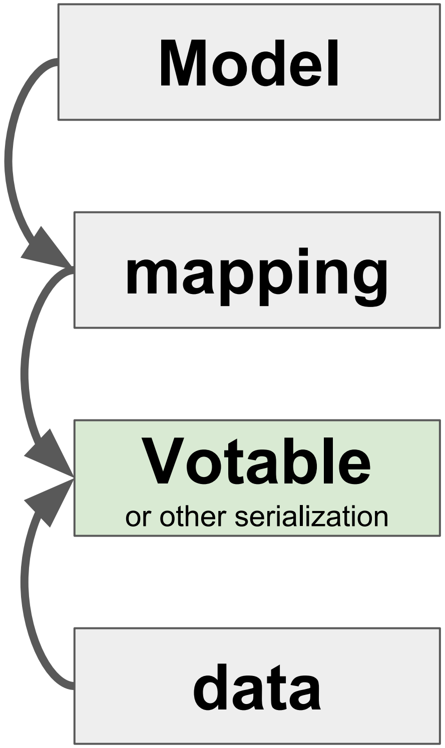
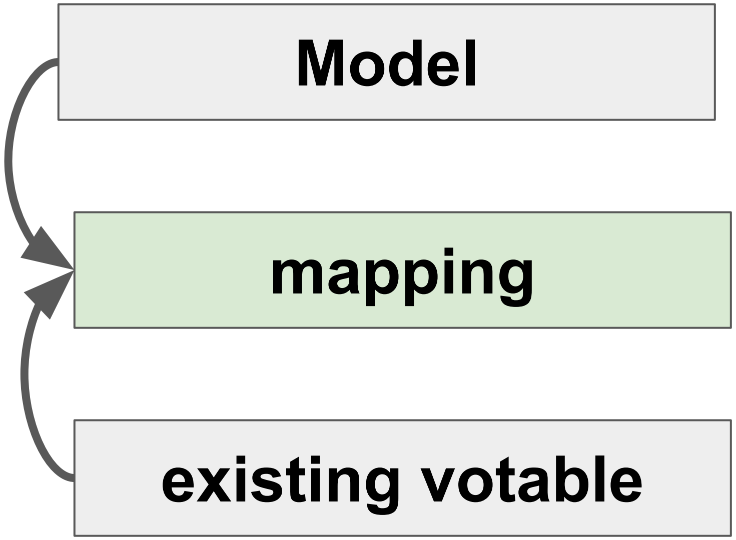
Annotating GAIA Time Series with VO-DML

https://github.com/lmichel/vodml-lite-mapping

2 Ways of Seeing Things



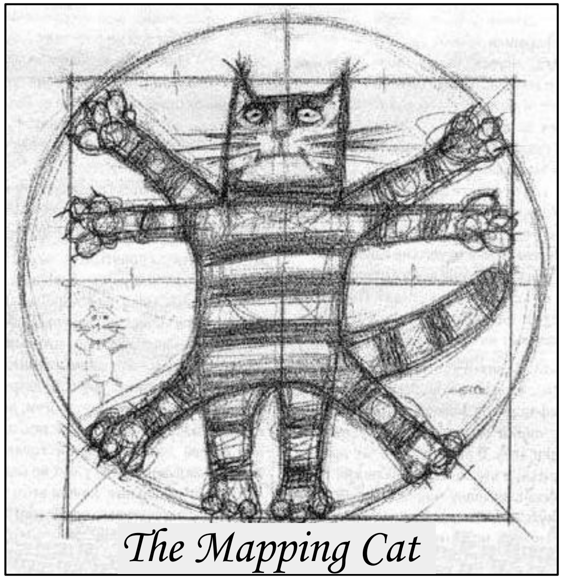


- Data can be put in a VOtable in a way they can be mapped onto the model.
- Might put limitations on the VOTable structure

- The mapping must be applicable to any existing dataset.
- This impacts the mapping syntax
- The mapping has also to drive the parser

Mapping *Any* Existing VOTable

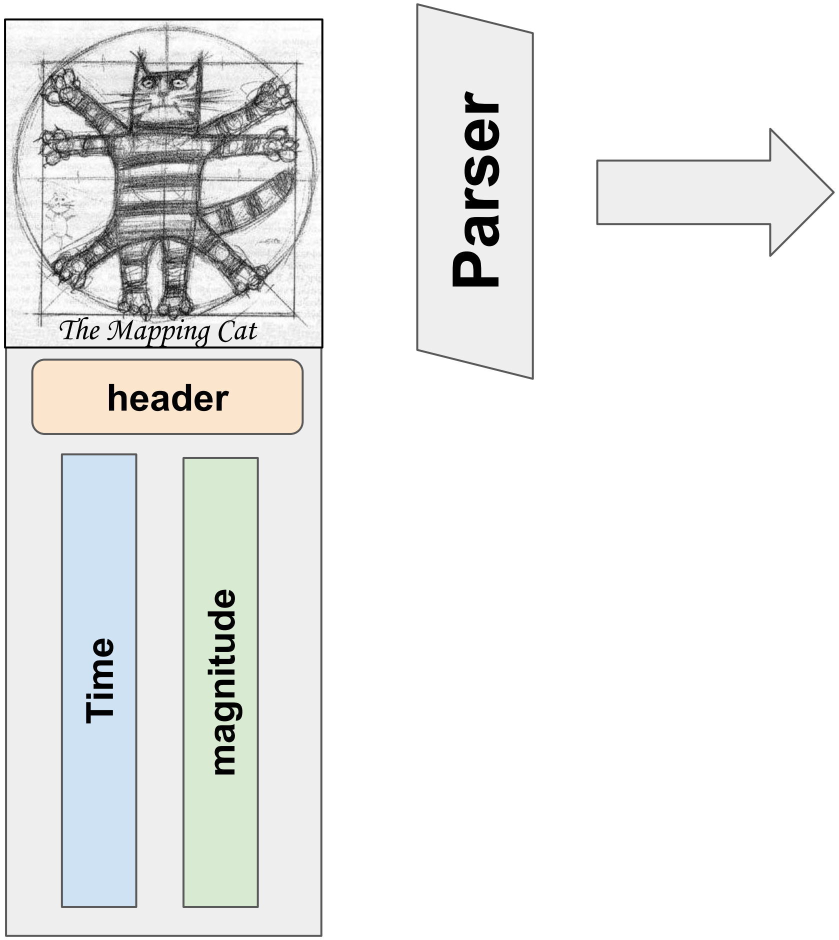
Model


The Mapping Cat

existing votable

- Mapping any model on any VOTable is like squaring the circle.
- Should mix model elements with directives for the parser
- **But time domain gives us some reasonable examples yet**

