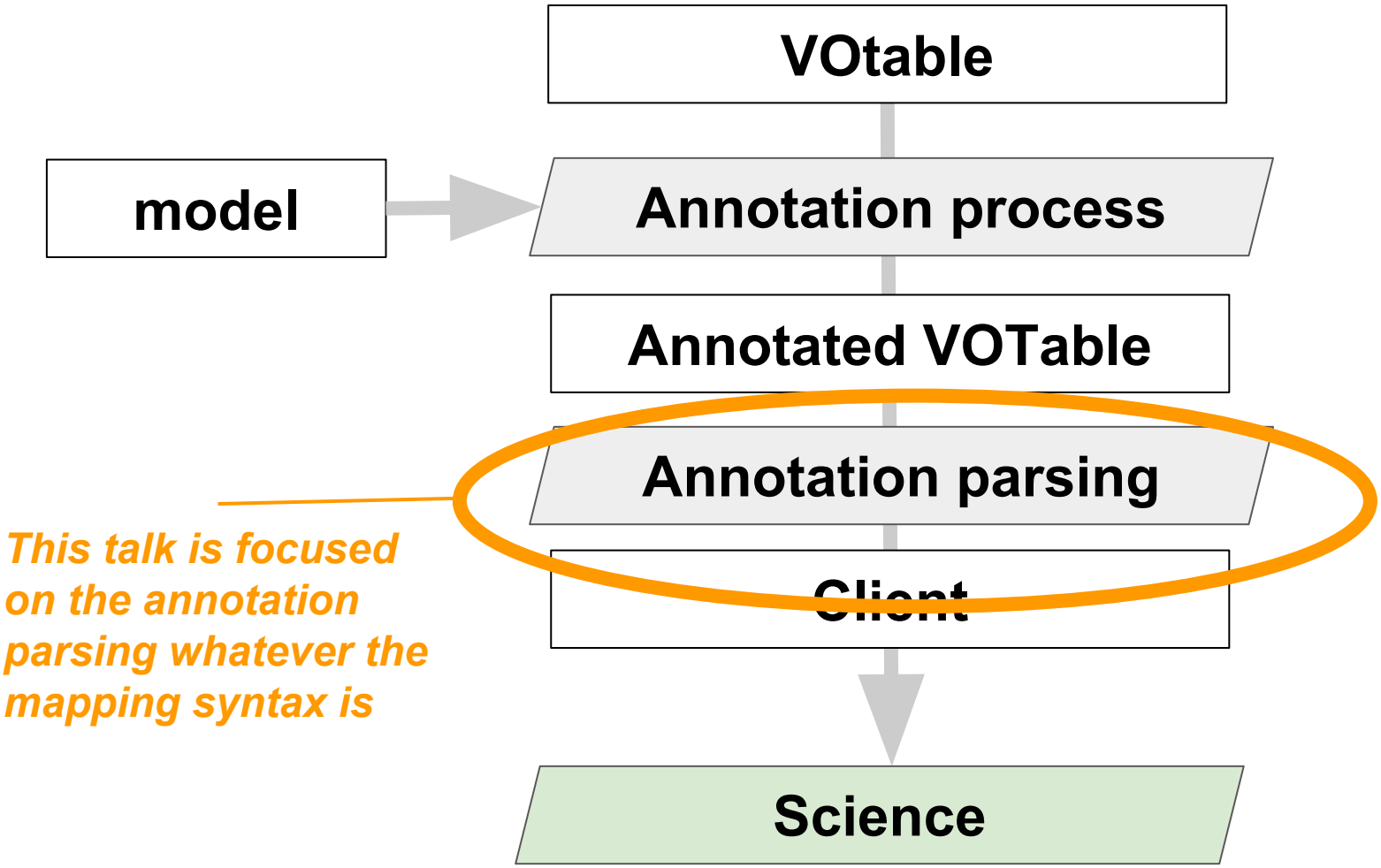
Reading VO-DML Annotations With Java Code

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

The VO-DML Stack



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

