Coosys, Mivot, AstroPyvo and Mango

-

Laurent MICHEL on the behalf of all DMers

Tucson Reminder (VOTable session - Apps)

Tucson 2024
Closing remarks

Mivot approach for the Epoch propagation L. Michel
Presentation of 4 approach to represent information of COOSYS

VOTable 1.5 Status T. Donaldson
Presentations of evolution and discussion on the process
we go for VOTable 1.5 and let COOSYS to 1.6

https://wiki.ivoa.net/internal/IVOA/InterOpNov2023CloseTCG/apps_cloture.pdf

CSP
Sydney 2024

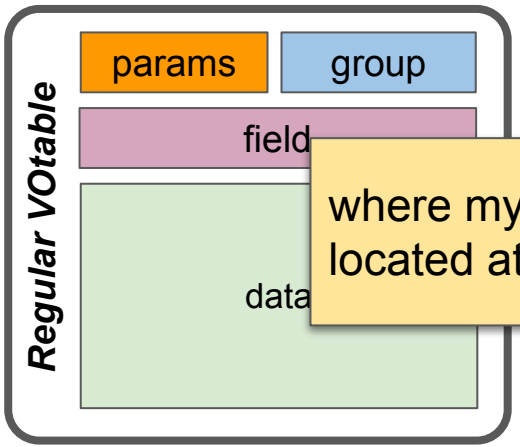
- Engagement with the community
  - Continuous work on PyVO:

→ Eg. improve the discoverability of astronomical data through the Registry in an end user client such as PyVO

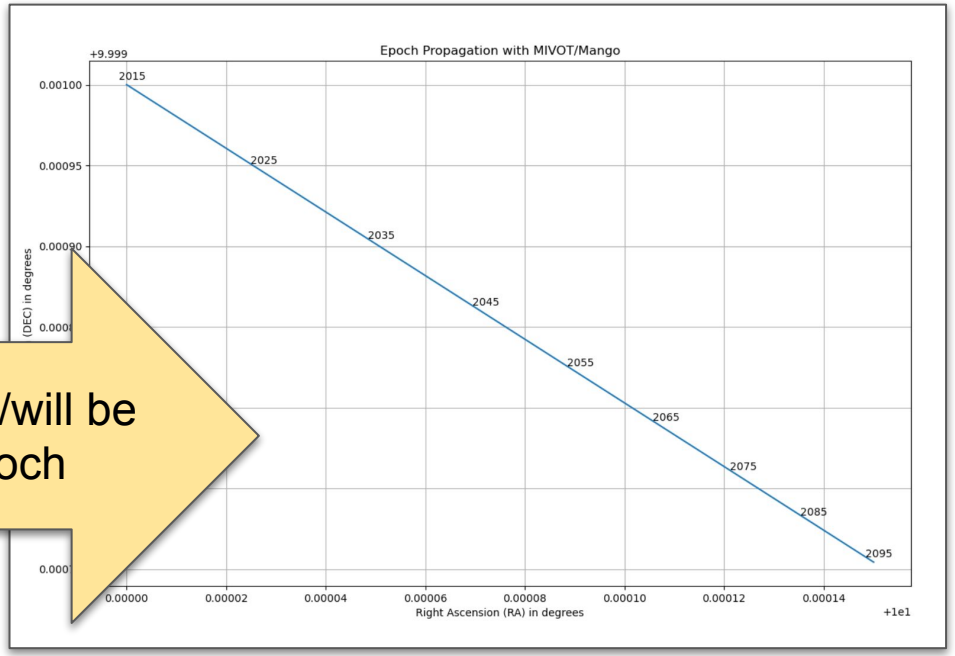
→ Support coordination activities to implement the MIVOT feature into astropy and PyVO Python packages

https://wiki.ivoa.net/internal/IVOA/InterOpMay2024/CSP-Presentation-interop-2024A_FC_2.pdf

The Challenge



where my object was/will be located at a given epoch



In this VOTABLE, I have somewhere:

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

The challenge: **Agree** on an appropriate **way to present data** to make the process **standard**

Legacy: Position + Proper Motion

```
<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"/>
<FIELD name="pm_RA" ucd="pos.pm.ra;meta.main" datatype="double" unit="mas/y" ref="J2000"/>
<FIELD name="pm_DEC" ucd="pos.pm.dec;meta.main" datatype="double" unit="mas/year" ref="J2000"/>
```

- Can see with UCDs that **`pos_RA`** and **`pos_DEC`** do work together
- Can see with UCDs that **`pm_RA`** and **`pm_DEC`** do work together
- The 4 columns refer to the **`COOSYS`** element

- The role of the **`@ref->@ID`** link is implicit (**`@ref`** to what?)
- No clear way to see that **`pos_RA/pos_DEC`** and **`pm_RA/pm_DEC`** relate to the same quantity

Legacy:Limitation of the COOSYS solutions

- The `@ref` semantic or **role is not well defined**
- The way fields connected to `COOSYS` **do interact** together is **poorly defined**
- **No** convenient way to show up **errors with correlations**
- According the the VOTable document, solution based on `COOSYS` are **short term solution**

> 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.

