Mivot AstroPyvo et Mango

-

Laurent MICHEL on the behalf of all DMers

Reading VOTables Column per Column

Parcourir toutes les **colonnes** pour identifier leurs rôles

Sélectionner les **colonnes** d'intérêt

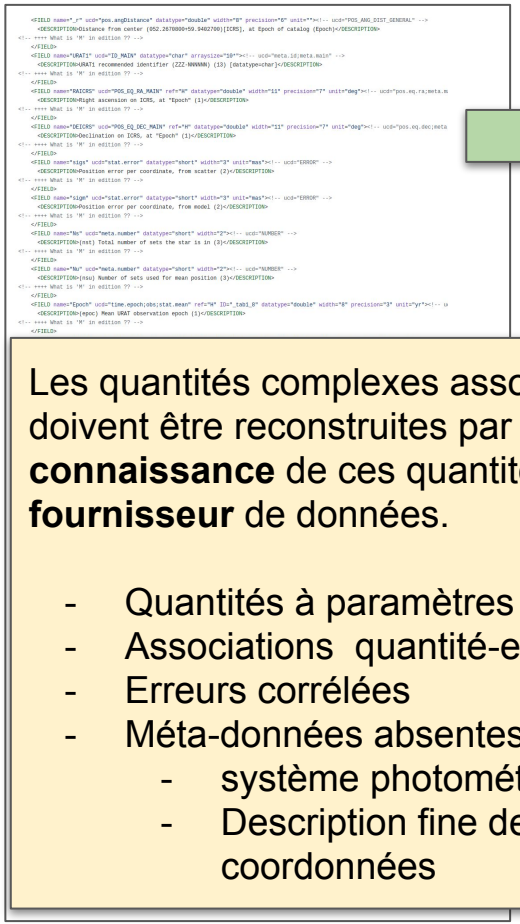
```
<FIELD name="RAICRS" ucd="POS_EQ_RA_MAIN" ref="H" datatype="double" width="11" precision="7" unit="deg"><
  <DESCRIPTION>Right ascension on ICRS, at "Epoch" (1)</DESCRIPTION>
<FIELD name="DEICRS" ucd="POS_EQ_DEC_MAIN" ref="H" datatype="double" width="11" precision="7" unit="deg">
  <DESCRIPTION>Declination on ICRS, at "Epoch" (1)</DESCRIPTION>
<FIELD name="pmRA" ucd="pos.pm;pos.eq.ra" ref="H" datatype="float" width="6" precision="1" unit="mas/yr"><
  <DESCRIPTION>?(pmr) Proper motion RA*cosDec (from 2MASS) (7)</DESCRIPTION>
<FIELD name="pmDE" ucd="pos.pm;pos.eq.dec" ref="H" datatype="float" width="6" precision="1" unit="mas/yr">
  <DESCRIPTION>?(pmd) Proper motion in Declination (7)</DESCRIPTION>
```

Selected <FIELDS>

Construire le code scientifique à partir des **colonnes** isolées

```
for rec in table.array:
    skycoord = SkyCoord(longitude=rec["RAICRS"],
                        latitude=rec["DEICRS"],
                        pmLongitude=rec["pmRA"],
                        pmLatitude=rec["pmDEC"],
                        frame="icrs")
```

Reading VOTables Column per Column



Parcourir toutes les **colonnes** pour identifier leurs rôles

Sélectionner les **colonnes** d'intérêt

```
<FIELD name="RAICRS" ucd="POS_EQ_RA_MAIN" ref="H" datatype="double" width="11" precision="7" unit="deg"><
  <DESCRIPTION>Right ascension on ICRS, at "Epoch" (1)</DESCRIPTION>
<FIELD name="DEICRS" ucd="POS_EQ_DEC_MAIN" ref="H" datatype="double" width="11" precision="7" unit="deg">
  <DESCRIPTION>Declination on ICRS, at "Epoch" (1)</DESCRIPTION>
<FIELD name="pmRA" ucd="pos.pm;pos.eq.ra" ref="H" datatype="float" width="6" precision="1" unit="mas/yr"><
  <DESCRIPTION>?(pmr) Proper motion RA*cosDec (from 2MASS) (7)</DESCRIPTION>
<FIELD name="pmDE" ucd="pos.pm;pos.eq.dec" ref="H" datatype="float" width="6" precision="1" unit="mas/yr">
  <DESCRIPTION>?(pmd) Proper motion in Declination (7)</DESCRIPTION>
```

Selected <FIELDS>

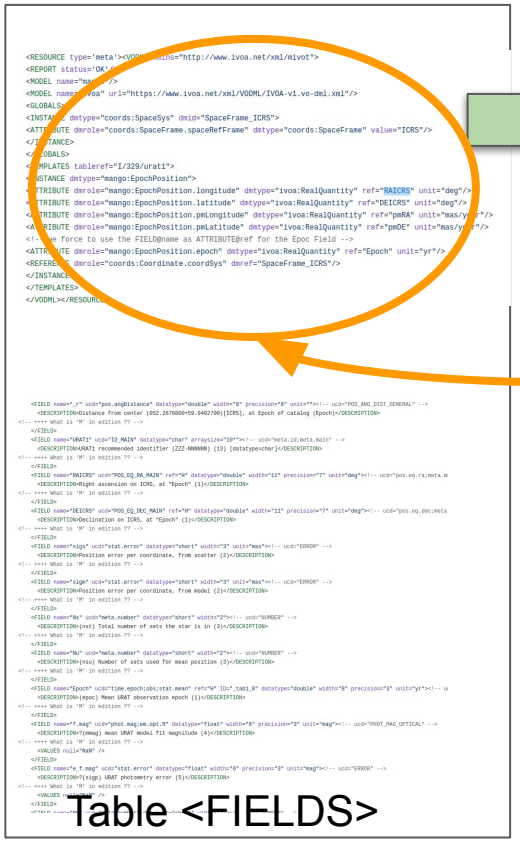
Construire le code scientifique à partir des **colonnes** isolées

```
for rec in table.array:
    skycoord = SkyCoord(longitude=rec["RAICRS"],
                        latitude=rec["DEICRS"],
                        pmLongitude=rec["pmRA"],
                        pmLatitude=rec["pmDEC"],
                        frame="icrs")
```

Les quantités complexes associant plusieurs colonnes doivent être reconstruites par le **code client** alors que la **connaissance** de ces quantités est du ressort du **fournisseur** de données.

- Quantités à paramètres multiple
- Associations quantité-erreurs
- Erreurs corrélées
- Méta-données absentes du schéma VOTable
  - système photométrique
  - Description fine des systèmes de coordonnées

Reading VOTables Quantity per Quantity



Parcourir toutes les **quantités** disponibles

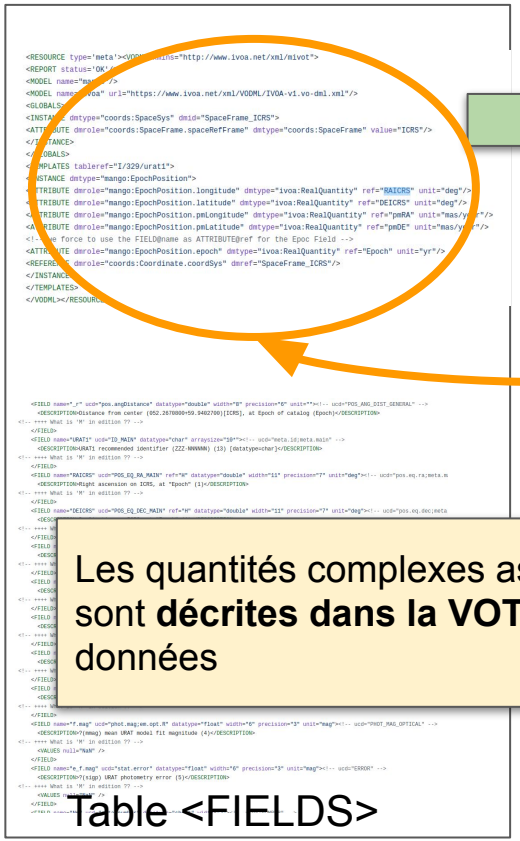
Sélectionner les **quantités** d'intérêt

La manière de reconstruire les quantités complexe est décrite dans la VOtable elle-même

Construire le code scientifique à partir de ces **quantités**

```
sky_coord = SkyCoord(
    longitude=complex_quantity.longitude.value,
    latitude=complex_quantity.latitude.value,
    pmLongitude=complex_quantity.pmLongitude.value,
    pmLatitude=complex_quantity.pmLatitude.value,
    frame=mivot_object.Coordinate_coordSys.spaceRefFrame.value,
)
```

