Data Models: Homogeneous view for heterogeneous data

Jesús Salgado
Laurent Michel
On behalf of Data Model Working Group

Why Data Models

- XXI century astronomy is plenty of data in different formats
- Both new astronomy and legacy data need to be combined
- Applications analyze heterogeneous data and extract results. This could be erroneous if data is not properly combined
- Serialization of data is only partially standardized

"Babel" from the Hebrew verb בָּלַל (*bālal*), meaning to jumble or to confuse.

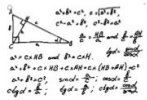
IVOA Architecture

LEVEL 1

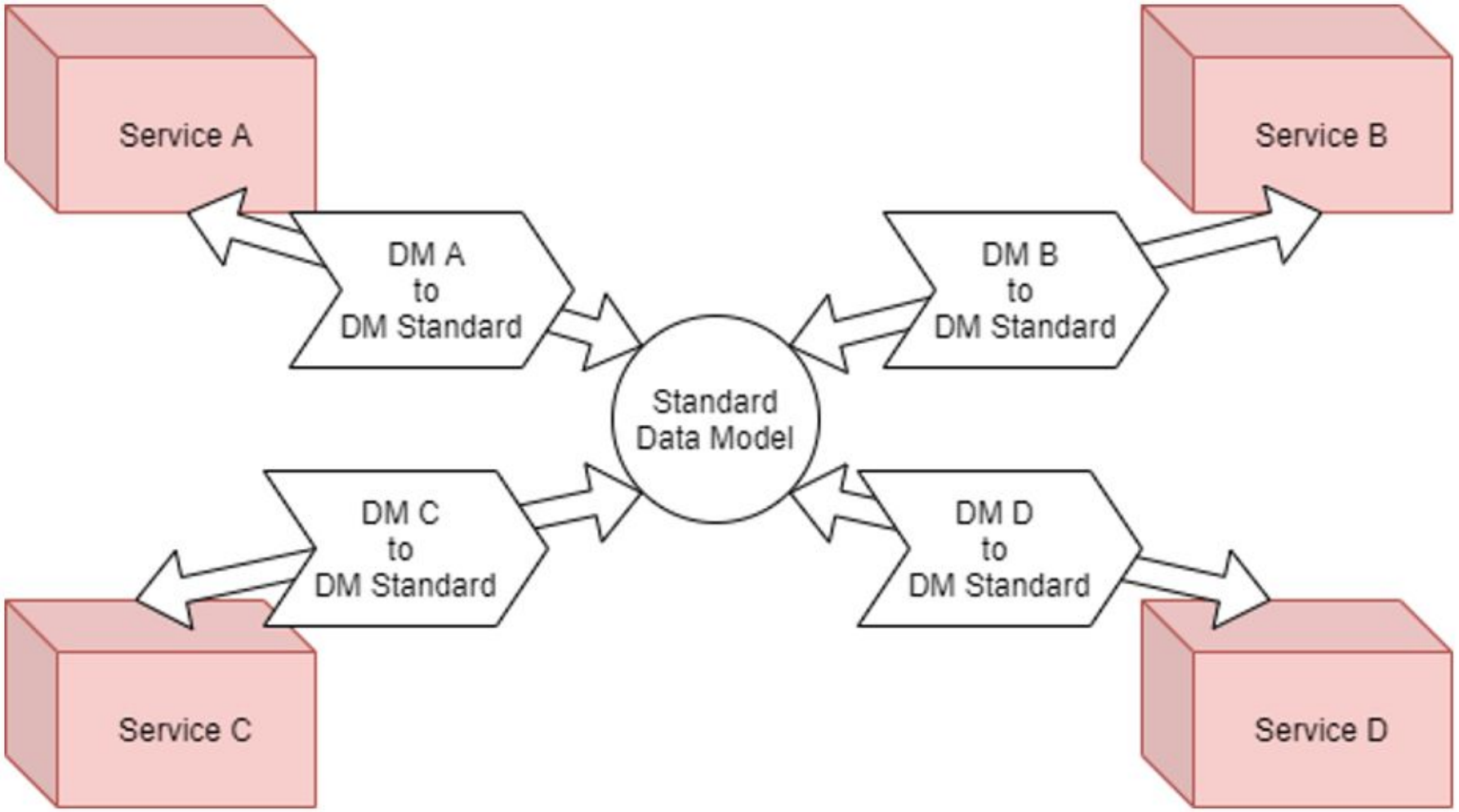
		USER LAYER		
	Browser Based Apps	*Desktop Apps*	*Script Based Apps*	
		USING		
REGISTRY	*Semantics*	*VO Query Languages* VO CORE *Formats*	*Data Models*	DATA ACCESS PROTOCOLS
		SHARING		
	Storage	*Data and Metadata Collection* RESOURCE LAYER	*Computation*	

