Reading VO-DML Annotations With Java Code

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

Laurent MICHEL - College Park - 2018

The VO-DML Stack

VOtable

model → Annotation process

Annotated VOTable

Annotation parsing

Client

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

Science
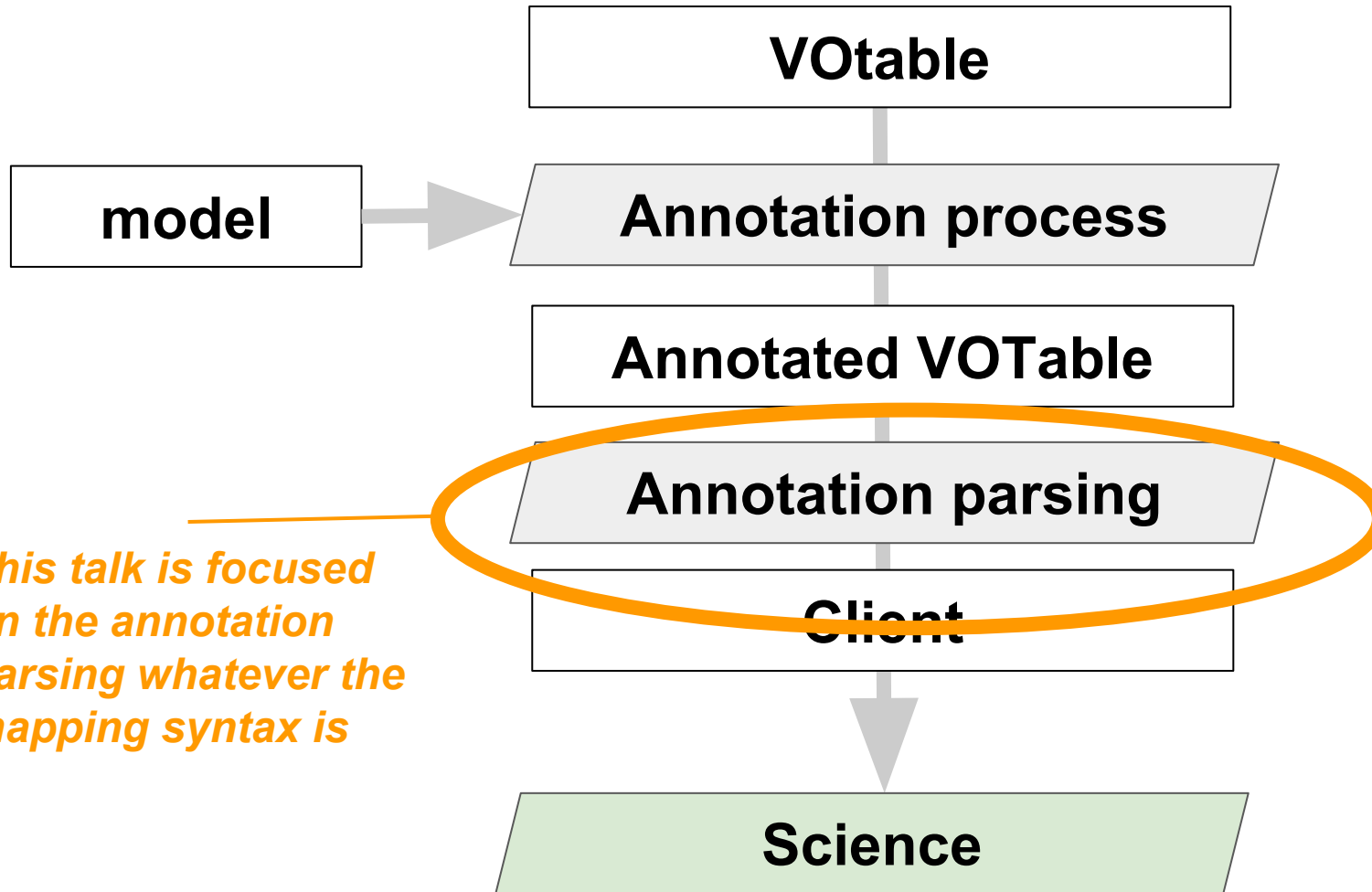
Client Expectations for Using Models

- ## 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
  - Python API (O.L. Victoria 2018 https://olaurino.gitlab.io/ivoa-dm-examples)

- ## A clear way to finally get the VOTable content
  - This feature is still a lack for the VOTable schema

