VIRTUAL ASTRONOMICAL OBSERVATORY

UTYPEs

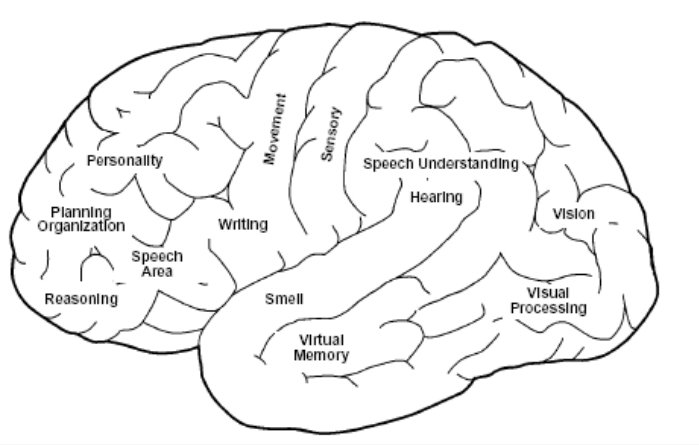
the art of moving objects around

Omar Laurino
SAO/VAO
UTYPEs Tiger Team

“Where is the science here?”

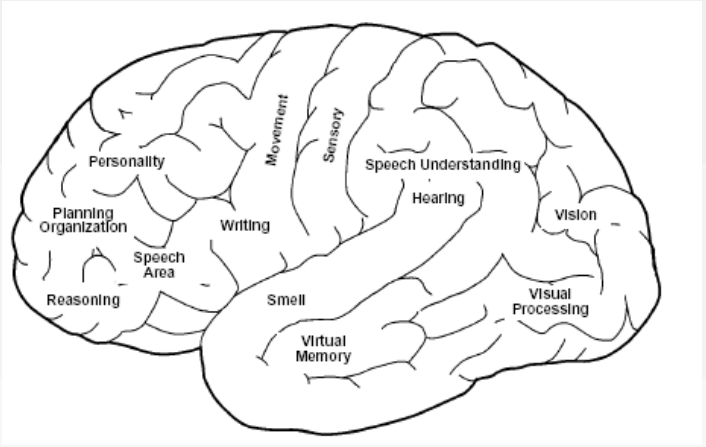
“Where is science here?”

Where is the General Relativity Theory here?

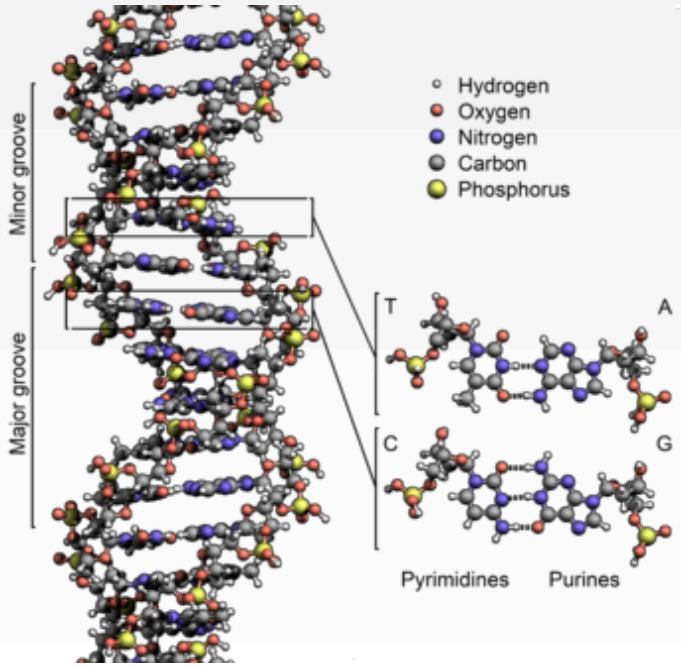


"Where is science here?"

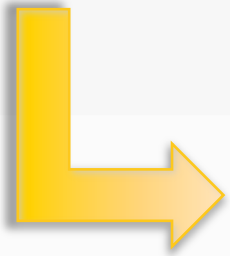
Where is the General Relativity Theory here?

