1. Time Series, VO-DML, DaCHS

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

- DaCHS Annotation...
- ... for time series...
- ... and how it ends up in VOTable.

(cf. Fig. 3)

2. DaCHS Annotation I

DaCHS is a general VO publishing framework.

Each resource is described using a resource descriptor (RD) containing, among lots of other stuff, table metadata, e.g.,

```xml
<table id="instance">
    <column name="hjd" type="double precision" unit="d" ucd="time.epoch" tablehead="Time" description="Time this photometry corresponds to.">
    <column name="df" type="double precision" unit="adu" ucd="phot.flux" tablehead="Diff. Flux" description="Difference flux as defined by 2008MNRAS.386.L..77B."">
    <column name="e_df" unit="adu" ucd="stat.error;phot.flux" tablehead="Err. DF" description="Error in difference flux."
</table>
```

From this, a human can work out that it's a time series with one value and its error. A machine could perhaps based on UCDs, but we want a less ambiguous and more explicit annotation.

3. Prior Art

Before VO-DML, DaCHS understood one DM: STC. Annotation used a slight variant of STC-S:

```xml
<table id="sample">
    <stc>
        Time UTC BARYCENTER "t_0" Position ICRS Epoch J2000.0 "raj2000" "dej2000"
    </stc>
</table>
```

The compact, text-based annotation with informal name-using referencing worked reasonably well.

Plan: Move that into the VO-DML age.
4. DaCHS Annotation II

<dm type="ivoa:Measurement">
  <value>@dfstatError: @e_df</value>
</dm>

<dm type="stc2:Coords">
  <time>
    <frame>
      <timescale>UTC</timescale>
      <refPosition>BARYCENTER</refPosition>
      <kind>JD</kind>
      <loc>@hjd</loc>
    </frame>
  </time>
  <space>[...]</space>
</dm>

<dm type="ndcube:Cube">
  <independent_axes>@hjd</independent_axes>
  <dependent_axes>@df @mag</dependent_axes>
</dm>

Independent, task-specific annotations.
Ad-hoc annotation language specific to DaCHS rather than XML on input (though DaCHS RDs are XML otherwise; DM annotation just explodes if you do that, and that’s bad for something routinely written and reviewed by humans).

Literals, sequences, and references.
This particular annotation not backed up by VO-DML. Once that’s there, validation is possible.
The full annotation can be obtained as part of the embedding RD.

5. Simplified VOTable Mapping

This is an excerpt of the resulting VOTable’s annotation.

```
<INSTANCE dmtype="ivoa:Measurement">
  <ATTRIBUTE dmrole="statError">
    <COLUMN ref="e_df"/>
  </ATTRIBUTE>
  <ATTRIBUTE dmrole="value">
    <COLUMN ref="df"/>
  </ATTRIBUTE>
</INSTANCE>

<INSTANCE dmtype="stc2:Coords">
  <ATTRIBUTE dmrole="space">
    <INSTANCE dmtype="stc2:Coord">
      <ATTRIBUTE dmrole="loc">
        <INSTANCE dmtype="stc2:SphericalPoint">
          <ATTRIBUTE dmrole="latitude">@dfstatError: @e_df</ATTRIBUTE>
        </instance>
      </ATTRIBUTE>
    </INSTANCE>
  </ATTRIBUTE>
</INSTANCE>
```

6. Proposed NDCube Processing

1. client parses STC annotations by looking for `stc2:Coords`-typed annotation.
2. client looks for `ndcube:Cube`-typed annotation. Here, there’s just one independent axis, `hjd`, so we have a 1-D dataset.
3. client inspects existing STC annotation of `hjd`. It’s a temporal coordinate, hence we have a time series.
4. client pulls the set of dependent axes from `ndcube:Cube` annotation. Perhaps let the user choose which one to plot?
5. plotting component looks for `ivoa:Measurement`-typed annotation of `df` to work out what to use as error in the plot.

7. Conclusion

Annotation isn’t rocket science …… but there’s lots of little engineering problems. Let’s try and work them out.