# Simplifying VO standards for improving interoperability

Fabien Chéreau – Virtual Observatory Program office - ESO

October-29-2008

## My story in the VO world

- I work since 2 years on creating a visual browser (VirGO) using VO standards.
  - Coded a testing SIA server
  - Coded SIA/SSA client
- SIA is reasonably simple, however... interop problems arose rapidly

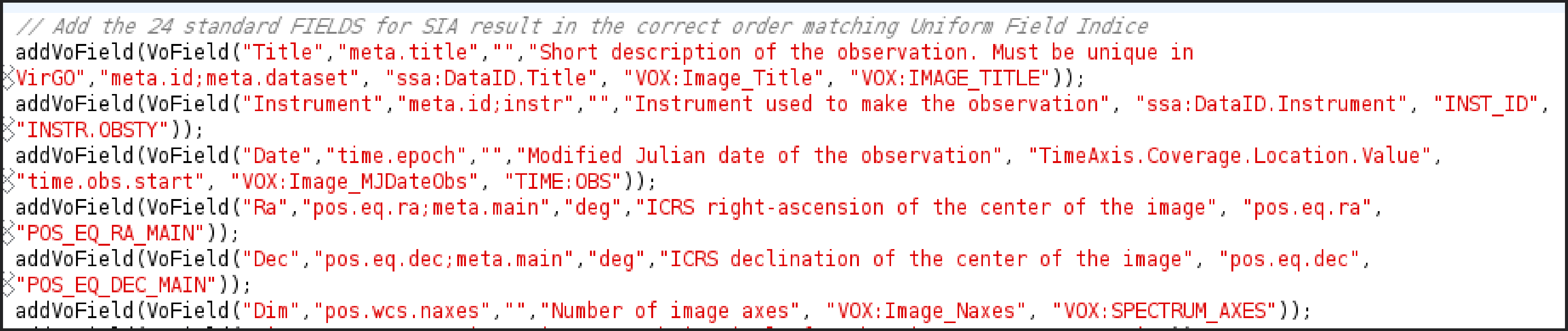
- My first (naive!) VOTable/SIA parser:
  - Generated automatically in C++ by a tool (xsdcpp) from the VOTable Schema
    - Xml validation + strict parsing
    - Just 3 servers could be used: ESO S*A and ST-ECF SSA
  - Why?
    - VOTable format generally not fully respected
    - Ambiguities in SIA standard: I reported some of them in http://wiki.eurovotech.org/twiki/bin/view/VOTech/SiaComments

- My second (pragmatic) VOTable/S*A parser:
  - Coded by hand to be as permissible as possible
    - No validation, no checks, in case of ambiguities/incompatible versions of the standards, try all solutions in the code.
    - Still not a single external S*A could be used without code change!!
  - Why?
    - Mismatching UCD/UTypes names
    - Mismatching Units names
    - + unsupported reference frame
    - etc...

- Why is interop not working properly:
  - 1. Engineers lack resources
  - 2. The current VO standards are not engineers-friendly
    - 2.1 “OK, but we cannot change that, the problems are by nature very complex.”
    - 2.2 OK, but still, could we make our life easier?

## 3 proposals to make VO standards engineers-friendly

- The UCD/UTypes as seen by an engineer
  - In the code, they are both used as simple key string
    - The usage is not rigorous: ad hoc code is literally made for each server (≠interoperability)



```
// Add the 24 standard FIELDS for SIA result in the correct order matching Uniform Field Indice
addVoField(VoField("Title","meta.title","","Short description of the observation. Must be unique in
VirGO","meta.id;meta.dataset", "ssa:DataID.Title", "VOX:Image_Title", "VOX:IMAGE_TITLE"));
addVoField(VoField("Instrument","meta.id;instr","","Instrument used to make the observation", "ssa:DataID.Instrument", "INST_ID",
"INSTR.OBSTY"));
addVoField(VoField("Date","time.epoch","","Modified Julian date of the observation", "TimeAxis.Coverage.Location.Value",
"time.obs.start", "VOX:Image_MJDateObs", "TIME:OBS"));
addVoField(VoField("Ra","pos.eq.ra;meta.main","deg","ICRS right-ascension of the center of the image", "pos.eq.ra",
"POS_EQ_RA_MAIN"));
addVoField(VoField("Dec","pos.eq.dec;meta.main","deg","ICRS declination of the center of the image", "pos.eq.dec",
"POS_EQ_DEC_MAIN"));
addVoField(VoField("Dim","pos.wcs.naxes","","Number of image axes", "VOX:Image_Naxes", "VOX:SPECTRUM_AXES"));
```