20101004
IVOA Architecture

PROVIDERS



European Connected Factory Platform for Agile Manufacturing

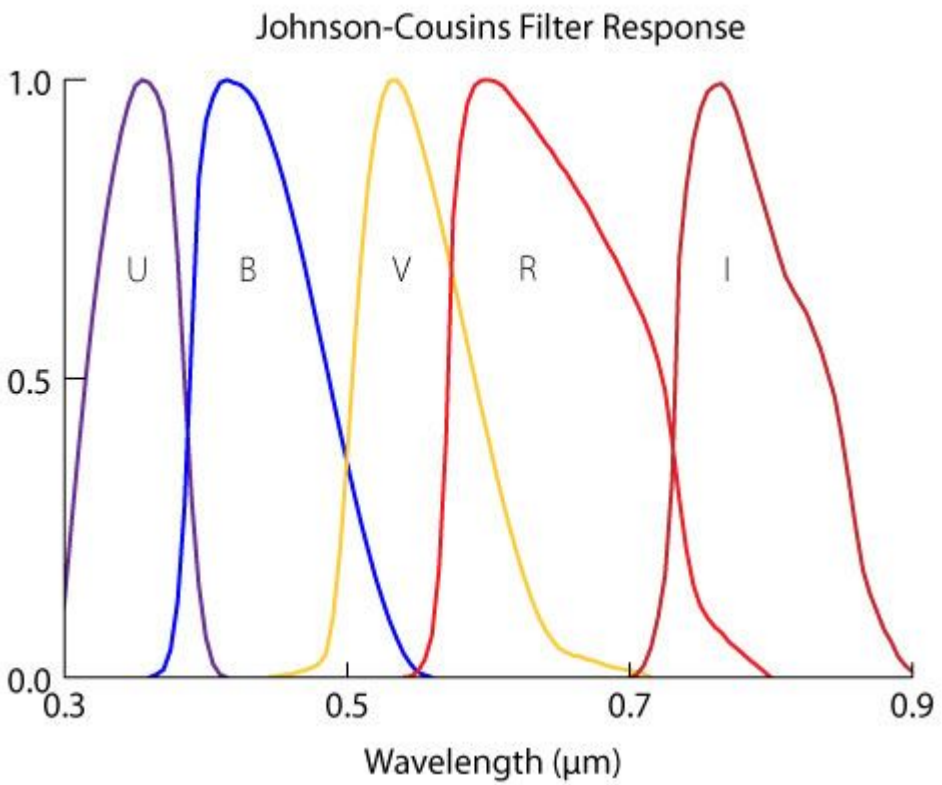


Data Models

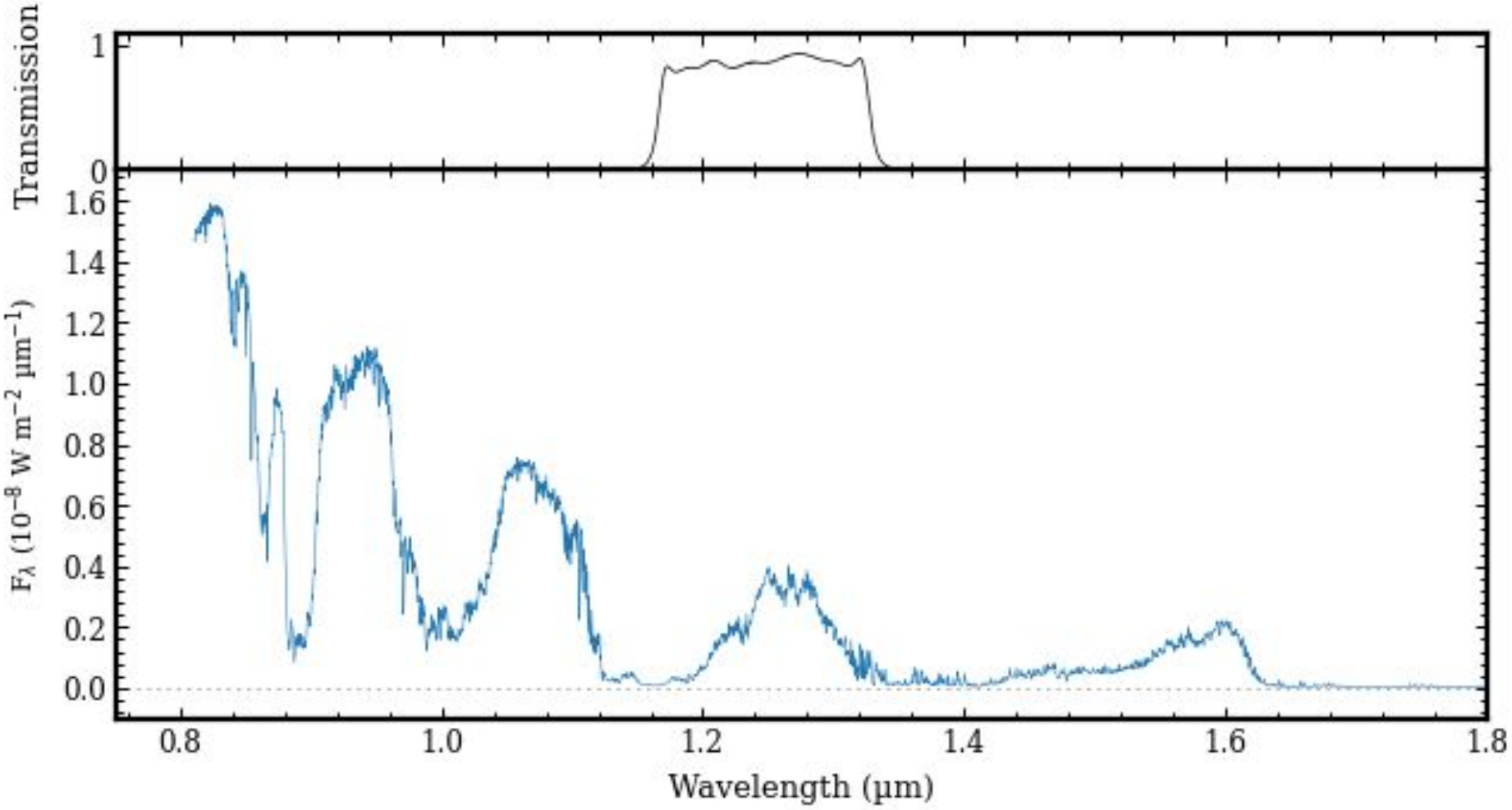
- **Data Model:** A data model is a visual representation of data elements and the relationships between them
- **Benefits:**
  - Improve discovery
    - -> IVOA discovery protocols
  - Standardization and documentation of data sources
    - -> Ensure metadata is comparable
  - Successfully design and implement databases
    - -> Catalogues
  - Improve interoperability
    - All IVOA