Reading VOTables Quantity per Quantity



Parcourir toutes les **quantités** disponibles

Sélectionner les **quantités** d'intérêt

La manière de reconstruire les quantités complexe est décrite dans la VOtable elle-même

Les quantités complexes associant plusieurs colonnes sont **décrites dans la VOTable** par le **fournisseur** de données

Construire le code scientifique à partir de ces **quantités**

```
SkyCoord(
    longitude=complex_quantity.longitude.value,
    latitude=complex_quantity.latitude.value,
    pmLongitude=complex_quantity.pmLongitude.value,
    pmLatitude=complex_quantity.pmLatitude.value,
    frame=mivot_object.Coordinate_coordSys.spaceRefFrame.value,
)
```

La feuille de route

- **Pour décrire les quantités complexes dans la VOTable, il faut:**
  1. Un modèle décrivant ces quantités
  2. Un moyen de connecter les données particulières au modèle
  3. Intégrer ce pont entre données et modèles dans le schéma de la VOTable
  4. Développer des outils capables d'utiliser ce mécanisme d'annotations
  5. Des services capables de fournir des données annotées
  6. Des outils de validation

1 MANGO 2 3 MIVOT 4 Aladin Desktop, Python, Rust 5 Vizier 6 mivot-validator

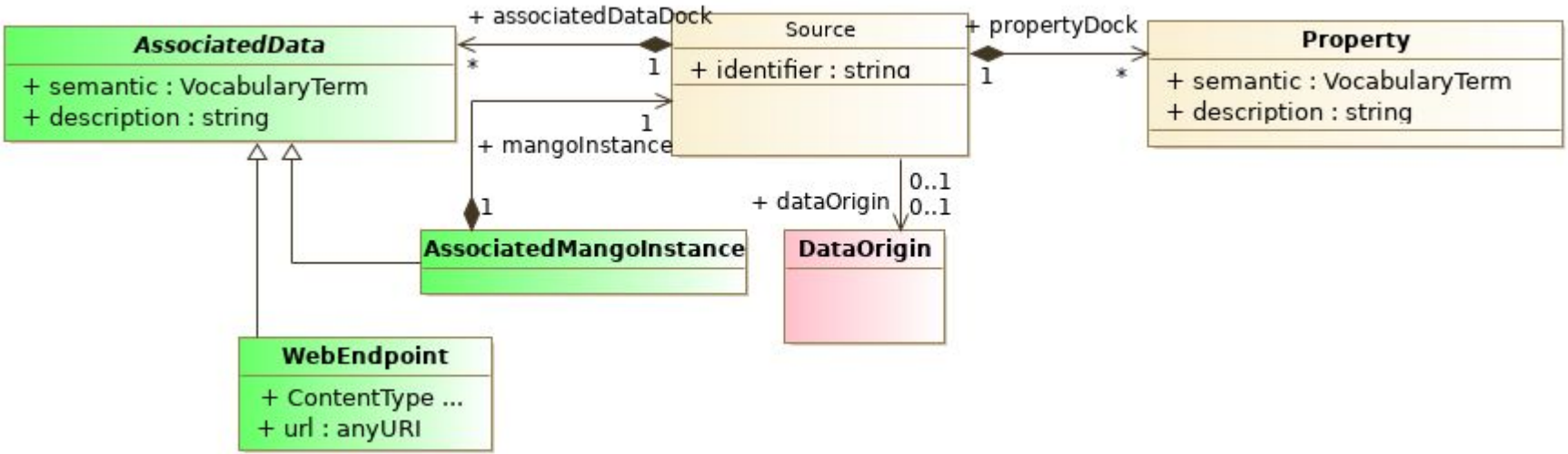
1 Le modèle: MANGO

- **Un modèle générique pour décrire les quantités contenues dans une table**
  - **M**odel for **AN**notating **G**eneric **O**bjects : **MANGO**
  - `Source` est la classe racine du modèle

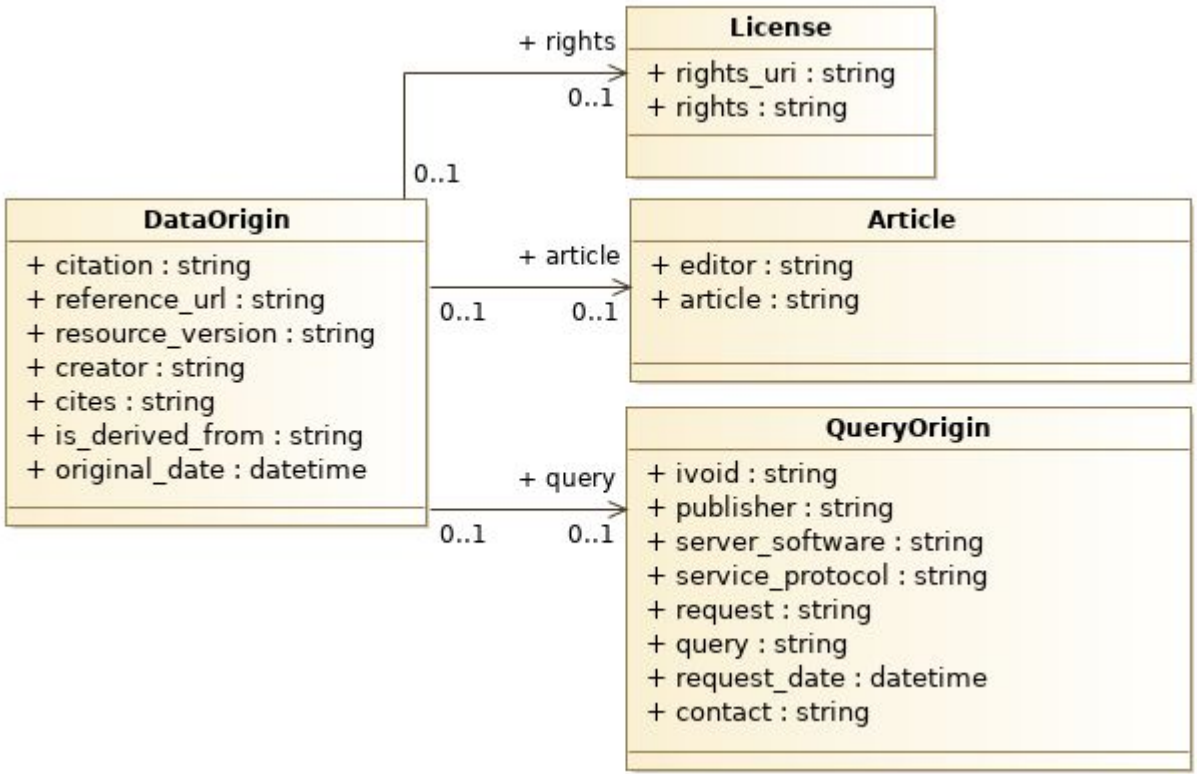
- **Une source est définie par**
  - Un identifier
  - Une origine
  - Un ensemble de propriétés
  - Un ensemble de données associées

Chaque ligne de la VOtable peut-être interprétée comme un instance de la classe `Source`