- We **miss a model** showing how all of these **parameters** do **work together.**

MIVOT: The Model Based Alternative

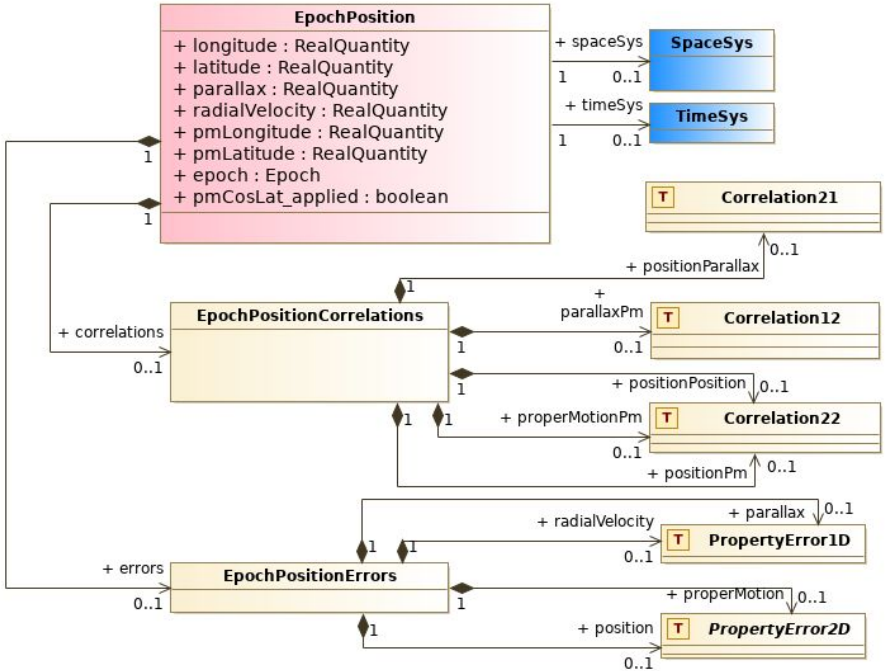
0. **Keep the VOTable as delivered by the DAL engine**
   - No FIELDREF tuning
   - No GROUP to create

1. **Use a VO model describing the EPOCH propagation**
   - 6 parameters
   - Errors
   - Space Frame
   - Time System

2. **Write an XML serialization of that MODEL in the VOTable**
   - MIVOT syntax

3. **Insert that XML piece above the TABLE**
   - Put the reference of the matching FIELDs into the model leaves

1 The Daunting Step: Build a Model.



- The role of any component is perfectly defined in the model

- The class is part of MANGO draft
  - DM2 Thursday
  - The MANGO overview is not shown here
- Import coordinate systems from **`Coords`**
  - For Space and time axis
- Support complex errors
  - Per parameter errors
  - Axis errors
  - Correlated errors
- No need to use all the features proposed by the model
  - only use model elements that match data

② *MIVOT*: 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="meas" url="https://www.ivoa.net/xml/Meas/20200908/Meas-v1.0.vo-dml.xml" />
    <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 *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 *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

MIVOT: Design Baselines

- **Easy to build**
  - The annotation **structure** only **depends** on the **model**
  - It does **not depends** on the way **data are arranged**
- **Easy to parse**
  - A simple XPath query returns a whole object

```
epoch_propagation = mivot_xmltree.findall(
        "INSTANCE[@dmtype='mango:EpochPropagation]']"
        )
```