The Basic Case



header

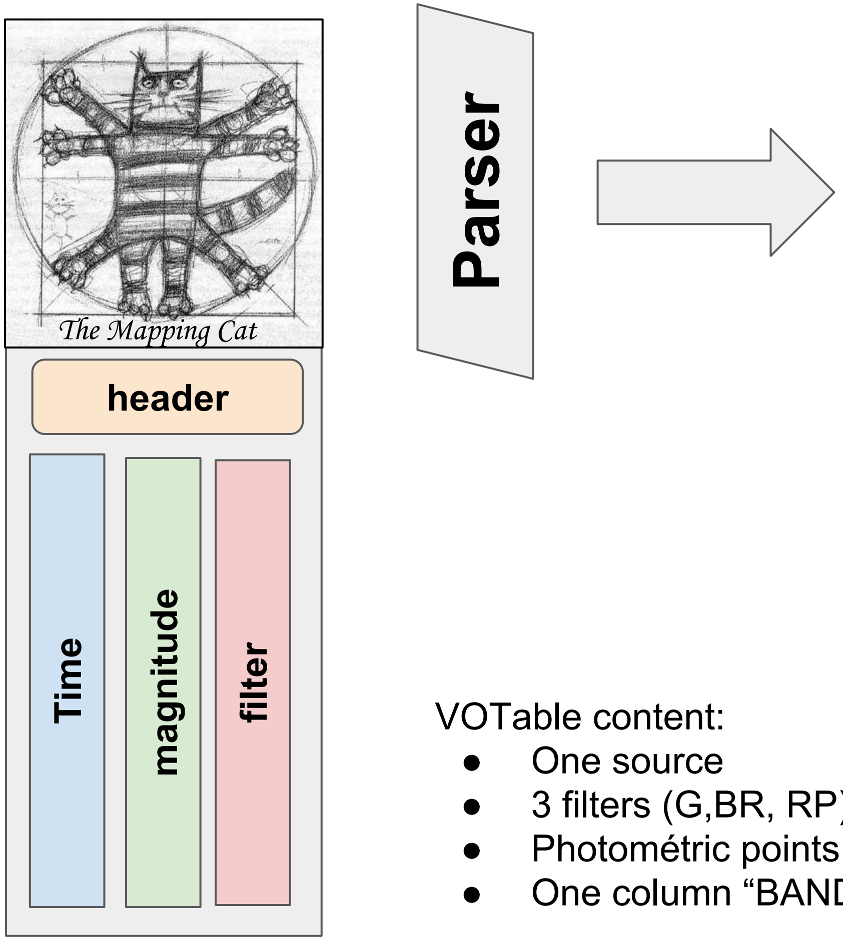
Time

magnitude

Parser

`SparseCube`
One instance
One Light Curve

The Case of the Day: GAIA



`TimeSeries`

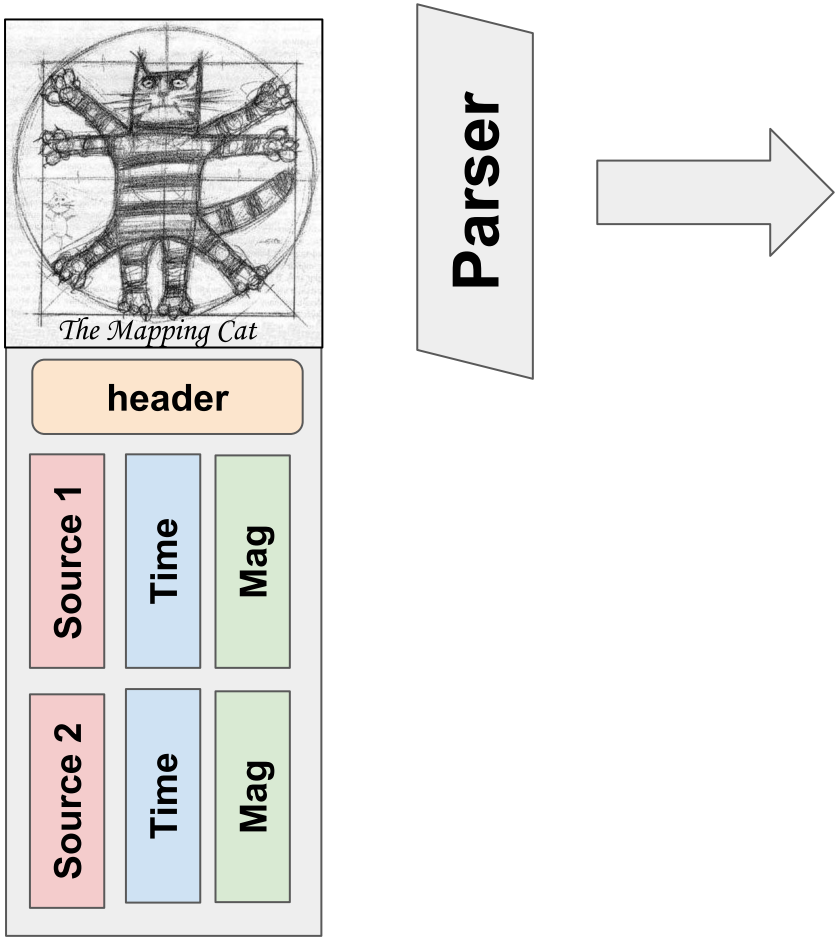
One instance

Several Light Curves

VOTable content:

- One source
- 3 filters (G,BR, RP)
- Photométric points mixed in one <DATATABLE>
- One column “BAND” identifying the filter for each measurement

Another Gaia Case ?



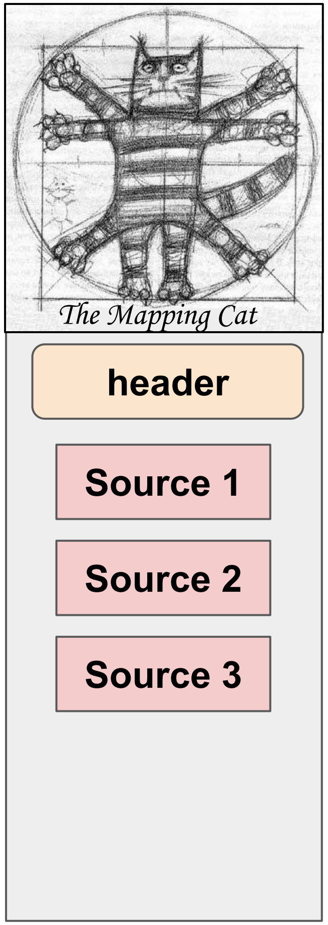
`[TimeSeries]`

. List of instances

. The number of instances results from the data grouping

. Each instance owns a subset of the dada rows

And So Forth ...

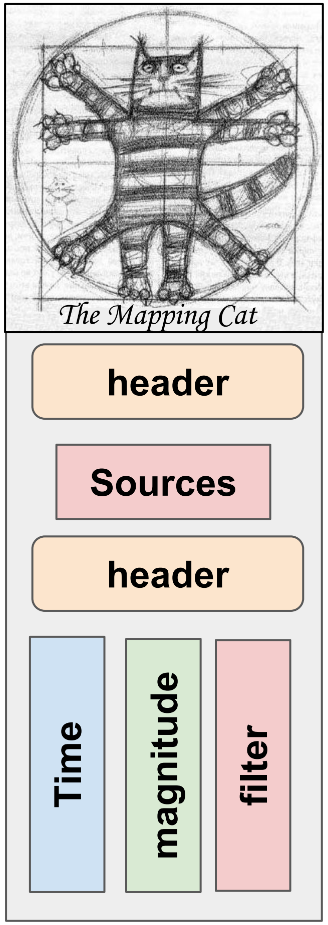


header

Source 1

Source 2

Source 3



header

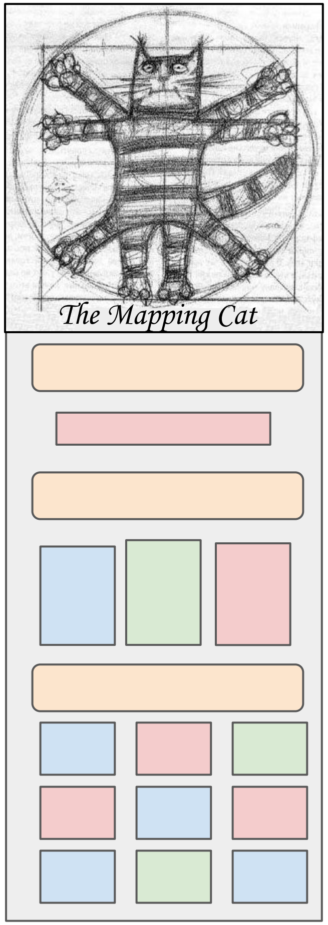
Sources

header

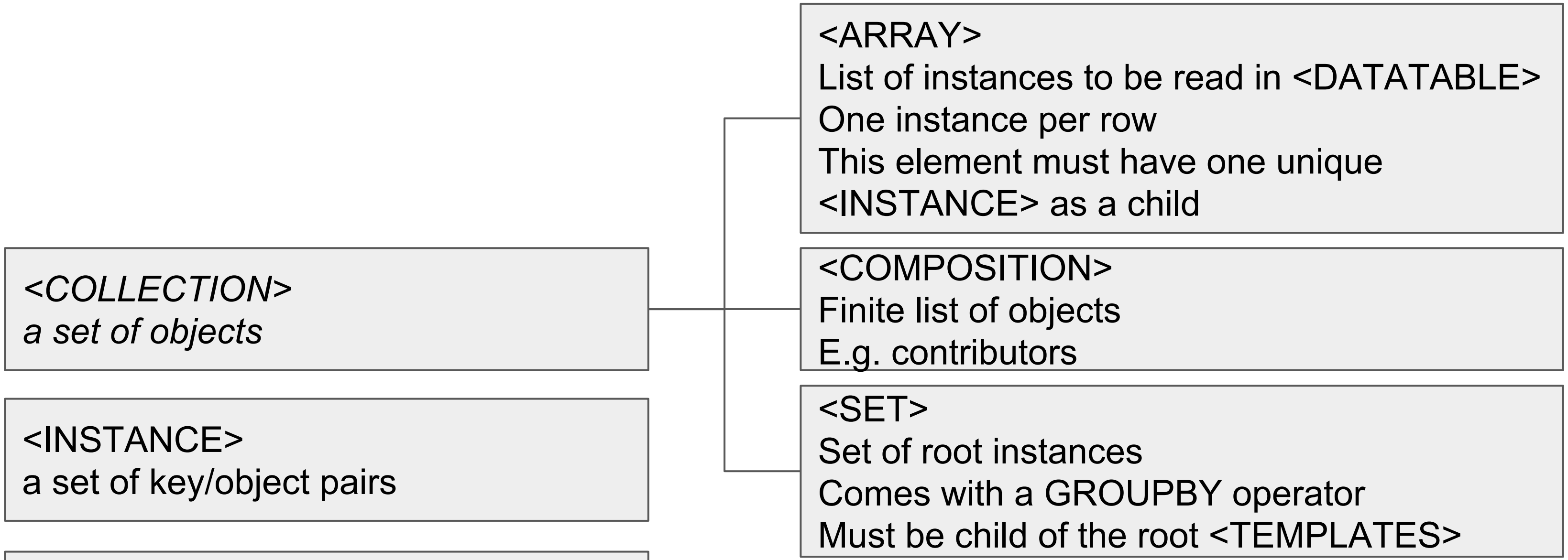
Time

magnitude

filter



Lite Syntax at a Glance



<COLLECTION>
a set of objects

<ARRAY>
List of instances to be read in <DATATABLE>
One instance per row
This element must have one unique
<INSTANCE> as a child

<COMPOSITION>
Finite list of objects
E.g. contributors

<SET>
Set of root instances
Comes with a GROUPBY operator
Must be child of the root <TEMPLATES>