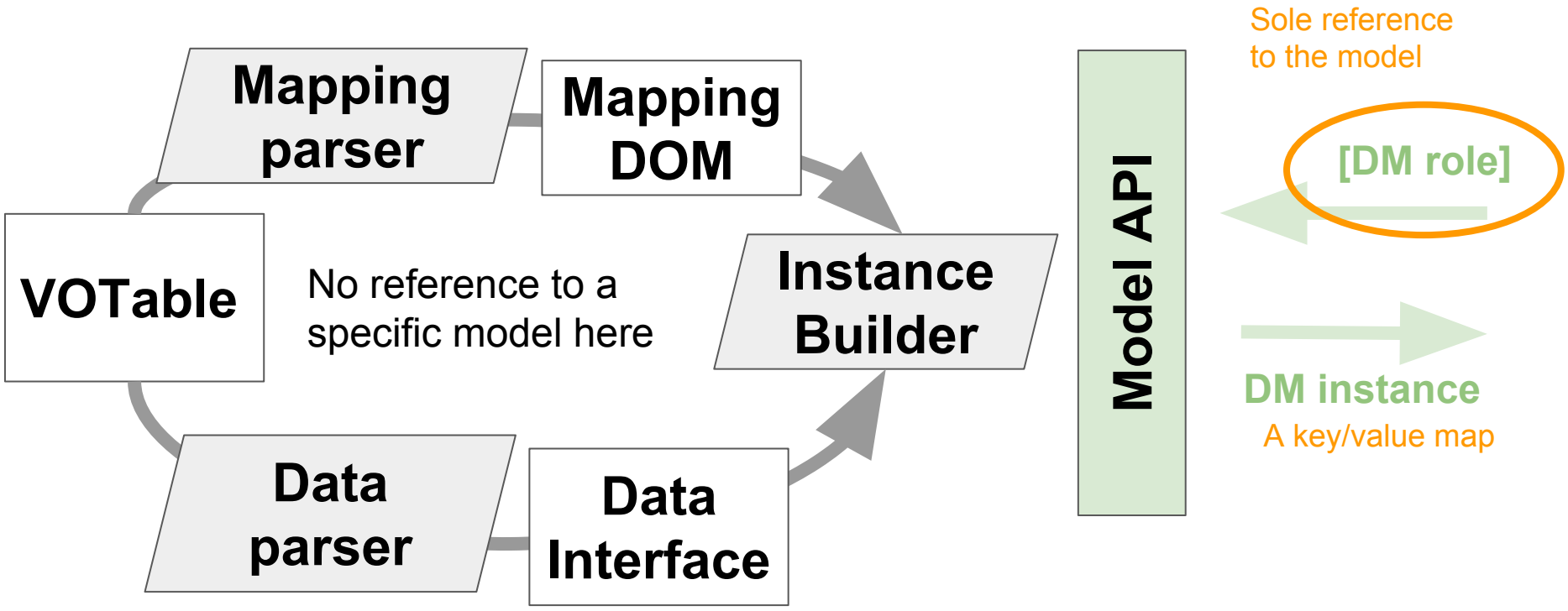
```
vodmlParser = new VodmlParser("Myvotable");

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

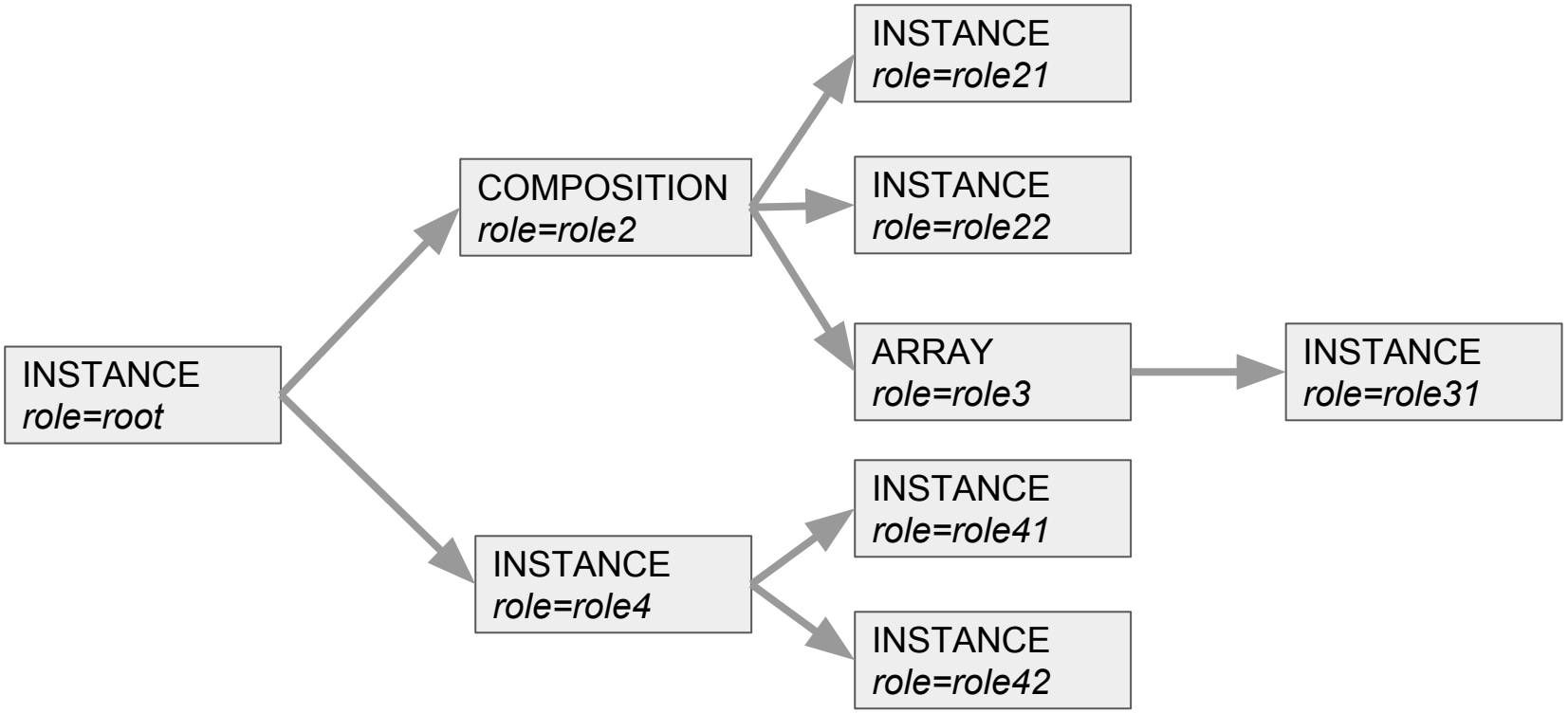
Architecture



Model API:

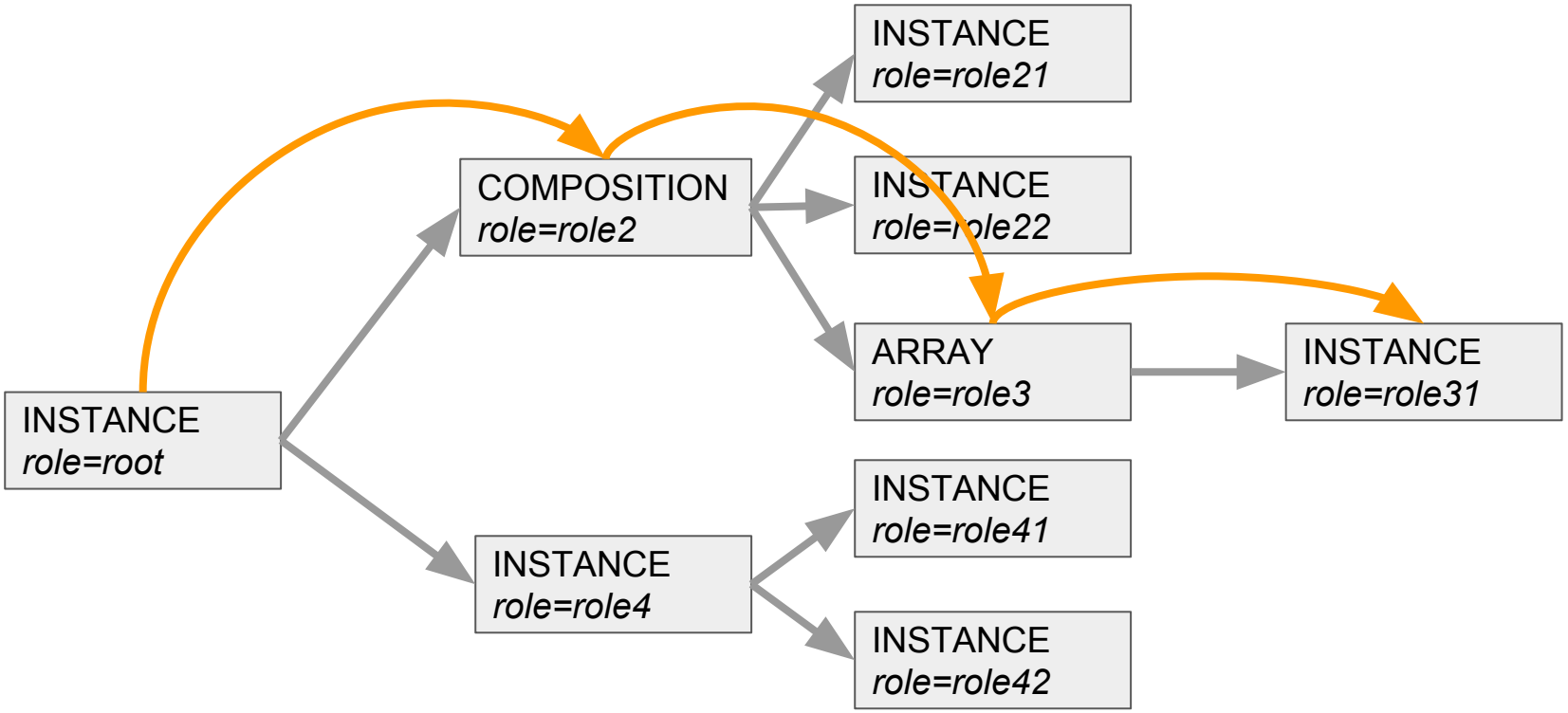
