Fig. 1

Fig. 2

Fig. 3

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.,

```
<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.386L..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:

```
<table id="sample">
  <stc>
    Time UTC BARYCENTER "t_0"
    Position ICRS Epoch J2000.0 "raj2000" "dej2000"
  </stc>
  <column name="raj2000" type="double precision" ...
  <column name="dej2000" type="double precision" ...
  <column name="t_0" type="double precision"
</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>
  (ivoa:Measurement) {
    value: @df
    statError: @e_df} </dm>

<dm>
  (stc2:Coords) {
    time: (stc2:Coord) {
      frame:
        (stc2:TimeFrame) {
          timescale: UTC
          refPosition: BARYCENTER
          kind: JD }
      loc: @hjd }
    space: [...]}</dm>

<dm>
  (ndcube:Cube) {
    independent_axes: [@hjd]
    dependent_axes: [@df @mag]}</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[1].

5. Simplified VOTable Mapping

This is an excerpt of the resulting VOTable[2]'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">
            <CONSTANT ref="ndstsswpdpsa"/></ATTRIBUTE>
        ...
      <ATTRIBUTE dmrole="frame">
        <INSTANCE dmtype="stc2:SpaceFrame">
          <ATTRIBUTE dmrole="epoch">
            <LITERAL dmtype="ivoa:string">J2000.0</LITERAL></ATTRIBUTE>
          <ATTRIBUTE dmrole="orientation">
            <LITERAL dmtype="ivoa:string">ICRS</LITERAL></ATTRIBUTE>
```

Modifications vs. plans as gleaned from Mark's draft serialisations:

[1] http://svn.ari.uni-heidelberg.de/svn/gavo/hdinputs/k2c9vst/q.rd

[2] http://dc.zah.uni-heidelberg.de/getproduct/k2c9vst/data/OGLE-2016-BLG-0126_VST_r_SDSS68.t

- No GLOBALs (actually, could do without an explicit TEMPLATE element, too).
- Attributes are always ATTRIBUTE independent of model details (no COMPOSITION or so)
- actually, I'd much prefer if there'd just be a single kind of reference rather than COLUMN, CONSTANT and whatnot – a client will see what it is right after referencing, and I don't see a major benefit in knowing that before dereferencing.
- I'm using LITERAL with dmtype, but only with reservations. I'd much rather see PARAM here.

Modifications vs. DM patterns:

- Lots of minor ones I don't much care about (e.g., if you insist stripping the J off of J2000.0, I'm ok with it; don't care too much whether the attributes in the spherical point are longitude and latitude or something else).
- STC frame is a dataType-valued attribute of the coordinate (no referencing). Re-use of these beasts doesn't buy that much, and there's no point modelling them as objects.
- Just one coordinate type with weakly-typed attributes `location` (could be 1D, 2D, 3D, though that's probably only an issue for spatial) and `frame` (could be SpaceFrame, TimeFrame, or perhaps other frames); so, you don't have to define and maintain separate types for the various "axes".
- STC structures are referencing actual coordinates (VOTable FIELDs or PARAMs) rather than measurements (with errors and all); to understand the STC structure, it's not important if a coordinate is a measurement or has a different provenance.

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.