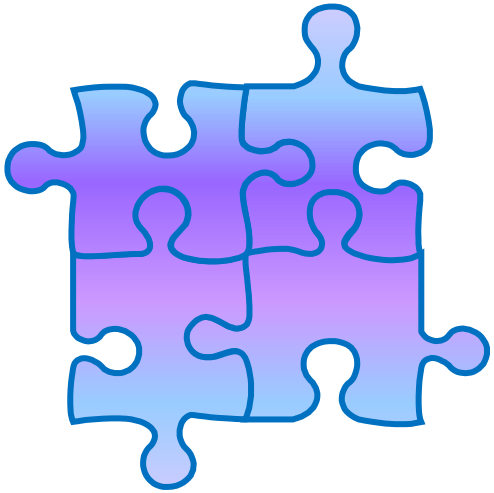


Utypes : Points to discuss

Data Model Working Group , M.Louys,
IVOA Interop Heidelberg, mai 2013

Application modeling or Utype definition

Where do we put the line ?

- Serialisation / deserialisation should not bother the archive end
  - Data models try to be very comprehensive : consider as many use-cases as possible
  - Archives offer a partial set of metadata depending of the mission, instrument, data products, etc.
- Applications evolve quicker than archive management – Apps are driven by evolving science questions
- Archives need to be stable for the long term and cannot completely and precisely anticipate and forsee the next applications paradigms
- The needs are different
- Clear and rich documentation is the common basis

How to derive a utype from UML

DeSerialisation needs

- Object Types (classes)
- Associations between classes too (collection, composition, inheritance (?)
- At the finest level, for classes attributes, we need :
- names , data type, and often unit and ucd

We agreed that we need a unique identifier for a piece of metadata and that we derive it from a UML data model representation

- logical path to a data model item
- Same UML diagram , two different path definitions
  - VO-DML defines a relative path , for classes and any level of nesting
  - Legacy utypes used root-to-leave path , for leaves

- The trade-off could be to have both co-exist ***in the spec***

Combining two levels of annotation

Legacy utype :

- Keep on FIELD and PARAM in VOTable serialisation
- They correspond to a long path traversing the DM graph

→ Simple to check for the data provider for short data model instances

→ Needed for a transition period

VODML utypes

- Defined for Groups → any hierarchy in the DM tree
- Contain the role of a group with respect to its parent level in a hierarchy
- Hooked to a FIELD by a FIELDref

→ Is their a consistency issue?

Utype as labels in 2012

- Up_to_now:
  - a utype is a label that tells where a metadata value can be located in an existing IVOA DM
  - It has a path-like structure
  - It goes from single value element to classes descriptions in a DM
- For deserialisation :
- Build-up classes instances from an IVOA data model and fill their attributes with the values stored in VOTable fields.
- Object types were defined in an XML schema attached to the IVOA REC (no explicit tag in VOTable serialisation))

Utype as labels in 2013

- VO-DML offers to describe any kind of VO DM in a machine readable format
- It needs labels for all data model items to express their nature:
  - Classes: a label for an object type name
  - Attributes: a label for the name
  - Collection, a label from an element to the container object.
  - Reference: a label for the link between two classes
  - Inheritance: a label for the derivation link (??)
- These are structure information , as embedded in the XMI format used internally by any UML modeler (CASE tool).
- This is different from the semantic flavored usage of Utypes defined previously.

The semantic role of Utype

- The semantic role of the former utype *specialised*
  - char: characterisation.spatialAxis.accuracy.staterror.value
- Is different from Accuracy.statError *generic*
  - which represents an object type, that can be used in a relation to any kind of measurement or Axis calibration
- How to interpret the Accuracy.StatError label in a data cube for instance ?