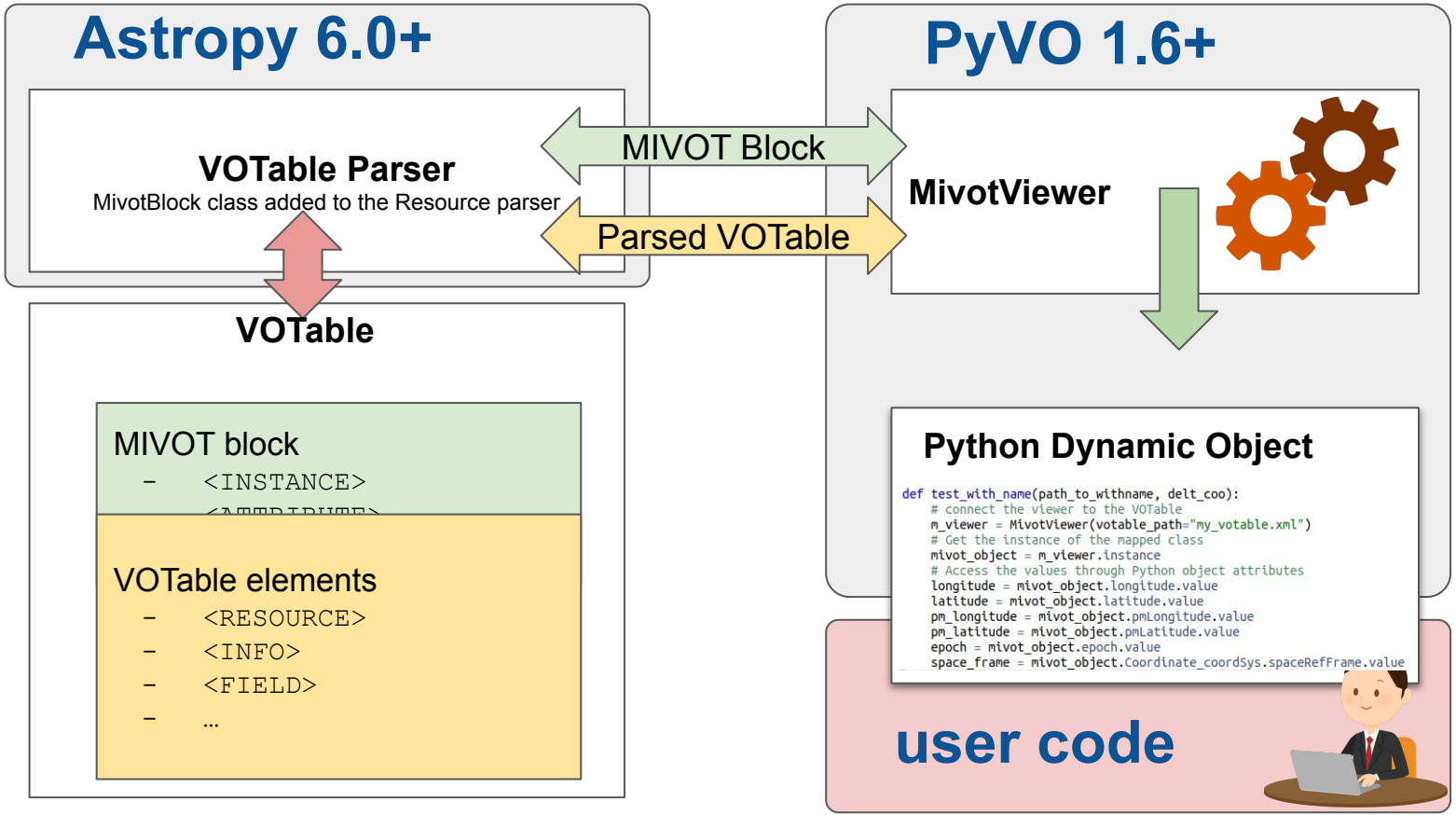
5 Server Side: Vizier ConeSearch (G. Landais)

- **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&amp;DEC=59.94027&amp;SR=0.05'
```

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



Run the Notebook

```
% git clone git@github.com:astropy/pyvo.git

% git clone git@github.com:ivoa/dm-usecases.git

% cd dm-usecases/notebooks

% jupyter notebook
```

Python Implementation (L. Michel, G. Landais, S. Floret)

```python
votable = "../pyvo-ci-sample/gaia_epoch_propagation_full.xml"
# init the MIVOT viewer from a VOTable file, parsedVOTavle od DAL response
mivot_viewer = MivotViewer(votable)

# get the reference on the dynamic MIVOT instance
mivot_object = mivot_viewer.dm_instance
```

- **`MivotViewer`: annotation parser**
  - Parse the annotations
  - Iterate over the table rows
  - Apply the mapping on data rows

- **`MivotInstance`: dynamic Python object**
  - Dynamic object whose structure matches the model and whose attribute are set with the values in the current row.

4 Browse the Model View through a Python Object