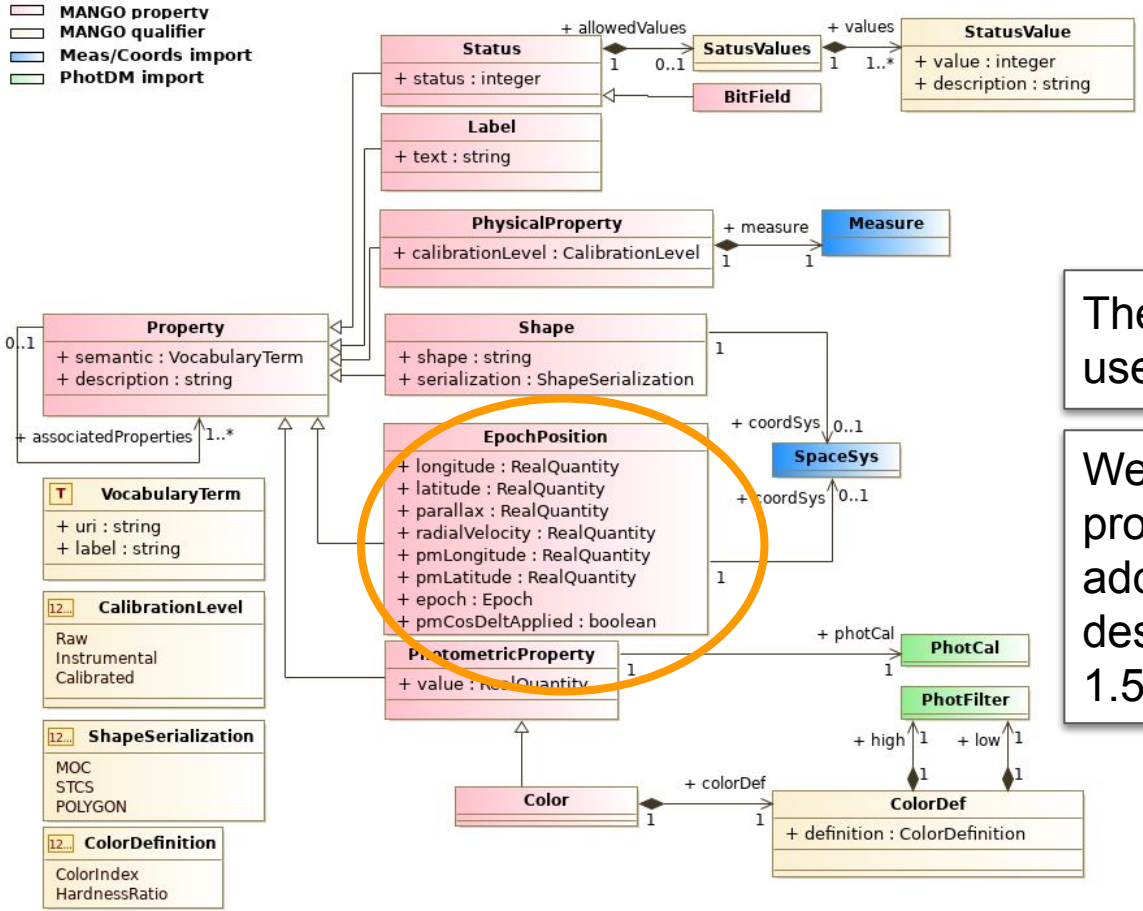
1 Le modèle: La Source



1 Le modèle: L'origine des données (DCP IG)



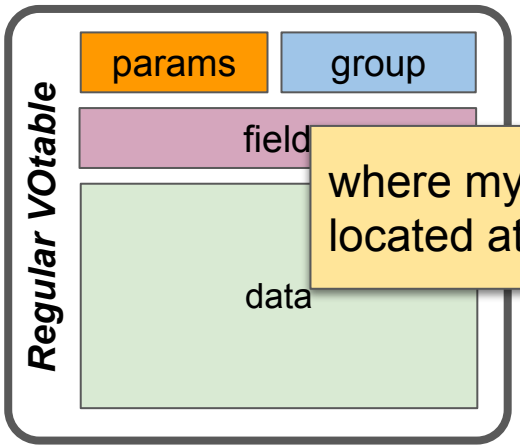
1 Le modèle: Les `properties` MANGO



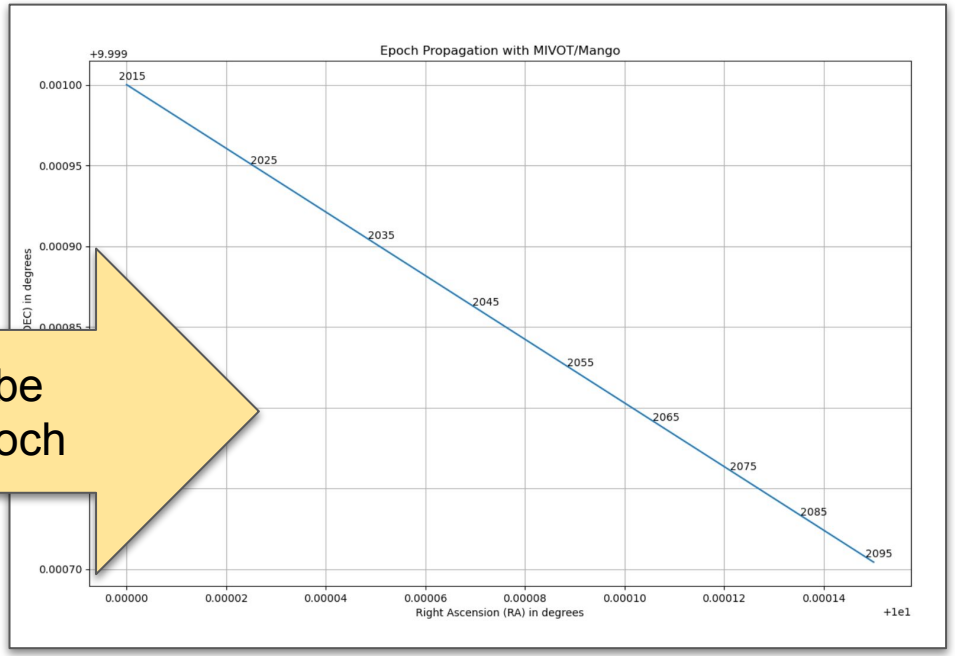
The workflow is validated against on use-case: **The Epoch propagation**

We have to demonstrate that our proposal works so that it can be adopted as a standard way to describe moving position on VOTable 1.5

The IVOA Challenge



where my object will be located at a given epoch



In this VOTABLE, I have somewhere:

- Position
- Proper motion
- Parallax
- Radial velocity
- Desired metadata

The challenge: **find an appropriate way to do it**

2 3 *MIVOT*: The Model Based Option

- Keep the VOTable as delivered by the ADQL engine
  - No FIELDREF tuning
  - No GROUP to create
- Take a VO model describing the EPOCH propagation
  - 6 parameters
  - Errors
  - Space Frame
- Write an XML serialization of that MODEL in the VOTable
  - MIVOT syntax
  - Put the proper FIELD references into model leaves
- Insert that XML piece above the TABLE

2 3 *MIVOT*: Add a Mapping Block above the Data Table

- The space coordinate system is a **`GLOBAL`** object that can be referenced by any other MIVOT element

- Each table row can be interpreted as an instance of the class **`EpochPosition`** of the MANGO model

```
    <MODEL name="coords" url="https://www.ivoa.net/xml/STC/20200908/Coords-v1.0.vo-dml.xml" />
    <MODEL name="mango" />
    <MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
    <GLOBALS>
        <INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceSys">
            <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:SpaceFrame">
                <INSTANCE dmrole="coords:SpaceFrame.refPosition" dmtype="coords:StdRefLocation">
                    <ATTRIBUTE dmrole="coords:StdRefLocation.position" dmtype="ivoa:string" value="NoSet" />
                </INSTANCE>
                <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame" dmtype="ivoa:string" value="ICRS" />
            </INSTANCE>
        </INSTANCE>
    </GLOBALS>
    <TEMPLATES>
        <INSTANCE dmtype="mango:EpochPosition">
            <REFERENCE dmrole="coords:Coordinate.coosys" dmref="SpaceFrame_ICRS"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.longitude" dmtype="ivoa:RealQuantity" ref="pos_RA"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.latitude" dmtype="ivoa:RealQuantity" ref="pos_DEC" />
            <ATTRIBUTE dmrole="mango:EpochPosition.pmLongitude" dmtype="ivoa:RealQuantity" ref="pm_RA" />
            <ATTRIBUTE dmrole="mango:EpochPosition.pmLatitude" dmtype="ivoa:RealQuantity" ref="pm_DEC"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.pmCosDeltApplied" dmtype="ivoa:boolean" value="true"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.radialVelocity" dmtype="ivoa:RealQuantity" ref="RV"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.parallax" dmtype="ivoa:RealQuantity" ref="PARALLAX" />
            <ATTRIBUTE dmrole="mango:EpochPosition.epoch" dmtype="coords.Epoch" value="J2016.0" unit="year"/>
        </INSTANCE>
    </TEMPLATES>
</VODML>
```

2 3 *MIVOT*: Add a Mapping Block above the Data Table

- The space coordinate system is a **GLOBAL** object that can be referenced by any other MIVOT element

- Each table row can be interpreted as an instance of the class **EpochPosition** of the MANGO model

```
<MODEL name="coords" url="https://www.ivoa.net/xml/STC/20200908/Coords-v1.0.vo-dml.xml" />
<MODEL name="mango" />
<MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
<GLOBALS>
    <INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceSys">
        <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:SpaceFrame">
            <INSTANCE dmrole="coords:SpaceFrame.refPosition" dmtype="coords:StdRefLocation">
                <ATTRIBUTE dmrole="coords:StdRefLocation.position" dmtype="ivoa:string" value="NoSet" />
            </INSTANCE>
            <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame" dmtype="ivoa:string" value="ICRS" />
        </INSTANCE>
    </INSTANCE>
</GLOBALS>
<TEMPLATES>
    <INSTANCE dmtype="mango:EpochPosition">
        <REFERENCE dmrole="coords:Coordinate.coosys" dmref="SpaceFrame_ICRS"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.longitude" dmtype="ivoa:RealQuantity" ref="pos_RA"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.latitude" dmtype="ivoa:RealQuantity" ref="pos_DEC" />
        <ATTRIBUTE dmrole="mango:EpochPosition.pmLongitude" dmtype="ivoa:RealQuantity" ref="pm_RA" />
        <ATTRIBUTE dmrole="mango:EpochPosition.pmLatitude" dmtype="ivoa:RealQuantity" ref="pm_DEC"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.pmCosDeltApplied" dmtype="ivoa:boolean" value="true"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.radialVelocity" dmtype="ivoa:RealQuantity" ref="RV"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.parallax" dmtype="ivoa:RealQuantity" ref="PARALLAX" />
        <ATTRIBUTE dmrole="mango:EpochPosition.epoch" dmtype="coords.Epoch" value="J2016.0" unit="year"/>
    </INSTANCE>
</TEMPLATES>
</VODML>
```