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
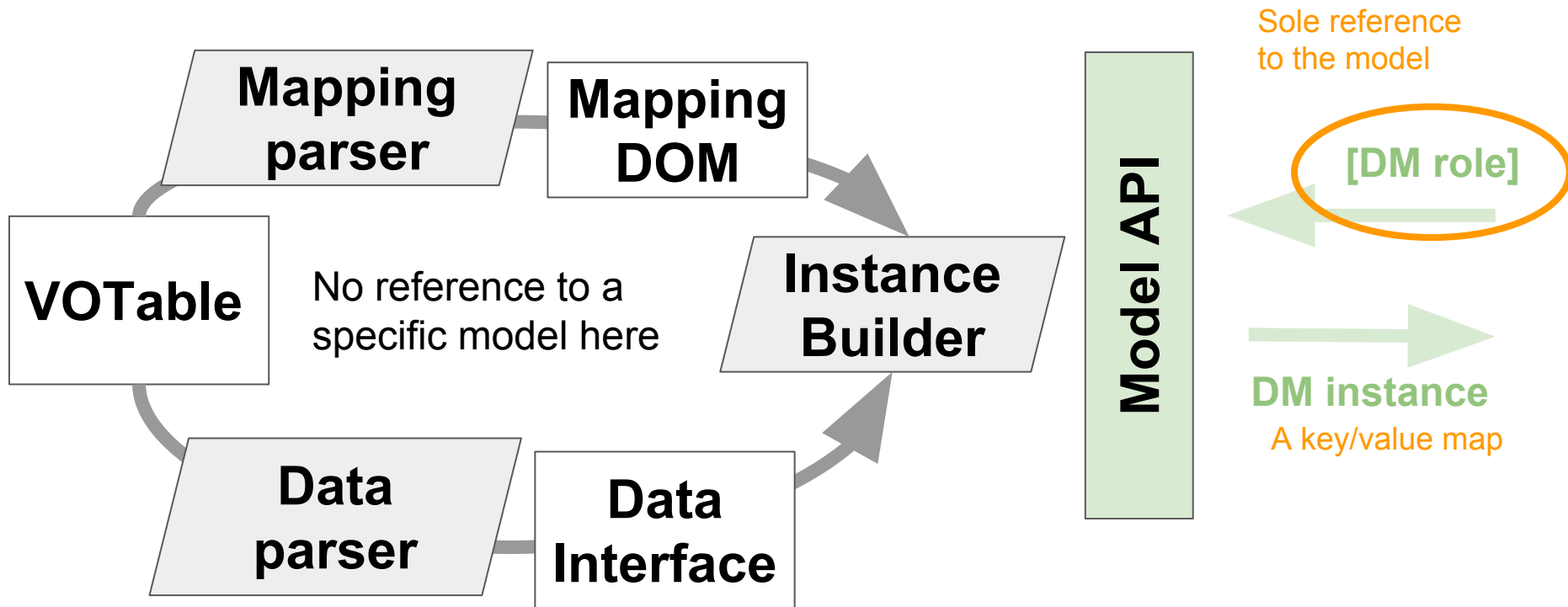
I Would Like to Have Something Like This

```java
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
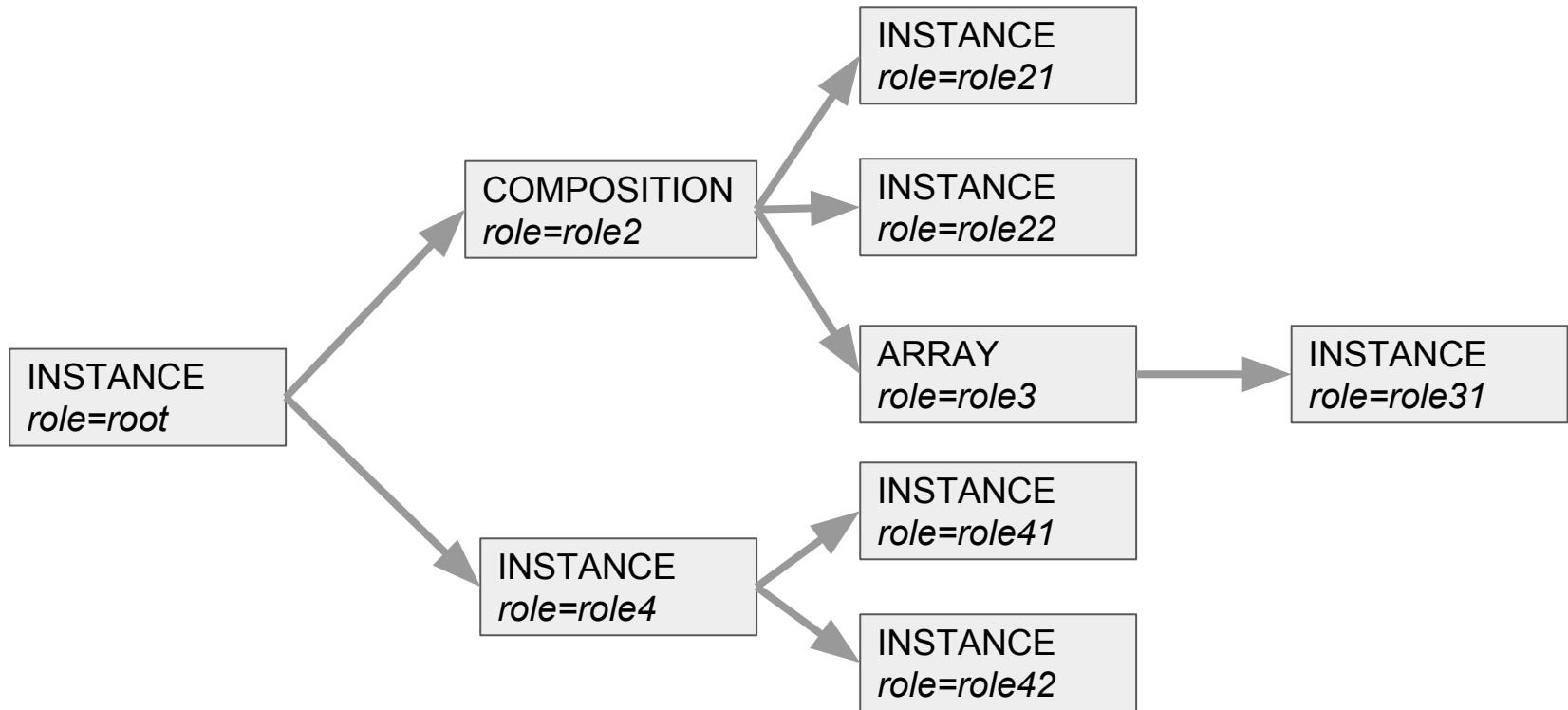- **In "green" :** Model related quantities, strings only

Architecture

Mapping parser

Mapping DOM

VOTable

No reference to a specific model here

Instance Builder

Data parser

Data Interface

Model API

