Mapping model compatibility with STC.

Motivation:

This study is in response to the IVOA discussion thread on the Mapping object in the ImageDM.

It is my assertion that the Mapping object defines coordinate systems which are relative to other coordinate systems. As such, this information should be stored under the CoordSys object of the model. The CoordSys object in the Image, Char, and Spectral models is based on the STC AstroCoordSys object, as are the Frame definitions, though greatly simplified. Further, the STC model allows for the definition of coordinate systems via transform in the full definition of the Frames.

As the IVOA Recommendation for representing coordinate systems and the basis for many of the model elements, I feel it is important that we evaluate the applicability of the full STC representation before creating new mechanisms. The purpose of this document is to migrate the Mapping model from the ImageDM draft to objects in terms of STC, and determine if STC fully supports the encapsulated information.

ImageDM Draft:

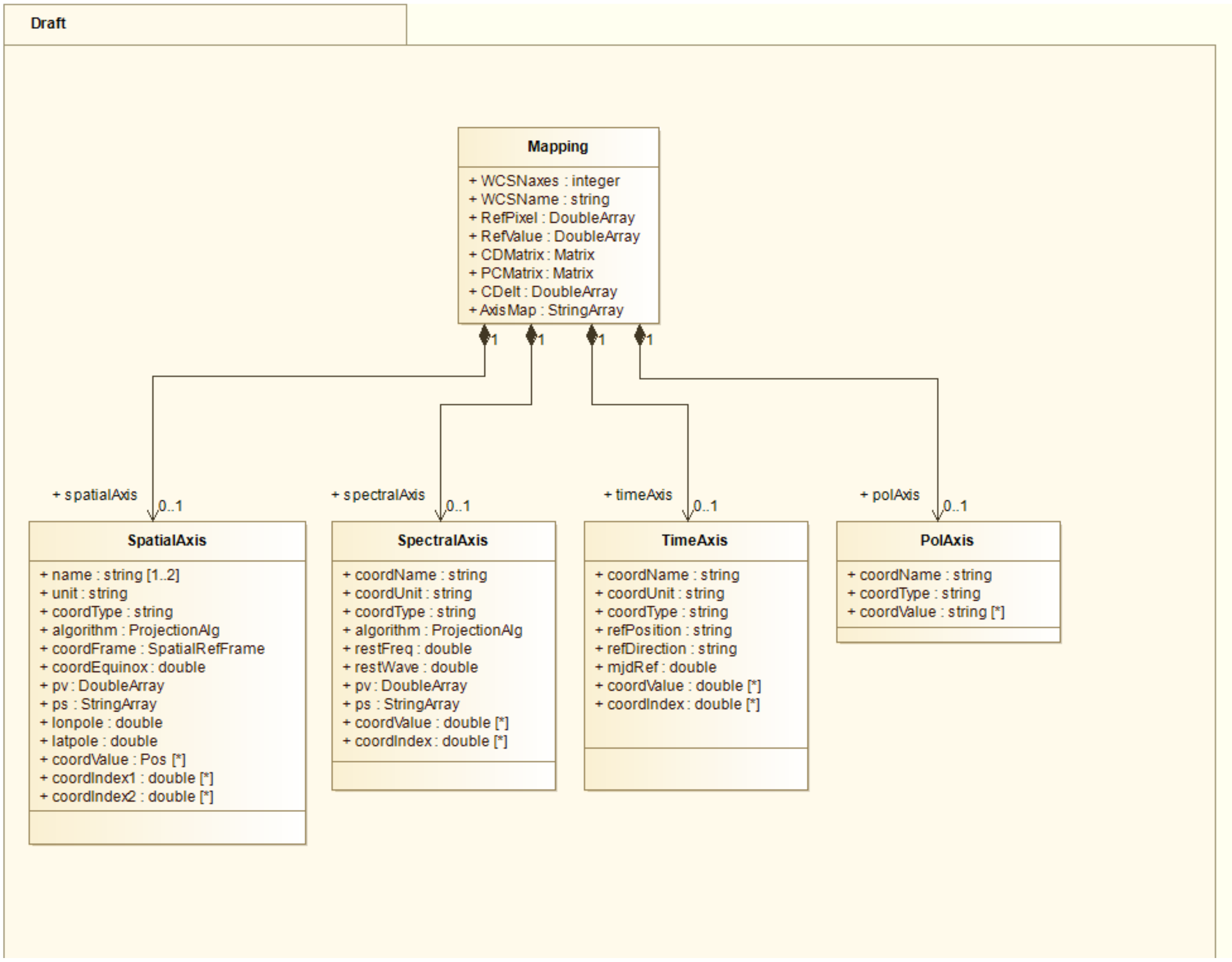


Illustration 1: ImageDM draft Mapping model

Figure 1 shows the Mapping model as described in the ImageDM draft document. It is essentially an encapsulation of the FITS WCS keyword set. The object flattens a great deal of structure, which will have to be explained in the text portion of the model.

- Linear transforms: can provide CDMatrix or PCMatrix + CDelt, but not both
- Non-Linear transforms: have Lookup table, or projection
- Frames: has Frame information in each Axis, including attributes which may or may not be relevant (equinox).

Step 1: Transforms and Frames

Extract the content describing the Transforms into a more generalized model.

- Lookup = "TAB" projection type
- Projection = WCS projection types
- SimpleScale
- CDMatrix
- PCMatrix

* Note: PV and PS are described in the FITS WCS text as parameters to certain non-linear transformations and are defined by the particular algorithm. As such, I would model them as attributes to the specific Transform type which are serialized as PV/PS keys and drop them from this model.

Extract Frame information from each Axis, use ImageDM definitions for the Frame objects for now. Note that the Mapping object holds partial Frame information, presumably the full definition of the coordinate system would be stored somewhere, creating a redundancy.