2 3 *MIVOT*: Add a Mapping Block above the Data Table

- The space coordinate system is a **GLOBAL** object that can be referenced by any other MIVOT element

- Each table row can be interpreted as an instance of the class **EpochPosition** of the MANGO model

```
<MODEL name="coords" url="https://www.ivoa.net/xml/STC/20200908/Coords-v1.0.vo-dml.xml" />
<MODEL name="mango" />
<MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
<GLOBALS>
    <INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceSys">
        <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:SpaceFrame">
            <INSTANCE dmrole="coords:SpaceFrame.refPosition" dmtype="coords:StdRefLocation">
                <ATTRIBUTE dmrole="coords:StdRefLocation.position" dmtype="ivoa:string" value="NoSet" />
            </INSTANCE>
            <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame" dmtype="ivoa:string" value="ICRS" />
        </INSTANCE>
    </INSTANCE>
</GLOBALS>
<TEMPLATES>
    <INSTANCE dmtype="mango:EpochPosition">
        <REFERENCE dmrole="coords:Coordinate.coosys" dmref="SpaceFrame_ICRS"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.longitude" dmtype="ivoa:RealQuantity" ref="pos_RA"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.latitude" dmtype="ivoa:RealQuantity" ref="pos_DEC" />
        <ATTRIBUTE dmrole="mango:EpochPosition.pmLongitude" dmtype="ivoa:RealQuantity" ref="pm_RA" />
        <ATTRIBUTE dmrole="mango:EpochPosition.pmLatitude" dmtype="ivoa:RealQuantity" ref="pm_DEC"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.pmCosDeltApplied" dmtype="ivoa:boolean" value="true"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.radialVelocity" dmtype="ivoa:RealQuantity" ref="RV"/>
        <ATTRIBUTE dmrole="mango:EpochPosition.parallax" dmtype="ivoa:RealQuantity" ref="PARALLAX" />
        <ATTRIBUTE dmrole="mango:EpochPosition.epoch" dmtype="coords.Epoch" value="J2016.0" unit="year"/>
    </INSTANCE>
</TEMPLATES>
</VODML>
```

- Class attributes refer to the columns that are used to set their values

- Some class attributes can have fixed values, they don't hold column references

2 3 MIVOT: C'est un standard VO

IVOA Recommendation

International Virtual Observatory Alliance

IVOA Documents

Model Instances in VOTables
Version 1.0

IVOA Recommendation 20 June 2023

Interest/Working Group:
http://www.ivoa.net/twiki/bin/view/IVOA/IvoaDataModel
Author(s):
Laurent Michel, Mark Cresitello-Dittmar, François Bonnarel, Gilles La
Editor(s):
Laurent Michel, Mark Cresitello-Dittmar

Abstract

Abstract: Model Instances in VOTables (MIVOT) defines a syntax to map VOTable d
VOTable to the data model elements (class, attributes, types, etc.) of a standardiz
missing in the table metadata. The data model elements are grouped in an indepe
The MIVOT syntax allows to describe a data structure as a hierarchy of classes. It
VOTable. Missing metadata can also be provided using MIVOT, for instance by con
both client and server sides. The adopted design does not alter the original VOTab

Status of this document

This document has been produced by the Data Model Working Group.
It has been reviewed by IVOA Members and other interested parties, and has beer
reference from another document. IVOA's role in making the Recommendation is t
Community.

Available formats: pdf

The standard comes with:

- An IVOA standard document
- An XML schema allowing computer to validate documents.
  - The schema has been written in XSD1.1 to support different syntax patterns depending on the local context
  - The schema is independent from the VOTable: tools not supporting MIVOT are still working on annotated VOtables

5 Coté serveur: ConeSearch Vizier (G. Landais)

- **Service**
  - Maps epochs, positions and proper motions on the `EpochPosition` MANGO class
    - Data origin, errors, radial velocities and parallax not supported yet
  - Works on any Vizier catalogue
    - https://cdsarc.cds.unistra.fr/beta/viz-bin/mivotconesearch/TABLE_ID
      - Ex: TABLE_ID = I/239/hip_main
- **Used to validate various tools**

% curl 'https://cds/viz-bin/mivotconesearch/I/329/urat1?RA=52.26708&DEC=59.94027&SR=0.05'

5 Coté serveur: Filter Profile Service (Carlos Rodrigo *RIP*)

- **Service**
  - Return MIVOT serializations of photometric calibrations
    - `PhotDM` instances
  - Works on all filter references by the SVO filter profile service
    - http://svo2.cab.inta-csic.es/svo/theory/fps/fpsmivot.php?PhotCalID=FILTER_ID
      - Ex: FILTER_ID = GAIA/GAIA3.G/Vega

% curl 'http://svo2.cab.inta-csic.es/svo/theory/fps/fpsmivot.php?PhotCalID=GAIA/GAIA3.G/Vega'

6 Validator (L. Michel, J. Abid, M. Louys, F. Bonnarel)

- **IVOA project**
  - https://github.com/ivoa/mivot-validator
- **Main features**
  - Document validation: 3 independent processes
    - The VOTable part is validated against the VOTable 1.3 schema
    - The MIVOT block is validated against the MIVOT schema
    - The structure of the mapped classes is validated against the model they refer to.
  - Side benefit: the snippet generation
    - The tool can generate snippets for all class of a model
      - These snippets can be used by other stakeholder to build annotations

6 Validator (L. Michel, J. Abid, M. Louys, F. Bonnarel)

Validate an annotated VOTable

```
mivot-votable-validate  <VOTable path>
```

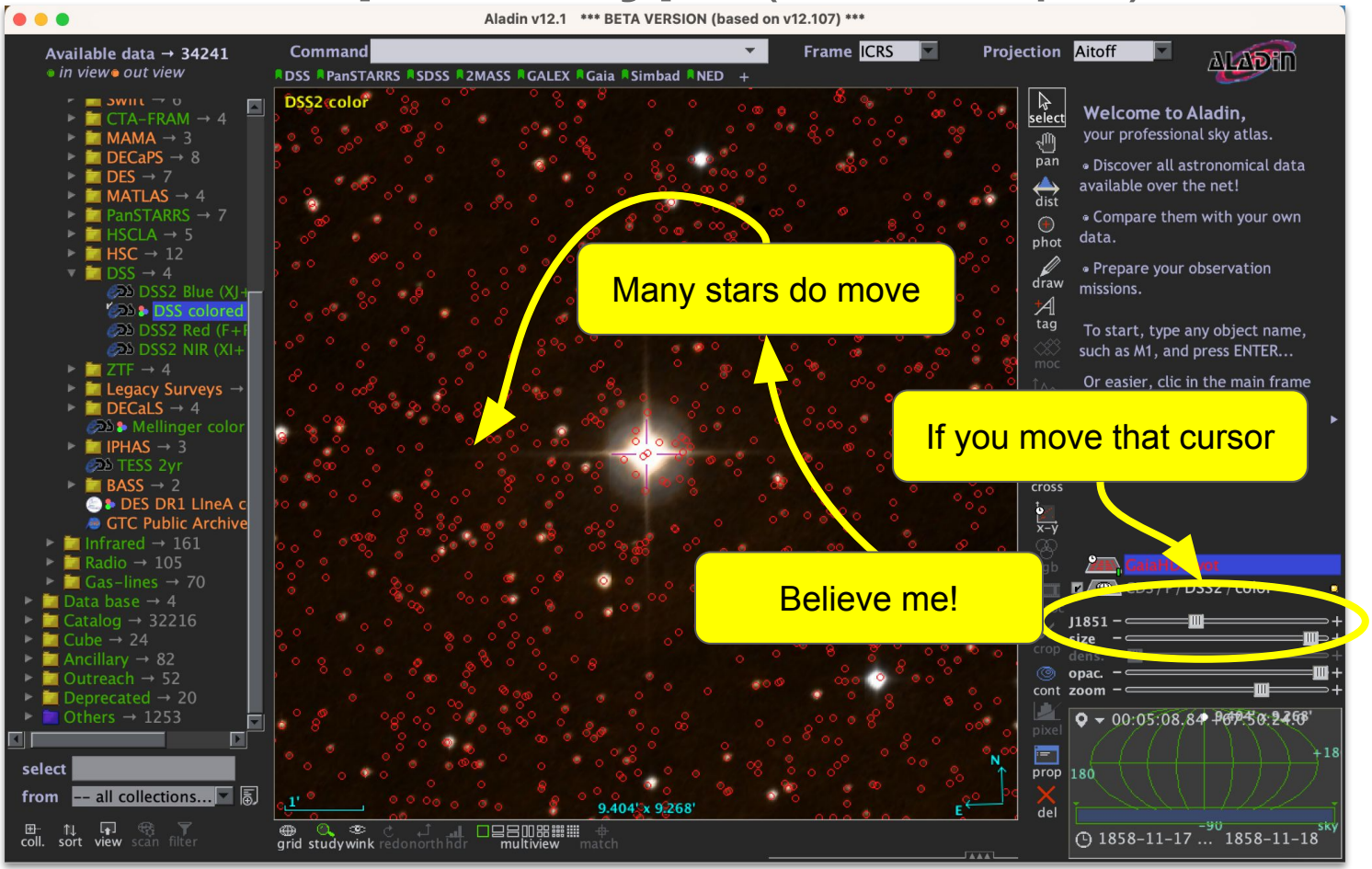
Validate an XML file containing just a MAPPING block

```
mivot-mapping-validate  <XML path>
```

