Fig. 1

Fig. 2

1. Tackling utypes

(vgl. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(vgl. Fig. 2)

Thoughts on what should and should not be in a utype REC.

- Data models
- utype syntax
- utype semantics
- VOTable embedding
- Appendices

2. Data Models

Disclaimer: I'm parsecs from being a DM expert.

What's a data model? The current spec mentions UML, but that's a trife vague (class diagram?). I'd say:

Abstractly, DMs are tree-shaped entity-relationship graphs (i.e., there's a root, they are acyclic, and there's a unique parent).

Operationally, current VO practice has been to give XSDs, which has the advantage of giving serialization rules for values.

But other formalisms are fine, too.

Or do we really want to standardize on XMI?

3. utype Syntax

Let's *not* have "packages".

I frankly don't understand much of the distinction made between class, attribute, reference, and collection names in the current WD. I can see we'd like to have some sort of native referencing mechanism, but I think that would clutter the spec and we should rather rely on whatever the container format gives us.

Thus, I propose:

utype ::=*modelName* `":"` *rootName* { `"."` *itemName* }

I'm not convinced *modelName* is terribly useful, but it's too late to get rid of it now.

The item names would correspond to the names of the nodes within the ER diagram traversed when going from the root node to the destination node, including start and end.

Since ER diagrams for utype generation are tree-like, the path being traversed is unique.

I'd also propose to include the algorithm given in the STC note (sans the mess on substition groups) in case the data model is represented in XSD.

It's ok to have utypes ending in tree-internal nodes.

If we could start again, I'd plead for case-sensitiveness (there's a reason no modern programming language has case-insensitive identifiers), but there's no point arguing here any more.

4. Model Reuse

If you need to do it, then just pull in (sub-) trees into the ER model or the main data model.

However, what I'd like best would be something like:

```
<GROUP utype="char:SpatialAxis">
  <GROUP utype=char:SpatialAxis.Coverage.location.coord"
    ref="cl"/>
</GROUP>
<GROUP ID="cl">
  <FIELDRef utype="stc:Position2D.Value2D.C1"
    ref="ra"/>
</GROUP>
```

(but ok, that only works in VOTable). The big advantage of such an embedding would be that clients that know STC can at least understand the STC part without having to parse utypes. That's useful because that would let them at least gather, say, RA, dec, and their errors together even if they wouldn't know it's a coverage since they don't understand the char utypes.

If reuse requires parsing utypes, then let's not use reuse.

5. utype Semantics

Values should always be strings. Serialization is defined by the models.

Each block of utype-value pairs should come with a data model declaration like `modelName:DataModel.URI`. Do *not* use the xml namespace declaration for that. XML namespace prefixes are not meant to be fixed, and we break XML by requiring a given prefix to point to some URI. This has already bitten us in SimpleDALRegExt.

The value of DataModel.URI could be the URL of an XSD file or an IVORN with a RR that contains utypes in StandardKeyEnumerations. Or still something else.

The REC probably shouldn't say anything about documentation and publication; that's up to the DM producers.

6. Using utypes

In VOTable, I'm sure we should have the PARAM/FIELDref mechanism proposed in STC-in-VOTable, one group per DM instance:

```
<GROUP utype="stc:CatalogEntryLocation">
   <PARAM datatype="char" arraysize="*" value="ICRS"
    utype="stc:AstroCoordSystem.SpaceFrame.CoordRefFrame"
    name="CoordRefFrame"/>
  <FIELDref utype="stc:AstroCoords.Position2D.Error2.C1"
    ref="raErr"/>
</GROUP>
<GROUP utype="stc:CatalogEntryLocation">
  <PARAM datatype="char" arraysize="*" value="GALACTIC_II"
    utype="stc:AstroCoordSystem.SpaceFrame.CoordRefFrame"
    name="CoordRefFrame"/>
  <FIELDref utype="stc:AstroCoords.Position2D.Value2.C2" ref="glat"/>
</GROUP>
```

7. utypes and FITS

I think “standard” shortenings of utypes won't work generically.

Let's rather define that each HDU requiring utypes can have a header element UTYPEEXT giving the FITS extension of a mapping header name to utypes.

8. Appendices

In the future, each DM REC should contain a chapter on utypes generated from it. For the existing DMs (STC, Char, Spectral, possibly Obs, Sim, Photometry), there could/should be appendices to the utypes document. These could list the legal utypes (where that's compact enough), stipulate special rules (e.g., substitution group handling), etc.

I *might* volunteer to play editor for the utypes document if you all don't disagree *too* strongly with me. . .