<INSTANCE>
a set of key/object pairs

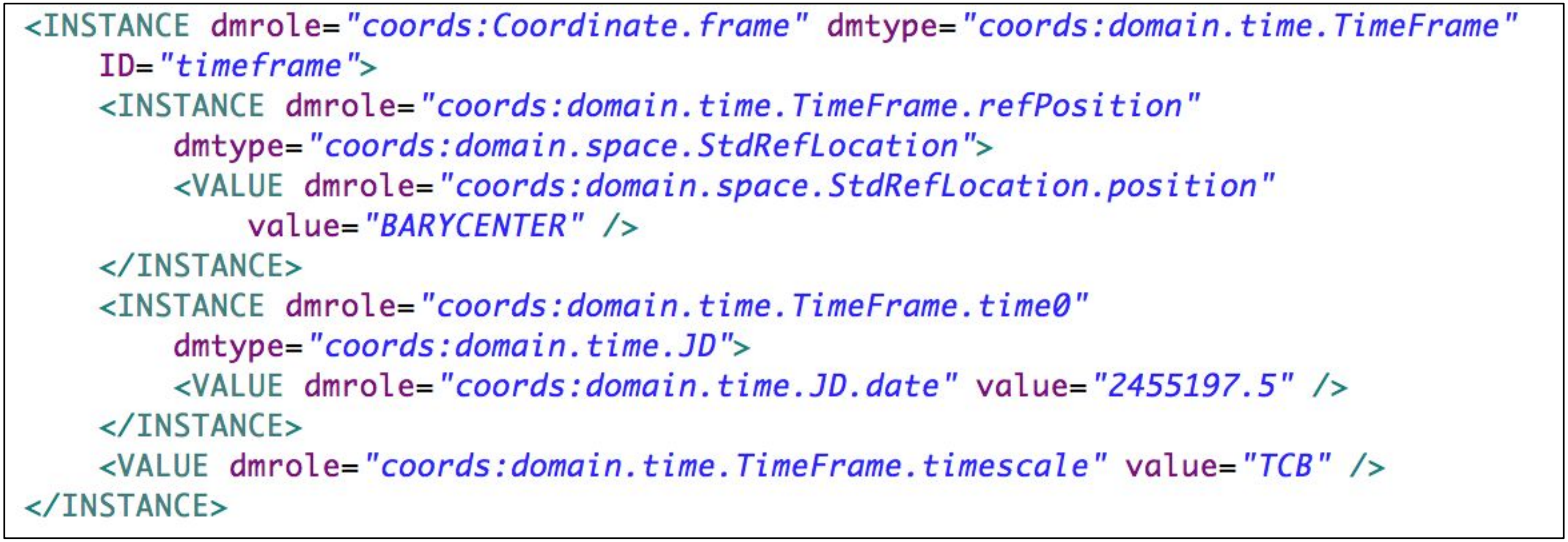
<VALUE>
An atomic value (string or numerical)

- *Each one of these elements has a* **`dmrole`**
- **`dmtypes`** *are supported by not used yet*

<FILTER>
Filter the values read in <DATATABLE>
Must be after the <INSTANCE> contained in a <ARRAY>

<FOREIGNKEY>
Not implemented yet

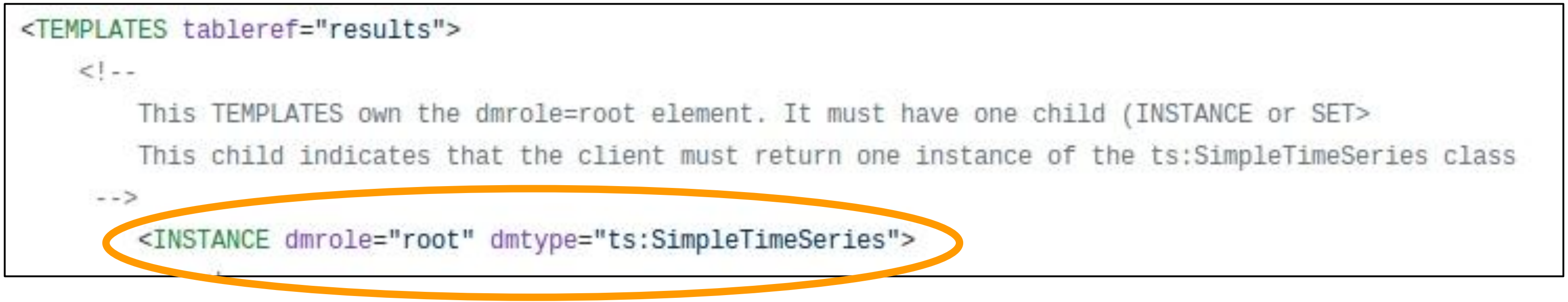
Compact Syntax



```
<INSTANCE dmrole="coords:Coordinate.frame" dmtype="coords:domain.time.TimeFrame"
    ID="timeframe">
    <INSTANCE dmrole="coords:domain.time.TimeFrame.refPosition"
        dmtype="coords:domain.space.StdRefLocation">
        <VALUE dmrole="coords:domain.space.StdRefLocation.position"
            value="BARYCENTER" />
    </INSTANCE>
    <INSTANCE dmrole="coords:domain.time.TimeFrame.time0"
        dmtype="coords:domain.time.JD">
        <VALUE dmrole="coords:domain.time.JD.date" value="2455197.5" />
    </INSTANCE>
    <VALUE dmrole="coords:domain.time.TimeFrame.timescale" value="TCB" />
</INSTANCE>
```

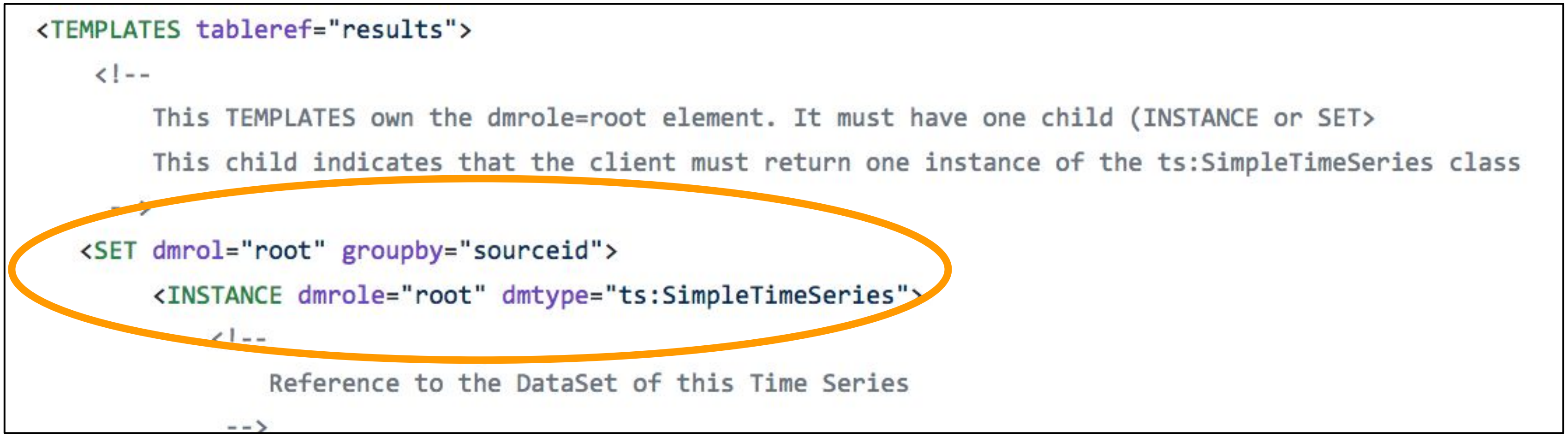
Example: STC time frame

`dmrole=root` indicates the VOTable Content



```
<TEMPLATES tableref="results">
    <!--
        This TEMPLATES own the dmrole=root element. It must have one child (INSTANCE or SET>
        This child indicates that the client must return one instance of the ts:SimpleTimeSeries class
    -->
        <INSTANCE dmrole="root" dmtype="ts:SimpleTimeSeries">
```

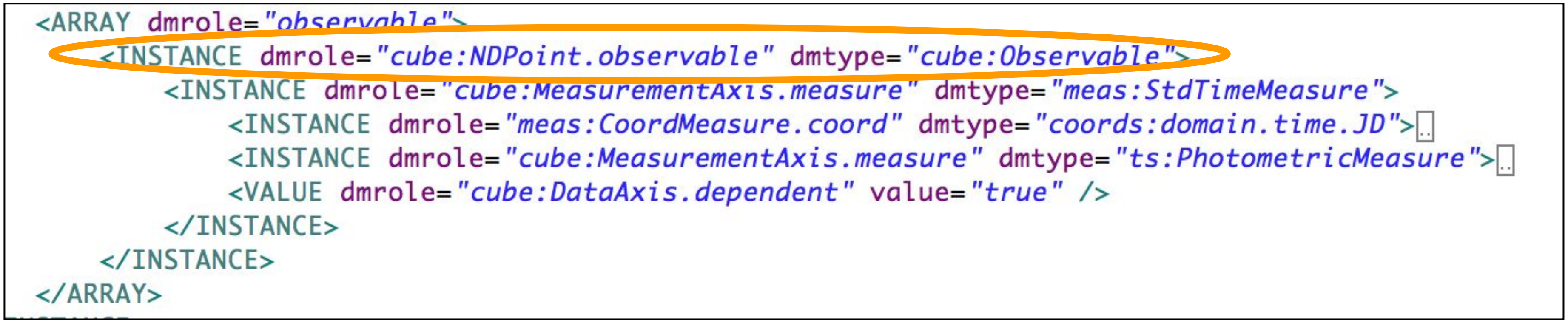
*This VOTable contains **one instance** of class* `ts:SimpleTimeSeries`



```
<TEMPLATES tableref="results">
    <!--
        This TEMPLATES own the dmrole=root element. It must have one child (INSTANCE or SET>
        This child indicates that the client must return one instance of the ts:SimpleTimeSeries class
    -->
   <SET dmrol="root" groupby="sourceid">
        <INSTANCE dmrole="root" dmtype="ts:SimpleTimeSeries">
            <!--
                Reference to the DataSet of this Time Series
            -->
```

