International
Virtual
Observatory
Alliance

IVOA TimeSeries data modelling and representation

Version 1.0

IVOA Note 2018-05-22

Working group
TimeDomain
This version
http://www.ivoa.net/documents/TSSerializationNote/20180522
Latest version
http://www.ivoa.net/documents/TSSerializationNote
Previous versions

Author(s)
Francois Bonnarel, Mireille Louys, Ada Nebot, Laurent Michel, Mark Cresitello-Dittmar
Editor(s)
Ada Nebot

Version Control
Revision 4724, 2018-01-29 15:20:52 +0100 (lun. 29 janv. 2018)
https://volute.g-vo.org/svn/trunk/projects/ivoapub/ivoatexDoc/ivoatexDoc.tex

Abstract

Purpose of the note

Status of this document

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

Acknowledgments

1 Introduction

This is the intro of the Note.

2 Time Series

In this section we describe what Time Series data is in a wide context, defining the most relevant parameters that define it. We describe the common requirements of the different science use cases collected by the SPC **?**. A common frame for time is defined with the minimum set of parameters taken from and compatible with STC. We then compare the defined fields describing time with the fields content of Obscore and EPNcore.

2.1 Definition

Time Series can be defined in a very large sense as a collection of any kind of data over time for a particular source (e. g. star, binary, QSO) or part of a source (e. g. sun spots), independent on the type of data (images, light-curves, radial velocity, polarisation estates or degrees, positions, number of sunspots, densities,...), the duration of the observation or the cadence.

Independent on the type of data we can sketch Time Series data as shown in Fig. 1. Time Series data is composed of a set of observations (n_observations = 3 in this example), each with a different exposure or integration time (t_exp). Although in some cases the cadance or time span between each observation (delta_t) is fixed, in the general case it can be different and we can therefore define a minimum and a maximum value (delta_t_min, delta_t_max). Each observation has it's own time stamp (t_i) with a given precision or resolution (t_resolution). As can be seen from this figure the duration of the observation can be defined in different ways: a) as the total integration or exposure time, i. e. the sum of all the

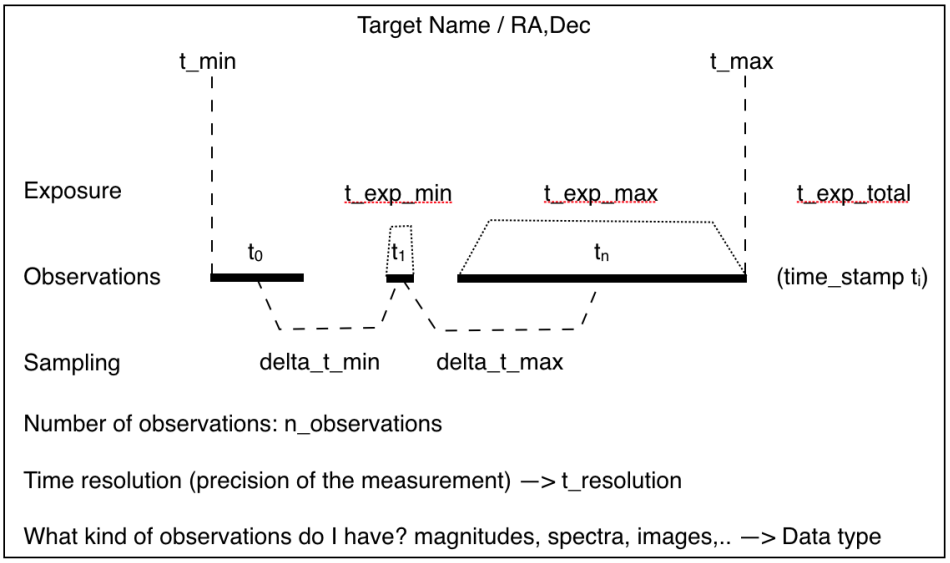

Figure 1: Simple representation of Time Series data.

exposure times: t_exp_total = $\sum$t_exp; or b) as the time spam between the beginning and the end of the observations: t_exp_total = t_max - time_min). Note that in the case that the exposure time is constant for all the observations then t_exp_total = n_observations $\times$t_exp. The situation can be more complicated, for instance during the observation there could be clouds and we therefore pause the exposure for a while and resume once the cloud has passed or we might want to remove parts of the observation due to artefacts in the data. In any case these values can be taken as approximative of the minimum and the maximum value this specific field can have. The most relevant fields of Time Series metadata are summarized in Table 1.

