VIRTUAL ASTRONOMICAL OBSERVATORY

Utypes
Issues, Questions and Suggestions

Omar Laurino (SAO/VAO)

VIRTUAL ASTRONOMICAL OBSERVATORY

Utypes are the answer. What was the question, BTW?

Omar Laurino (SAO/VAO)

Some other interesting questions

- Do we care about interoperability?
- Do we care about requirements?
- Do we care about technological uptake?

Parsability

- To parse means to “resolve into its elements … and their relation to each other”
- Parsability is “the state or condition of being parsable”
- Parsable is anything that *is able to be parsed*

- Parsability *does not* require a reader to parse it, but can be convenient
  - For instance, an XML file is not parsed (according to its XML nature) by a simple text editor. An advanced XML editor offers tools for e.g. syntax highlighting and validation, by parsing the file according to the XML grammar.

– *Definitions from Wiktionary*

IVOA practices involving Utypes

- Spectrum 1.1 REC, Quality Flags: Data.FluxAxis.Quality.n, where n is an integer.
- Photometry 1.0 PR, Access class: “we use the Access class defined in ObsTAP and inherited from SSA” -> PhotometryFilter.transmissionCurve.Access.*
- Photometry 1.0 PR, Spectrum is *imported* using the **spec** namespace (notice the difference with the previous approach).
- Namespace (in several DMs): the namespace must be parsed out of the Utype string… but then again which is the actual Utype string?
- Extensibility (e.g. NED SED): Data.FluxAxis.Published.Value: is this Utype by any chance related to the standard Data.FluxAxis or to Target.Name? (How can I infer it?)

Utypes in current RECs (1/2)

- Introduced as an attribute for FIELD and PARAM in VOTable 1.2:
  - Maps FIELD/PARAM to a DM attribute
  - Encourages use of the XML namespace convention for avoiding name collisions
  - Encourages use of the XML xmlns for linking to the DM
  - Highlights the usefulness of utypes for space-time coordinates and provides an example for STC
  - Does not say anything about parsability
- Redefined in SSA 1.1:
  - The goal of utypes is to “flatten a hierarchical data model so that all fields are represented by fixed strings in a flat namespace”
  - They are introduced as “fixed” strings, but no explanation is given on the meaning of “fixed”.
  - “Of course, if a data model becomes complex enough this will no longer be possible”
  - Introduces a serialization mechanism for multiple instances (multiple equal Utypes in the same file), providing an example using serialization specific features, for VOTable.
  - Does not say anything explicit about parsability, however…
  - In others sections (e.g. query response metadata) other features are introduced:
    - Utype is built with the pseudo-grammar “<component-name>”.”<field-name>”
    - spec:Spectrum.Target.Name and ssa:Target.Name are the same thing.
  - More information about utypes in Section 4.2.7 (Metadata Extension Mechanism)

Utypes in current RECs (2/2)

- Redefined in Spectrum 1.1, also introducing Data Model inheritance:
  - Analogy with XPATH ('.' instead of '/'). "a.b.c.d", dots indicate "has-a" relationship (3.5)
  - 'Data Model Field' and 'Utype' interchangeable (3.5)
  - "Other IVOA standards may use a different prefix instead of "Spectrum." … This represents Data Model inheritance." (3.5)
  - "the utypes can be used to infer the data model structure" (8.2)
- Most DMs define utypes in tables, using different conventions
- Utypes strings can change when DMs are reused. Also, the namespace changes globally for each DM (spec:Target.Name, ssa:Target.Name)
- Utypes are only partially used in FITS serializations: they can be used for columns, not for parameters: in this case, an arbitrary 8 char string is provided by the DM document.
- DMs do not define an "xmlns" link to the DM URI

Utypes related issues

- Serialization of multiple instances of the same class
- Extensibility:
  - sedNed:Data.FluxAxis.Published.Value
  - ned:Data.FluxAxis.Value
- FITS arbitrary keywords (round-tripping)

- Inconsistencies between Utypes and XSD
- Clumsy UML

- Serialization of DM instances depends on both the format and the DM:
  - some can be serialized in VOTable but not FITS
  - some require specific FITS serializations

Some other interesting questions

- Do we care about interoperability?
- Do we care about requirements?
- Do we care about technological uptake?

Suggestions

- Abstraction of the Utypes definition process:
  - From XSD to Utypes
  - From UML to XSD, Utypes, etc…
  - Abstraction and standardized description of Data Models
- Parsability of Utypes for:
  - Encoding more instances of the same class (w/ interop):
    - Dynamic utypes and DM standard description
    - Static utypes with qualifiers
  - Reconstructing object's structure
  - Generic meta-programmed VO libraries and tools (generic I/O, importers, publishers)
- VOTable GROUPs (drop FITS support + interop)
- Add FITS aux table for mapping Utypes to keywords (keep FITS support and interoperability)

Conclusions

- Introduction of a standard for utypes *will* break something, given the current entropy
- Parsability?
- FITS support?
- Standardization of Data Models?
- Standardization of Serialization?
- Use Cases?
- Tiger Team?
- …
- …
- 2-sigma consensus in the VAO

Tiger Team

- TCG suggested to the Exec the appointment of a Tiger Team with the following mandate:
  - Collect use cases
  - Study current usage of Utypes:
    - Used in a meaningful way?
    - Impact of changes on existing standards
  - Develop a standard

Tiger Team

- TCG suggested to the Exec the appointment of a Tiger Team with the following mandate:
  - Collect use cases
  - Study current usage of Utypes:
    - Used in a meaningful way?
    - Impact of changes on existing standards
  - Develop a standard
  - ~~Wear a fancy tiger t-shirt~~

Think positive

- With few changes and a clear-cut solution for Utypes we can enable:
  - A standardized, versioned, straightforward Data Modeling framework (new Data Models are coming!)
  - A single basic VO library that works with the abstract framework, therefore with all DMs past, present and future
  - A single model-agnostic VO importer
  - A single model-agnostic VO publisher

Technological uptake: make it embarrassingly easy using VO DMs, by extending a simple, generic API

VIRTUAL ASTRONOMICAL OBSERVATORY

What was the question, by the way?