```
vizier_url = "https://cdsarc.cds.unistra.fr/beta/viz-bin/mivotconesearch/I/239/hip_main"
scs_srv = SCSService(vizier_url)

# init the viewer from a DAL response.
mivot_viewer = MivotViewer(
    scs_srv.search(
        pos=SkyCoord(ra=52.26708 * u.degree, dec=59.94027 * u.degree, frame="icrs"),
        radius=0.05,
    )
)

# get the reference on the MIVOT instance
mivot_object = mivot_viewer.dm_instance
while mivot_viewer.next():
    # get the space frame reference (could be done out of the loop)
    frame = mivot_object.Coordinate_coordSys.spaceRefFrame.value
    # get the position values and units
    ra = mivot_object.longitude.value
    ra_unit = mivot_object.longitude.unit
    dec = mivot_object.latitude.value
    # get the proper motion values and units
    pmra = mivot_object.pmLongitude.value
    pmra_unit = mivot_object.pmLongitude.unit
    pmdec = mivot_object.pmLatitude.value
    # get the epoch
    epoch =  mivot_object.epoch.value
    # Print out a summary of the row
    print(f"Year {epoch}: position({frame})=[{ra} {dec} {ra_unit}] proper motion = [{pmra} {pmdec} {pmra_unit}]")
```

- **MIVOT instance set from SCS output**
  - Done backstage by the MIVOT viewer
  - Access model components as Python object attributes

```
Year 1991.25: position(ICRS)=[52.26722684 59.94033461 deg] proper motion = [-0.82 -1.85 mas/yr]
```

Apply the Space Motion

```
position = SkyCoord(ra*u.deg, dec*u.deg,
                    frame=frame.lower(),
                    pm_ra_cosdec=pmra*u.mas/u.year, pm_dec=pmdec*u.mas/u.year,
                    obstime=Time(epoch, format='decimalyear', scale='utc'))
dt = 10. * u.year

for cpt in range(5):
    print(f"year {position.obstime} {position.to_string(style='hmsdms', sep=':', precision=6)}")
    position = position.apply_space_motion(dt=dt)
```

```
year 1991.25 03:29:04.134442 +59:56:25.204596
year 2001.2486299467275 03:29:04.133350 +59:56:25.186096
year 2011.2499997463217 03:29:04.132259 +59:56:25.167596
year 2021.2486297881785 03:29:04.131168 +59:56:25.149096
year 2031.2499996511922 03:29:04.130076 +59:56:25.130596
```

- **Apply the Astropy space motion to the MIVOT instance (set above)**
  - To be added to the package soon

Apply the Mapping on Numpy Data Rows

```
# Iterate over all data rows
for rec in table.array:
    # update the instance with the new data row
    mivot_instance.update(rec)
    # get the position values and units
    ra = mivot_instance.longitude.value
    ra_unit = mivot_instance.longitude.unit
    dec = mivot_instance.latitude.value
    # get the proper motion values and units
    pmra = mivot_instance.pmLongitude.value
    pmra_unit = mivot_instance.pmLongitude.unit
    pmdec = mivot_instance.pmLatitude.value
    # get the epoch
    epoch =  mivot_instance.epoch.value
    # Print out a summary of the row
    print(f"Year {epoch}: position({space_frame}) = [{ra} {dec} {ra_unit}] proper motion = [{pmra} {pmdec} {pmra_unit}]")
```

MivotInstances can be updated with Numpy data

```
Year 2016.5: position(ICRS) = [307.79115807079 20.43108005561 deg] proper motion = [-2.557 -5.482 mas/yr]
Year 2016.5: position(ICRS) = [307.79582391363 20.43253083107 deg] proper motion = [-2.78 -3.853 mas/yr]
Year 2016.5: position(ICRS) = [307.79737366047 20.43558827154 deg] proper motion = [None None mas/yr]
Year 2016.5: position(ICRS) = [307.78856137441 20.43011713943 deg] proper motion = [-2.781 -3.61 mas/yr]
Year 2016.5: position(ICRS) = [307.78977228741 20.43150439876 deg] proper motion = [-1.975 -3.712 mas/yr]
```

Get the MIVOT Object as a Python Dict

```
# get the reference on the dynamic MIVOT instance
mivot_object = mivot_viewer.dm_instance