In many cases time series data is composed of only three colums:

time, magnitude, magnitude error

For this data to be fully exploitable and reusable (interoperable) it has to be properly documented. In this specific case the minimum information that needs to be provided is: the object coordinates (or name), the filter in which the observations have been carried out, and the time frame and offset (if applicable).

2.2 Science use cases

Different science use cases for Time Series have been collected and described in by **?** and can be found under `http://wiki.ivoa.net/twiki/bin/view/`

Table 1: Time Series metadata fields.

Field	**Explanation**
(RA,Dec)	Coordinates[1]
target_name	Target name[1]
t_min	Date of the begining of the of observation
t_max	Date of the end of the observation
t_exp_min	Minimum exposure time
t_exp_max	Maximum exposure time
t_exp_total	Total exposure time
delta_t_min	Minimum time sampling / cadence
delta_t_max	Maximum time sampling / cadence
t_resolution	Time resolution/precision
n_observations	Number of observations
type_of_data	Type of data (fluxes, radial velocities, images,...)

Note: [1]For SSO or moving objects coordinates might not be enough or relevant.

Science Case	**Target(s)**	**Datatype**	**Time**	**Brightness**	**Photometric Band**
Group A	yes	lightcurves	yes	yes	one
Group B	yes	lightcurves	yes	yes	several
Group C	yes	other	yes	no	no

IVOA/CSPTimeSeries. Science cases are grouped according to their common requirements:

- **Group A** Common requirement: Combine photometry and light curves of a given object/list of objects in the same photometric band
- **Group B** Common requirement: Combine photometry and light curves of a given object/list of objects in different photometric bands
- **Group C** Common requirement: Time series other than light curves

As highlighted by the different science use cases described in http://wiki.ivoa.net/twiki/bin/view/IVOA/CSPTimeSeries, there are astrophysical phenomenae that vary in different timescales and hence, in order to study the different physical underlying mechanims a user might need to collect and analyse data from different missions and of different nature. Answering all the possible science cases is a difficult task. We would therefore

like to keep a practical approach to the problem, solving the simplest cases in a first step and allowing having incremental solutions for more complex systems at later stages.

Looking at the different science cases we simplify the questions to two:

1. *Have these two missions observed this object within these two dates?*
2. *Is it possible to discover long/short term variability within the data?*

To answer the first question a user needs to be sure that dates are comparable, that is time has to be brought into a common time frame. To answer the second question we need to keep track of the minimum and maximum time span. We aim in a first step to answer these fundamental questions and, later on we will move to answer the specific science cases, which focus of the nature of the Time Series data, giving priority to lightcurves which represent the majority of the cases, while having in mind a wider approach.

2.3 Using a common time frame

To compare datasets from different missions or archives a common representation of time is needed. In order to do so we propose to map time into a pivot format. Following **?** and Rots (2007) we propose a set of minimum metadata to be added for serializations of Time Series (see Table 2).

We recommend to be specific on the time frame and we suggest to use:

JD(TDB;BARYCENTER)

We also give some values that can be used as default in the case that some information is not known and impossible to recover. We minimize the impact of doing this by adding a systematic error to time when those values are unkown.

2.4 Extension of ObsCore based on EPNCore

Some of the fields described for Time Series data have already been explecitely defined and used in the context of data discovery using ObsCore Louys and Bonnarel et al. (2011), and the remaining ones have been defined in the context of EPNcore **?**. In Table 3 we show the equivalence between the fields we define here and those by ObsCore and EPNcore.