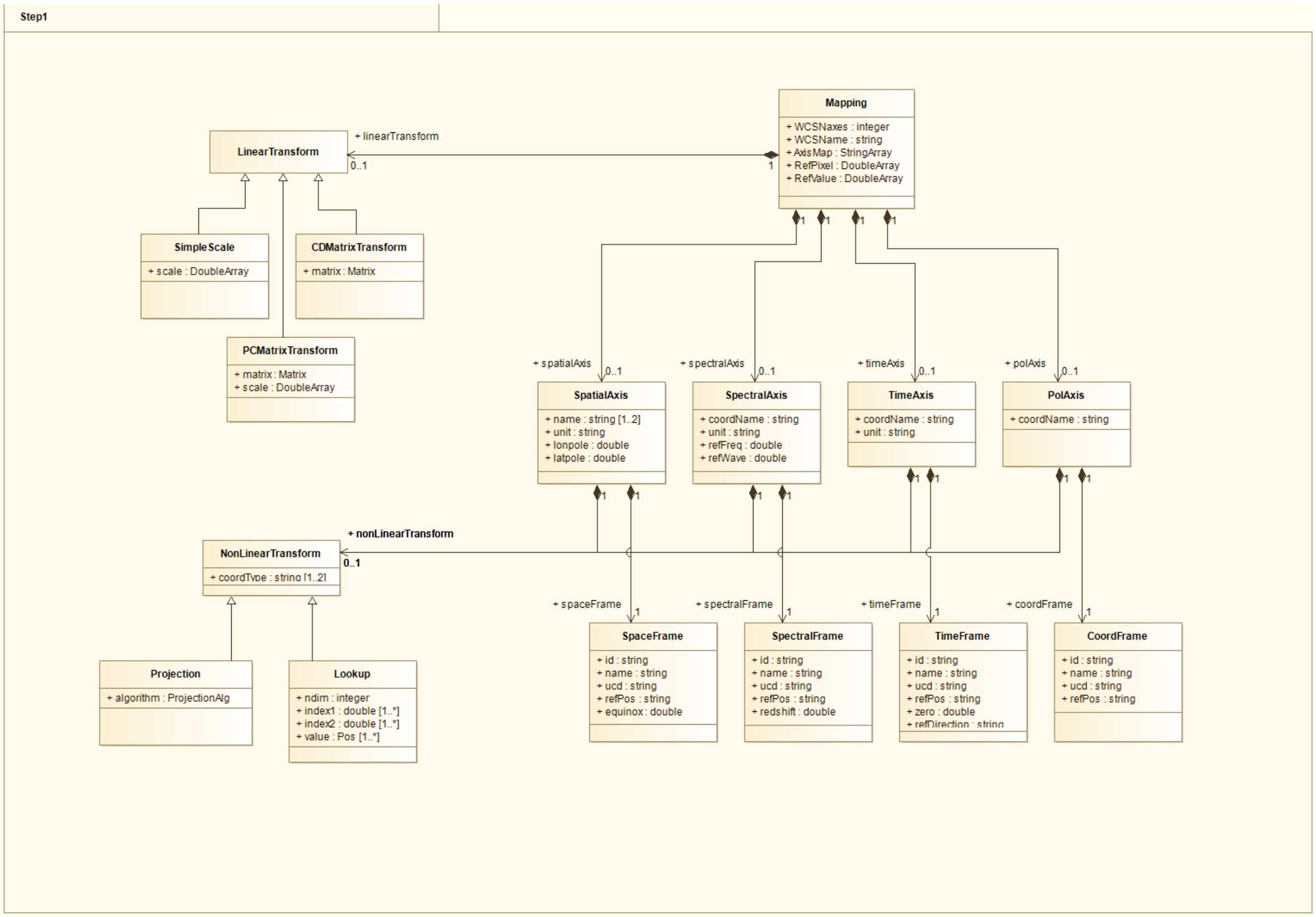


Illustration 2: Extract Transform and Frame

Step 2: Pixel Axes and Systems

The set of Frames defines a coordinate system (CoordSys).
- Add CoordSys object to hold the full definition of the WCS System
- Move ownership of the Frame objects to CoordSys, leaving a reference to them in each Axis

This will reside with the top level metadata, the Image dataset will have 1..* CoordSys objects describing all the coordinate systems in the dataset. These may be referenced by the Axis, or Char.

Note: there is no Observable Axis in this model.. the image values cannot be scaled.

Mapping provides "transformation from Image pixel coordinates to the specified World Coordinate System". The <Type>Axis objects provide the means for obtaining information about and along these axes, yet have no Axis for the pixel system to provide the same capability.
- Add PixelAxis, PixelFrame and PixelCoordSys analagous to <Type>Axis. For now, use a simplified STC definition for PixelFrame. Attach directly to Data as the 'source' of the Mapping link.