Sole reference to the model

[DM role]

DM instance
A key/value map

Model API:
- A reference to the root object
- A set of selectors to browse it

Browsing the Mapping DOM

Browsing the Mapping DOM

INSTANCE
role=role21

COMPOSITION
role=role2

INSTANCE
role=role22

ARRAY
role=role3

INSTANCE
role=role31

INSTANCE
role=root

INSTANCE
role=role4

INSTANCE
role=role41

INSTANCE
role=role42
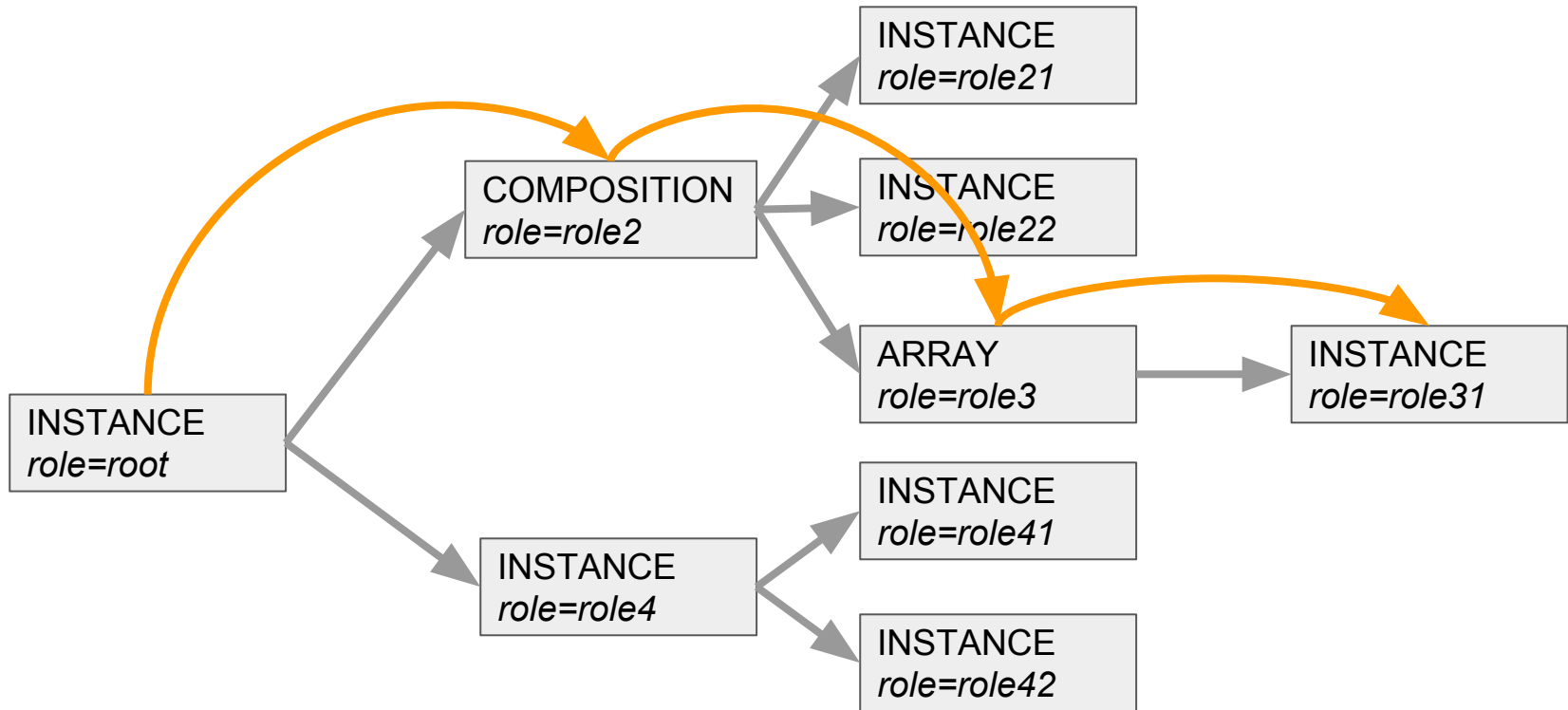
What the parser did

What the client does