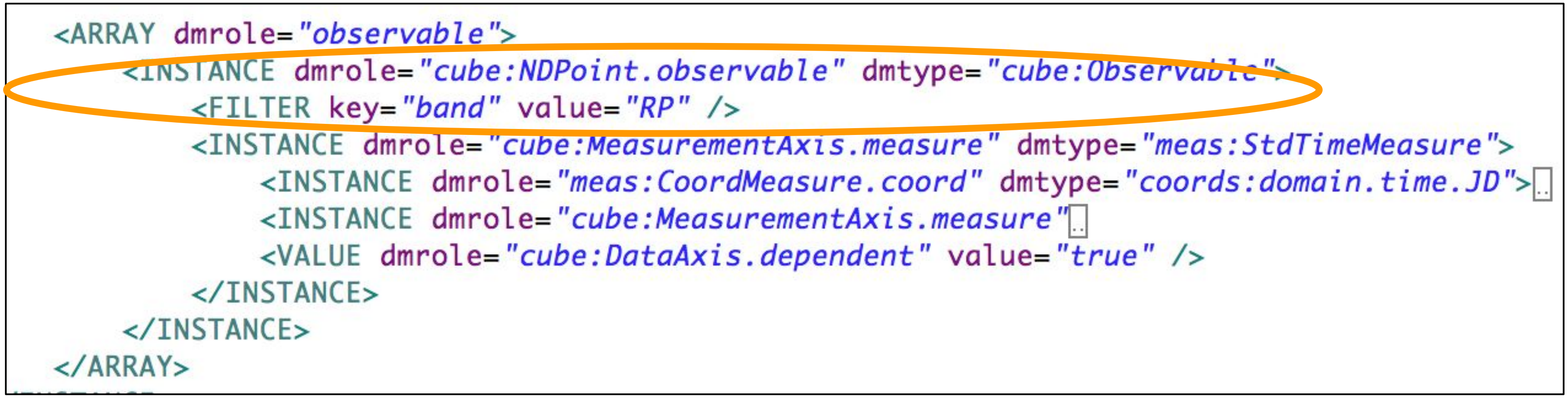
*This VOTable contains a **set of instances** of class* `ts:SimpleTimeSeries`
(work in progress)

<DATATABLE> Mapping



```
<ARRAY dmrole="observable">
    <INSTANCE dmrole="cube:NDPoint.observable" dmtype="cube:Observable">
        <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:StdTimeMeasure">
            <INSTANCE dmrole="meas:CoordMeasure.coord" dmtype="coords:domain.time.JD">□
            <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="ts:PhotometricMeasure">□
            <VALUE dmrole="cube:DataAxis.dependent" value="true" />
        </INSTANCE>
    </INSTANCE>
</ARRAY>
```

Each <DATATABLE> row is mapped as an instance of the class `cube:Observable`

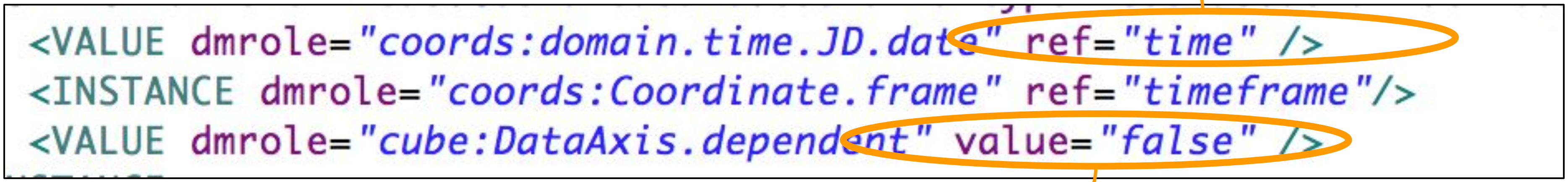


```
<ARRAY dmrole="observable">
    <INSTANCE dmrole="cube:NDPoint.observable" dmtype="cube:Observable">
        <FILTER key="band" value="RP" />
        <INSTANCE dmrole="cube:MeasurementAxis.measure" dmtype="meas:StdTimeMeasure">
            <INSTANCE dmrole="meas:CoordMeasure.coord" dmtype="coords:domain.time.JD">□
            <INSTANCE dmrole="cube:MeasurementAxis.measure"□
            <VALUE dmrole="cube:DataAxis.dependent" value="true" />
        </INSTANCE>
    </INSTANCE>
</ARRAY>
```

Each <DATATABLE> row with `band=RP` *is mapped as an instance of the class* `cube:Observable`

One Tag for Both Values and Literals

Value resolved by reference



```
<VALUE dmrole="coords:domain.time.JD.date" ref="time" />
<INSTANCE dmrole="coords:Coordinate.frame" ref="timeframe"/>
<VALUE dmrole="cube:DataAxis.dependent" value="false" />
```

Value resolved as a literal

If both `ref` and `value` attributes are present, `ref` is first resolved and then `value` is taken in case of failure

Validation

- **Mapping Validation**
  - `SimpleTimeSeries` model
  - Gaia 3 bands time series
  - Ongoing tests on multi-source datasets

- **Client Validation**
  - See *app1* talk
  - Everything is available on GitHub

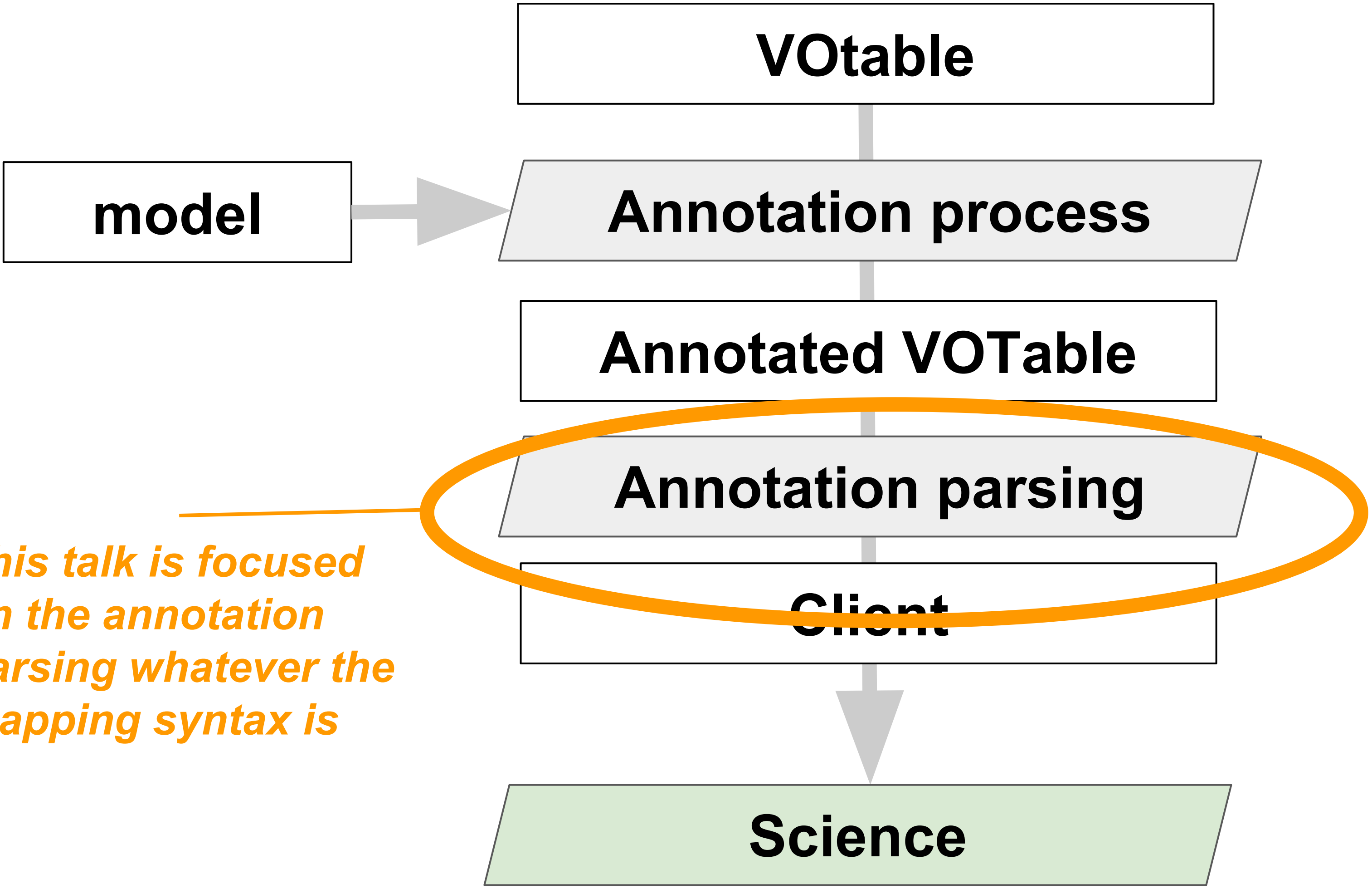
https://github.com/lmichel/vodml-lite-mapping

