Data Model Posture

Comments

Mark Cresitello-Dittmar

Nov. 18, 2020

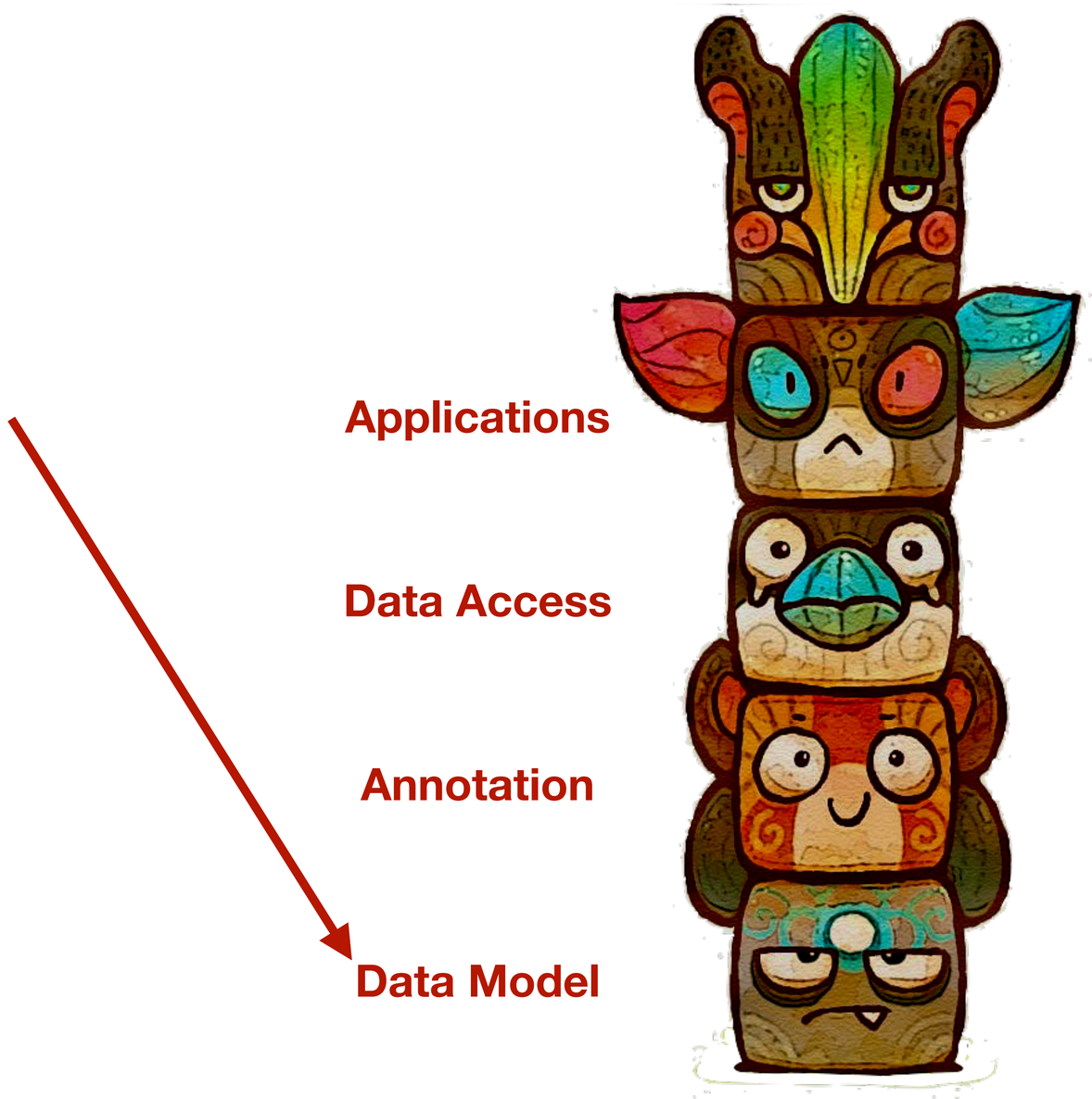
The issue

- The concern is valid; We can’t make it too difficult for data providers to maintain/generate ‘valid’ instances.
- The solution is in question.. where do we address this concern?

IVOA Hierarchy

My opinion

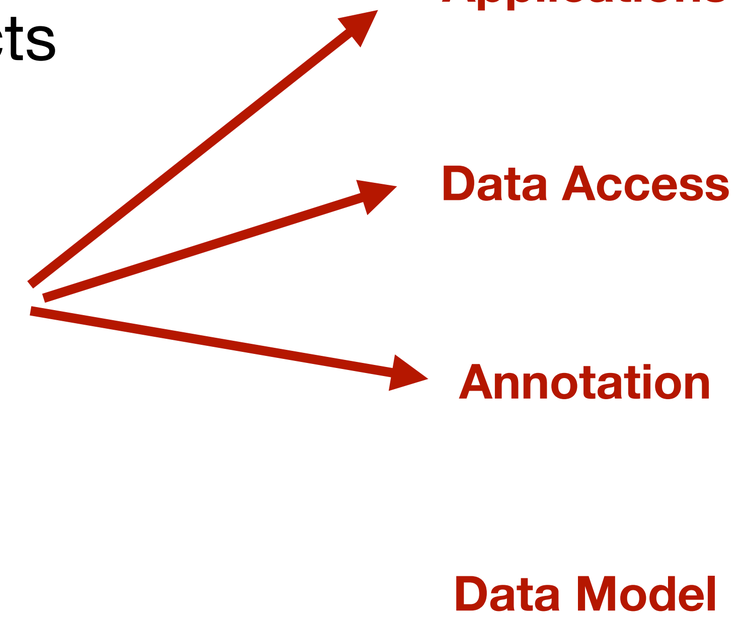
- The data model layer MUST be explicit. It informs all users of the expectations for all objects and relations.
- The higher layers can be as flexible as their requirements allow.