Omar Laurino (SAO/VAO)

Use Cases / Requirements

- Requirements:
  - "*we can't re-engineer the VO every year*"
  - save both the goat and the cabbages

Use Cases / Requirements

- Requirements:
  - "*we can't re-engineer the VO every year*"
  - save both the goat and the cabbages

Once upon a time a villager wanted to cross a river with a basket of cabbages, a wolf and a goat. His boat allowed him to carry only one item at a time. He couldn't leave the wolf alone with the goat, and the goat alone with the cabbage. How could he get across the river?

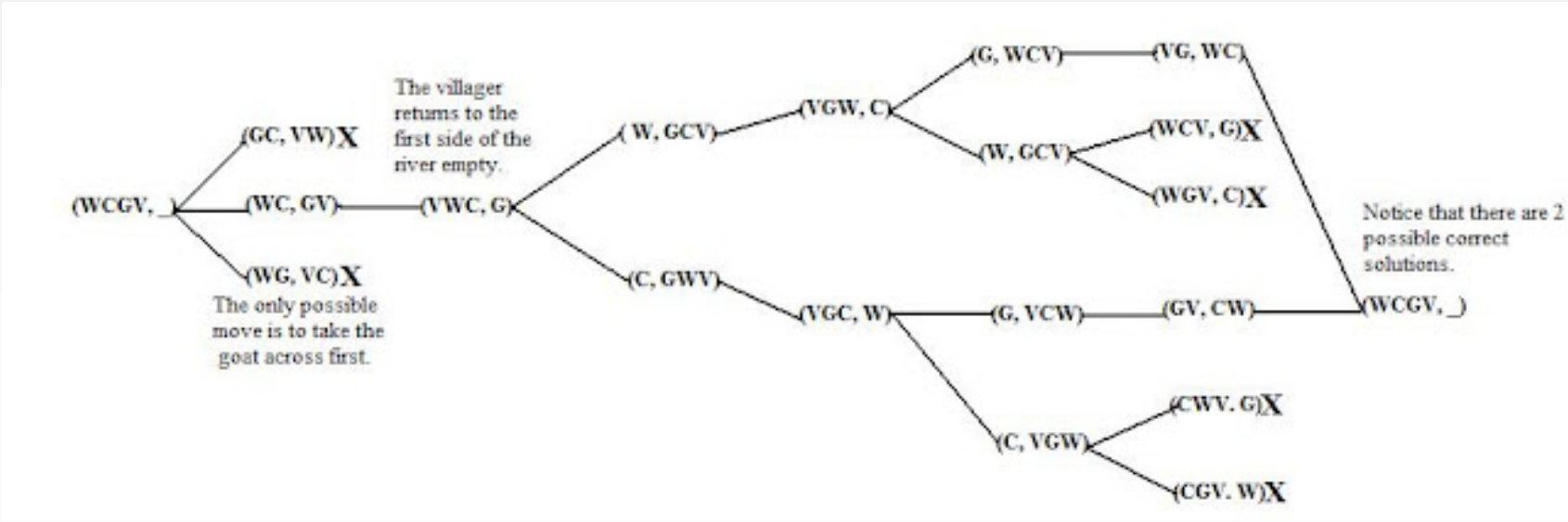
Alcuin of York, 8th century AD

Problem 18 in *Propositiones ad Acuendos Juvenes*

(Problems to Sharpen the Young)

Use Cases / Requirements

- Requirements:
  - "*we can't re-engineer the VO every year*"
  - save both the goat and the cabbages



Use Cases / Requirements

- Requirements:
  - "*we can't re-engineer the VO every year*"
  - save both the goat and the cabbages

- You can replace the goat and cabbage with any tantalizing metaphor you like

- Please, set aside any biases. This discussion is a constructive effort aimed to enable useful features and to solve pernicious issues

Use Cases / Requirements

- UC #1. Serialize DM instances into a file
- UC #2. Deserialize a DM instance from a file

- UC #3. Embed STC information into VOTables
- UC #3.1. Embed STC information in FITS

- UC #4. Provide an abstract (de)serialization strategy that can work with any expressive enough file format. A client can instantiate an object equivalent to the object that was originally serialized
- UC #4.1 Trivial roundtripping

Use Cases / Requirements

- UC #5. Link columns in a relational model of the registry to VOResource schema elements
- UC #6. Tag metadata in a DAL query response
- UC #7. Render datasets/archives VO-compliant
- UC #8. Extensions of standard DMs

Use Cases / Requirements

- UC #9. Support serialization of multiple instances of the same DM class
- UC #10. Standard, machine readable DM description
- UC #10.1 Versioning of DM descriptions
- UC #10.2 DM descriptions should express relationships between DMs (reuse, extensions)
- UC #11. Documentation of DM fields
- UC #12. Query archives by DM attribute (e.g. by observation's target name)

Use Cases / Requirements

- A service must be able to tag metadata with a fixed string which uniquely identifies a field of a data model
- UTYPEs are used as simple strings that could be matched against, in a case insensitive way.
- UTYPEs are unique within the context of the specified data model
- New DM efforts should not re-invent concepts/ UTYPEs that have already been described and prescribed in other DMs. It must be easily possible to reuse existing UTYPEs.