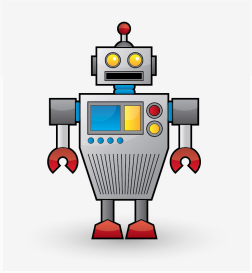


Or here?



A relocation robot





The Problem

- Problem:
  - **Tag objects (and boxes?) with labels, so that a robot can (un)pack**

The Problem

- Problem:
  - **Tag objects (and boxes?) with labels, so that a robot can (un)pack**

- Some constraints:
  - You want to make sure that you can tell different clients' objects apart:
    - "This is Omar's mug"; "This is Jesus's mug"
  - You want to be able to query the inventory in the robot's memory:
    - "How many rooms are there in Pat's house?"
    - "Get the mug on the coffee table in Jesus's Living room"
  - You want to minimize your effort: the same "algorithm" must work with all clients. You don't want ad hoc solutions.
  - When the robots unpack, you want them to place objects exactly where they were.

The solutions

- Problem:
  - **Tag objects (and boxes?) with labels, so that a robot can (un)pack**

- Solution 1:

omar:Kitchen.Shelf.Mug

jesus:LivingRoom.CoffeeTable.Mug

…

The solutions

- Problem:
  - **Tag objects and boxes with labels, so that a robot can (un)pack**

- Solution 1:

omar:Kitchen.Shelf.Mug

jesus:LivingRoom.CoffeeTable.Mug

…

- Solution 2:

Type = pottery:Mug

Role = omar:Shelf.mug (in omar:Kitchen box)

Role = jesus:CoffeeTable.mug (in jesus:LivingRoom box)

…

Testing for Generality

Testing for generality

- Problem: **Ask a robot to find a mug (“pottery:Mug”) in Pat’s house**

Testing for generality

- Problem: **Ask a robot to find a mug (type="pottery:Mug") in Pat's house**
- Solution 1:

pat:DiningRoom.Table.Mug

- Solution 2:

Type = pottery:Mug

Role = pat:Table.redMug (in pat:DiningRoom box)

Testing for generality

- Problem: **Ask a robot to find a mug (type="pottery:Mug") in Pat's house**
- Solution 1:

pat:DiningRoom.Table.Mug

It works w/ a LabelScanner and a formal Grammar, a Vocabulary, a Parser.

Testing for generality

- Problem: **Ask a robot to find a mug (type="pottery:Mug") in Pat's house**
- Solution 1:

pat:DiningRoom.Table.Mug

It works w/ a LabelScanner and a formal Grammar, a Vocabulary, a Parser.

- Solution 2:

Type = pottery:Mug

Role = pat:Table.redMug (in pat:DiningRoom box)

It works with a LabelScanner and a string equality check.

Testing for generality

- Problem: **Ask a robot to find a mug (type="abc:def") in Pat's house**
- Solution 1:

wre:rt45wh.2wdf5t.prtg4

It works w/ a LabelScanner and a formal Grammar, a Vocabulary, a Parser.

- Solution 2:

Type = abc:def

Role = wre:2wdf5t.qazwsx (in wre:rt45wh box)

It works with a LabelScanner and a string equality check.

Let’s finally go back to Astronomy

Let's compare the two solutions

- Problem: **Find the SDSS.g magnitude:**

"path utypes"

```
<GROUP>
  <FIELDref utype="phot:Catalog.PhotometryPoint.Value" ref="refField1"/>
  <FIELDref utype="phot:Catalog.PhotometryPoint.Filter" ref="refGroup1"/>
 </GROUP>
```

"type&role utypes"

```
<GROUP utype="sdss+2mass:Catalog.sdssG">
    <PARAM utype="Instance.type" value="phot:PhotometryPoint"/>
    <FIELDref utype="phot:PhotometryPoint.value" ref="refField1"/>
    <GROUP utype="phot:PhotometryPoint.filter" ref="refGroup1"/>
</GROUP>
```