Contributor are Welcome

Reading VO-DML Annotations With Java

https://github.com/lmichel/vodml-lite-mapping

The VO-DML Stack



This talk is focused on the annotation parsing whatever the mapping syntax is

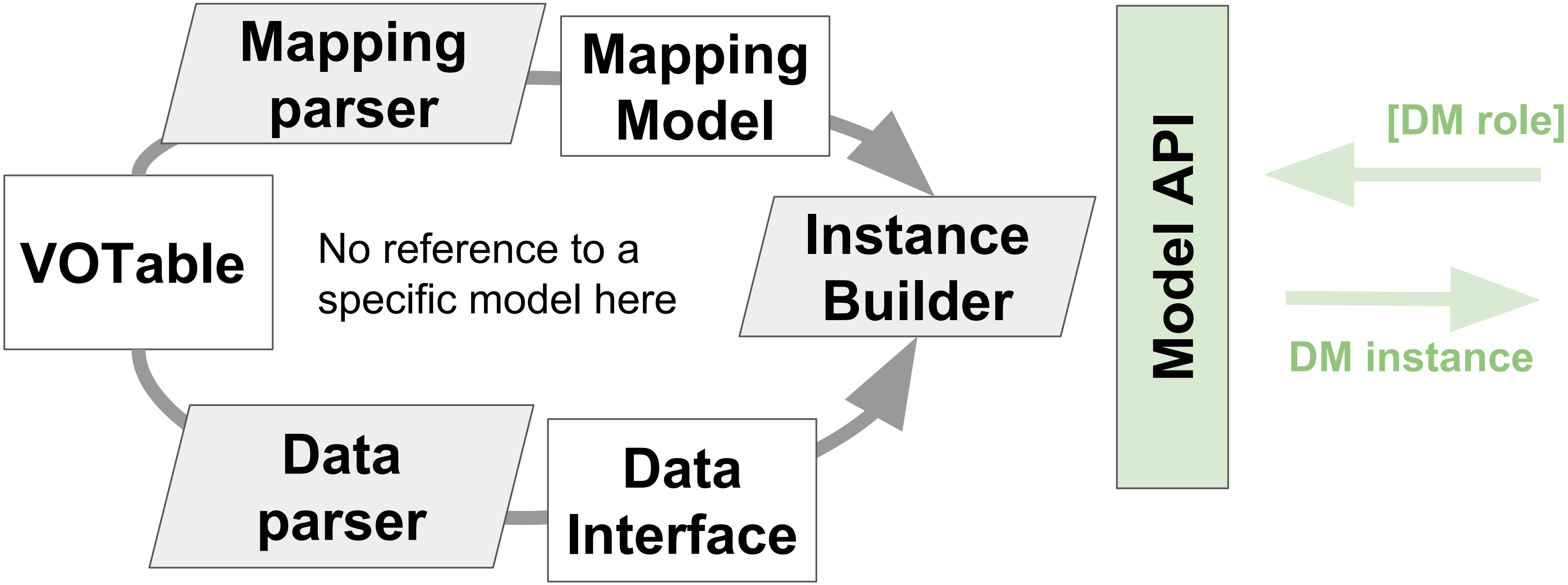
Client Expectations for Using Models

- **Hiding the data complexity**
  - Only see the model structure whatever the data are
  - Avoiding Inferences for Retrieving Data
  - No specific code for specific data sets
- **A clear way to finally get the VOTable content**
  - This feature is still a lack for the VOTable schema
- **Python API (OL)**
  - Victoria 2018 https://olaurino.gitlab.io/ivoa-dm-examples/

Java Client Expectation

- **Avoiding Application Update**
  - Adding new modules in Java implies software upgrades
    - Developers have to validate the upgrade
    - Users have to download it
- **Parser Code Independent from any Particular Model**
  - A unique parser for the VODML block
  - Paths leading to model nodes set by the caller
    - Something expressed with strings
    - Can be stored as external resources

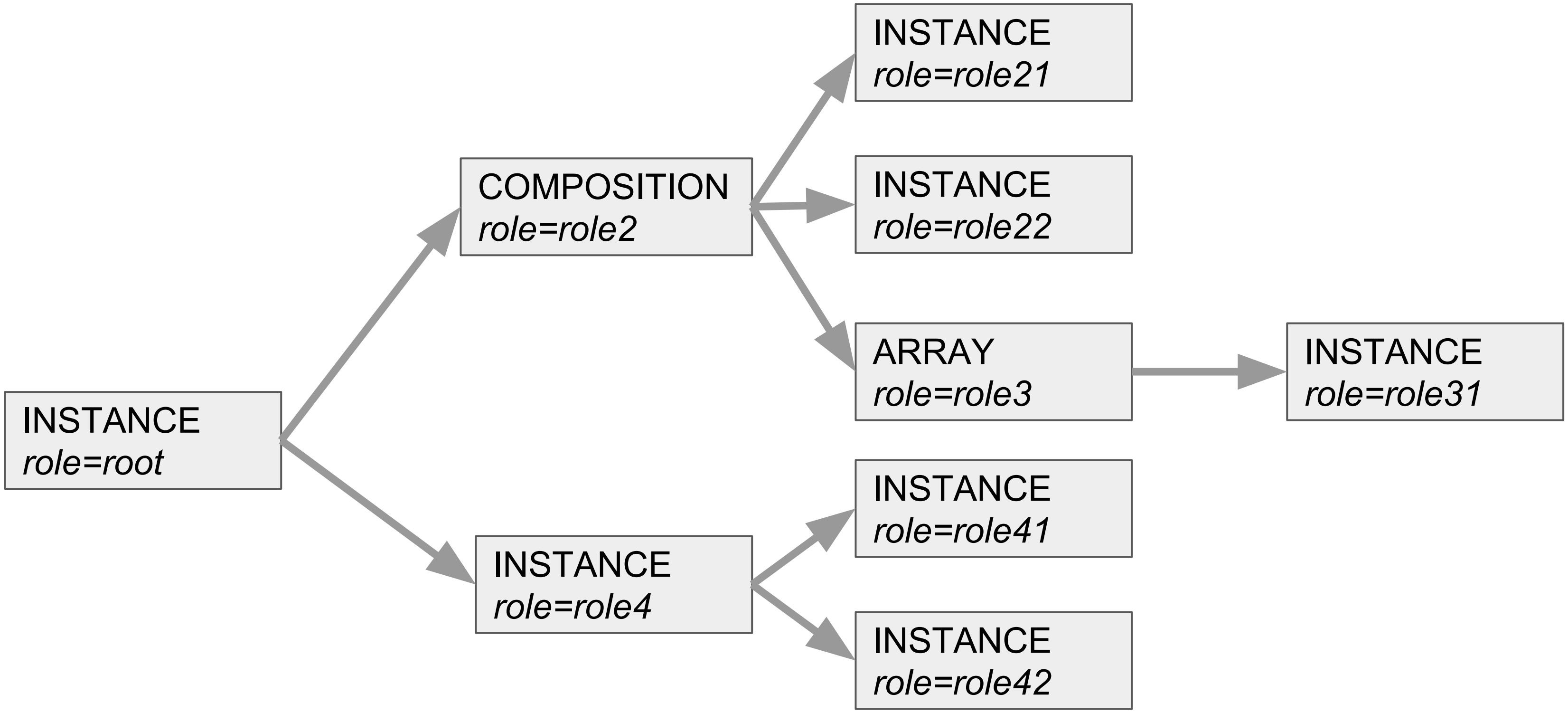
Architecture



Model API:

- Nothing specific to a model
- A reference to the root object
- A set of selectors to browse it