```
Node(role=role31) = Node(role=root)
                      ->Node(role=role2)
                        ->Node(role=role3)
                          ->Node(role=role31)
```
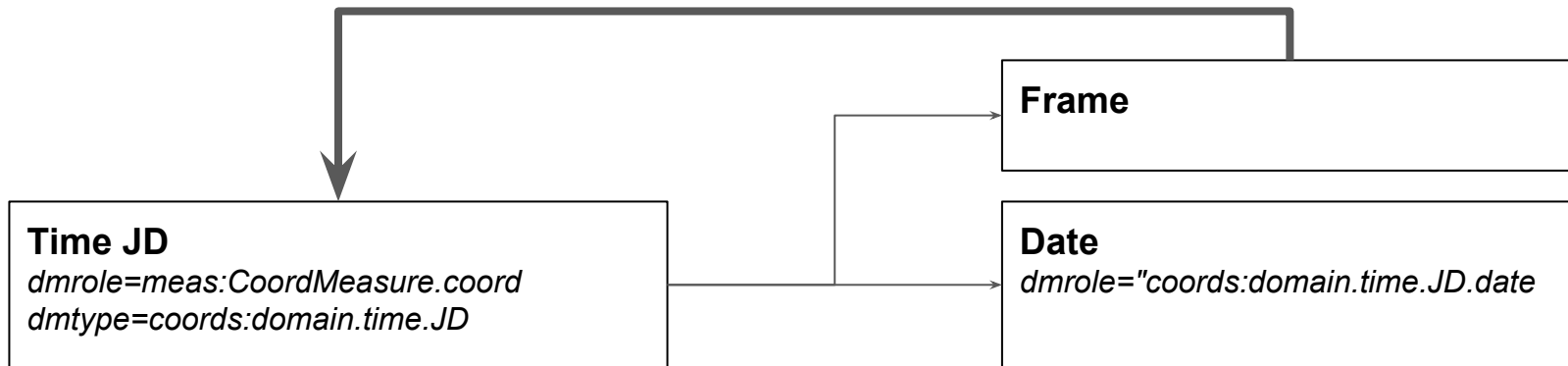
My API as it Is Now

Points onto the collection of contributors