# iterate over one data row
while mivot_viewer.next():
    # print the dictionary that has been use dto buil
    print(mivot_object)
    break
```

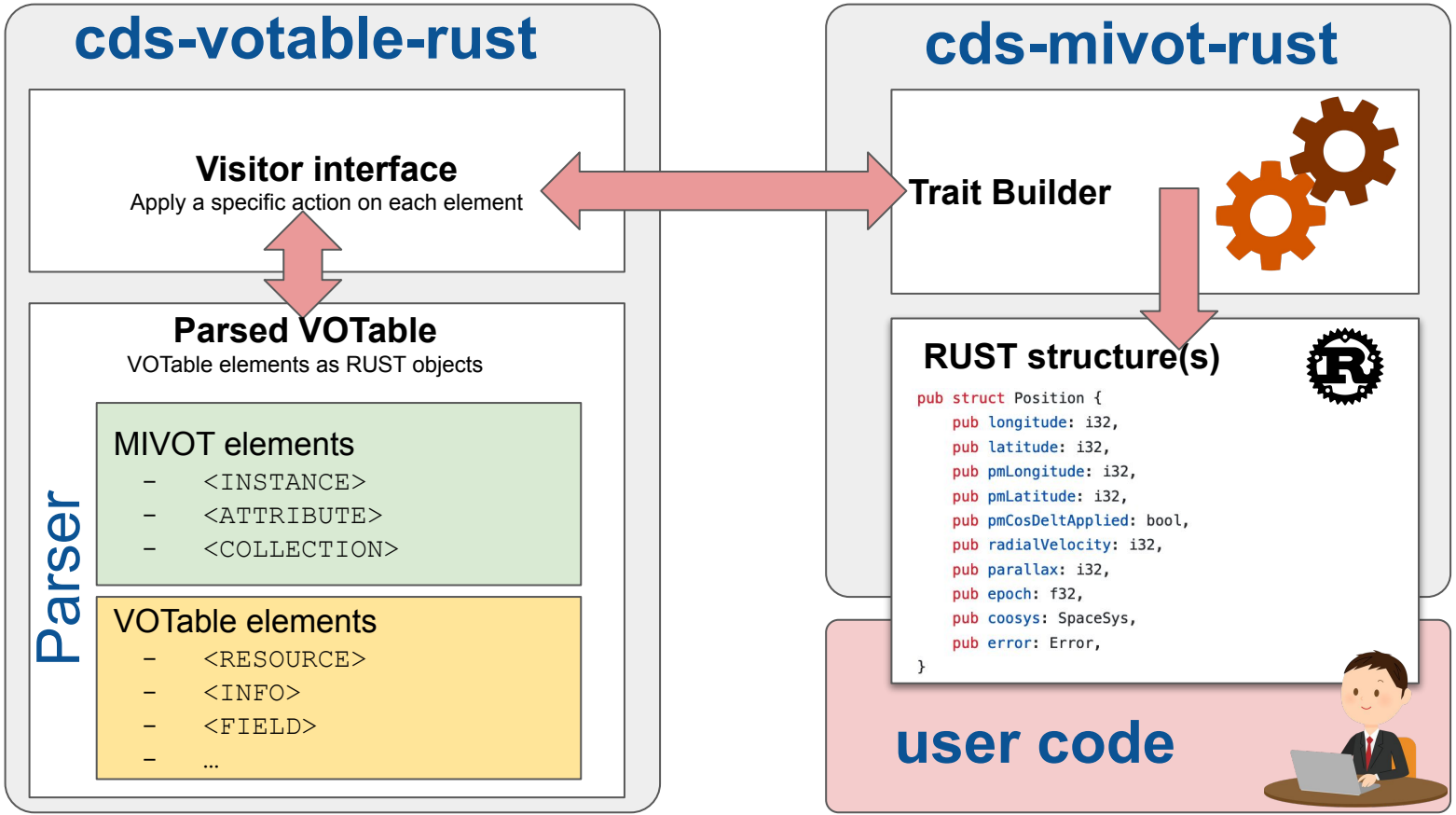