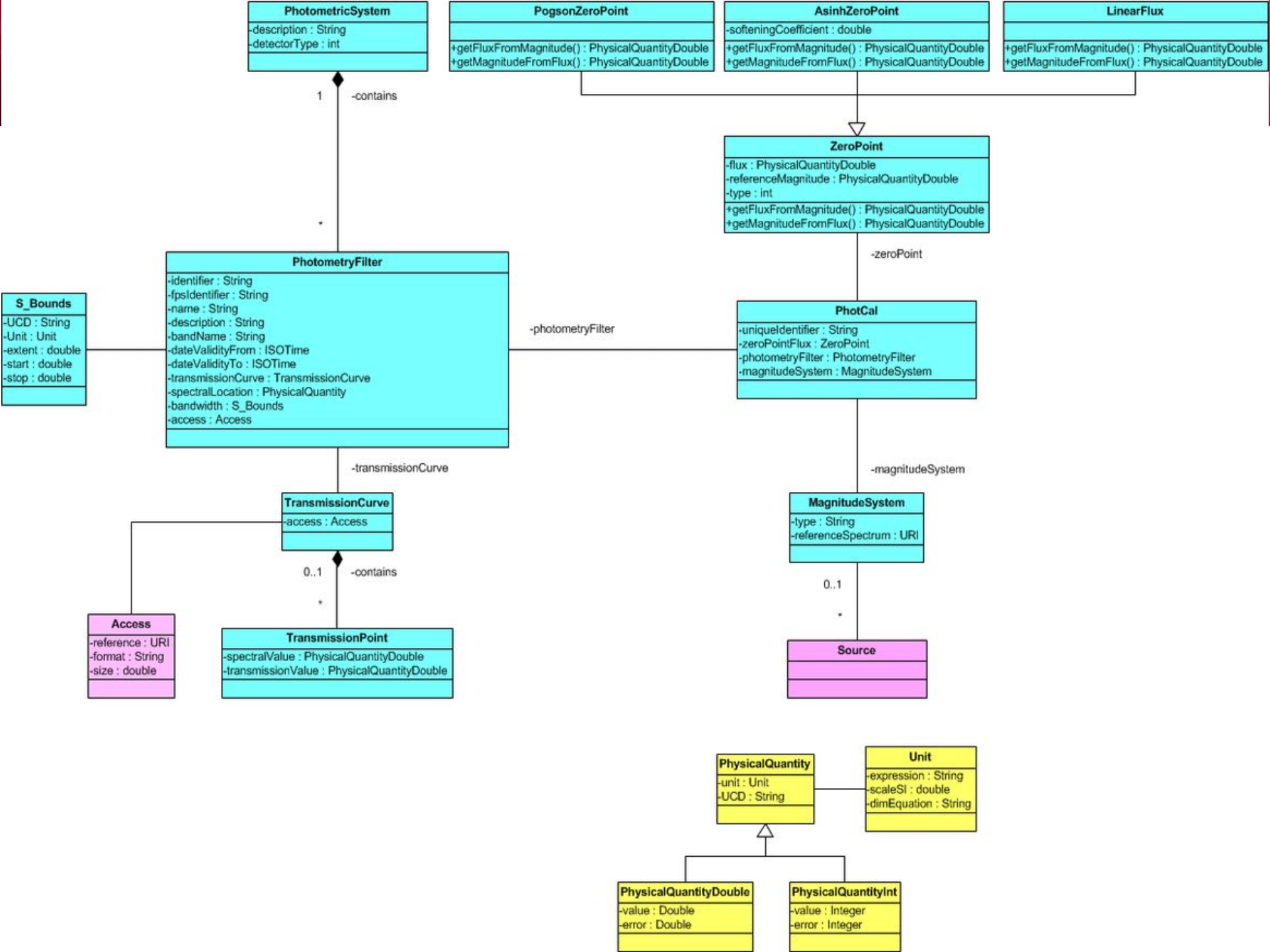
A case of success: Photometry

- Astronomical magnitudes used historically in all astronomy
- Magnitudes are relative measurements for a certain observatory
- How to compare magnitudes from different observatories?



