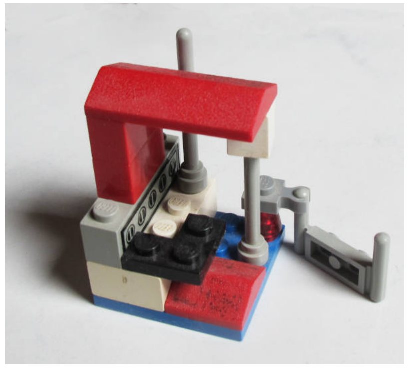
Fig. 1

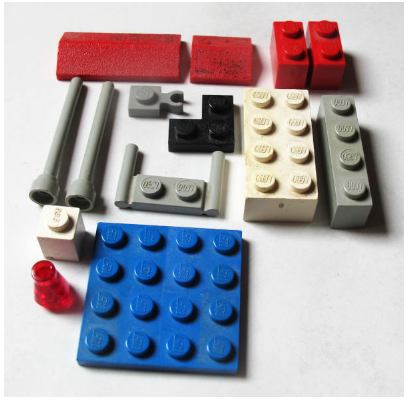
Fig. 2

1. Data Model Strategy

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 1)

Integrated?

(a.k.a. "entangled")

(cf. Fig. 2)

Isolated?

Fig. 3

(cf. Fig. 3)

2. Sketch of the Problem

Measure (value/error) vs. Coords (frames, grouping positions and PMs).

Meas has a meas:Time with a coord attribute of type coords:TimeStamp.

Why is that bad?

- Clients have to know the coords DM just to figure out the error for a time coordinate.
- When we incompatibly change coords to coords2, we will have to introduce meas2 even if *nothing changes* in Measurement at all.

3. A Way Out

Do DM just like we write programmes: in small, maximally isolated blocks. I claim that works for all actual *use* cases; see my astropy fork[1]'s README.

A.k.a.: Avoid the God model antipattern.

Let's do a quick run-down of our DMs to see what they'd do and not do.

4. Coords

Deals with:

- Location and motion in time and space
- Reference frames necessary for that

Has no idea of

- Errors
- Coverage
- Provenance

Governing use case: Epoch slider

[1] `https://github.com/msdemlei/astropy`

5. Measurement

Deals with:

- Matching up errors and values
- In the future, distributions
- Possibly, already now correlations between errors

Has no idea of

- Physics
- Coverage
- Provenance

Governing use case: Plot error bars (sci-fi: error propagation)

6. Photometry

Deals with:

- Location on the spectral axis
- Filter metadata
- Going between magnitudes and flux

Has no idea of

- Coverage
- Provenance

Governing use case: Plot SED.

7. Characterisation

Deals with:

- What values are possible and given on an axis.

Has no idea of

- Physics
- Time and space

Governing use case: Not sure if there's a place for this between Registry and VOTable

8. Dataset

Deals with:

- What is this (timeseries, spectrum, eventlist. . .)?
- Why was it made (e.g., target position)?
- Who do I ask for help?

Has no idea of

- Axes
- Coverage

Governing use case: Routing datasets to matching applications; figuring out who to ask if things go wrong

9. Cube

Deals with:

- Independent and dependent axes in a dataset

Has no idea of

- The physical nature of these axes
- Coverage
- Frames

Governing use case: Fill plotting dialogues

10. Provenance

Deals with:

- What was done to what to produce this data?

Has no idea of

- Table structure and content (except insofar as specific columns might contain results ProvDM talks about).

Governing use case: Debugging

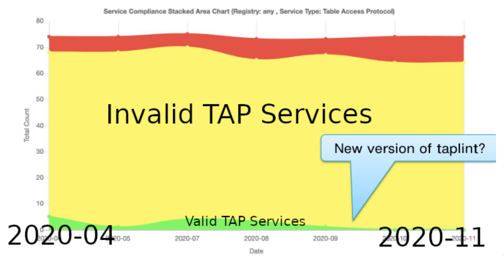

Fig. 4

11. So, a time series?

A time series, for instance, is just a cube with time as independent axis.

You'll quickly know it from the outside by inspecting Dataset annotation.

You'll know the time axis and the variables from cube annotation.

You'll know time metadata from coords annotation.

You'll know, e.g., zero points from PhotDM annotation.

You'll know the errors from Meas annotation.

All with generic code for the respective DMs!

12. Why bother?

1. Make DM adoption easy and rewarding. That's a lesson from STC-1 (and, really, all other IVOA MDs): data providers get scared of many boxes, and they lose all motivation when all the annotation they put in doesn't do anything. With a near-trivial Meas, they can just quickly mark up what's value and what's error, and TOPCAT can trivially plot error bars. Getting automatic error bars in TOPCAT will do a lot for DM street cred. Let's make it as easy as we possibly can for Mark and the data providers.
2. Meaningful validation. That's a lesson from DAL validation: Whenever you're doing something halfway complex, it's going to "break" in exotic ways; as of this writing almost all of the TAP services in the VO are "invalid" according to the EuroVO validator. (cf. slide 23 of this talk[2]). Except: they work. An integrated DM has about the same complexity as TAP. There's no reason to believe it'll behave differently from TAP, and so you'll have a huge lake of invalid annotations where nobody can say if it's just a small detail that's wrong of if it's utter breakage. With small, isolated DMs, it's clear what works just fine; if they're good, they work, if they're broken, they probably fail fast.
3. Separate evolvability. In a graph of DMs linked through typed references, when you incompatibly change one, you have to incompatibly change all (see the Coord/Meas example). You will thus break all annotations, although they are perfectly good and clients that do not need the one incompatibly changed annotation would work just fine.
4. Flexibility and Consistency. A client that knows how to deal with Coords annotation can do that whether these Coords are in a time series, a catalogue, or a SIAP response – always in the same way. And as a data provider, you don't need anything else at all: If it's a position, you can slap a Coords annotation on it, and it'll just work.

[2] `https://wiki.ivoa.net/internal/IVOA/InterOpNov2020Ops/20201118-Euro-VOResourcesValidationStatus.pdf`

(cf. Fig. 4)

13. Throw Everything Away?

Not at all. Disentangling DMs does not take a lot.

- Drop a few classes (e.g., per-physics Measurements)
- Introduce a few explicit references to columns (e.g., a value in PhotCal)

Oh, and the astropy implementation needs a few robustness fixes and write support. And the stamina for a PR.

14. But Most Importantly

Bring in the client authors. Now.

My proposition: Use a minimal Meas and the error bar plotting use case to make inroads.

Without their takeup, all we do here is pointless. They need to have a big say in what they want to consume.