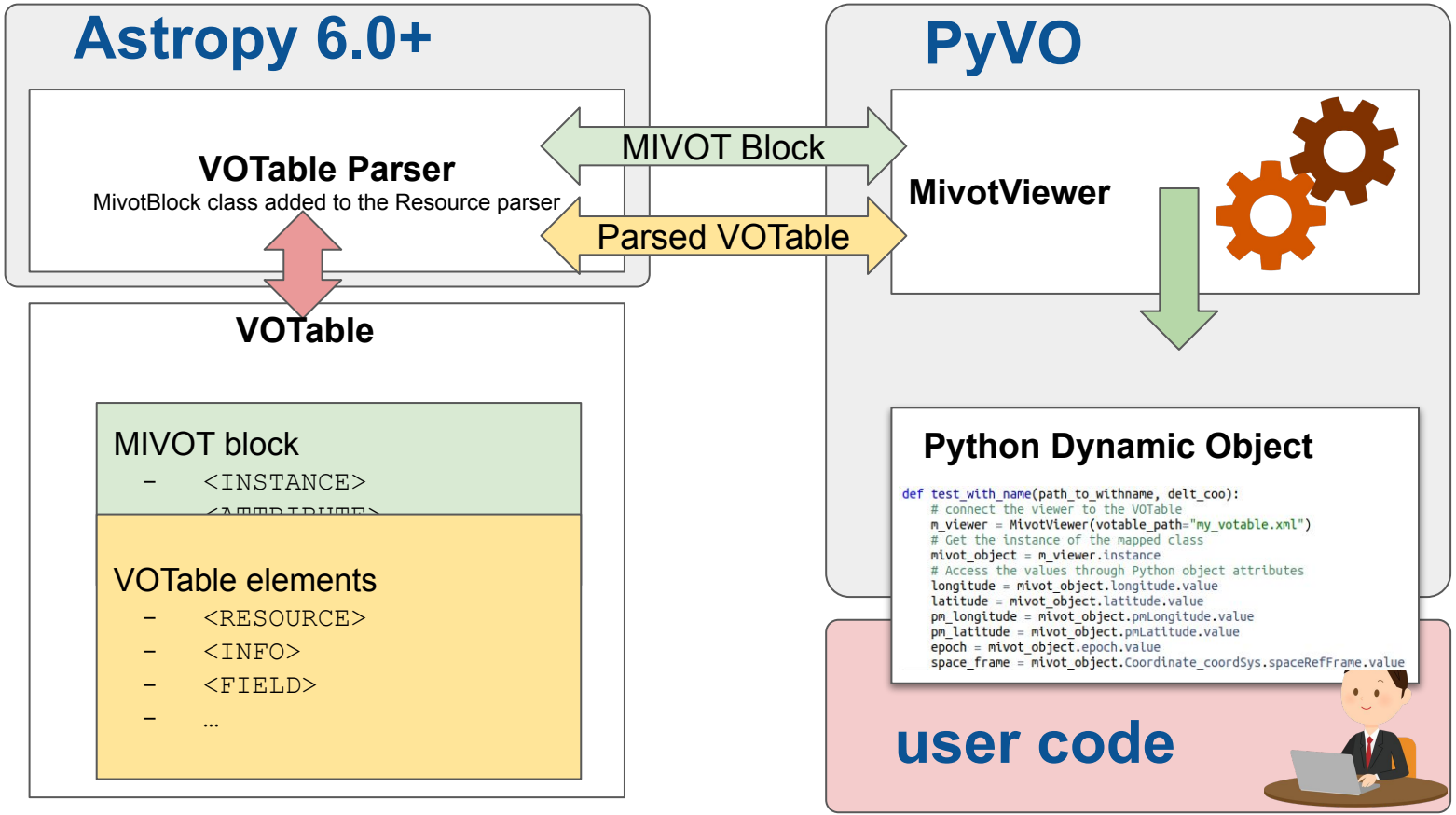
```
mivot-instance-validate <VOTABLE path>
```

```
USAGE: mivot-snippet-model [path]
   Create MIVOT snippets from VODML files
   path: either a simple file to any VODML-Model or an url
   exit status: 0 in case of success, 1 otherwise
```

4 Aladin Desktop Prototype (P. Fernique)



4 PYTHON implementation (L. Michel S. Floret G. Landais)



4 Python implementation (L. Michel, G. Landais, S. Floret)

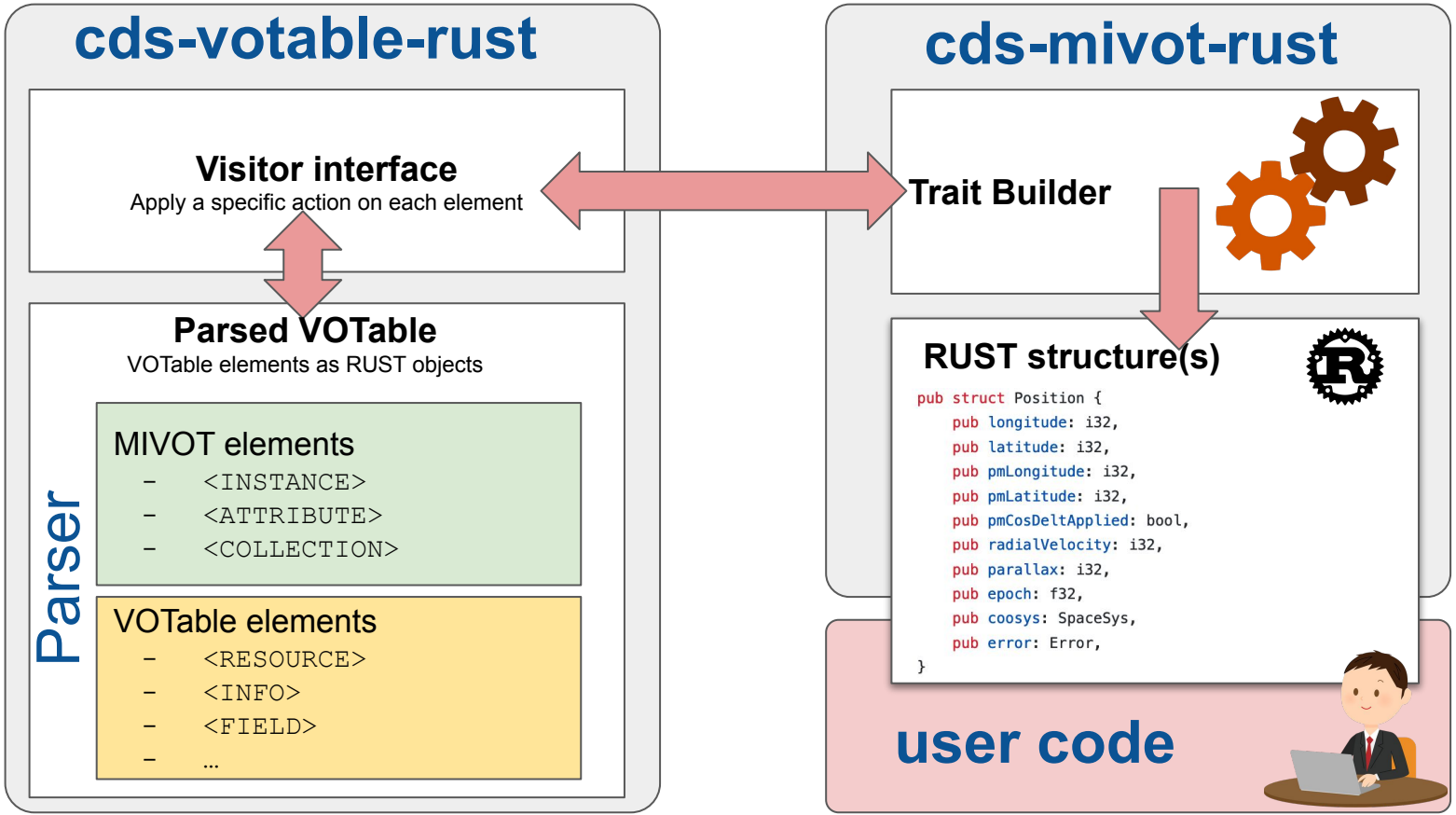
```
def test_cone_search(vizier_url):
    scs_srv = SCSService(vizier_url)
    m_viewer = MivotViewer(
        scs_srv.search(
            pos=SkyCoord(ra=52.26708 * u.degree, dec=59.94027 * u.degree, frame='icrs'),
            radius=0.05
        )
    )
    mivot_instance = m_viewer.instance
    assert mivot_instance.dmtype == "EpochPosition"
    assert mivot_instance.Coordinate_coordSys.spaceRefFrame.value == "ICRS"
    ra = []
    dec = []
    while m_viewer.next():
        ra.append(mivot_instance.latitude.value)
        dec.append(mivot_instance.longitude.value)
    assert ra == [59.94033461]
    assert dec == [52.26722684]
```