- A reference to the root object
- A set of selectors to browse it

Browsing the Mapping DOM



What the parser did

Browsing the Mapping DOM



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

The dataset object is supposed to be unique

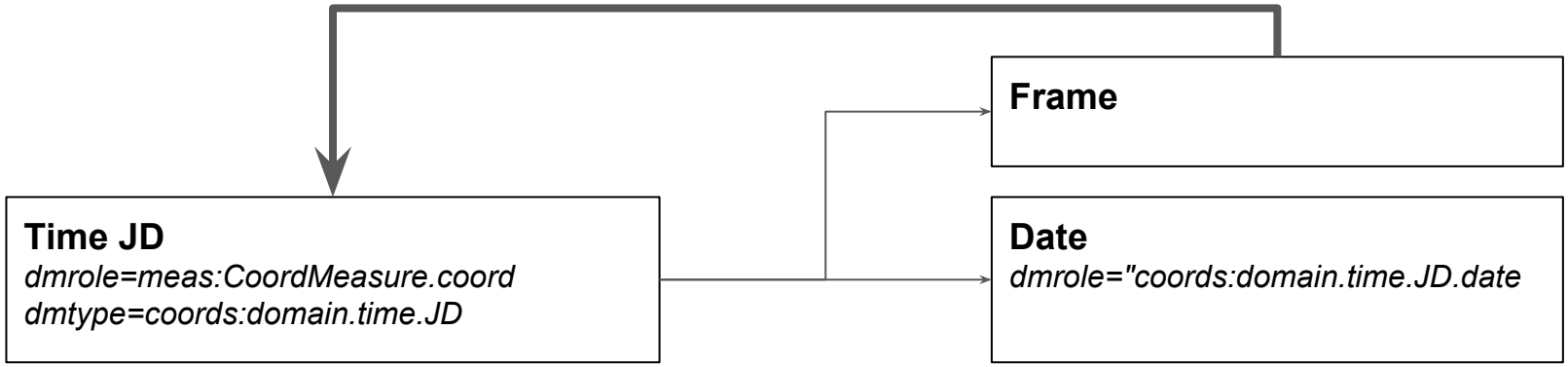
Points onto the collection of contributors

```
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());
    }
}
```

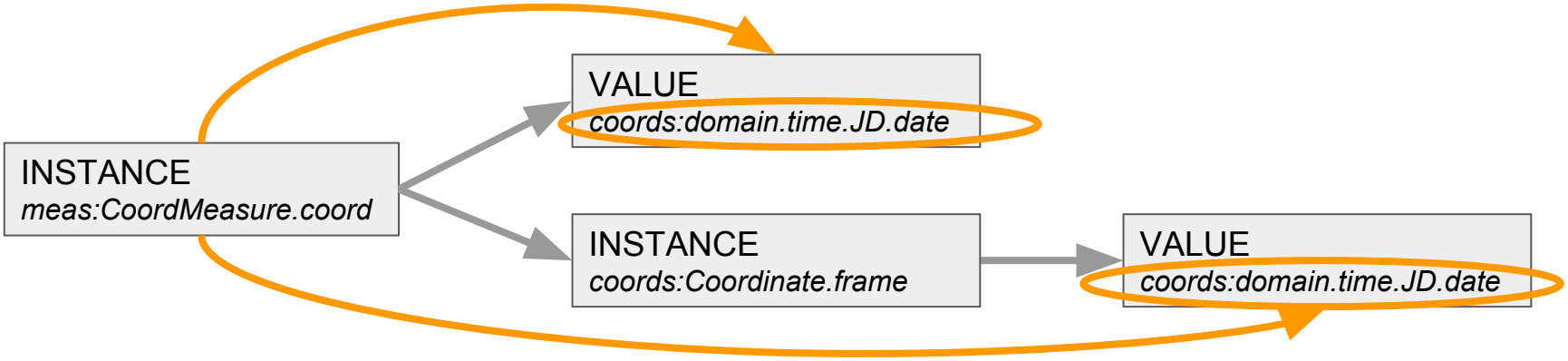