```java
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());
    }
}
```
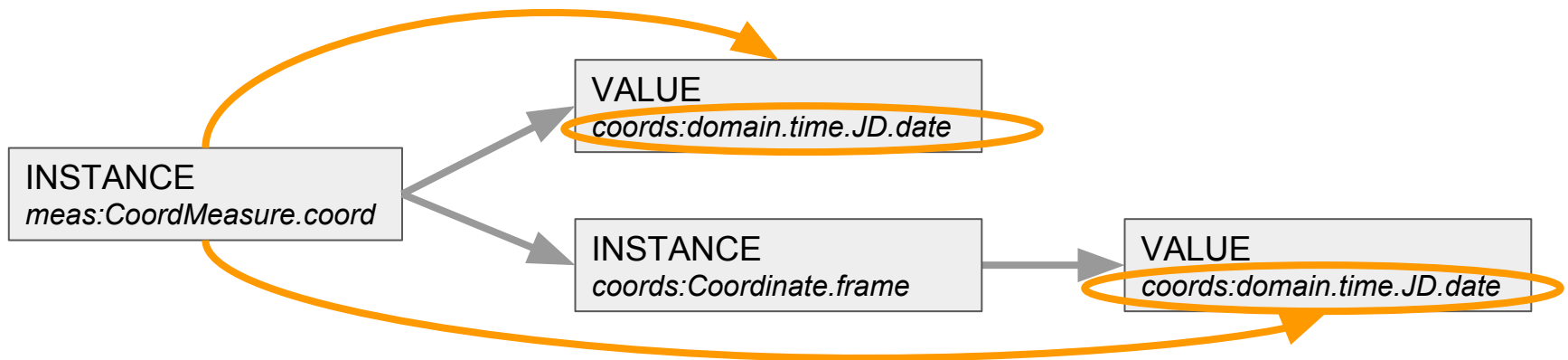
Retrieving the list of contributors

Take all acknowledgements of all contributors
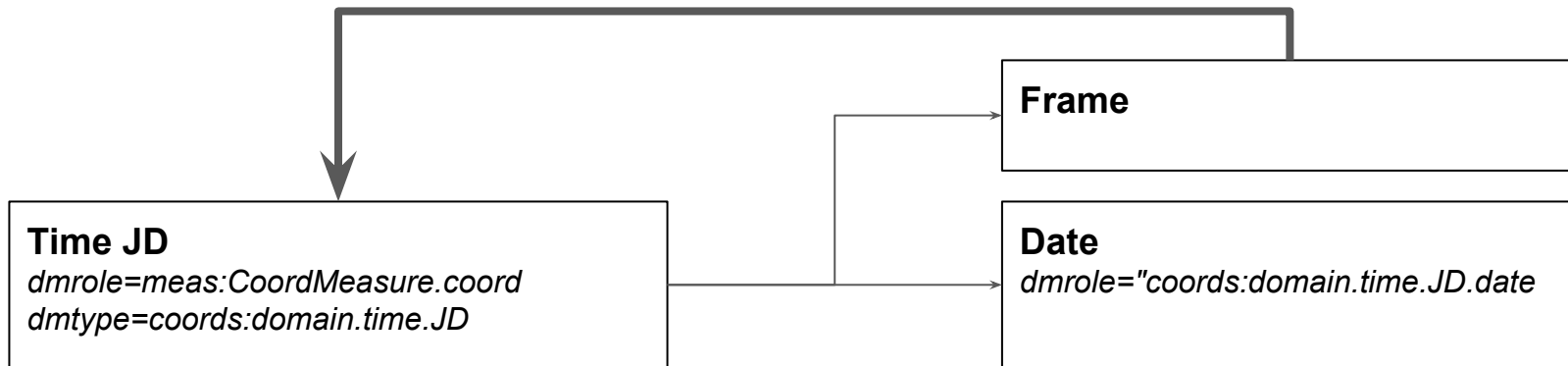
Sometime, a Risk of Confusions

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

Sometime, a Risk of Confusions

Frame

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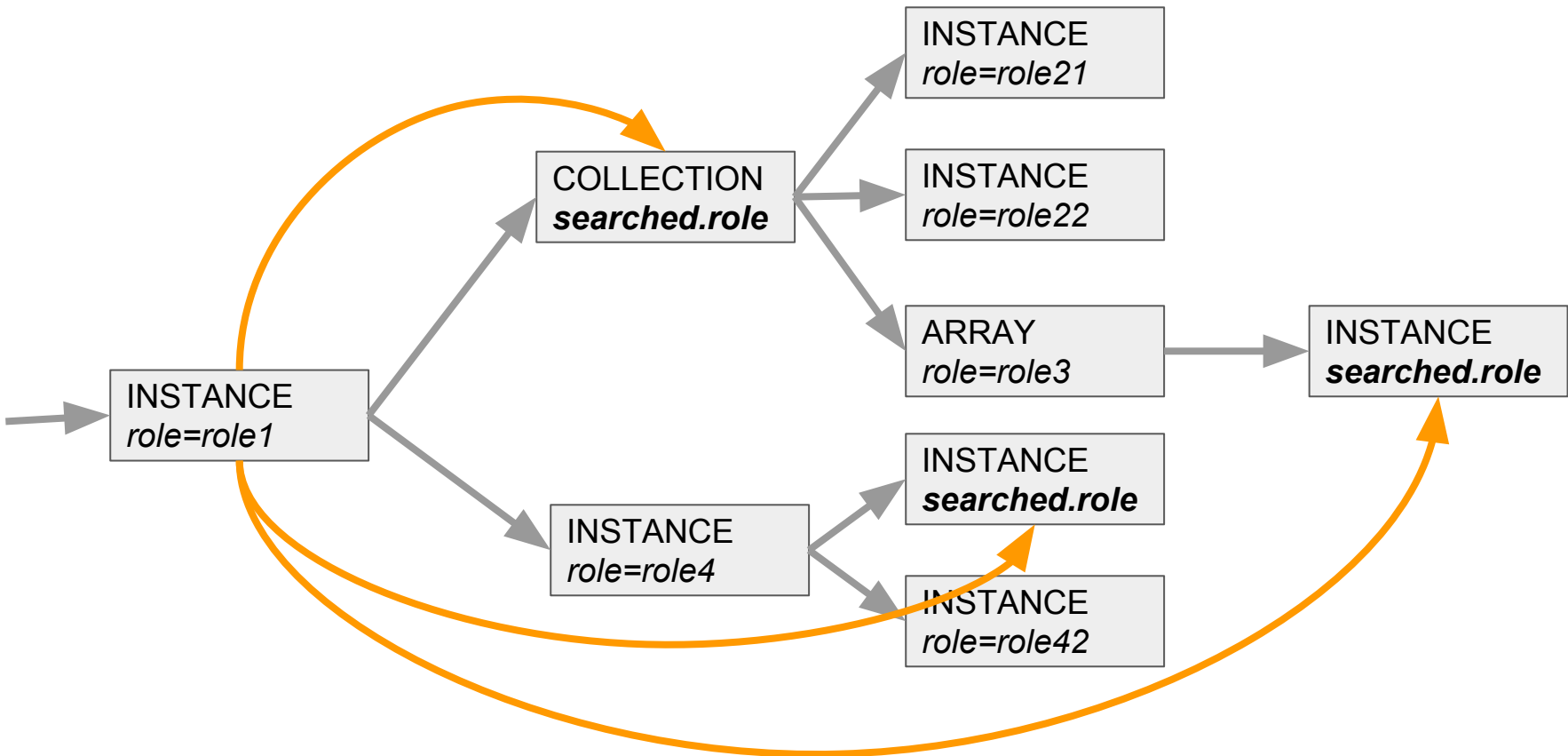*dmtype=coords:domain.time.JD*