This implementation is model-agnostic

Same code for whatever model

Model knowledge is the charge of the user

4 RUST implementation (F.X. Pineau J. Abid)



④ RUST implementation (F.X Pineau J. Abid)

```rust
use std::path::Path;
use crate::mivot::ModelLayer;

let mut model_layer = ModelLayer::from_file(Path::new("my-votable.xml"), true).unwrap();

model_layer.init_epoch_positions();

// Retrieve the EpochPosition instance from the mapping block
let epoch_positions = model_layer.get_epoch_positions_as_ref().get("EpochPosition").unwrap();

// Browse the instance
print!("{}", epoch_positions.longitude.value);
print!("{}", epoch_positions.latitude.value);
print!("{}", epoch_positions.coosys.frame.spaceRefFrame);
```

This implementation is model-dependant

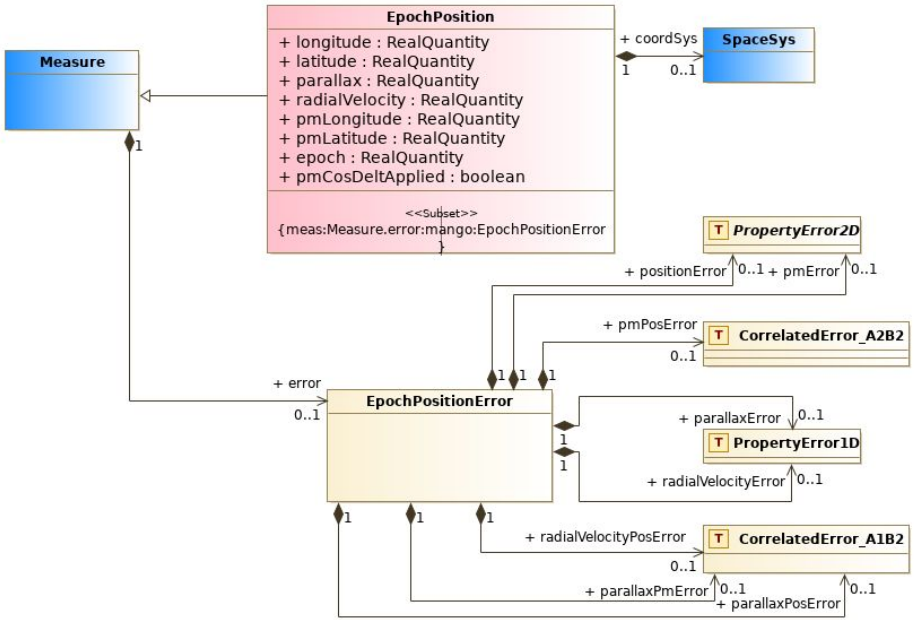
New/change model => New code

Ou en est-on?

- **Python code**
  - Annotation readout merged in Astropy 6.0 (10/2023)
  - MIVOT viewer should be part of Pyvo 1.6
    - Wait on MR#497 to be merged
- **RUST code**
  - Will be published when more MIVOT-enable services will be available
- **Validator**
  - Require some polishing before to be published in Pypi.
- **MANGO**
  - Major evolution following the MIVOT discussion
  - Current draft to be presented in Sydney

BACKUP - BACKUP

MIVOT: The Daunting Step: Build a Model



- The role of all model components is perfectly defined by the model

- The class is part of MANGO draft
  - The MANGO overview is not shown here
- Use of Meas/Coord
  - Extends *Measure* classes from the Measure Model
  - Use *Coords* classes to describe coordinate systems
- Support complex errors
  - Per parameter errors
  - Covariance errors
  - Correlated errors
- No need to use all the features proposed by the model
  - only use model elements that match data

MIVOT: Flexibilty

EPOCH defined in a **<FIELD>**
@ref to the FIELD identifier

```
<ATTRIBUTE dmrole="tucson:Position.epoch"
        dmtype="coords:Epoch"
        unit="year"
        ref="_EPOCH_FIELD"/>
```

EPOCH defined in a **<PARAM>**
@ref to the PARAM identifier

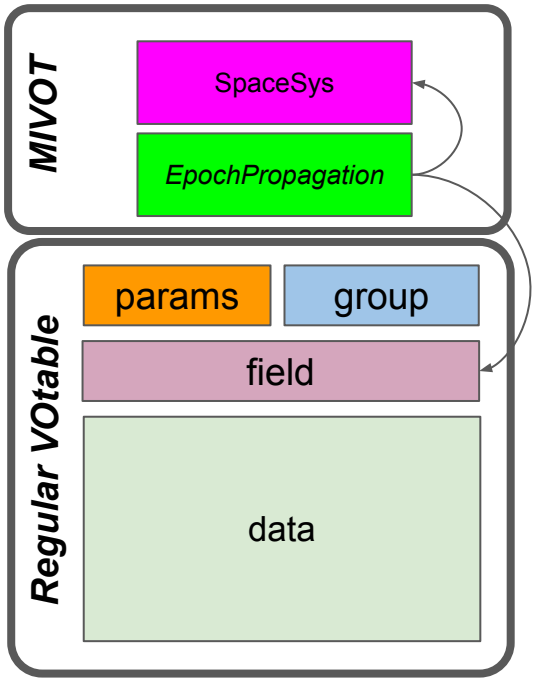
```
<ATTRIBUTE dmrole="tucson:Position.epoch"
        dmtype="coords:Epoch"
        unit="year"
        ref="_EPOCH_PARAM"/>
```

EPOCH defined in a non machine-readable element
No **@ref** but a fixed **@value**

```
<ATTRIBUTE dmrole="tucson:Position.epoch"
        dmtype="coords:Epoch"
        unit="year"
        value="J2023.88"/>
```