Retrieving the list of contributors

Take all acknowledgements of all contributors

Sometime, a Risk of Confusions

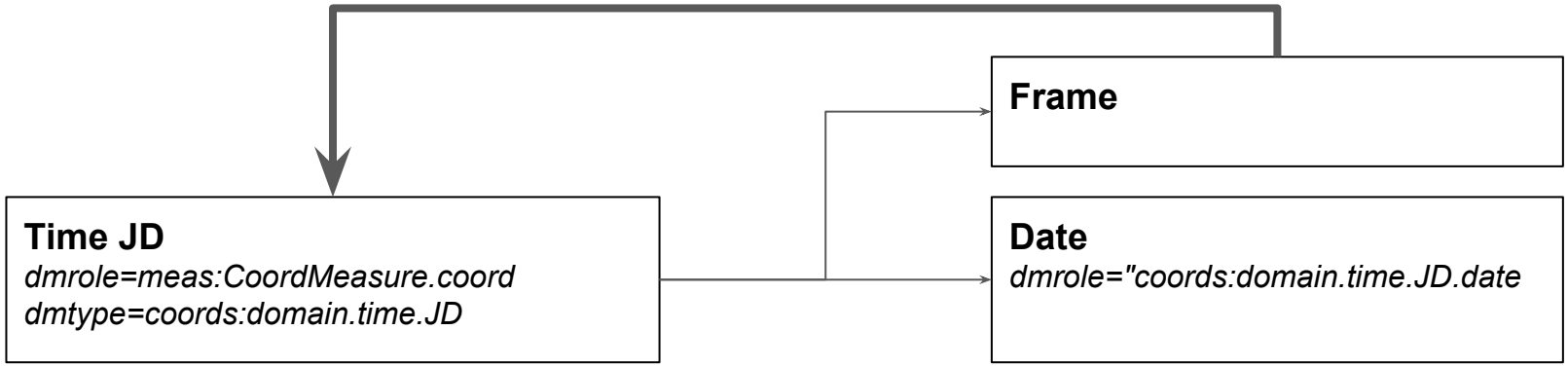


- Isolating the timestamp *date* with selectors based on *dmroles* may be confusing

