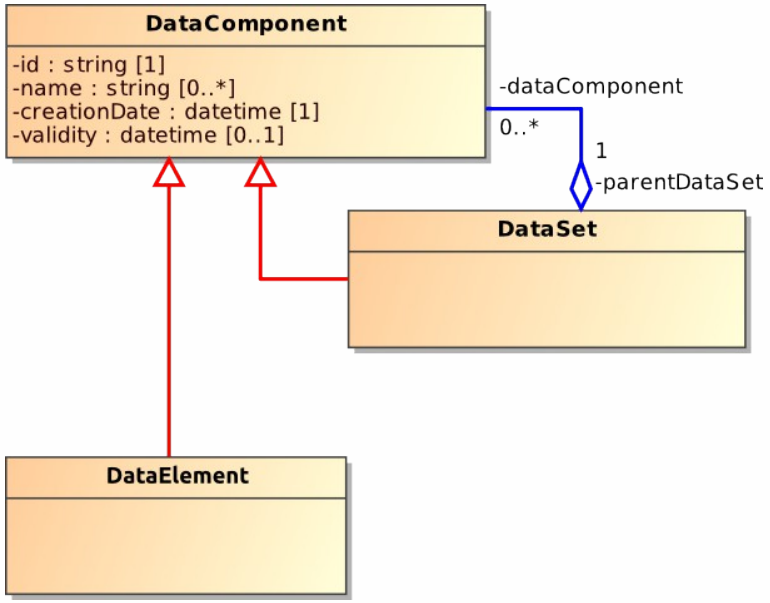


Activity/Data + descriptions



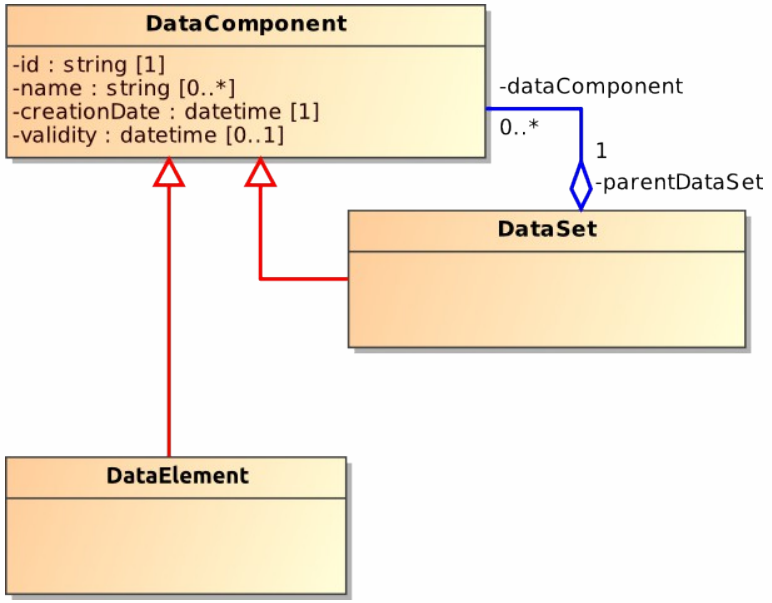
Data: composite pattern

- data can be grouped to DataSet, collection of data, e.g. RAVE DR4 tables, group all calibration data
- want to treat DataSet and DataElements the same way (same interface)
- Composite design pattern:
  - basic class: DataComponent
  - DataSet inherits from DataComponent
  - DataSet is a collection of DataComponents
  - DataComponents refer to the parentDataSet