Still too abstract, right?
Let's see what happens in practice

Finding mugs, in practice

A Python library can read a VOTable according to the two specs:

>> votable = volib.read(“sdss_2mass_xmatch.vot”)

Finding mugs, in practice

A Python library can read a VOTable according to the two specs:

```
>> votable = volib.read("sdss_2mass_xmatch.vot")
```

Solution 1 ("path" utypes):

```
>> g_array = list()
>> for group in votable.get_groups(): // let's assume there are Groups
       filter = group.get_by_utype("phot:Catalog.PhotometryPoint.Filter")
       if filter.name == "sdssG": // let's assume this is standardized
           for row in votable.get_rows():
             g = PhotometryPoint()
             g.value = row.get_by_utype("phot:Catalog.PhotometryPoint.Value")
             g.error = row.get_by_utype("phot:Catalog.PhotometryPoint.Error")
             g_array.append(g)
```

Finding mugs, in practice

A Python library can read a VOTable according to the two specs:

```
>> votable = volib.read("sdss_2mass_xmatch.vot")
```

Solution 1 ("path" utypes):

```
>> g_array = list()
>> for group in votable.get_groups(): // let's assume there are Groups
       filter = group.get_by_utype("phot:Catalog.PhotometryPoint.Filter")
       if filter.name == "sdssG": // let's assume this is standardized
           for row in votable.get_rows():
             g = PhotometryPoint()
             g.value = row.get_by_utype("phot:Catalog.PhotometryPoint.Value")
             g.error = row.get_by_utype("phot:Catalog.PhotometryPoint.Error")
             g_array.append(g)
```

Solution 2 ("role+type" utypes **+ VO-DML + mapping spec**):

```
>> g_array = votable.sdss_2mass.catalog.sdss_g
```

Finding mugs, in practice

Get all photometry points in a file

>> votable = volib.read("sdss_2mass_xmatch.vot")

Solution 1 ("path" utypes):
You can't, unless you parse utypes (e.g. phot:~~Catalog.~~PhotometryPoint.*)

Solution 2 ("role+type" utypes **+ VO-DML + mapping spec**):
>> points = votable.get_objects("phot:PhotometryPoint")

Why VO-DML is the “smart” robot

This utype points to the “sdss_g” attribute of the “Catalog” type
Python: sdss_2mass.catalog.sdss_g

```
<GROUP utype="sdss+2mass:Catalog.sdssG">
    <PARAM utype="Instance.type" value="phot:PhotometryPoint"/>
    <FIELDref utype="phot:PhotometryPoint.value" ref="refField1"/>
    <GROUP utype="phot:PhotometryPoint.filter" ref="refGroup1"/>
</GROUP>
```

Why VO-DML is the “smart” robot

This utype points to the “sdss_g” attribute of the “Catalog” type
Python: sdss_2mass.catalog.sdss_g

The sdss_g attribute **is a** PhotometryPoint type

```
<GROUP utype="sdss+2mass:Catalog.sdssG">
    <PARAM utype="Instance.type" value="phot:PhotometryPoint"/>
    <FIELDref utype="phot:PhotometryPoint.value" ref="refField1"/>
    <GROUP utype="phot:PhotometryPoint.filter" ref="refGroup1"/>
</GROUP>
```

Why VO-DML is the “smart” robot

This utype points to the “sdss_g” attribute of the “Catalog” type
Python: sdss_2mass.catalog.sdss_g

The sdss_g attribute **is a** PhotometryPoint type

```
<GROUP utype="sdss+2mass:Catalog.sdssG">
    <PARAM utype="Instance.type" value="phot:PhotometryPoint"/>
    <FIELDref utype="phot:PhotometryPoint.value" ref="refField1"/>
    <GROUP utype="phot:PhotometryPoint.filter" ref="refGroup1"/>
</GROUP>
```

This utype points to the “value” attribute of the “PhotometryPoint” type
Python: sdss_2mass.catalog.sdss_g.value

Why VO-DML is the “smart” robot

This utype points to the “sdss_g” attribute of the “Catalog” type
Python: sdss_2mass.catalog.sdss_g