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

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

VALUE
coords:domain.time.JD.date

INSTANCE
meas:CoordMeasure.coord

INSTANCE
coords:Coordinate.frame

VALUE
coords:domain.time.JD.date
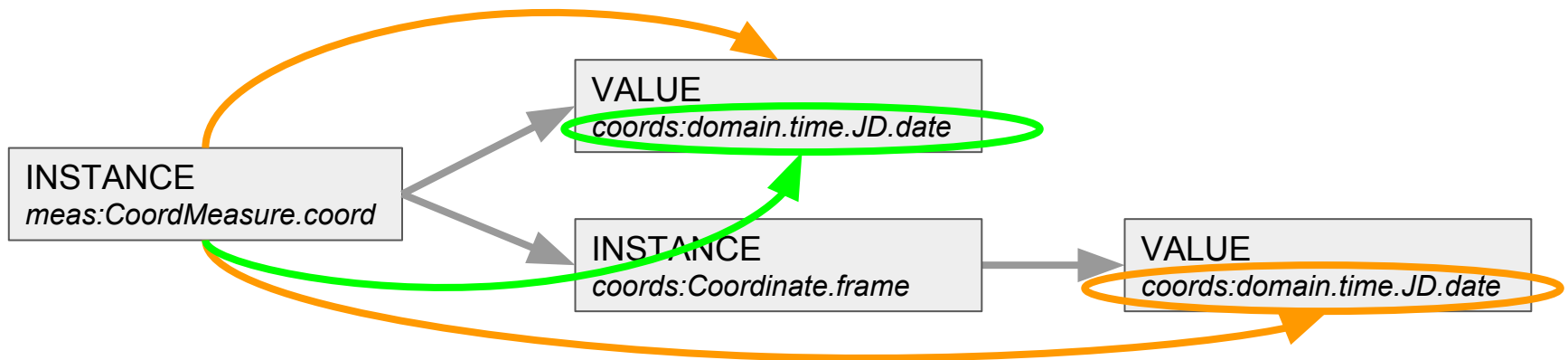
Very simplified model view

Mapping Element Selectors

INSTANCE
role=role1

COLLECTION
searched.role

INSTANCE
role=role21

INSTANCE
role=role22

ARRAY
role=role3

INSTANCE
searched.role

INSTANCE
role=role4

INSTANCE
searched.role