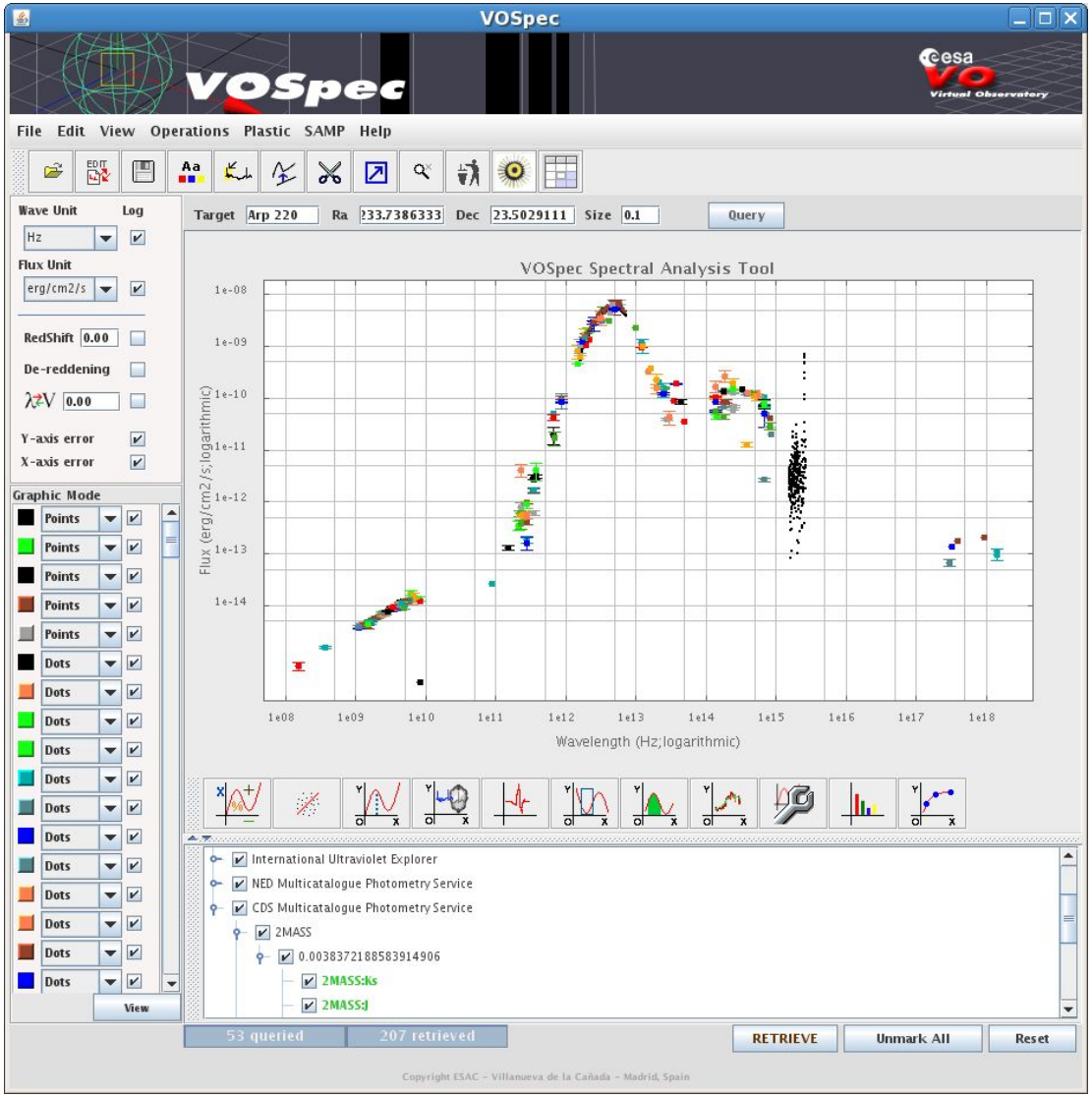
Synthetic photometry




PhotometricSystem
-description : String
-detectorType : int
PogsonZeroPoint
+getFluxFromMagnitude() : PhysicalQuantityDouble
+getMagnitudeFromFlux() : PhysicalQuantityDouble
AsinhZeroPoint
-softeningCoefficient : double
+getFluxFromMagnitude() : PhysicalQuantityDouble
+getMagnitudeFromFlux() : PhysicalQuantityDouble
LinearFlux
+getFluxFromMagnitude() : PhysicalQuantityDouble
+getMagnitudeFromFlux() : PhysicalQuantityDouble
1
-contains
*
ZeroPoint
-flux : PhysicalQuantityDouble
-referenceMagnitude : PhysicalQuantityDouble
-type : int
+getFluxFromMagnitude() : PhysicalQuantityDouble
+getMagnitudeFromFlux() : PhysicalQuantityDouble
-zeroPoint
PhotometryFilter
-identifier : String
-fpsIdentifier : String
-name : String
-description : String
-bandName : String
-dateValidityFrom : ISOTime
-dateValidityTo : ISOTime
-transmissionCurve : TransmissionCurve
-spectralLocation : PhysicalQuantity
-bandwidth : S_Bounds
-access : Access
S_Bounds
-UCD : String
-Unit : Unit
-extent : double
-start : double
-stop : double
-photometryFilter
PhotCal
-uniqueIdentifier : String
-zeroPointFlux : ZeroPoint
-photometryFilter : PhotometryFilter
-magnitudeSystem : MagnitudeSystem
-transmissionCurve
-magnitudeSystem
TransmissionCurve
-access : Access
MagnitudeSystem
-type : String
-referenceSpectrum : URI
0..1
-contains
*
0..1
*
Access
-reference : URI