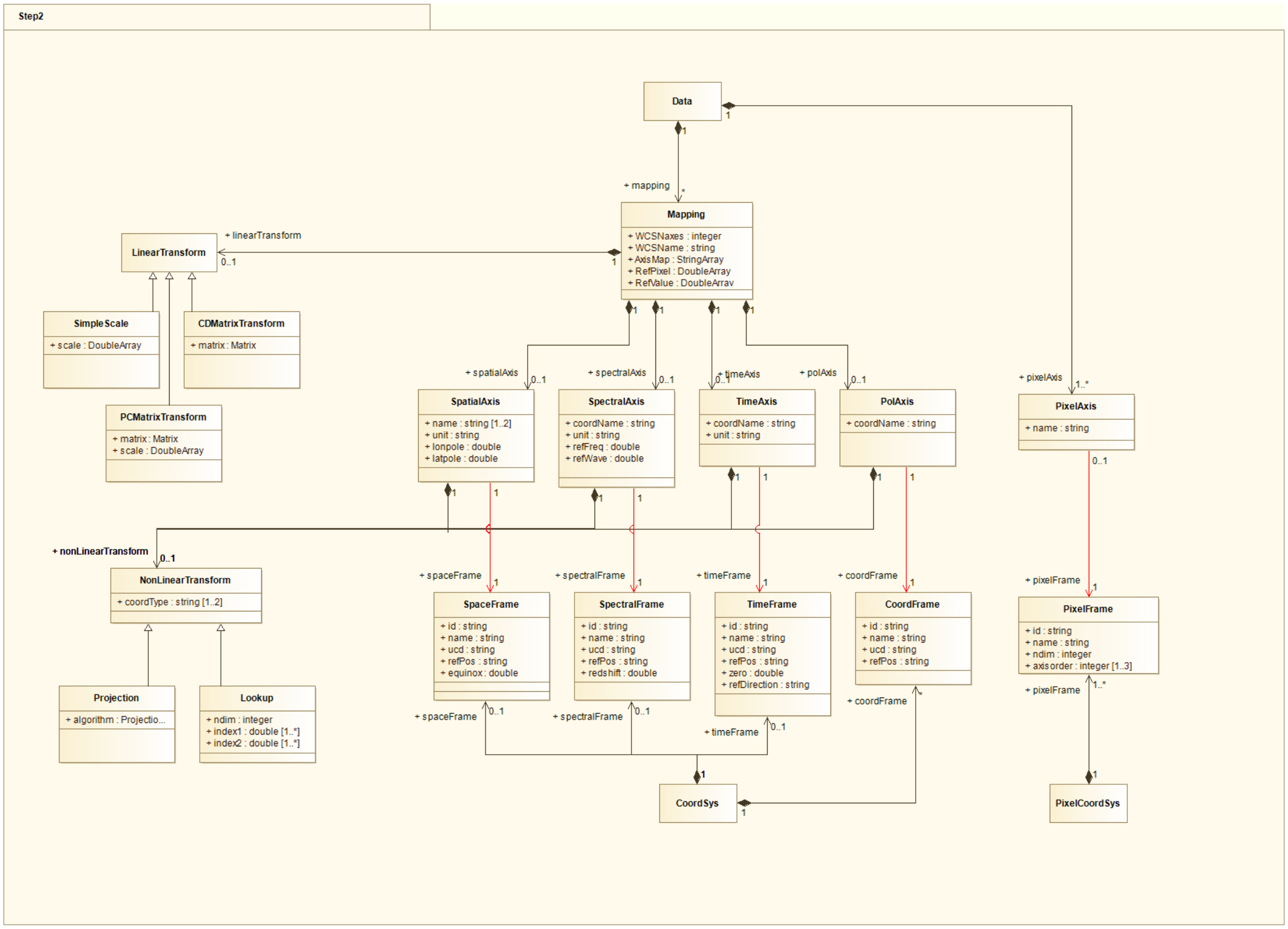


Illustration 3: Add PixelAxis and System objects

Step 3: Connecting the Systems.

Mapping.AxisMap identifies the "axes of the intermediate world coordinates". This is an unrealizable, hidden coordinate system and axis set in the diagrams. There is a Linear transform associated with the (source) pixel system to this intermediate system, and a Non-Linear transform associated with the (destination) WCS system from the intermediate system. Note: this association of Transform to different ends of the axis sets (source/destination) is inconsistent and difficult to model.

Adding another axis set to the diagram will get very ugly. Instead, apply some generalizations.
- Specific axis types are extensions of a generalized data axis. (Mapping.Axis exists in ImageDM).
  Note: Mapping.Axis only allows Lookup, but here I allow more.
- CoordFrame, PixelFrame, <Type>Frame are extensions of Frame. This is consistent with ImageDM Frame model and STC.

Move Transforms to DataAxis
- Allow 0..1 Linear plus 0..1 Non-Linear Transform per transition.
- add reference to associated Axis (replaces AxisMap), linking the source and destination axis.
- refPixel, refValue move with the Transforms.. but clearly don't belong here.

This enables one to 'realized' the intermediate axis set if desired, or apply both in one step.