For discovering Time Series data, an extension of ObsCore based on the fields listed in Table 3 and missing in ObsCore would suffice. For these extra fields we would recommend using the EPNcore name convention.

Table 2: Metadata for time in Time Series data serialisation.

Parameter	**Explanation**
time_frame_scale	Time frame scale is the scale used to measure time. IAU definition: "A time scale is simply a well defined way of measuring time based on a specific periodic natural phenomenon." http://aa.usno.navy.mil/publications/docs/Circular_179.pdf. Recognized time scale values and their meaning are listed in Table 9. If we don't know use "UNKOWN".
time_frame_position	Time Frame Position is the place where the time is measured. Standard values are liste in Table 10. If we don't know use "UNKOWN".
time_uncertainty	Resolution or uncertainty of the time stamps.
time_sys_error	Time Systematic Error to take into account our knwoledge of the time frame (scale and position). If time_scale is not known then 100s as DEFAULT value, if time_scale and time_frame_position are both not known then use 1000s as DEFAULT value. Approximately 100s is good for the time_scale since that's related to changes in the clock in space/earth; 1000s is good if we don't know if times are corrected for the position of the Earth/satellite on its orbit around the Sun since that's approximately twice the time it takes the light to travel the Sun-Earth/satellite distance.
time_representation	JD, MJD, ISO-8601.
time_offset	Offset that has been subtracted to the time. Time can be relative to a certain moment, e. g. time after the GRB that happened on date YYYYMMHHMMSS.SS or a random number the authors have subtracted from data to allow higher precision in the time stamps. Its default value is 0.0.
Description	A text briefly describing what is varying with time. "Photometric variability in filter V", "Radial velocity curve in HJD". This field is aimed to help the reader.

Table 3: Equivalence between Time Series data fields and ObsCore and EPNCore fields

Field	ObsCore field name	EPNCore field name
coordinates	s_ra, s_dec	-
target_name	target_name	target_name
t_min	t_min	time_min
t_max	t_max	time_max
t_exp_min	-	time_exp_min
t_exp_max	-	time_exp_max
t_exp_total	t_exp	-
delta_min	-	time_sampling_step_min
delta_max	-	time_sampling_step_max
n_observations	t_xel	-
type_of_data	dataproduct_type[2]	dataproduct_type

Note: [1] The explanation of t_resolution in Obscore as "Temporal resolution FWHM" should be modified to simply "Temporal resolution". [2] dataproduct_type should be set to timeseries in Obscore, but this would still not tell the user that the timeseries is composed of images or lightcurves or a combination of both in a more complicated case. **Q : is it possible to combine dataproduct_type=timeseries;images;sed for instance?**

3 Models

In this section we describe the usage and role of models in the context of this note (see (**?**)).

4 Time series representation: Data Model and UML diagram

4.1 Data modeling with CharacterizationDM and Cube DM

The goal of this modeling effort is to tackle various use-cases for characterizing time dependent datasets.

This model interprets the Cube Data Model (**?**) for describing the metadata of Time series necessary to discover and select a dataset. It also represents the data themselves, the time line and the kind of measurements taken for each time stamp and how they are grouped together by some common physical parameter.

It summarizes the properties of an observed dataset following the approach of the ObsCore Data Model and Characterization Data Model. The

time axis especially is characterized in coverage (span), resolution(uncertainty), sampling in order to easily identify a data set for studying one specific event. Spatial axis, Spectral axis, Observable (Flux), and Polarimetry axes are also characterized for data discovery.

The representation of data re-uses the SparseCube Class for representing a collection of measures organized as a multi-dimensional array with possibly empty values : a sparse cube. The axes covered by the SparseCube can be described from an axis profile listing which axes are present, which ones are combined and in which data product types and subtypes.