Internal Model



INSTANCE
role=root

COMPOSITION
role=role2

INSTANCE
role=role21

INSTANCE
role=role22

ARRAY
role=role3

INSTANCE
role=role31

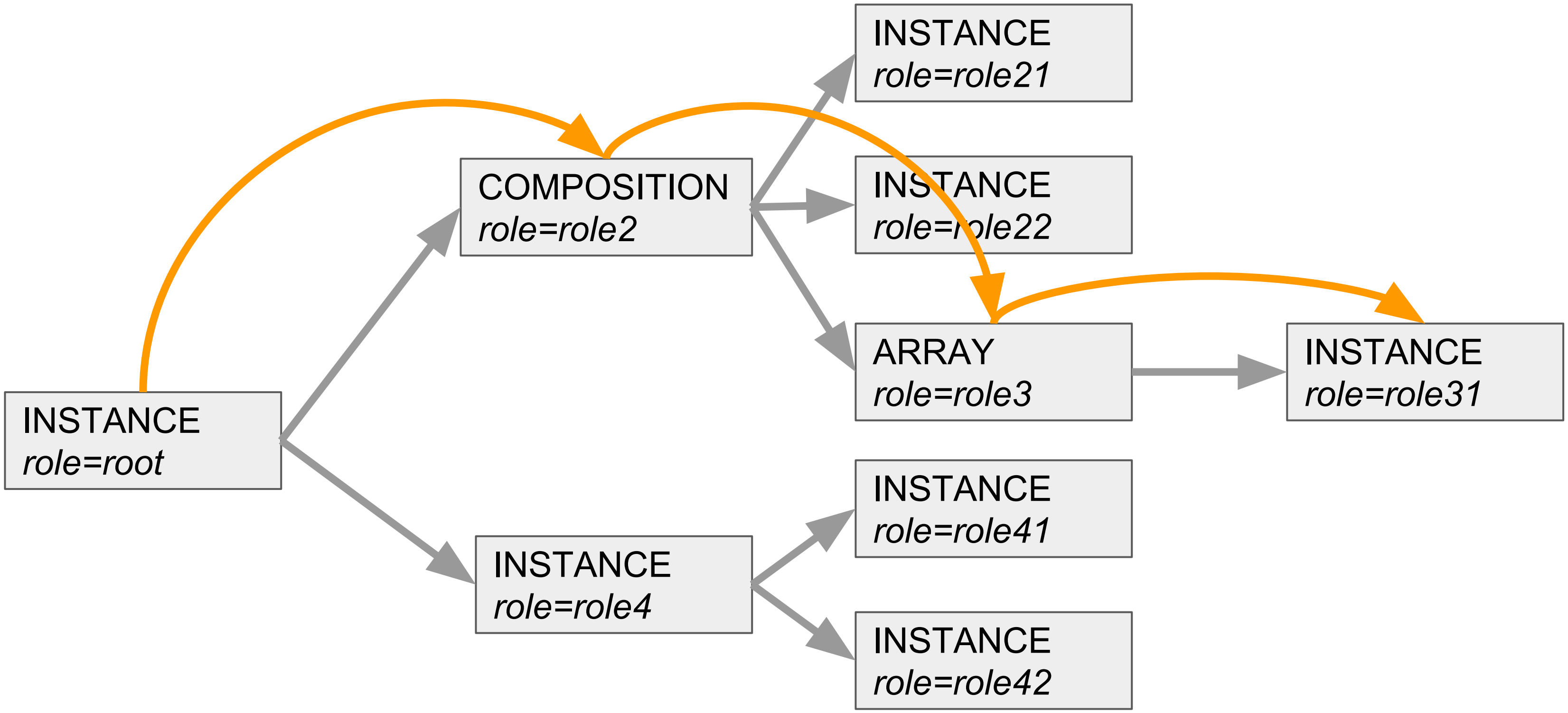
INSTANCE
role=role4

INSTANCE
role=role41

INSTANCE
role=role42

What the parser did

Internal Model



What the parser did ▭ →

What the client does →

```
Node(role=role31) = Node(role=root)
                       ->Node(role=role2)
                           ->Node(role=role3)
                               ->Node(role=role31)
```

Something Like This

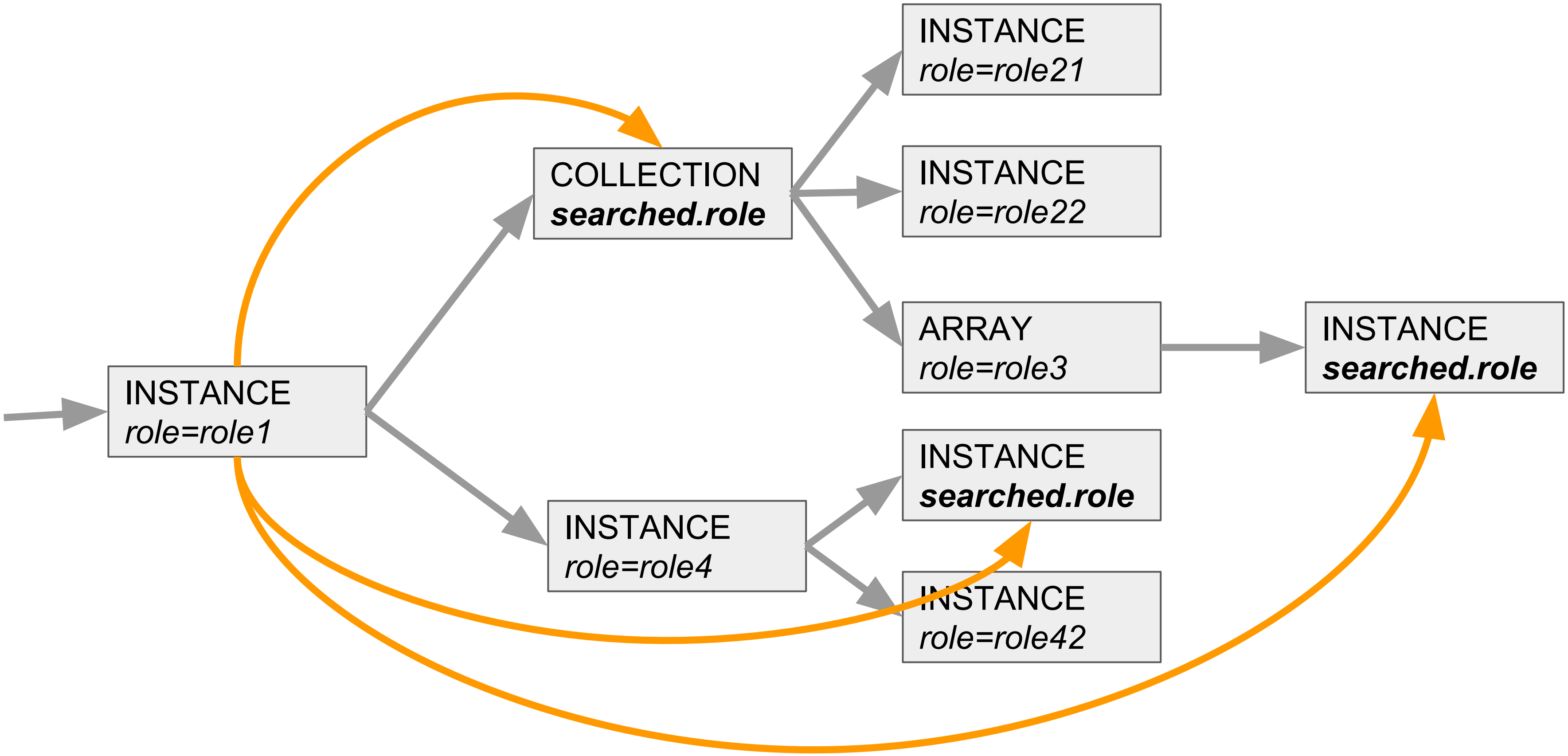
```
vodmlParser = new VodmlParser(“Myvotable”);

if( vodmlParser.implements(“TSmodel”) {
    /* getting the position object */
    Element position = vodmlParser.element(“model:Source.Position”)
    ra   = position.element(“Astro:position.lat”);
    dec  = position.element(“Astro:position.long”);
    /* browsing the photometric points */
    points = vodmlParser.element(“model:photometric.points”);
    for( int i=0 ; i<points.getLength() ; i++ ) {
        Element point = data.getValue(i);
        time  = point.element(“Astro:mes.time”);
        mag   = point.element(“Astro:mes.mag”);
    }
}
```

Resemblance to existing model roles is purely coincidental.

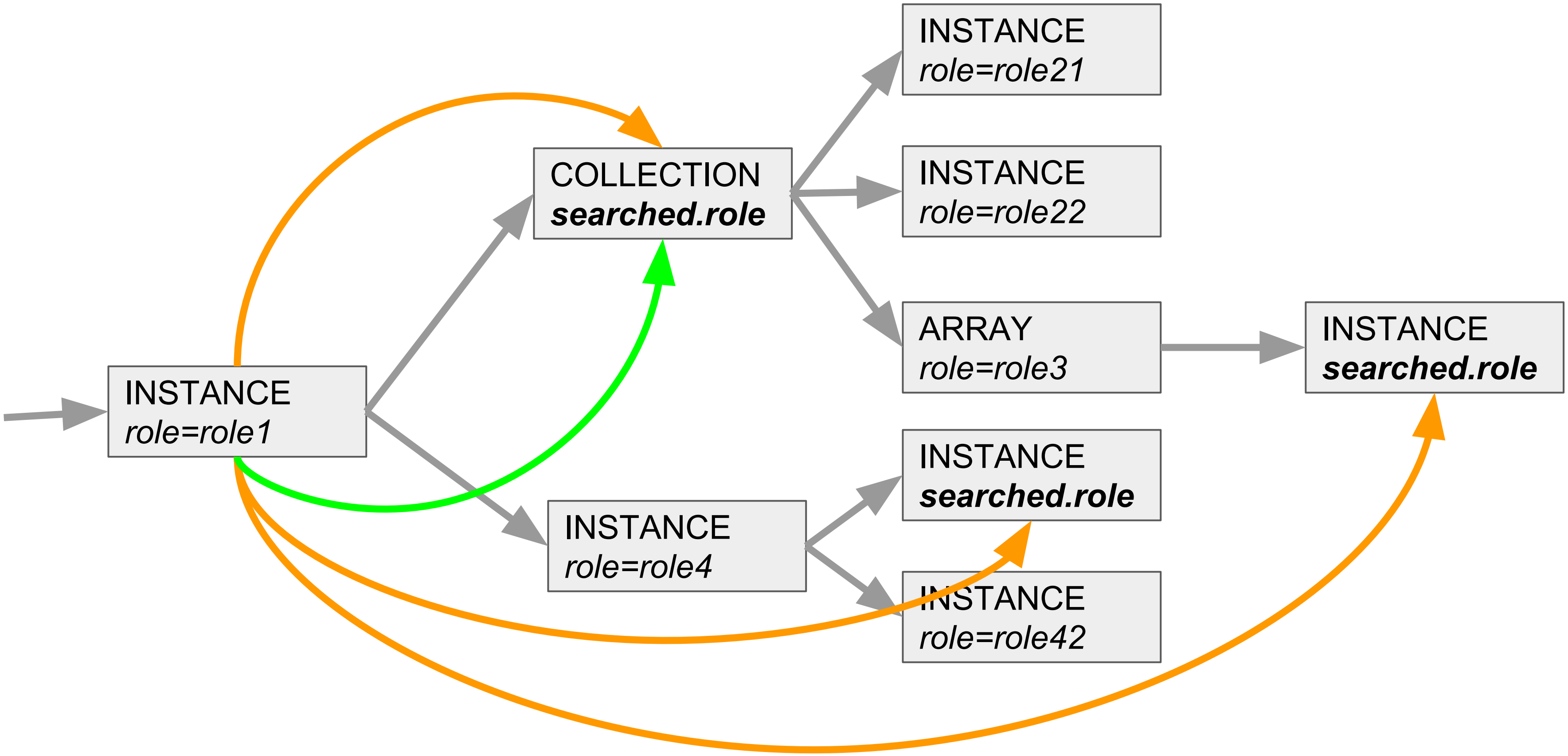
- In blue: Java words
- **In black:** VODML API code
- **In “green” :** Model related quantities, strings only