The sdss_g attribute **is a** PhotometryPoint type

```
<GROUP utype="sdss+2mass:Catalog.sdssG">
    <PARAM utype="Instance.type" value="phot:PhotometryPoint"/>
    <FIELDref utype="phot:PhotometryPoint.value" ref="refField1"/>
    <GROUP utype="phot:PhotometryPoint.filter" ref="refGroup1"/>
</GROUP>
```

This utype points to the “filter” attribute of the “PhotometryPoint” type
Python: sdss_2mass.catalog.sdss_g.filter

This utype points to the “value” attribute of the “PhotometryPoint” type
Python: sdss_2mass.catalog.sdss_g.value

UTYPEs are mere pointers

This utype points to the “sdss_g” attribute of the “Catalog” type
Python: sdss_2mass.catalog.sdss_g

The sdss_g attribute **is a** PhotometryPoint type

```
<GROUP utype="sdss+2mass:er$3qer$^sa">
    <PARAM utype="Instance.type" value="phot:04cg^5$sdrt"/>
    <FIELDref utype="phot:vf5$6&88*dsa@" ref="refField1"/>
    <GROUP utype="phot:axdf54rt^!wdrg987" ref="refGroup1"/>
</GROUP>
```

This utype points to the “filter” attribute of the “PhotometryPoint” type
Python: sdss_2mass.catalog.sdss_g.filter

This utype points to the “value” attribute of the “PhotometryPoint” type
Python: sdss_2mass.catalog.sdss_g.value

Meanwhile, in a Spectral Energy Distribution…

Sanity Check

- **Meanwhile, in a Spectral Energy Distribution…**

Solution 1 (“path” utypes)

```
<FIELD utype=“sed:Sed.PhotometryPoint.Value”/>
<FIELD utype=“sed:Sed.PhotometryPoint.Filter”/>
```

Solution 2 (“type&role” utypes)

```
<GROUP utype=“sed:Sed.photometryPoint”>
  <PARAM utype=“Instance.type” value=“phot:PhotometryPoint”/>
  <FIELDref utype=“phot:PhotometryPoint.value” ref=“refField1”/>
  <GROUP utype=“phot:PhotometryPoint.filter” ref=“refGroup1”/>
</GROUP>
```

Sanity Check

- **Meanwhile, in a Spectral Energy Distribution…**

Solution 1 (“path” utypes)

<FIELD **utype=“sed:Sed.PhotometryPoint.Value”**/>
<FIELD **utype=“sed:Sed.PhotometryPoint.Filter”**/>

Solution 2 (“type&role” utypes)

<GROUP **utype=“sed:Sed.photometryPoint”**>
<PARAM **utype=“Instance.type”** value=“**phot:PhotometryPoint”/>**
<FIELDref **utype=“phot:PhotometryPoint.value” ref=“refField1”**/>
<GROUP **utype=“phot:PhotometryPoint.filter” ref=“refGroup1”**/>
</GROUP>

Remember the photometry point in a catalog? (i.e. in a different box)

Sanity Check

- **Meanwhile, in a Spectral Energy Distribution…**

Solution 1 (“path” utypes)

```
<FIELD utype=“sed:Sed.PhotometryPoint.Value”/>
<FIELD utype=“sed:Sed.PhotometryPoint.Filter”/>
```

Solution 2 (“type&role” utypes)

```
<GROUP utype=“sed:Sed.photometryPoint”>
  <PARAM utype=“Instance.type” value=“phot:PhotometryPoint”/>
  <FIELDref utype=“phot:PhotometryPoint.value” ref=“refField1”/>
  <GROUP utype=“phot:PhotometryPoint.filter” ref=“refGroup1”/>
</GROUP>
```

Remember the photometry point in a catalog? (i.e. in a different box)

```
<GROUP utype=“sdss+2mass:Catalog.sdssG”>
  <PARAM utype=“Instance.type” value=“phot:PhotometryPoint”/>
  <FIELDref utype=“phot:PhotometryPoint.value” ref=“refField1”/>
  <GROUP utype=“phot:PhotometryPoint.filter” ref=“refGroup1”/>
</GROUP>
```