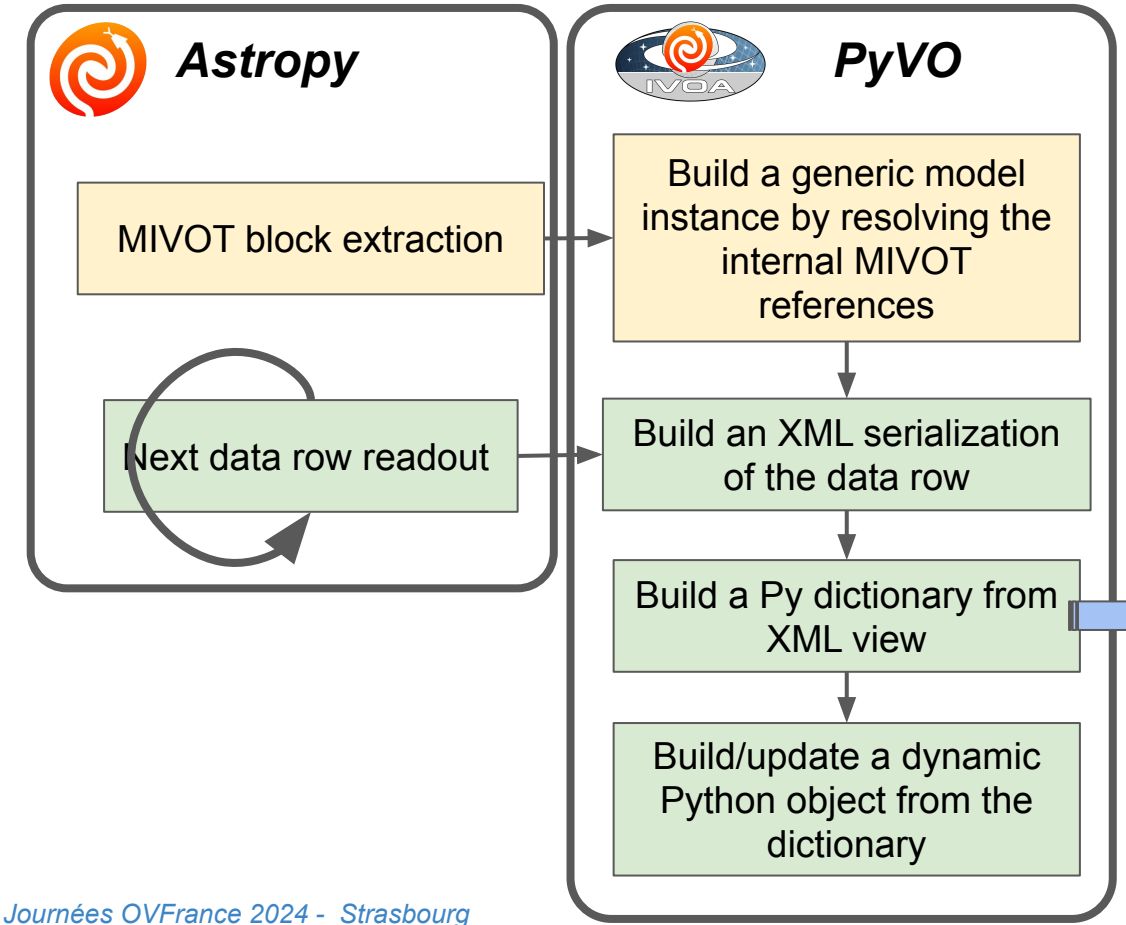
The structure of MIVOT block is not altered by the way the EPOCH is set in the VOTable
- Allow server code to be versatile
- Allow a same client code to process many different VOTable

MIVOT: MIVOT annotations

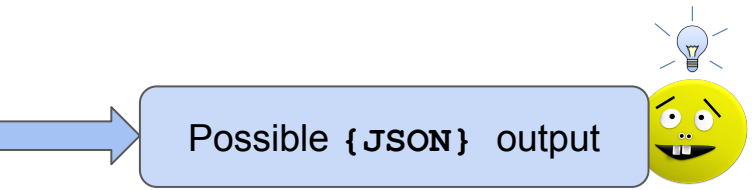


- **MIVOT block: an XML model view**
  - Above the data table
  - The hierarchy of the XML elements denotes the model structure
  - References to the appropriate columns
  - Syntax controlled by the MIVOT XML schema
- **The client can easily get model instances**
  - Read the MIVOT block
  - Resolve the reference to the FIELDS
  - Set the attribute values with the row data

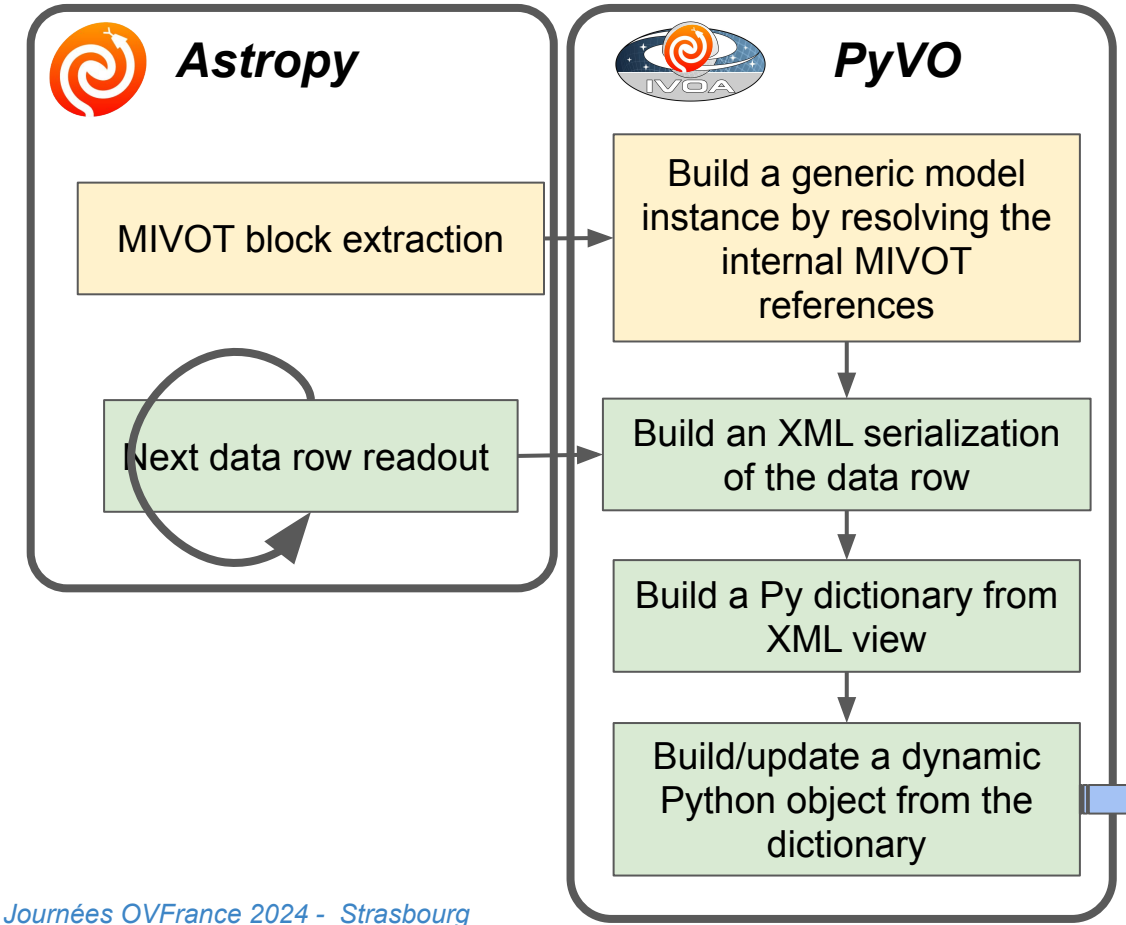
MIVOT: PyVO implementation,



- No model specific code
  - The compliance with the model structure is not checked at this level
  - Can be done with the validator



MIVOT: PyVO implementation,



- No model specific code
  - The compliance with the model structure is not checked at this level
  - Can be done with the validator

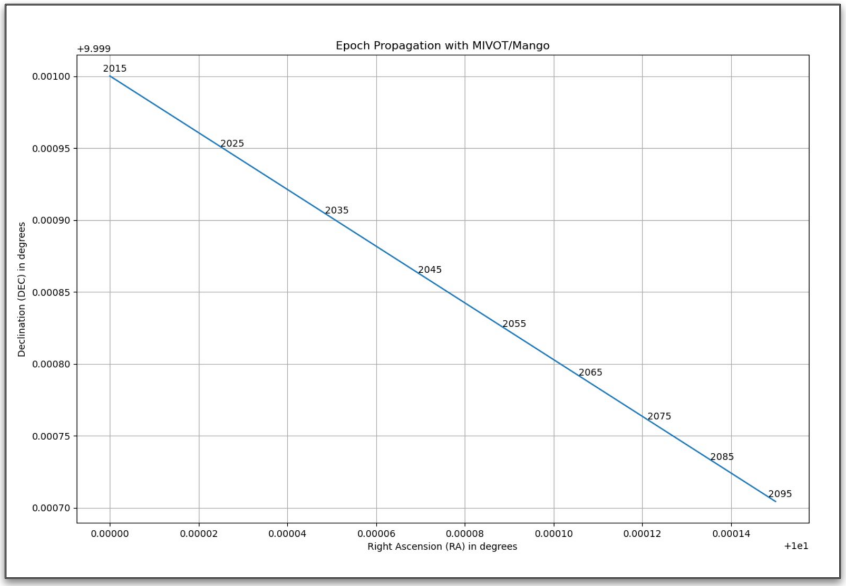
MIVOT: PyVO implementation: user point of view

- Go through the model view with Python object fields
- Field names match the model roles
  - Escape rules
- Based on public and documented models

```python
from pyvo.mivot.interpreter.model_view import ModelViewer

with ModelViewer("whatever-votable.xml") as m_viewer
    while (row_view := m_viewer.get_next_row_view):
        ra = row_view.EpochPostion.longitude.value
        dec = row_view.EpochPostion.latitude.value
        pm_ra = row_view.EpochPostion.longitude.value
        pm_dec = row_view.EpochPostion.latitude.value
        radial_velocity = row_view.EpochPostion.radialVelocity.value
        parallax = row_view.EpochPostion.parallax.value
        # Do whatever you want with those values
```