Mapping Element Selectors



`getSubElement...`Return one or all sub-element (s)matching the role

Mapping Element Selectors



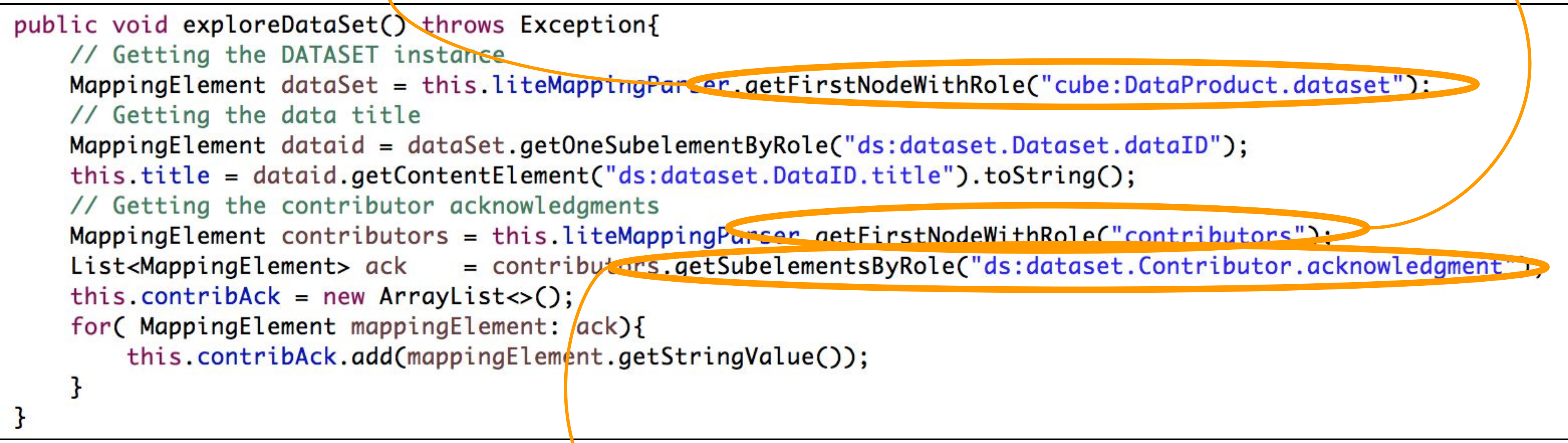
getSubElement... Return one or all sub-element (s)matching the role

getChild... Return one or all child(ern) matching the role

My API as it Is Now

The dataset object is supposed to be unique

Points onto the collection of contributors

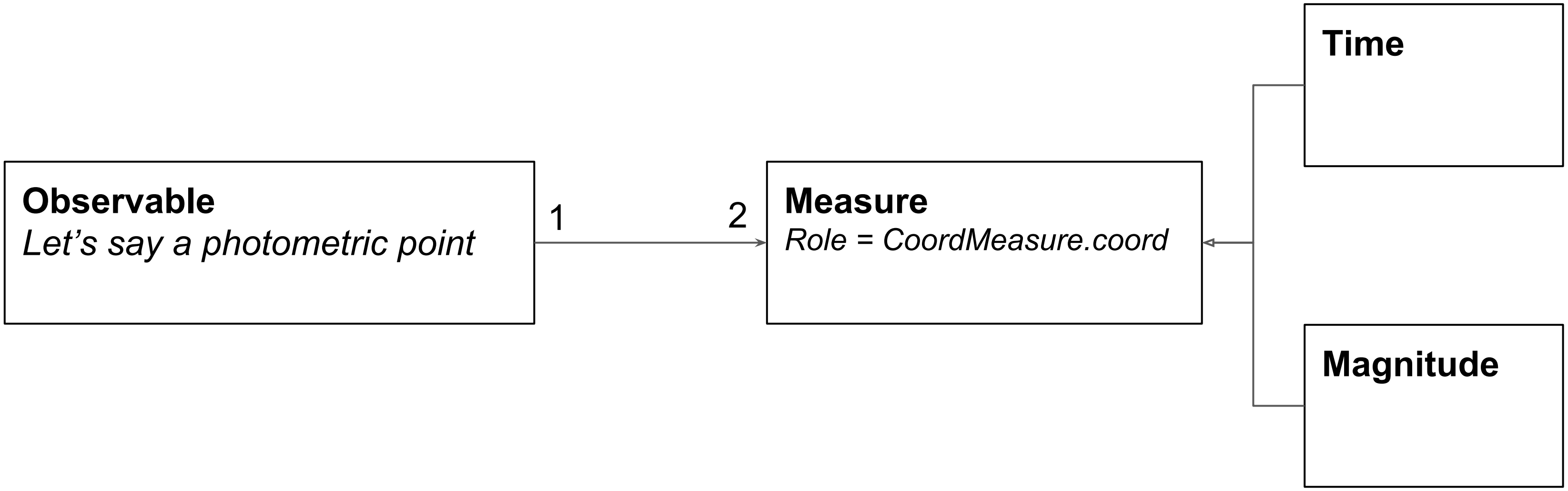


```
public void exploreDataSet() throws Exception{
    // Getting the DATASET instance
    MappingElement dataSet = this.liteMappingParser.getFirstNodeWithRole("cube:DataProduct.dataset");
    // Getting the data title
    MappingElement dataid = dataSet.getOneSubelementByRole("ds:dataset.Dataset.dataID");
    this.title = dataid.getContentElement("ds:dataset.DataID.title").toString();
    // Getting the contributor acknowledgments
    MappingElement contributors = this.liteMappingParser.getFirstNodeWithRole("contributors");
    List<MappingElement> ack     = contributors.getSubelementsByRole("ds:dataset.Contributor.acknowledgment");
    this.contribAck = new ArrayList<>();
    for( MappingElement mappingElement: ack){
        this.contribAck.add(mappingElement.getStringValue());
    }
}
```

Retrieving the list of contributors

Take all acknowledgements of all contributors

When Things Become Tricky

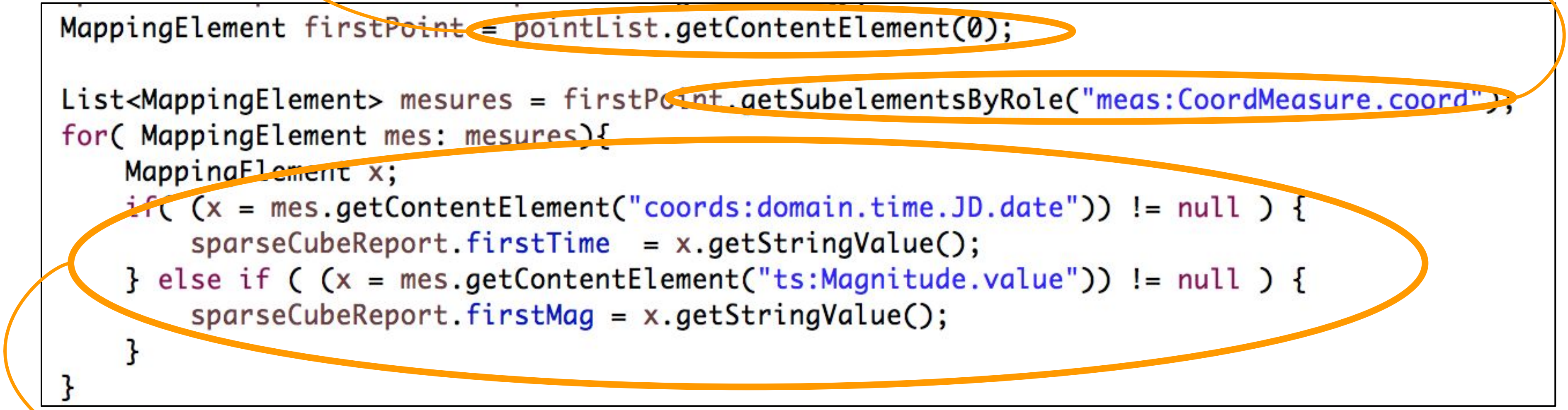


- The 2 Measures have the same role.
- To know what is what, we have to check the `dmtype` (class name) or to explore the inside of each instance