Sanity Check

- **Meanwhile, in a Spectral Energy Distribution…**

Solution 1 (“path” utypes)

```
<FIELD utype=“sed:Sed.PhotometryPoint.Value”/>
<FIELD utype=“sed:Sed.PhotometryPoint.Filter”/>
```

Solution 2 (“type&role” utypes)

```
<GROUP utype=“sed:Sed.photometryPoint”>
  <PARAM utype=“Instance.type” value=“phot:PhotometryPoint”/>
  <FIELDref utype=“phot:PhotometryPoint.value” ref=“refField1”/>
  <GROUP utype=“phot:PhotometryPoint.filter” ref=“refGroup1”/>
</GROUP>
```

Extract all photometry points with the python generic, not ad hoc, library:

“path utypes”

You can’t unless you parse the utypes

“type+role” utypes

```
>> points = votable.get_objects(“phot:PhotometryPoint”)
```

“I am concerned about backward compatibility”

Plenty of Room

Mappings not specified allow custom/legacy usage

- FIELDs.
- Standalone PARAMs.
- GROUPs with @utype of undeclared prefix.
- TABLE, RESOURCE

Transition

- No changes required in current stds, protocols.
- If Data Providers want to upgrade, they can **add** metadata, not change the current metadata.
- Services prototyped against WDs can simply **add** metadata.

New, complex Data Models can benefit

- Data nCubes and their projections/combinations: TimeSeries, SED, Spectral, Photometry

“Can we remove the boundaries between image, spectrum, SED, cube?”

We Must! The new UTYPEs proposal can help

- The needed bulding blocks are few

- Don’t reinvent the wheel:
  - spec:Segment.Char.SpectralAxis.Accuracy.StatErr
  - spec:Segment.Data.SpectralAxis.Accuracy.StatErr
  - spec:Segment.Char.FluxAxis.Accuracy.StatErr
  - char:SpectralAxis.Accuracy.StatErr
  - …on and on and on and on…
  - Why don’t we just put a char:Accuracy.StatErr in every box it is needed?

- If a Data Cube is described by the ImageDM, what is the Data Model describing a projection of the cube? What about Planetary science?

We Must! The new UTYPEs proposal can help

So, we **must** reuse Models consistently!

“Tools cannot do everything!!!!”

That's why we suggest a strict specification

- Right now clients do not know what to expect
- Thousands of UTYPEs and counting (often pointing to same concept)
- Where to look for a unit (ucd, value) string?
  - UTYPE.unit
  - FIELDref unit
  - FIELD unit
- How to refer to other objects?
- The UTYPEs proposal defines the mapping patterns

“I cannot represent STC using the new UTYPEs”

STC serialization

```
<GROUP utype=”vo-dml:Instance.root”>
   <PARAM utype="vo-dml:Instance.type" value="stc:AstroCoords"/>
   <PARAM utype="stc:AstroCoords.coord_system_id" value=”UTC-ICRS-TOPO” />
   <GROUP utype=”stc:AstroCoords.position2D”>
    <PARAM utype=”vo-dml:Instance.type” value=”stc:AstroCoords.Position2D”>
    <GROUP utype=”stc:Position2D.value2”>
      <PARAM utype="vo-dml:Instance.type" value="stc:Value2D"/>
      <FIELDref utype="stc:Value2.C1" ref="col1"/>
      <FIELDref utype="stc:Value2.C2" ref="col2"/>
    </GROUP>
   </GROUP>
</GROUP>

<AstroCoords coord_system_id=”UTC-ICRS-TOPO">
<Position2D unit="deg">
      <Value2 id="Center">
       <C1>148.88821</C1>
       <C2>69.06529</C2>
      </Value2>
</Position2D>
</AstroCoords>
```

“Back in 2009 I asked some questions about UTYPEs: I am glad to finally see a consistent answer to most of them, but do you spell it Utype, UTYPE, Utype, or utype?!”

Thank you!

"Let's not reinvent the wheel, please"