INSTANCE
role=role42

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

Mapping Element Selectors

INSTANCE
role=role1

COLLECTION
searched.role

INSTANCE
role=role21

INSTANCE
role=role22

ARRAY
role=role3

INSTANCE
searched.role

INSTANCE
role=role4

INSTANCE
searched.role

INSTANCE
role=role42
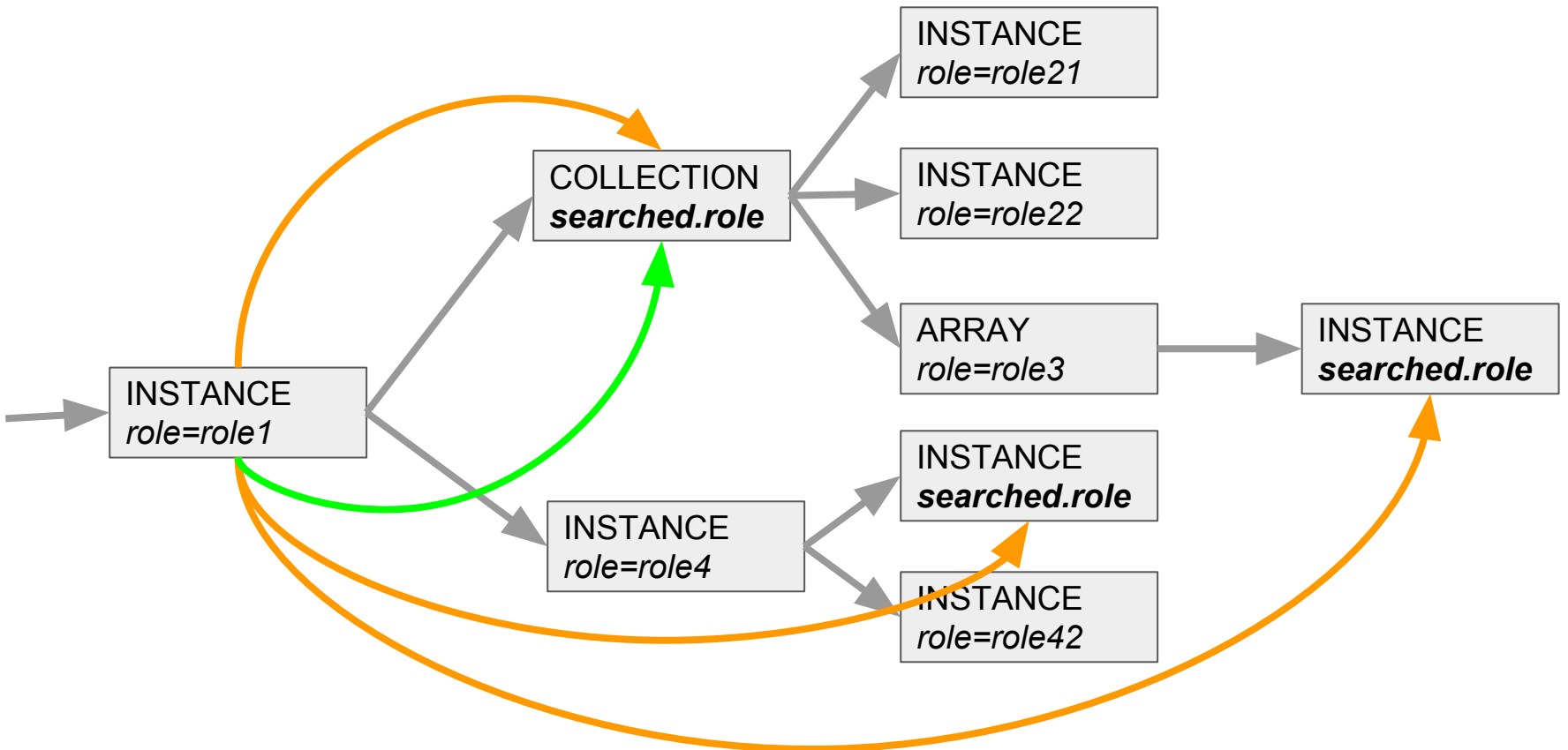
getSubElement... Return one or all sub-element (s) matching the role

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

Another Confusing Pattern