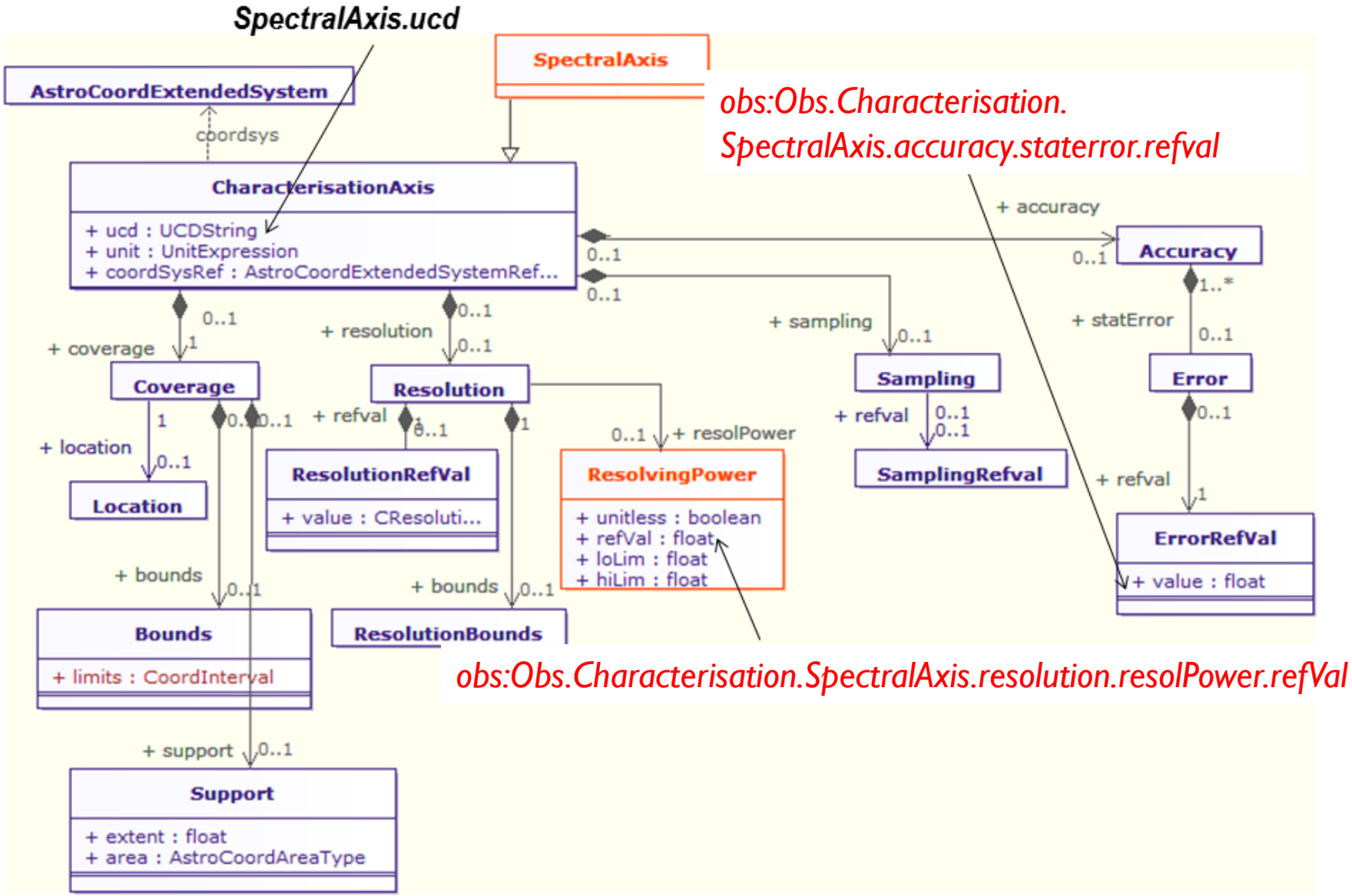
  Is it attached to a spectral, spatial, velocity, flux measurement?

  You need to interpret the group nesting to know the accuracy of what you are describing/using.

  Long strings or nested multi-level parsing ? Each « . » corresponds 'grosso modo' to GROUP

  → Same complexity

From Obscore DM



Need for well defined reusable blocks

- From experience , we can notice that most applications and protocols need some stable representations for current usage
- Coordinate system — STC
- Coordinates — STC
- Regions — STC
- Filter, Photometric calibration — PhotDM
- Data product identification — Dataset
- Access to linked data — Access
- Others?
  - These should be stabilised for all models

→ A dictionnary of common classes and their VOTable mapping with associated utypes.

→ The skeleton for re-usable libraries

Things to clarify

- Define a Vo-dml property for a GROUP

```
<GROUP utype="src:source.Source.position">
 <PARAM utype="vo-dml:Instance.type"
value="src:source.SkyCoordinate
"  name="datatype".../>
<GROUP>
```

Here utype is used for meta-information on data model definition:

This utype is part of the VODML translation mechanism of a data model instance in VOTable

Why not use a more specific dedicated annotation :

? otype as new attribute in VOTable GROUP

? <INFO

? <LINK ref='http://vodml/vodml_item#Instance.root'

Anything better ?

Issues to fix / things to clarify

- Support for data model item property : mandatory and optional
- Example Obstap
  - has mandatory datamodel elements
  - Need to be there in order to be compliant to Obs/TAP spec
  - Can also provide richer metadata descriptions with optional DM items
- A tag for mandatory status

Data model extension

Define new data model fields for a specific use-case

- If a data model does not cover sufficiently the needs of a specific service or data collection
- Define a new data model name (name space)
  - Define new classes by
    - derivation of existing classes
    - Addition of new classes
  - Provide documentation and utypes for the extended data model fields
- Could be VODML description
- Need some sort of IVOA validation to enter the interoperable ivoa domain and avoid redefinition