4.2 Perspective

The data are represented by a node with a time stamp and a bundle of observable data taken at this time slot. This bundle may contain only a simple measurement, f.i a flux or a magnitude in the case of a simple light curve, or a data varying along other parameters like wavelength or frequency, for a spectrum or like position for an image, or both like a hyperspectral data cube (2D position + lambda). More over these can also be multiband images, polarimetric spectra or images, etc.

The event list is the form of the most scattered format of measures, sampling all values separately, but the VO already offers a strategy for querying and handling dataproducts like images and spectra as well and we can build on it at least for data discovery.

4.3 TimeSerie representation use-cases

We identified various use-cases where data are taken following a series of time stamps. The data model allows to accommodate various cases, from the single light curve for one astronomical object to a series of advanced data sets depending on the Time axis. The following section describes how data are bound together for each use-case. It shows a simple block diagram and illustrates the case with real data sets. The more general form is TS = f(t, pos, em, pol, ..) represented as a node tupple and by setting one or more of these parameters to a constant or a set of discrete values, we can represent many cases .

4.3.1 Light curve

The node tupple is (t, m) with m beeing a photomeric measure which can be characterised by the Observable UCD tag o_ucd = *phot.mag* or $phot.flux$ for instance. In this case attributes from ObsDataset easily represent the type of dataset: $dataproduct\_type$ to be 'timeseries' and $dataproduct\_subtype$ to be 'lightcurve' for instance.

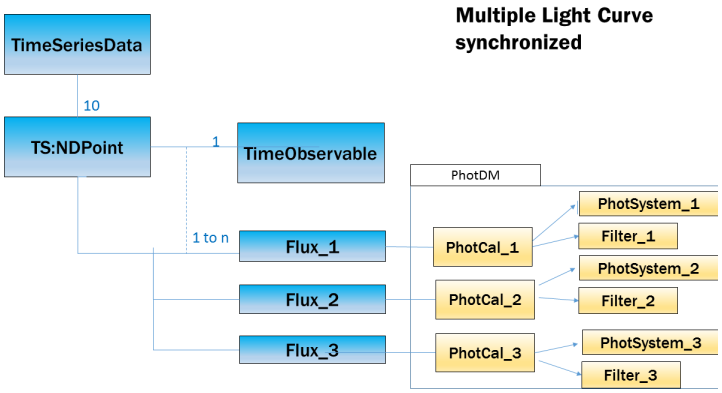

Figure 2: Light curve: simple if n equals 1 and multiband light curve if several fluxes have been recorded in various filters. Here they have been recorded simultaneously and are bound to the same time stamp. The Photometric calibration and information of the Photometric system together with the Filter details are linked to the Flux measures.

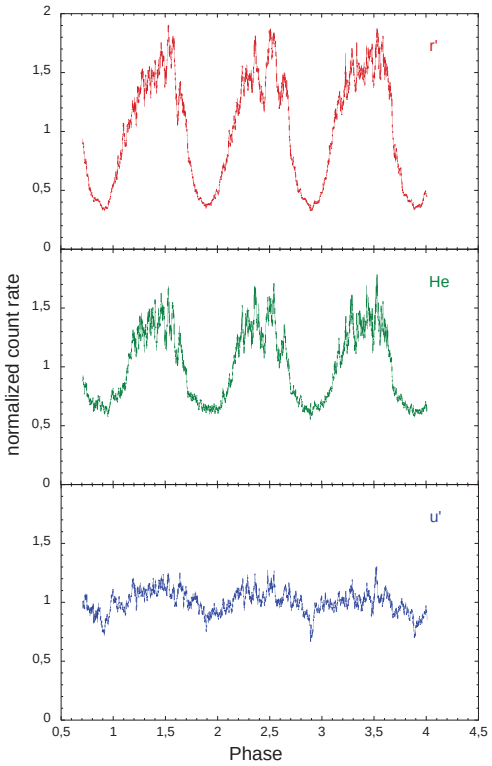

Figure 3: Time series data with multiple fluxes for one time stamp in three bands : ULTRACAM data