MIVOT: PyVO: Epoch Propagation Implementation

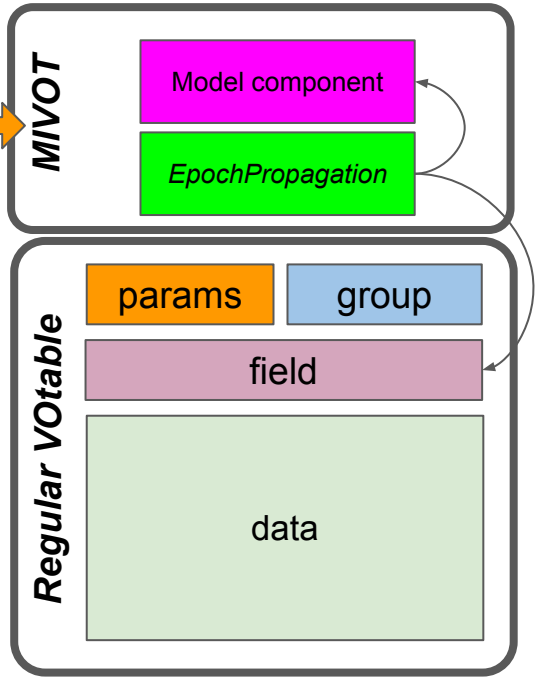
```python
import matplotlib.pyplot as plt
from pyvo.mivot.viewer.model_viewer import ModelViewer

years = np.arange(2015, 2030, 1)
with ModelViewer("path_to_my_votable") as m_viewer:
    # get the model view on the current data row
    row_view = m_viewer.get_next_row_view()
    # store the every year positions
    positions = [c.apply_space_motion(dt=year * u.year) for year in years]
    ra = [pos.icrs.ra.deg for pos in positions]
    dec = [pos.icrs.dec.deg for pos in positions]
    # Plot the object position over the years
    plt.figure(figsize=(14, 9))
    plt.plot(ra, dec)
    # do some plot polishing and exit
    break
```



Server Side Implementation

MIVOT block template

```
<VODML
    xmlns:dm-mapping="http://www.ivoa.net/xml/mivot">
    <MODEL name="mango" />
    <MODEL name="ivoa" url="https://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml" />
    <GLOBALS>
        <INSTANCE dmid="SpaceFrame_ICRS" dmtype="coords:SpaceSys">
            <INSTANCE dmrole="coords:PhysicalCoordSys.frame" dmtype="coords:SpaceFrame">
                <ATTRIBUTE dmrole="coords:SpaceFrame.spaceRefFrame" dmtype="ivoa:string" value="ICRS" />
            </INSTANCE>
        </INSTANCE>
    </GLOBALS>
    <TEMPLATES>
        <!-- One meas:Measure per row, no error -->
        <INSTANCE dmtype="mango:EpochPosition">
            <ATTRIBUTE dmrole="mango:EpochPosition.longitude" dmtype="ivoa:RealQuantity" ref="pos_RA"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.latitude" dmtype="ivoa:RealQuantity" ref="pos_DEC"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.pmLongitude" dmtype="ivoa:RealQuantity" ref="pm_RA"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.pmLatitude" dmtype="ivoa:RealQuantity" ref="pm_DEC"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.pmCosDeltApplied" dmtype="ivoa:boolean" value="true"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.radialVelocity" dmtype="ivoa:RealQuantity" ref="RV"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.parallax" dmtype="ivoa:RealQuantity" ref="PARALLAX"/>
            <ATTRIBUTE dmrole="mango:EpochPosition.epoch" dmtype="ivoa:RealQuantity" value="2015.0"/>
            <REFERENCE dmrole="coords:Coordinate.coosys" dmref="SpaceFrame_ICRS"/>
        </INSTANCE>
    </TEMPLATES>
</VODML>
```



Mapping rules

- Give the binding between table columns and model leaves
- Can be stored in the TAP_SCHEMA

- **The implementation is quite simple**
  - Do not alter the VOTABLE content
  - Just add a MIVOT block at the right place
- **Can easily extend existing services**
  - Can be done as a TAP query response post processing

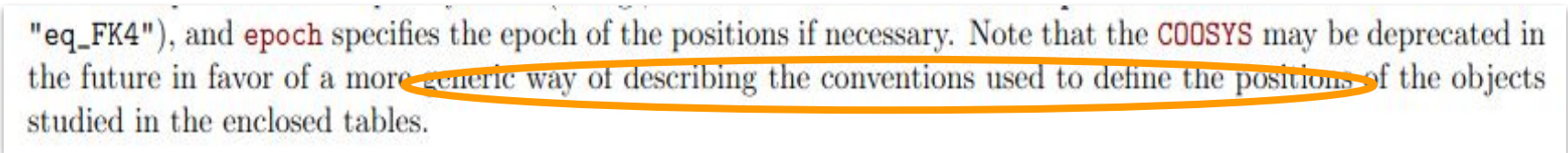
Conclusions

MIVOT+Mango: a seamless solution for the Epoch propagation

- Model supporting the complex errors
- Astropy/PyVO API
- Server side implementation preserving the original VOTable
- No change in the VOTable schema

The same mechanism can be used for many others quantities

- Versatile mapping syntax
- Photometric data
- Dataset meta data
- …


"eq_FK4"), and epoch specifies the epoch of the positions if necessary. Note that the COOSYS may be deprecated in the future in favor of a more generic way of describing the conventions used to define the positions of the objects studied in the enclosed tables.

Legacy: Connect Sky Position with a Space Frame

```
<COOSYS ID="J2000" equinox="J2000" epoch="J2000" system="eq_J2000"/>

<FIELD name="pos_RA" ucd="pos.eq.ra;meta.main" datatype="double" unit="deg" ref="J2000"/>
<FIELD name="pos_DEC" ucd="pos.eq.dec;meta.main" datatype="double" unit="deg" ref="J2000"/>
```

- Can see with the UCDs that **`RA`** and **`Dec`** do work together
- Both columns refer to the COOSYS element

- The `ref` attribute is used to quote another element of the document in the definition of a `FIELD` or `PARAM`. It is used in the example of section 3.1 to indicate the coordinate system in which the coordinates are expressed (reference to the `COOSYS` element which specifies the coordinate frame).

- The role of the **`@ref->@ID`** link is implicit
- **`@ref`** to what?

In the Hood

```
<instance dmtype="tucson:Position">
    <attribute dmrole="tucson:Position.ra"
        dmtype="ivoa:RealQuantity" value="10.876"
        ref="_RA2000"/>
    <attribute dmrole="tu
        dmtype="ivoa:Real
        ref="_DE2000"/>
</instance>
```

XML view transformed as a Py dict
MIVOT element names are removed
Only keep labels and values of interest

```
{
  "@dmtype": "tucson:Position",
  "tucson:Position.ra" : {
      "@dmtype": "ivoa:RealQuantity",
      "value": 10.876,
      "unit": "deg",
      "ref": "_RA2000"
  },
  "tucson:Position.dec" : {
      "@dmtype": "ivoa:RealQuantity",
      "value": -45.765,
      "unit": "deg",
      "ref": "_DEC2000"
  }
}
```

Py dict transformed as a dynamic Py object
Can be incorporated to the application logic

```
# class naming not defined yet
class MivotAttribute:
    def __init__(self, value, ref, unit)
        self.value = value
        self.ref = ref
        self.unit = unit

class Position:
    def __init__(self, ra, dec):
        self.ra = ra
        self.dec = dec
```

Work with the Column References

```
#get the model view
m_view m_viewer.get_next_row_view()

# Get the position from the model view
ra = m_view.EpochPropagation.longitude.value
dec = m_view.EpochPropagation.latitude.value

# get the column attached to a model leaf
colum_hosting_ra = m_view.EpochPropagation.longitude.ref
colum_hosting_dec = m_view.EpochPropagation.latitude.ref
```

The Python API give access to the reference of the columns that have been used to set attributes

```
# update the model view without redoing the parsing
table = votable.to_table()
for row in table:
    m_viewer.EpochPropagation.longitude.value
    = row[m_view.EpochPropagation.longitude.ref]
    m_viewer.EpochPropagation.latitude.value
    = row[m_view.EpochPropagation.latitude.ref]
```

This can be used to update thge Python object by skipping the parsing step

```
# get the next RA directly from the data row
table = votable.to_table()
for row in table:
    ra_value = row[m_view.EpochPropagation.longitude.ref]
    dec_value = row[m_view.EpochPropagation.latitude.ref]
```

This can be used to get attribute values without using the model view