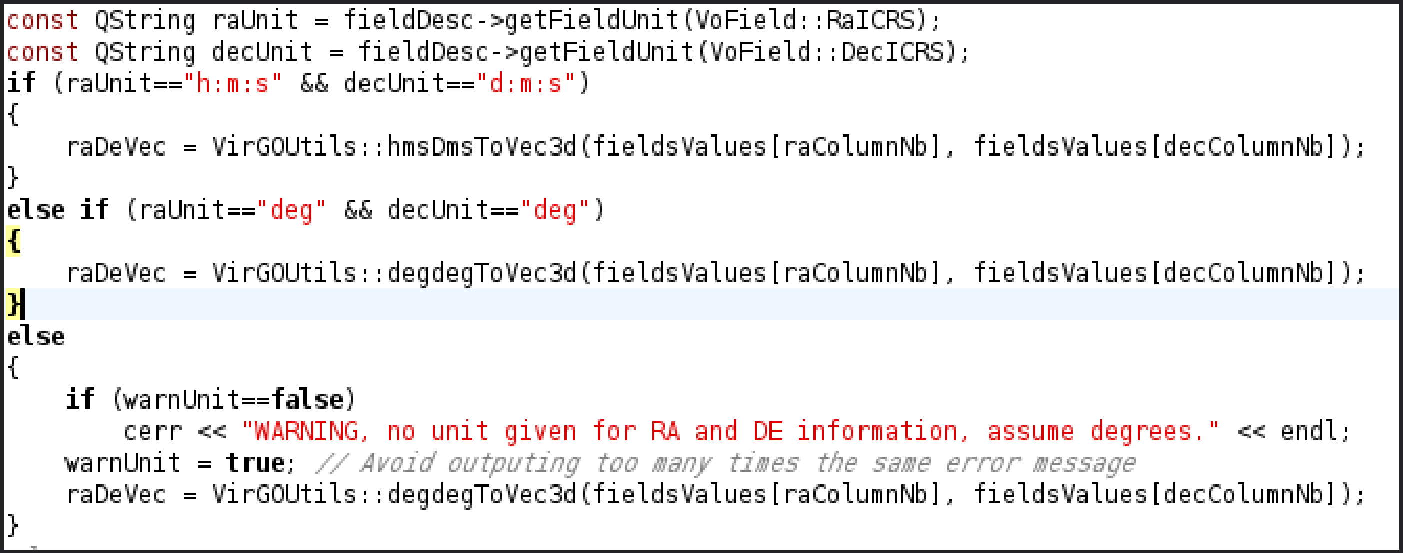
- 1. Never rely on UCD. Replace Utypes by pragmatic simple readable keys (but carefully chosen and precisely defined).

```
<characterizationAxis>
        <axisName>spatial</axisName>
                    <coordsystem id="TT-ICRS-TOPO"
xlink:type="simple" xlink:href="ivo://STClib/CoordSys#TT-ICRS-
TOPO"/>
                <coverage>
                    <location>
                      <coord coord_system_id="TT-ICRS-TOPO">
                       <stc:Position2D>
                        <stc:Name1>RA</stc:Name1>
                        <stc:Name2>Dec</stc:Name2>
                        <stc:Value2>
                           <stc:C1>
                              308.655620
                           </stc:C1>
                           <stc:C2>
                              60.211775
                           </stc:C2>
                         </stc:Value2>
                       </stc:Position2D>
                      </coord>
                    </location>
....
```

```
<characterizationAxis>
        <spatialAxis>
            <coordsystem id="TT-ICRS-TOPO" xlink:type="simple"
xlink:href="ivo://STClib/CoordSys#TT-ICRS-TOPO"/>
            <centralPosition unit="deg">308.655620, 60.211775</centralPosition>
....
```

For an engineer, it doesn't have to be matched 1-1 with a data-model.

- The Reference Frames and Units as seen by an engineer
  - Usually let free in the standard
    - Must code all possible conversions in EACH VO clients!
  - Understanding the standards requires advanced astronomy knowledge



```
const QString raUnit = fieldDesc->getFieldUnit(VoField::RaICRS);
const QString decUnit = fieldDesc->getFieldUnit(VoField::DecICRS);
if (raUnit=="h:m:s" && decUnit=="d:m:s")
{
    raDeVec = VirGOUtils::hmsDmsToVec3d(fieldsValues[raColumnNb], fieldsValues[decColumnNb]);
}
else if (raUnit=="deg" && decUnit=="deg")
{
    raDeVec = VirGOUtils::degdegToVec3d(fieldsValues[raColumnNb], fieldsValues[decColumnNb]);
}
else
{
    if (warnUnit==false)
        cerr << "WARNING, no unit given for RA and DE information, assume degrees." << endl;
    warnUnit = true; // Avoid outputing too many times the same error message
    raDeVec = VirGOUtils::degdegToVec3d(fieldsValues[raColumnNb], fieldsValues[decColumnNb]);
}
```