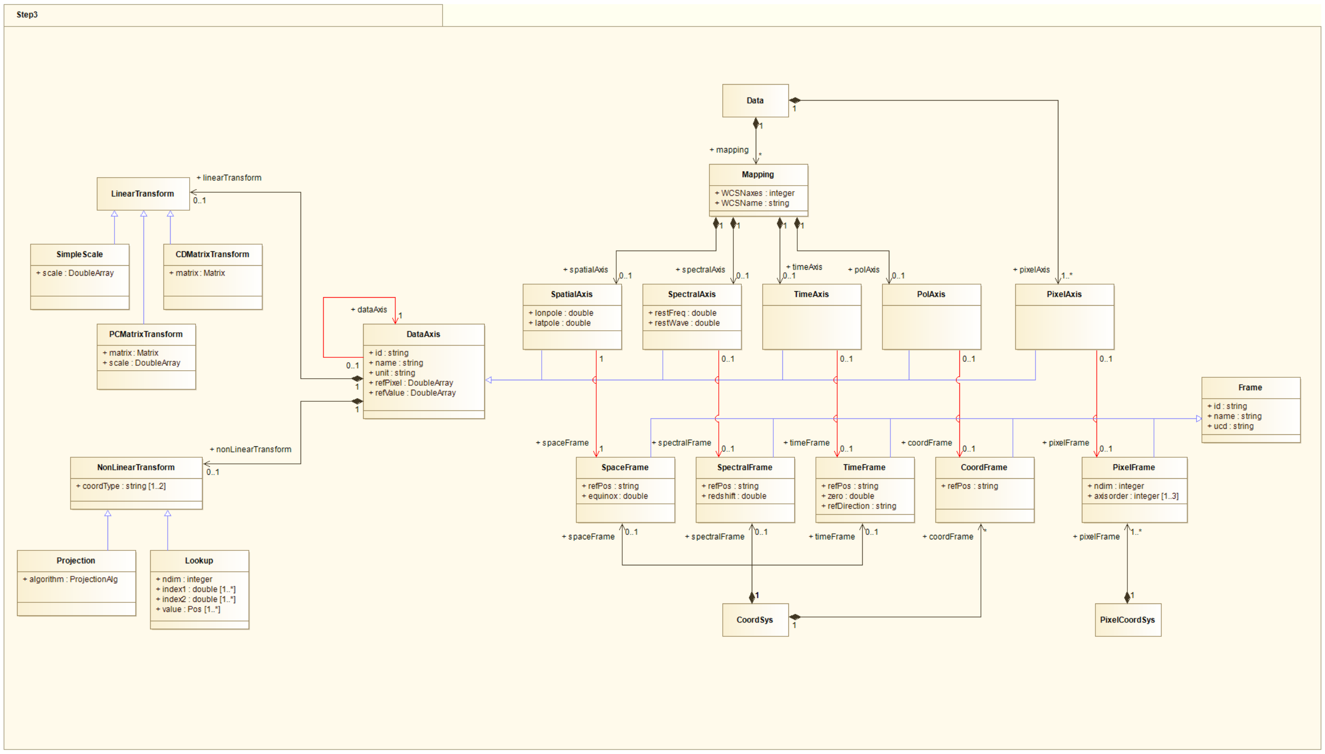


Illustration 4: Generalize axis and move Transforms

Step 4: Compare to STC

There are at least 2 good reasons not to leave Transforms hanging of the DataAxis.
  1) The Transforms define a transition between coordinate systems... not axes.
  2) Other 'data' may need to be converted between systems which are not Axis related.

STC associates the Transforms with the Frame via a ReferenceFrame. The ImageDM/Spectral/ObsCore models simplified the Frame objects at this level. I believe we are at a place where the simplifications no longer apply. Here we compare the ImageDM pieces to see if they all map to a corresponding STC element.

STC Frame has:

- Reference frame, either a standard one (ICRS, FK5, TT, UTC, etc) or one holding Transforms.
- Reference to the destination Frame
  (NOTE: I believe there is an error in the STC diagrams regarding the 'references')
- Reference Pixel or Position, either as as a standard origin (TOPOCENTER) or position.
- Reference Value or Offset.

