VIRTUAL ASTRONOMICAL OBSERVATORY

Utypes evolution and extension

Omar Laurino, SAO

What are Utypes?

- Map a piece of information (metadata) in a Data Model instance's serialization to an element in the DM definition:
  - transform unstructured information (table) in a structured domain object
  - Apply context to data ("this VOTable is a Spectrum")

UCDs vs Utypes

- Context vs Universality:
  - UCD: unstructured semantic tag
  - Utype: structured semantic tag
- Examples:
  - pos.eq.ra tags a coordinate: doesn't depend on the context.
  - Spectrum.Target.pos: attribute of the Target object in the Spectrum Data Model.

Use cases (1/3)

- Current use cases:
  - Build a domain object from a Data Table according to a DM specification
  - Query by Utypes
- Limitations:
  - Very specialized, hardly reusable libraries
  - Slow technological uptake

Utypes reflect the way in which DMs are defined, and DMs are tightly coupled (i.e. copied & pasted).

Use cases (2/3)

What is a VO-Aware application?

- Support for VOTables
- Support for a specific DM (e.g. the SED tool)
- SAMP
- VOSpace
- UCDs

An example:

- Is Sherpa VO-Aware? Well, it can't be:
  - Iris implements an adapter for the SED use case
  - For each use case a new adapter is needed

Use cases (3/3)

- Possible use cases no currently achievable:
  - A generic plotting program aggregates values, uncertainties (systematic and statistical), units and plot them
  - Most KDD use cases
  - A fitting program recognizes axes information and asks the user which axis to consider dependent, which independent. Results have units and uncertainties according to standard specs (e.g. CharDM)

A new breed of VO-Aware applications: e.g. VOSherpa

Utypes and DMs

- Utypes can become the key to:
  - Apply different contexts to the same data table
  - Apply the same context to tables from different services

What is needed:

- Standardize the way Utypes are defined and extended
- Redefine DMs as reusable components
- Abstract the DM definition process (a metaDM!)

VO-URP uses many of these concepts for the SimDM

Utypes and metaDM

- The Utypes document could be the leading part of an abstraction of the DM creation process
- Analysis must be requirements driven, not opinion driven (validate technical solution against empowered use cases)
- Object Oriented requirements:
  - Abstraction (the implementation/serialization is hidden)
  - Inheritance (can represent B as an extension of A)
  - Polymorphism (if B extends A, client can “see” B as an A instance => Dataset views)
  - Modularity (small building blocks enable flexible mash-ups)
- Example: see a generic TAP response as a combination of different DMs. Combine responses from different services, seamlessly

Summary

- Utypes can be merely a way of mapping unstructured tables to domain objects
- By standardizing their creation in an abstract metaDM process, Utypes and DMs can be more powerful and open the door to seamless, powerful use cases (Dataset mash-ups)
- This implies an abstraction of the DMs themselves (e.g. no reference to serialization, map to Utypes instead)
- We can discuss the technical options, as long as we validate them against use cases and reqs