Here is a representation for non synchronised multiband time series in Fig 4

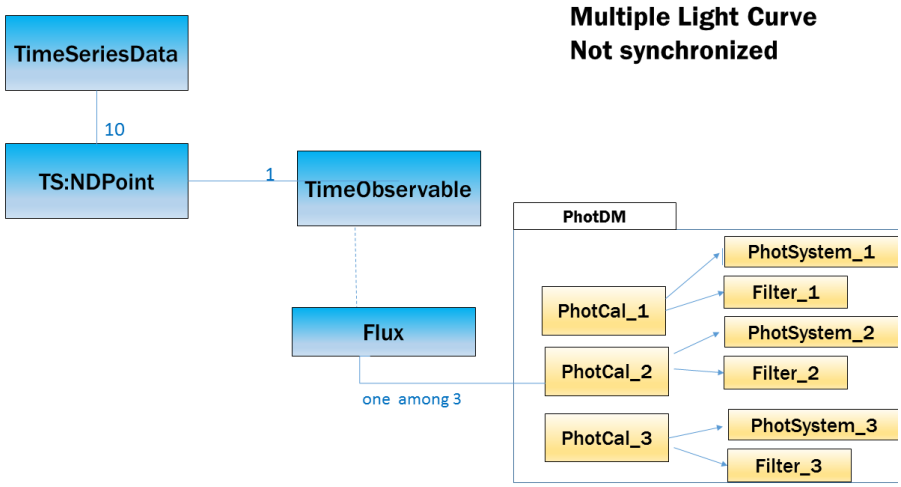

Figure 4: Time series data with multiband fluxes not synchronized : Can be seen as a compilation of three interlacing monoband time series .

4.3.2 Time Series of spectra

The node tupple here is (t, m) with m to be a spectrum $(t, flux_i = f(em_i))$. We deal with a list of time-stamped spectra. see Fig 5

4.3.3 Time series of images

This can be used in many fields for detecting changes (solar events, planetary applications, GRB events, ...) The sparse profile is $(t, flux_i = f(Pos_i))$ at some $em\_reference, em\_0, em\_1, em\_2, etc...$

Here we deal with a list of time-stamped images. The series can hold images taken with different filters taken simultaneously and have a multi-spectral coverage. Or this can be a compilation of multiple images in various bands at interlacing time intervals.

4.3.4 Time series of cubes

Sparse profile $= (t, Flux(em, pos))emin[em\_min, em\_max]$, pos in a sky bounding box This is typically the MUSE data collection which stacks a time series of hyperspectral cubes over a night and combine these to improve SNR of the reconstructed data cube.

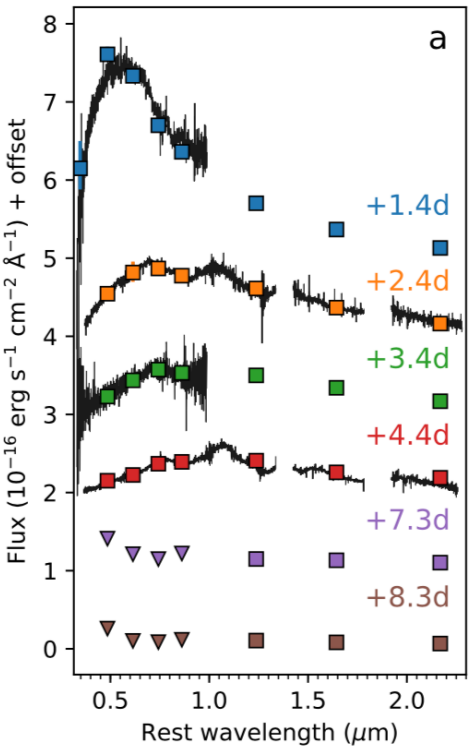

Figure 5: Time series of spectra and SED compared for the PESSTO data

4.3.5 Time series of combined data

A TS:NDPoint can be even more sophisticated and can gather several data to store interpretation results and highlight previous identification and analysis of the combined data. This is then up to the archive publisher/designer to link the science-ready data parts together. Time domain applications will then support appropriate browsing of these logical links and help astronomers for the science interpretation.