-format : String
-size : double
TransmissionPoint
-spectralValue : PhysicalQuantityDouble
-transmissionValue : PhysicalQuantityDouble
Source
PhysicalQuantity
-unit : Unit
-UCD : String
Unit
-expression : String
-scaleSI : double
-dimEquation : String
PhysicalQuantityDouble
-value : Double
-error : Double
PhysicalQuantityInt
-value : Integer
-error : Integer

Utypes approach

General Metadata				
Utype	**UCD 1+**	**Meaning**	**Default value**	**Data type**
Datamodel.name	meta.id	Data Model Identification	PhotCalDM-v1.0	string
Photometric System Metadata				
Utype	**UCD 1+**	**Meaning**	**Default value**	**Data type**
photDM:PhotometricSystem.description	meta.note	String representation Photometric System		string
photDM:PhotometricSystem.detectorType	meta.code	Type of detector (e.g energy or photon counter). Possible values defined by enumeration	0 (Energy Counter)	int
Photometry Filter General Metadata				
Utype	**UCD 1+**	**Meaning**	**Default value**	**Data type**
photDM:PhotometryFilter.identifer	meta.ref.ivorn	Unique identifer of filter within a Filter Profile Service (FPS)		string
photDM:PhotometryFilter.fpsIdentifier	meta.ref.ivorn	IVOA identifier of the Filter Profile Service		string
photDM:PhotometryFilter.name	meta.id;instr.filter	Filter Name in the instrumental configuration		string
photDM:PhotometryFilter.description	meta.note	Text description of the filter band		string