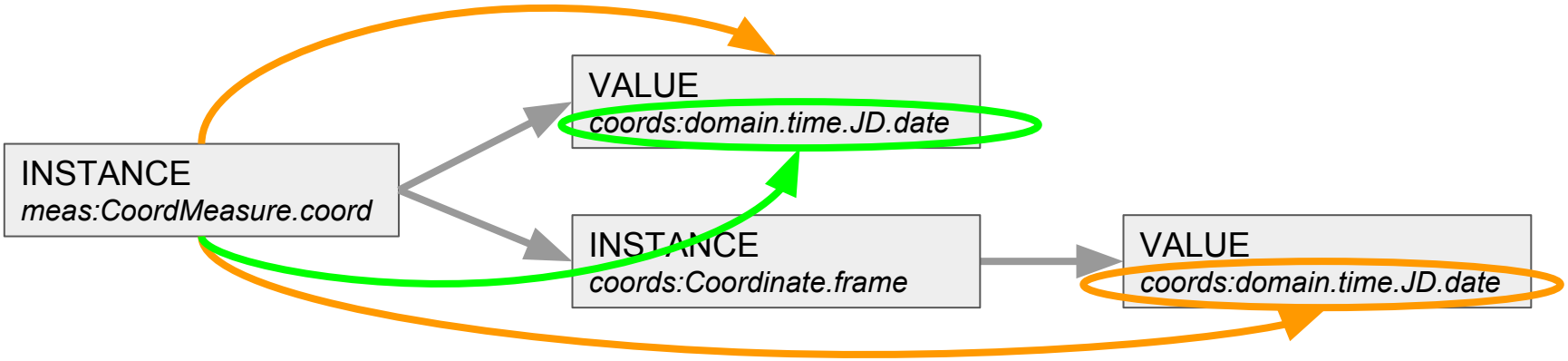


Very simplified model view

Sometime, a Risk of Confusions

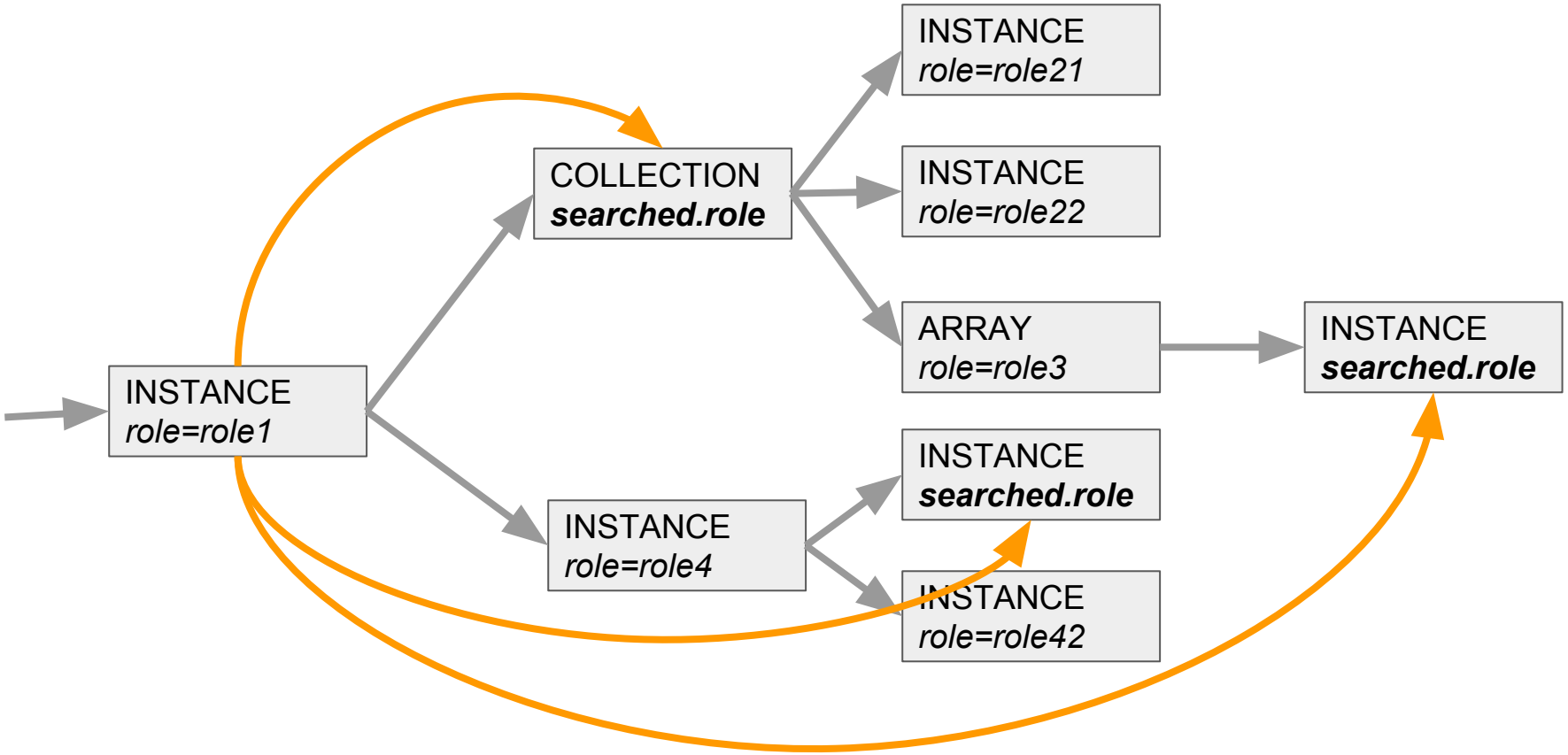


- Isolating the timestamp *date* with selectors based