```
{
  "dmtype": "EpochPosition",
  "longitude": {
    "value": 307.79582391363,
    "unit": "deg"
  },
  "latitude": {
    "value": 20.43253083107,
    "unit": "deg"
  },
  "parallax": {
    "value": -0.5553,
    "unit": "mas"
  },
  "radialVelocity": {
    "value": null,
    "unit": "km/s"
  },
  "pmLongitude": {
    "value": -2.78,
    "unit": "mas/yr"
  },
  "pmLatitude": {
    "value": -3.853,
    "unit": "mas/yr"
  },
```


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)



What's Next?

- **Let's continue with the PyVO API**
  - Automatically build `SkyCoord` instances from the VOTable

- **A stable MANGO version**
  - Requested to complete the RUST API
  - Requested to have compliant data sources
  - https://github.com/ivoa-std/MANGO

- **MIVOT Resources**
  - Standard
    - https://ivoa.net/documents/MIVOT/20230620/index.html
  - Online examples (J. Abid)
    - https://saada.unistra.fr/voexamples/show/MIVOT-Syntax
  - Validateur (L. Michel, J. Abid, M. Louys)
    - https://github.com/ivoa/mivot-validator

Conclusions

MIVOT+Mango: a seamless solution for the Epoch propagation

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

The same pattern can be apply to any other quantity

- Versatile mapping syntax
- Incoming models
  - Photometric data
  - Dataset metadata
- …

… And many thanks to the AstroPyvo teams for their support and the involvement in this huge PR effort.

"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.

Backup

Where do we stand?

- **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

MIVOT: A VO Standard


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 Server Side: 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>
```

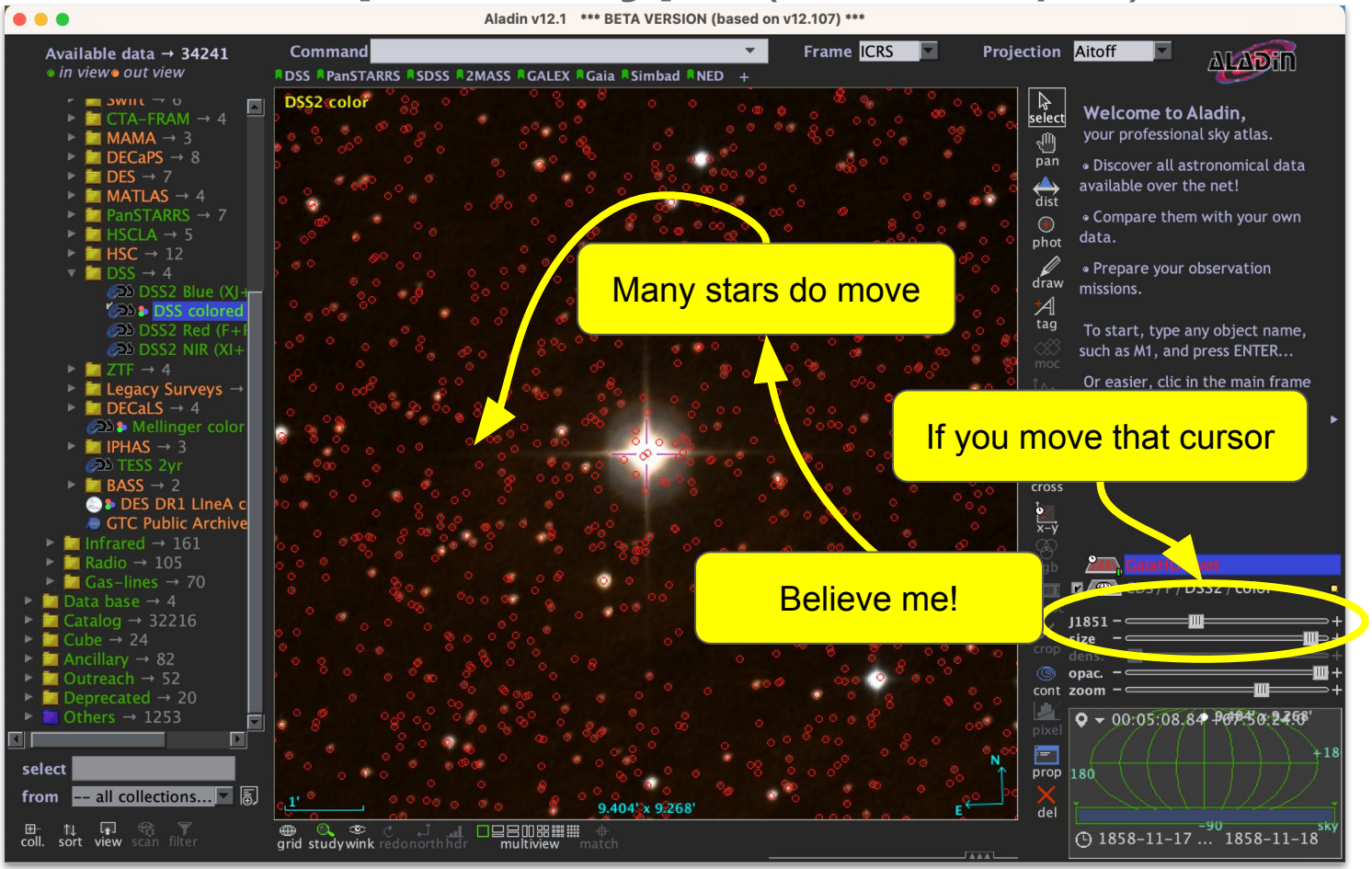
Validate an XML file containing just a MAPPING block

```
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 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

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

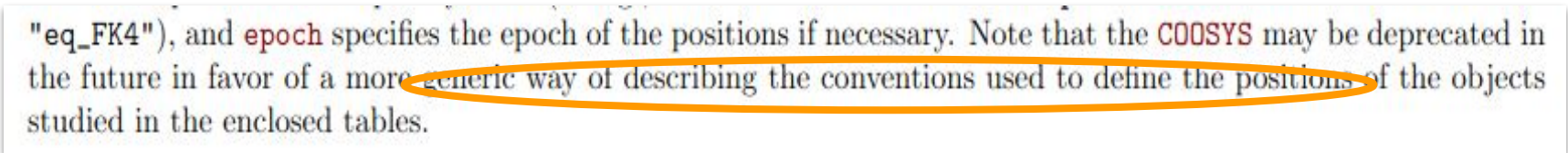
Conclusions

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.

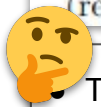
Legacy: Connect Sky Position with a Space Frame