VO/DML

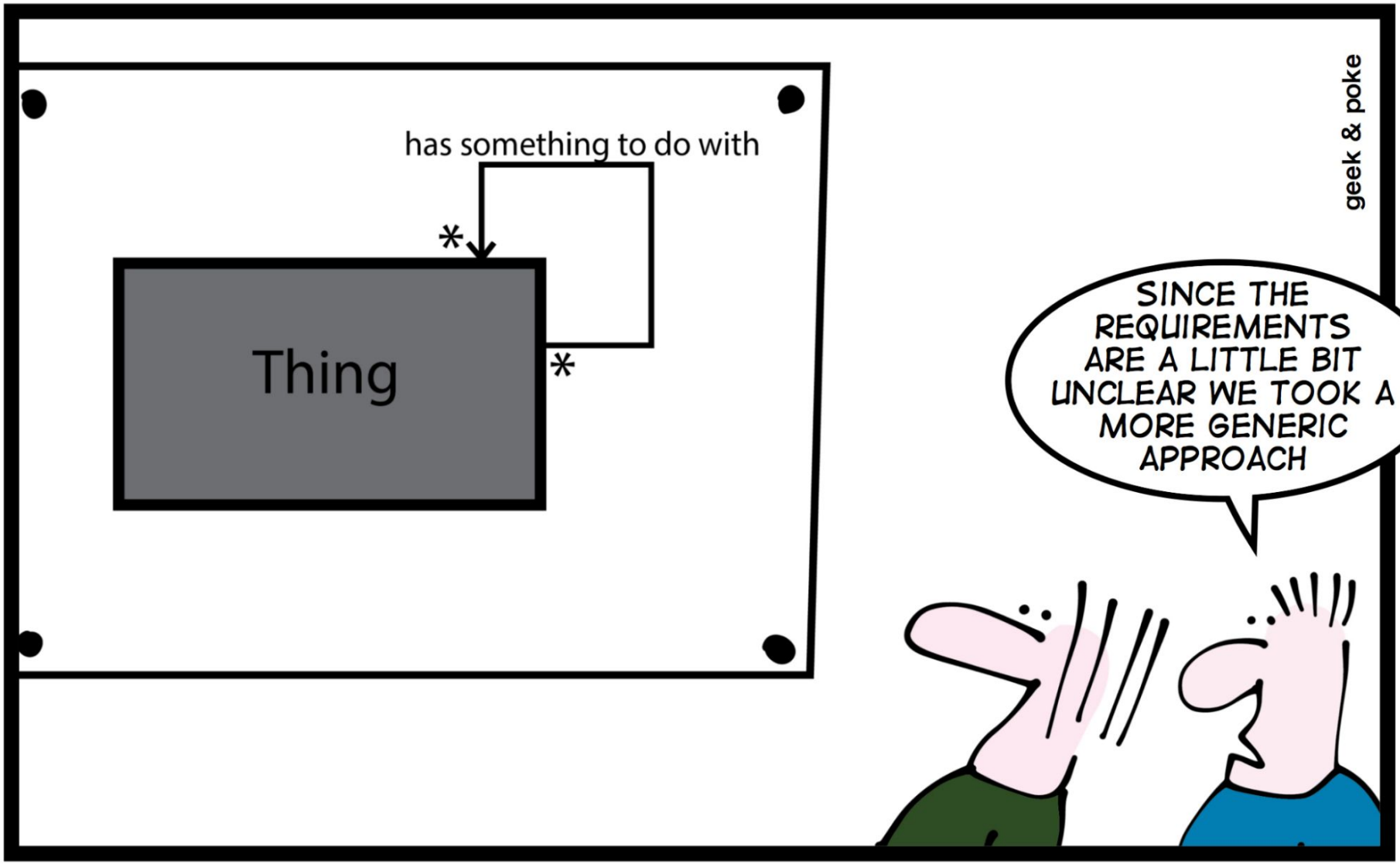
- **VO/DML** offers a way to document data models so they can be understood by a machine
- Includes diagrams (UML), xmi (XML DM description), .vo-dml.xml (IVOA DM XML description) and multiple htmls
- It is focused on the data model more than in the use case

  xmi (no standard)

  vo-dml.xml

  html

Question 1: How complex?



Question 1: How complex?

- **IVOA DM** should be complex enough to allow scientific use cases but simple enough so data providers can map their data
- A complex data model could sometimes needed. In other cases, a simplified set of elements could be used by **annotations**
- **Driven by scientific use cases**

Question 2: How to describe and annotate?

- Data models should be described in a machine readable format (VO/DML)
- Object instances should be annotated and/or serialized in a format that can be interpreted into client applications
- Most VO formats are tabular based (relational) more than object oriented
- Which technique should we used?
- **We will discuss some of them**