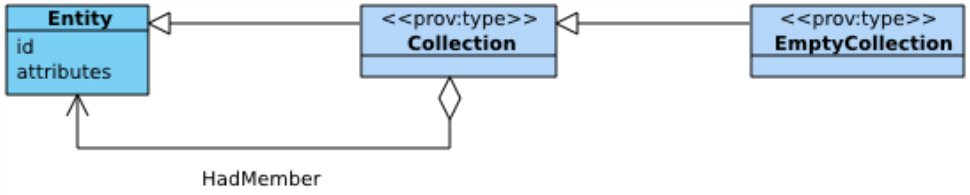


Data: composite pattern

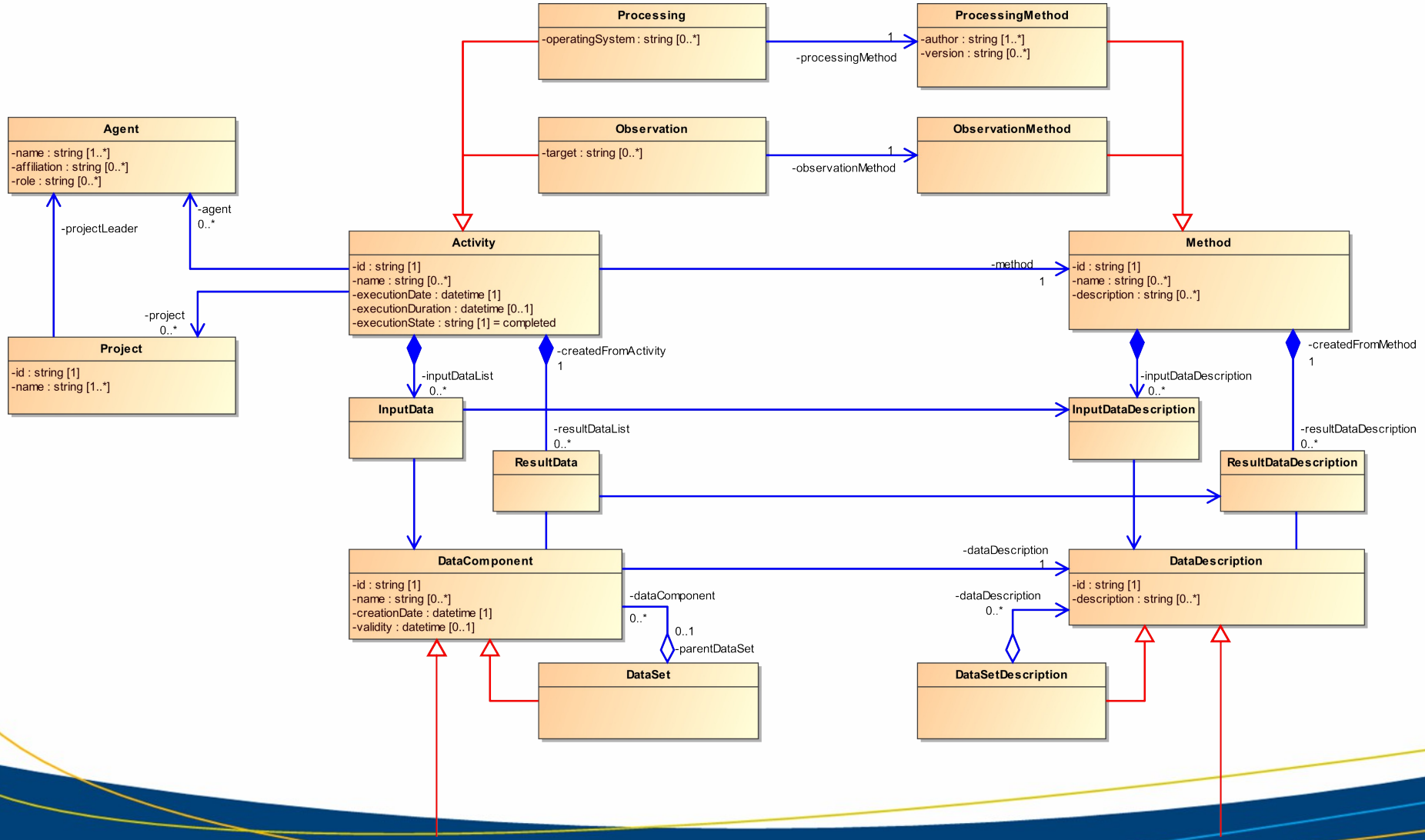
- data can be grouped to DataSet, collection of data, e.g. RAVE DR4 tables, group all calibration data
- want to treat DataSet and DataElements the same way (same interface)
- Composite design pattern:
  - basic class: DataComponent
  - DataSet inherits from DataComponent
  - DataSet is a collection of DataComponents
  - DataComponents refer to the parentDataSet