IVOA Hierarchy

My opinion

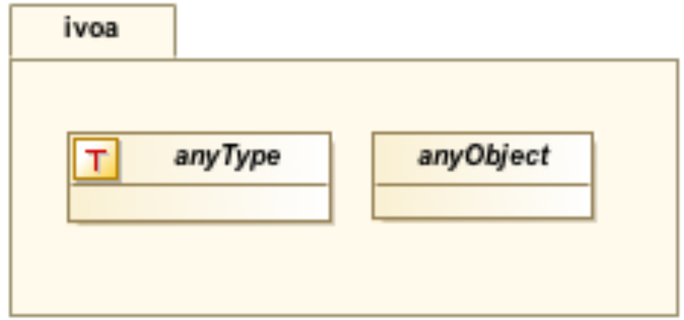
- The data model layer MUST be explicit. It informs all users of the expectations for all objects and relations.
- The higher layers can be as flexible as their requirements allow.



Un-Typing external dependencies

Technical

- Requires addition of ivoa:anyType and ivoa:anyObject
  - With explicit/implicit understanding that ALL DataTypes and ObjectTypes ultimately extend one of these two.

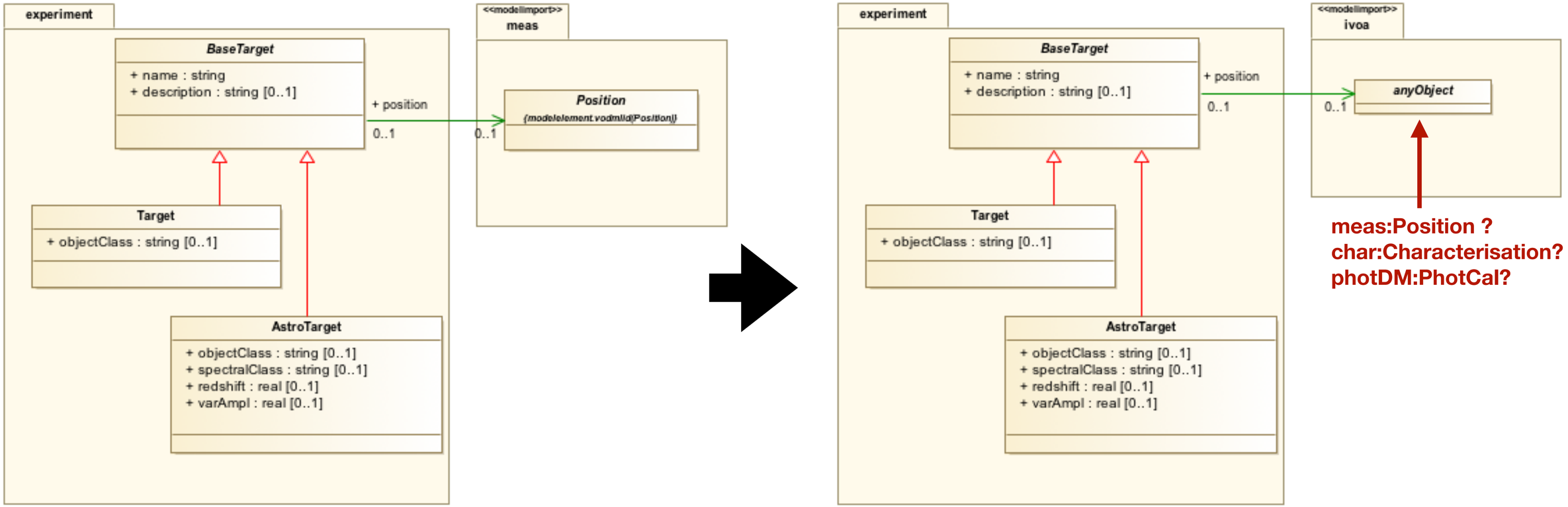


- Factoring of data models is somewhat arbitrary
  - If a model is split, suddenly explicit references become un-typed
- VO-DML rules permit any object to be in no more than ONE composition within the scope of a data model.

Un-Typing external dependencies

Application in models

- How does the ‘user’ know what the object SHOULD resolve to?



Solution?

I don’t know, maybe in annotation?

- VO-DML Annotations: very explicit, identifying specific versions of the contained models. But! We have already made serialization decisions which make things easier for ‘users’
  - ivoa:<types> => serialized per format types (ivoa:real => vot:double)
  - ivoa:Quantity => serialized by VOTable PARAM or FIELD elements
- VOTable COOSYS and TIMESYS: very safe
  - standard serializations map to coords model SpaceSys and TimeSys respectively
  - no formal relation to an underlying model
  - not scalable. We cannot define a serialization for every model object

Conclusion

- I think that something in the system MUST be explicit if we are going to interoperably exchange data. The client/provider must know what to expect at any given attribute.

- If we feel a need to protect clients/providers from version dependencies in serializations, I think this puts additional requirements on the annotation syntax, not on the underlying model.

- The general framework of developing smaller component models which are imported/used to build the more complex models is the proper approach. If you understand “meas:Position” you understand it when you see it as a Target position, as a column in a Cube, or as a Source property.