```
<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?

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

Legacy: Solution 2: add FIELDREF into COOSYS

On behalf of François B.

- The Epoch propagation components are packed **into a GROUP** element
- The **GROUP** role is set with a UType reusing names of VO model elements
- The **GROUP** refers to the **COOSYS**

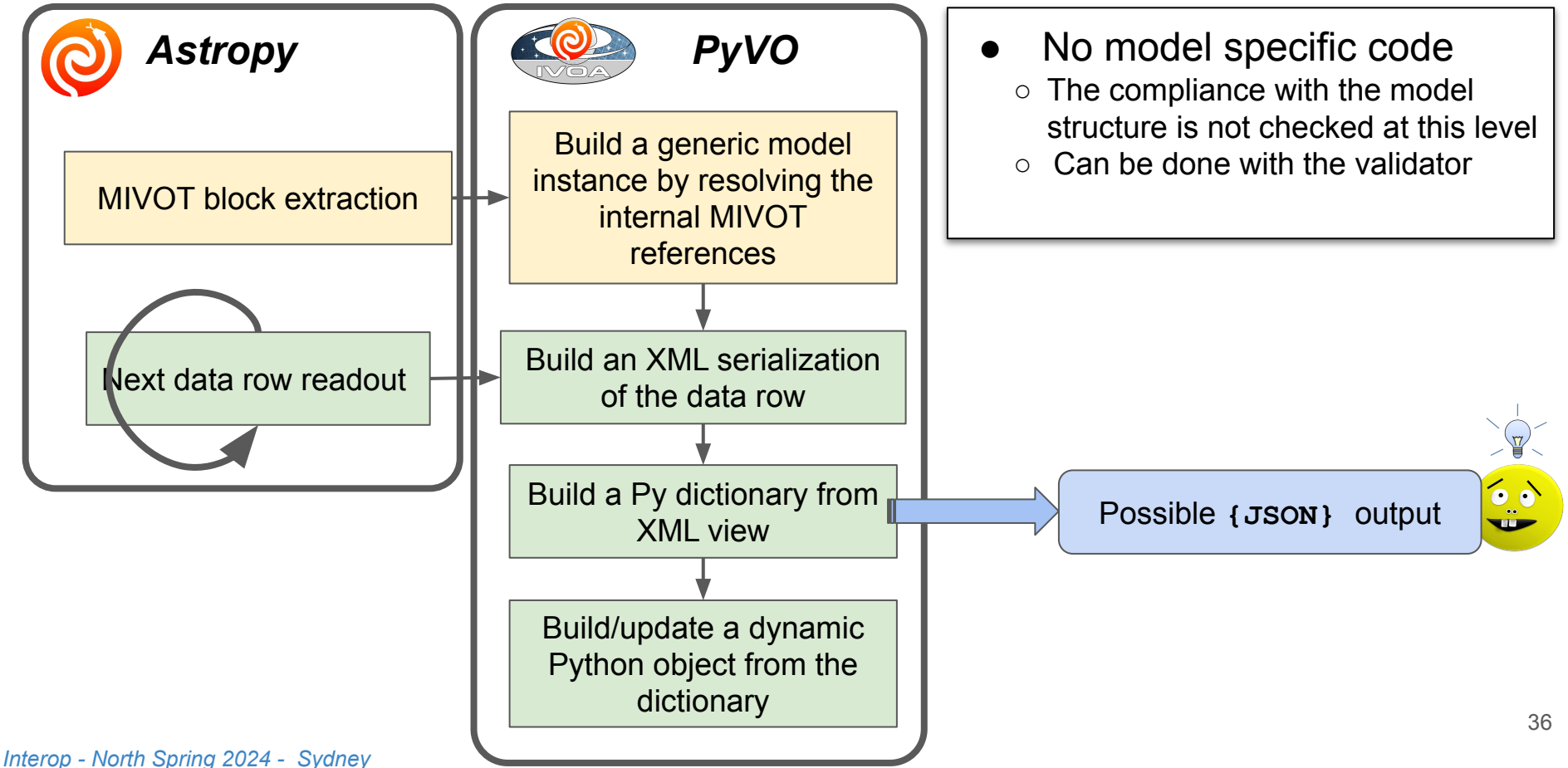
```
<COOSYS ID="J2000" equinox="J2000" epoch="J2000" system="eq_J2000"/>
<GROUP utype="demo:epoch.propagation" ref="J2000">
  <FIEDREF ref="pos_RA" ucd="pos.eq.ra;meta.main"/>
  <FIEDREF ref="pos_RA" ucd="pos.eq.dec;meta.main"/>
  <FIEDREF ref="pm_RA" ucd="pos.pm.ra;meta.main"/>
  <FIEDREF ref="pm_DE4C" ucd="pos.pm.dec;meta.main" />
  <FIEDREF ref="RV" ucd="spect.dopplerVeloc;pos.heliocentric"/>
  <FIEDREF ref="PARALLAX" ucd="pos.parallax"/>
</GROUP>
<FIELD name="pos_RA" ucd="pos.eq.ra;meta.main" datatype="double" unit="deg" />
<FIELD name="pos_DEC" ucd="pos.eq.dec;meta.main" datatype="double" unit="deg" />
<FIELD name="pm_RA" ucd="pos.pm.ra;meta.main" datatype="double" unit="mas/y" />
<FIELD name="pm_DEC" ucd="pos.pm.dec;meta.main" datatype="double" unit="mas/year" />
<FIELD name="RV" ucd="spect.dopplerVeloc;pos.heliocentric" datatype="double" unit="km/s"/>
<FIELD name="PARALLAX" ucd="pos.parallax" datatype="double" unit="mas" ref="J2000"/>
```