"Collections" in W3C

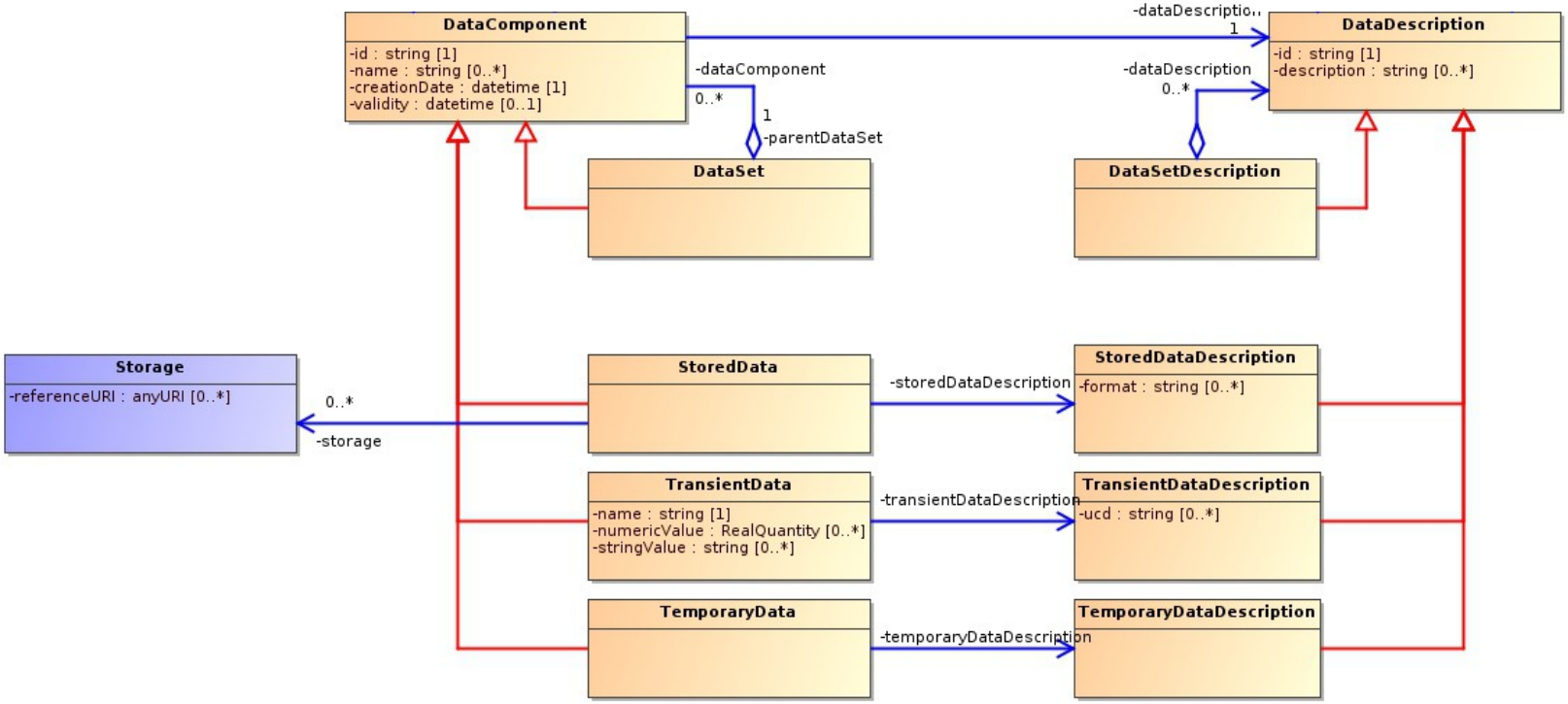


Class diagram



Data Subclasses

- stored: is stored somewhere, could in principle still be available, more permanent result data
- transient: command line parameter, user input
- temporary: created between two activities but not stored
  (file overwritten, data just existed in memory, …)
  => probably don't want a provenance record for those



Steps towards a VO-DML-compliant Data Model

- created an UML class diagram by using MagicDraw CE 12.1
- used Gerard's XSLT stylesheet to …
  - create a VO-DML document from the UML-XMI
  - generate HTML class documentation
- committed our code to volute, see http://volute.googlecode.com/svn/trunk/projects/dm/provenance

➔ translation of ProvDM's UML-XMI into VO-DML works nicely

➔ special treatment for aggregation would be needed

Outlook

- still many things to discuss:
  - allow hierarchical grouping of workflow?
    (e.g. reference from activity 'pipeline' to the individual steps)
  - tag 'main' input data item from which the result was derived?
- use general pattern for more concrete models for observation/processing
  - make ambient conditions, instrument characteristics more explicit
- refine & complete list of attributes per class
- define keywords for activity (=> see volute, activity_semantics.txt),
  keywords for ambient conditions, instrument characteristics
- link to other data models: ObsCore, ImageDM, DataLink, …
- check with real use cases