on *dmroles* may be confusing



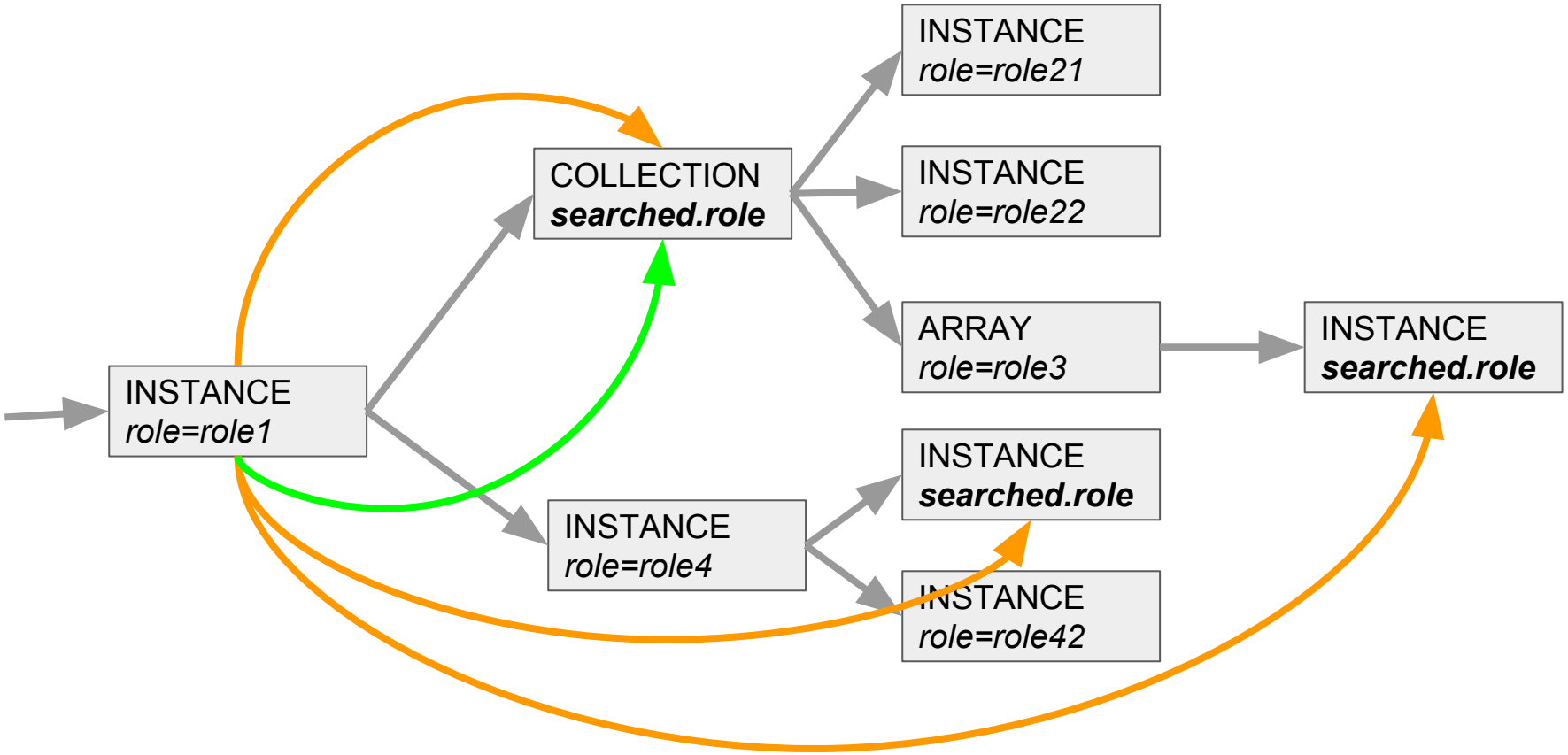
Very simplified model view

Mapping Element Selectors



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

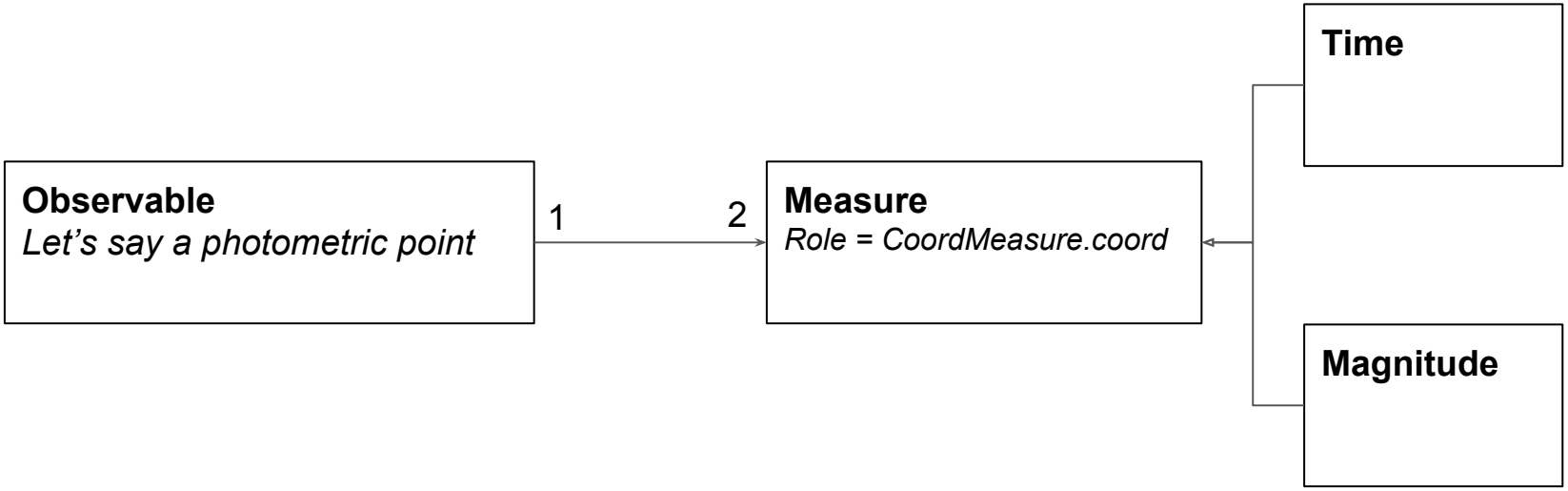
Mapping Element Selectors



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

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

Another Confusing Pattern



- 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