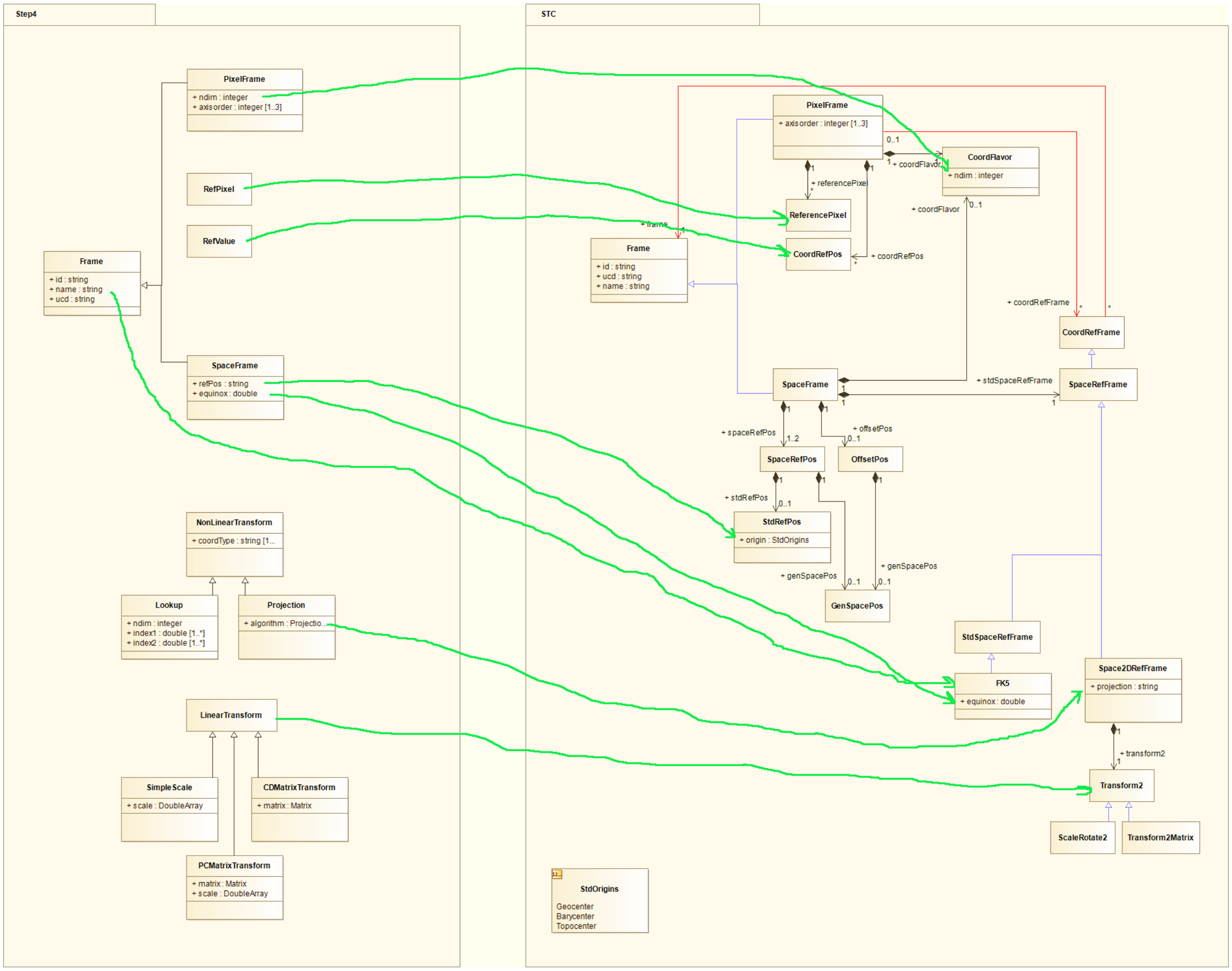


Illustration 5: Compare ImageDM with STC

The above diagram shows the SpaceFrame case, which is the most complex. The other axis types have similar structure.