- **No change** in the VOTable **schema**
- **No bi-directional** links
- Can have **multiple GROUPS** for multiple parameter sets
- **Easy to implement** in actual parsers

- **Alternative** model mapping syntax **less featured** than MIVOT
- Do not rely on any documented data model
- Complex errors **not supported**
  - Covariance, correlation
- **Short term** solution

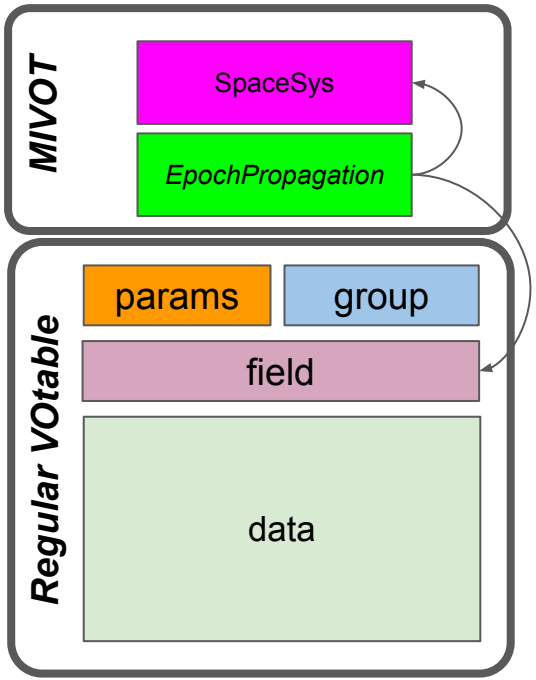
Note that the COOSYS may be deprecated in the future in favor of a more generic way of describing the conventions used to define the positions of the objects studied in the enclosed tables.

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: MIVOT annotations



- **MIVOT block: an XML model view**
  - Above the data table
  - The hierarchy of the XML elements matches 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