- 2. Fix by convention the reference frames and units for all standardized descriptors. Please no freedom!
    - E.g. centralPosition must be in ICRS(deg), time must be in TT (MJD), etc..
    - A descriptor not compliant, e.g. galactic pos could be defined in a different axis, e.g. “spaceAxisGalactic”
    - *It is much easier to converge on a realistic technical solution when you actually have to implement it!*

```
<characterization>
        <spatialAxis>
            <coordsystem id="TT-ICRS-TOPO" xlink:type="simple"
xlink:href="ivo://STClib/CoordSys#TT-ICRS-TOPO"/>
            <centralPosition unit=”deg”>308.655620,
60.211775</centralPosition>
....
    </spatialAxis>
</characterization>
```

```
<characterization>
        <spatialAxis>
            <centralPosition>308.655620, 60.211775</centralPosition>
....
        </spatialAxis>
</characterization>
```

- 3. Consider using simpler serializations. E.g. JSON
  - Extremely easy to parse and write, map directly in computer memory (no need for xml DOM)
  - Light weight
  - Fast learning curve (human readable)
  - Ready to use for Ajax applications (JSONP)

```
<characterization>
        <spatialAxis>
            <centralPosition>308.655620, 60.211775</centralPosition>
....
        </spatialAxis>
</characterization>
```

```
“characterization”: {
        “spatialAxis”:{
            “centralPosition”: [308.655620, 60.211775],
            ....
           }
}
```

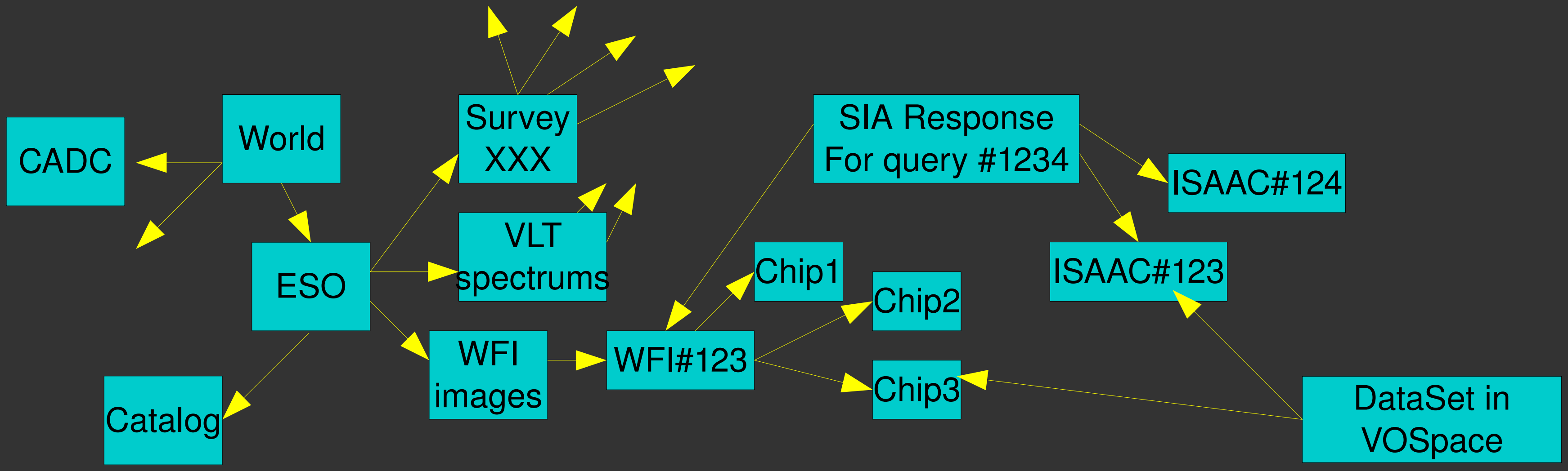
Utype = characterization.spatialAxis.centralPosition(0)

## Summary

- We must recognize that there is a serious inter-operability problem and try to improve that.
- Simple designs for engineers produce robust applications for astronomers.
- This is urgent: we are currently defining the corner stone of the future VO: a serialization of the observation DM (Generic DataSet)

## Why will a generic DataSet description file format be the corner stone of the VO

- World astronomical data form a big graph structure. This file format is for VO what HTML is for the web.
- A logical group of datasets **IS** a dataset



Thank you!

Questions?