Notes:
In the case of SpaceFrame -> SpaceFrame, I believe the mappings of RefPixel/RefValue would go to the SpaceFrame GenSpacePos/Offset objects for holding crpix and crval respectively.

There is an excellent mapping of objects to the STC model.
The STC model also provides the linkages between Frames.. the transforms are associated with the source frame, and the describe the transition From the destination To the source. The ReferenceFrame holds reference to the destination Frame. Again, this is not properly shown in the STC diagrams, but the XML seems to indicate so.

Issues:
- STC 'projection' seems insufficient, it holds just the projection type (e.g. TAN), I'm not sure this is sufficient description for the "TAB" case.

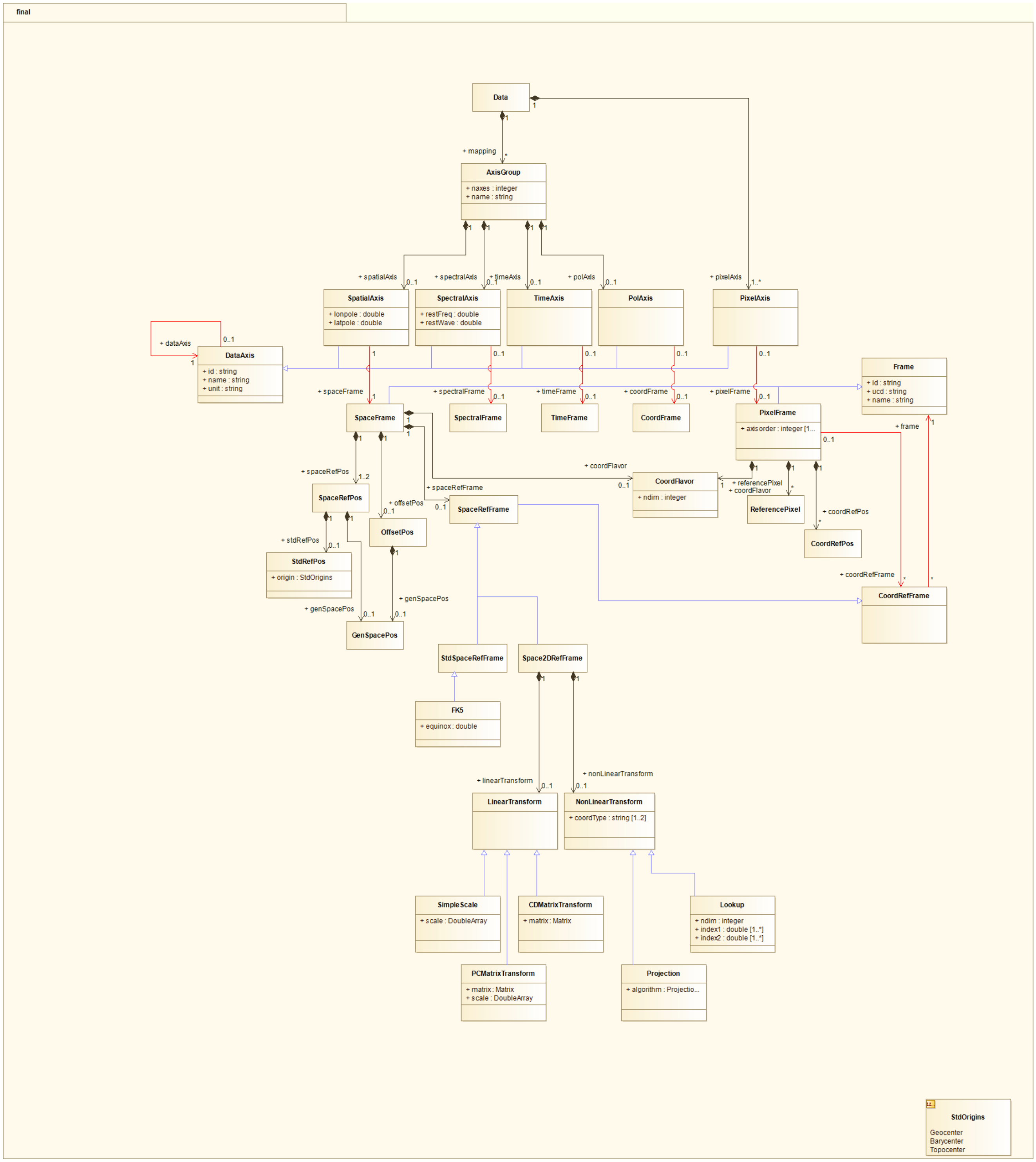
- There are some items which I did NOT map.. (lonpole, latpole, restFreq, restWave). Lonpole/Latpole have clear locations in STC. The spectral rest frequency is a bit harder to map, but I think is described as a singular valued Spectral Frame (?). I think these are mappable.