```
┌────────────────────────────┐              ┌────────────────────────────┐           ┌──────────────────┐
│ Observable                 │      1    2  │ Measure                    │           │ Time             │
│ Let's say a photometric    │─────────────▶│ Role = CoordMeasure.coord  │◀──────────│                  │
│ point                      │              │                            │    │      └──────────────────┘
└────────────────────────────┘              └────────────────────────────┘    │      ┌──────────────────┐
                                                                               └──────│ Magnitude        │
                                                                                      └──────────────────┘
```
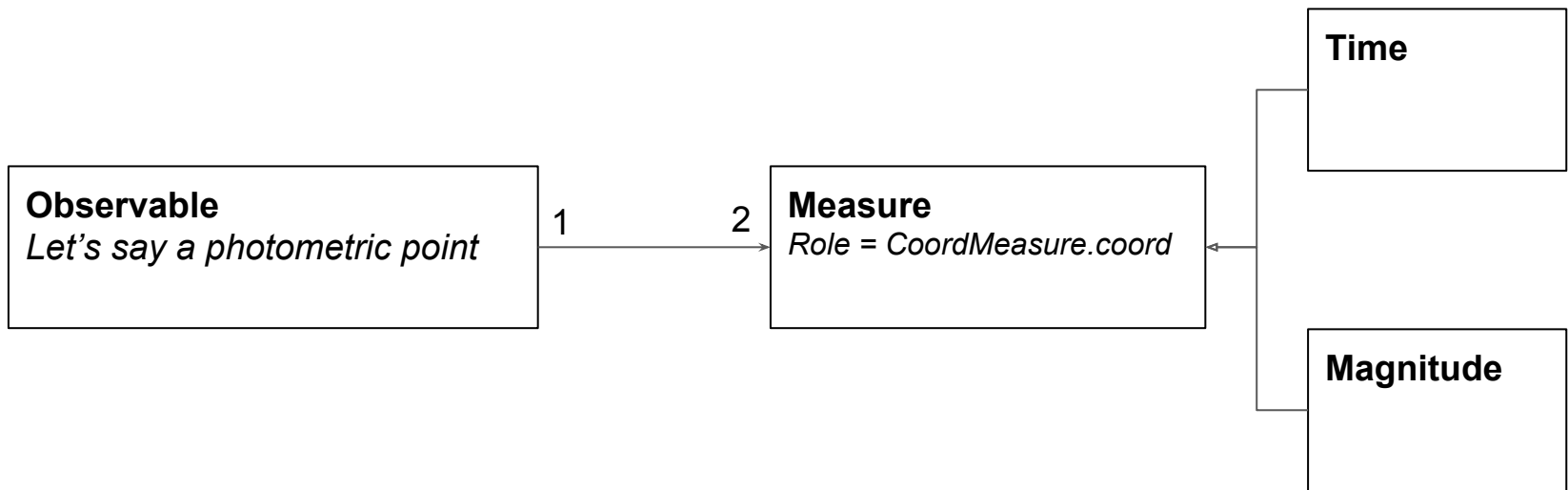
- The 2 Measures have the same role.
- To know what is what, we have to check the `dmtype` (class name) or to explore the inside of each instance

Another Confusing Pattern

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 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 *(see TDIG talk on Nov 10)*

- **Being Done**
  - Group by facility *<SET groupby=".."> (see TDIG talk on Nov 10)*

- **Todo**
  - Simplify the API
  - Using *DMTypes*
  - Foreign keys implementation
  - Test on an extented data sample

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 **`MappingElement`** 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

The Mapping Cat
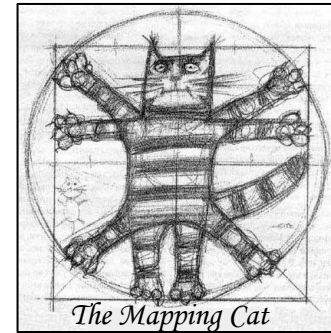
- ## Helping Clients to see what the VOTable Content Is

- ## 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