When Things Become Tricky

Take the first photometric point

Take all measures of that point

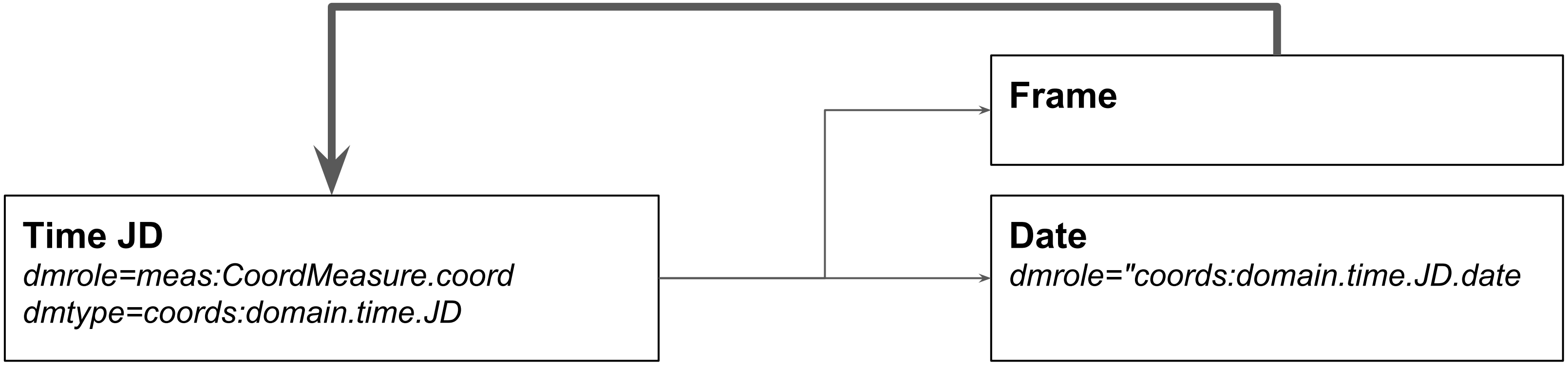


```
MappingElement firstPoint = pointList.getContentElement(0);

List<MappingElement> mesures = firstPoint.getSubelementsByRole("meas:CoordMeasure.coord");
for( MappingElement mes: mesures){
    MappingElement x;
    if( (x = mes.getContentElement("coords:domain.time.JD.date")) != null ) {
        sparseCubeReport.firstTime  = x.getStringValue();
    } else if ( (x = mes.getContentElement("ts:Magnitude.value")) != null ) {
        sparseCubeReport.firstMag = x.getStringValue();
    }
}
```

Explore the measure objects to see what they are

A Bit More Tricky

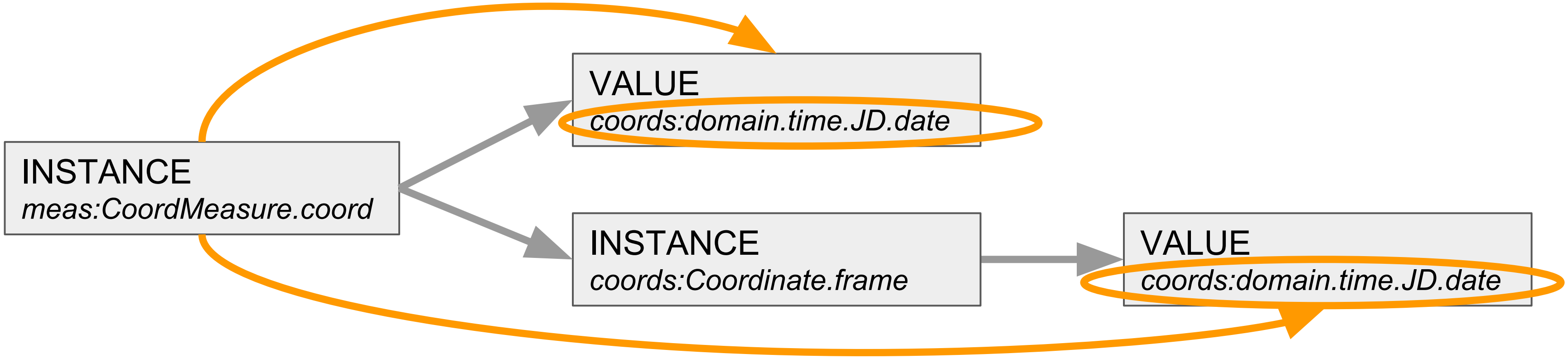


Time JD
dmrole=meas:CoordMeasure.coord
dmtype=coords:domain.time.JD

Frame

Date
dmrole="coords:domain.time.JD.date

- Isolating the timestamp *date* with selectors based on *dmroles* may be confusing



INSTANCE
meas:CoordMeasure.coord

VALUE
coords:domain.time.JD.date

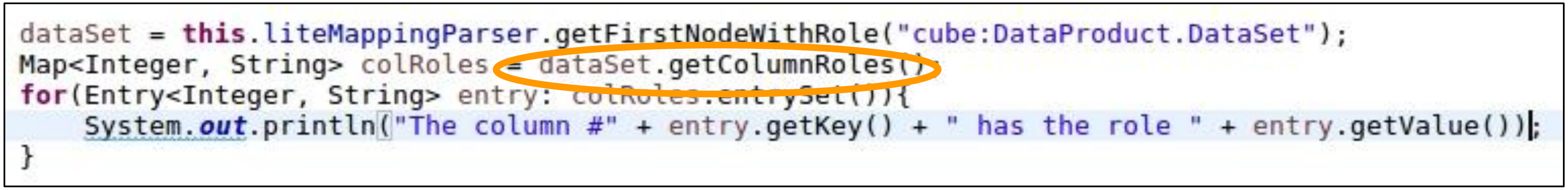
INSTANCE
coords:Coordinate.frame

VALUE
coords:domain.time.JD.date

Very simplified model view

A Shortcut

- **Bypassing Object Instantiation**
  - No need to systematically build an instance for each row
    - E.g. for plotting data
  - Knowing the `dmrole` of each column must be enough
    - Simple time series example:
      Column `#1` has the role “`coords:domain.time.JD.date`”
      Column `#3` has the role “`ts:Magnitude.value`”
  - This allow the client to use its own readout engine
    - Mapping used to extract meta-data
    - Standard way to read data tables with roles set for some columns



```
dataSet = this.liteMappingParser.getFirstNodeWithRole("cube:DataProduct.DataSet");
Map<Integer, String> colRoles = dataSet.getColumnRoles();
for(Entry<Integer, String> entry: colRoles.entrySet()){
    System.out.println("The column #" + entry.getKey() + " has the role " + entry.getValue());
}
```

Done/BeingDone/2Do

- **Done**
  - Works with *SimpleTimeSeries* model
  - Data filtering
- **Being Done**
  - Group by facility *<SET groupby=”..”>*
- **Todo**
  - Simplify the API
  - Implementing *DMTypes*
  - Foreign keys implementation

https://github.com/lmichel/vodml-lite-mapping

Contributors are Welcome

Mapping Nodos vs Java Classes

Mapping Node	Java Class	
`<INSTANCE>`	`Instance`	Set of key/value pairs Key are the **`dmrole`** of the values
`<VALUE>`	`Textual or Numerical`	Atomic value
`<COMPOSITION>`	`MultiInstanceCollection`	A collection of instances
`<SET>`	`GroupByCollection`	Set of “grouped by” instances
`<ARRAY>`	`DataTableCollection`	Iterator on <DATATABLE>

All of these classes inherit from the **Mapping`Element`** abstract class

VODml serialization

The structure of VODML instance has nothing more than complex JSON messages

It can be modeled as a tree of Tuple/Collection/Value

As we are not constrained by the JSON formalism (STring) we can had some metadata at each node

Test Results

Test achieved on hand-annotated VOTable and validated with my Java API

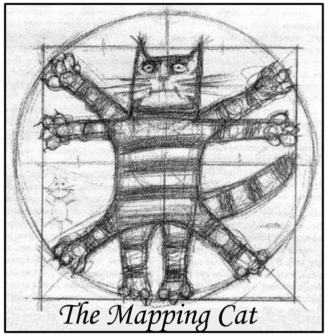
Test Case	Status	Comment
Simple model without **<DATATABLE>**	OK	
Simple model with **<DATATABLE>**	OK	Use of **<ARRAY>**
Simple model with **<DATATABLE>** and **<GLOBALS>**	OK	Use of **ID/ref**
Complex model: **TS** data model, a mix of **STC, DatasetMetadata, PhotDM** + time domain classes but one single light curve	OK	Model provided by Mark C.D. VOTable provided by ESAC
Complex model: **TS** data model, a mix of **STC, DatasetMetadata, PhotDM** + time domain classes but 3 light curves	OK	Use of **<ARRAY>** **<INSTANCE>** **<FILTER>**
Set of Time Series, one light curve each and grouped by bands	Work in progress	Use of **<SET groupby=”band”>**

My Proposal

- **JSON: my leitmotiv**
  - Incredibly complex data are exchange with JSON messages
  - JSON messages rely on 3 concepts
    - Values
    - Tuple
    - Collection
  - We must be able map our data with these 3 concepts
    - Could lose some ORM features
    - Will gain lot of expressivity
  - I do not propose to use JSON for the mapping
  - I propose to apply the JSON philosophy to our XML syntax
- **dmrole=root, my other leitmotiv**
  - Tagging the root object of the mapping with dmrole=root allows to clearly show what is the content of the VOTable

What I’m Experimenting with TD Data

- **Keeping the proposed workflow**
  - Reference to VODML models
  - VODML/MODELS/GLOBALS/TEMPLATES pattern
  - Mapping block below <VOTABLE>
  - A syntax reflecting the model structure



- **Helping Clients to see what the VOTable Content Is**

- **Supporting sa Much Existing Data Files as Possible**
  - Include directives for the parser such as aggregation operators

- **Syntax More Human Readable, then More Reliable**

My Guidelines

- **Syntax Simplification**
  - Just writing what the client really needs
  - Making it more human readable, then more reliable

- **Client Oriented**
  - Helping clients to identify what the actual content of the votable
  - Making easier the design of generic API (my talk in apps)

- **Versatility**
  - Supporting as much existing data files as possible
  - Making easier a possible templating