Conclusion:

I think this thread shows that STC adequately handles the desired capability. By "not" simplifying the Frame objects, we get the added capability desired, using existing IVOA recommendations. The ImageDM can leave the definitions to STC, and spend its time providing support information to assist users (use case diagrams, etc). Further, the DataAxis concept is one I also introduced to the Spectral model. With the combination of these elements, I believe we can have greater consistency in describing the various Datasets which we are modeling (NDCube, Image, Spectral, etc).

I fully concede that this scenario does not provide a self-contained object holding all the WCS information for a coordinate system, as the Mapping object does. If such an object is needed to assist data discovery and/or a query response, the access protocol document can define Mapping, how to populate it, and attach it to the query response. The Image dataset model should have a fully consistent and detailed model for the coordinate systems.

The final diagram below, shows the Transform information moved to a more STC like structure. This is just to give a feel for the structure, I don't want to re-diagram STC. Everything below the DataAxis line would be STC.


final
Data
1
1
+ mapping
*
AxisGroup
+ naxes : integer
+ name : string
1
1
1
1
+ spatialAxis
0..1
+ spectralAxis
0..1
+ timeAxis
0..1
+ polAxis
0..1
+ pixelAxis
1..*
SpatialAxis
+ lonpole : double
+ latpole : double
SpectralAxis
+ restFreq : double
+ restWave : double
TimeAxis
PolAxis
PixelAxis
+ dataAxis
0..1
1
DataAxis
+ id : string
+ name : string
+ unit : string
1
0..1
0..1
0..1
0..1
Frame
+ id : string
+ ucd : string
+ name : string
+ spaceFrame
1
+ spectralFrame
0..1
+ timeFrame
0..1
+ coordFrame
0..1
+ pixelFrame
0..1
SpaceFrame
SpectralFrame
TimeFrame
CoordFrame
PixelFrame
+ axisorder : integer [1...
1
1
+ frame
1
0..1
1
1
+ coordFlavor
0..1
CoordFlavor
+ ndim : integer
1
+ referencePixel
+ coordFlavor
1
1
1
*
*
ReferencePixel
+ coordRefPos
CoordRefPos
+ spaceRefPos
1..2
SpaceRefPos
+ spaceRefFrame
0..1
SpaceRefFrame
+ offsetPos
0..1
OffsetPos
1
1
+ stdRefPos
0..1
StdRefPos
+ origin : StdOrigins
1
+ genSpacePos
+ genSpacePos
0..1
0..1
GenSpacePos
+ coordRefFrame
*
*
CoordRefFrame
StdSpaceRefFrame
Space2DRefFrame
1
1
FK5
+ equinox : double
+ linearTransform
0..1
+ nonLinearTransform
0..1
LinearTransform
NonLinearTransform
+ coordType : string [1..2]
SimpleScale
+ scale : DoubleArray
CDMatrixTransform
+ matrix : Matrix
Lookup
+ ndim : integer
+ index1 : double [1..*]
+ index2 : double [1..*]
PCMatrixTransform
+ matrix : Matrix
+ scale : DoubleArray
Projection
+ algorithm : Projectio...
StdOrigins
Geocenter
Barycenter
Topocenter