LightCurve enriched with preview images Light curve may be enriched with preview images showing the source in spatial context and in appropiate filters for some critical time stamps. The linked images can be progenitor images from which the photometry was computed, or an illustration of how the source can appear at some specific time and /or spectral band. lSee 6 and 7

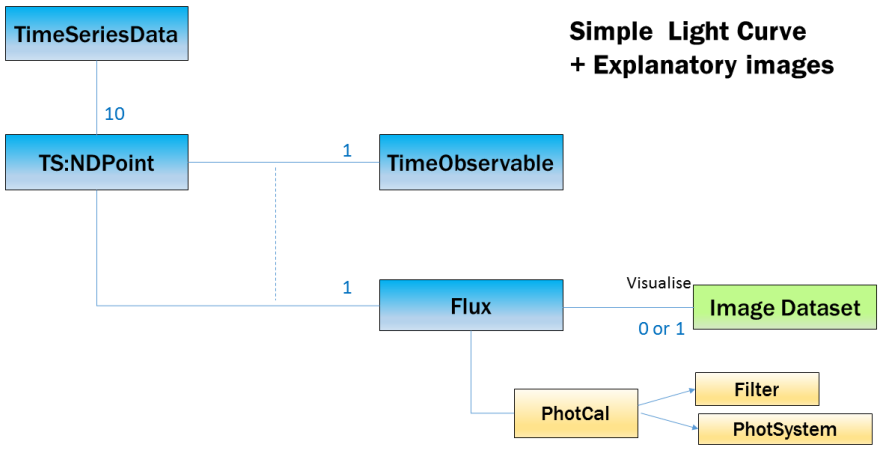

Figure 6: The TS:NDpoint now has a time stamp and a flux attached . Images of the source are linked to the various time stamps corresponding to luminosity peaks and illustrate the source in its temporal and spatial context.

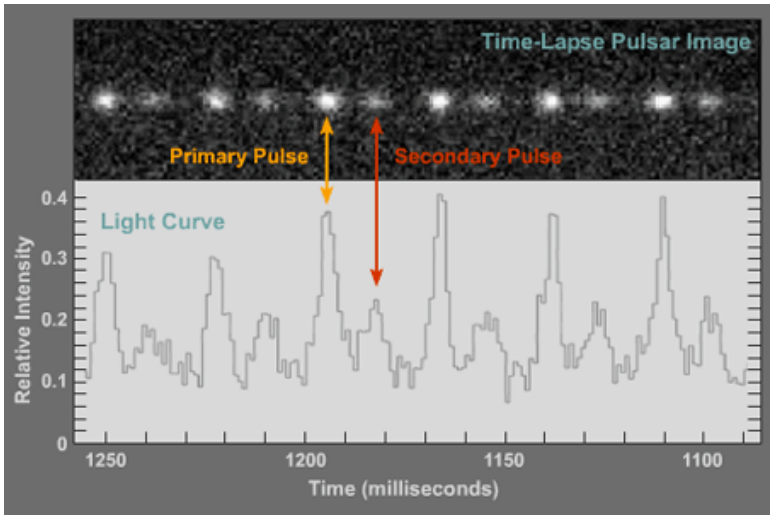

Figure 7: Time series data with flux measure and illustrating images at relevant data points. Application to Pulsar data analysis

5 TS Data Model

The overview of the data model is given in the following class diagram in fig. 8. The details of the Characterisation along the Time axis , which will be selection criteria used for timeseries discovery are modeled in the Characterisation part and represented in Fig. 9.

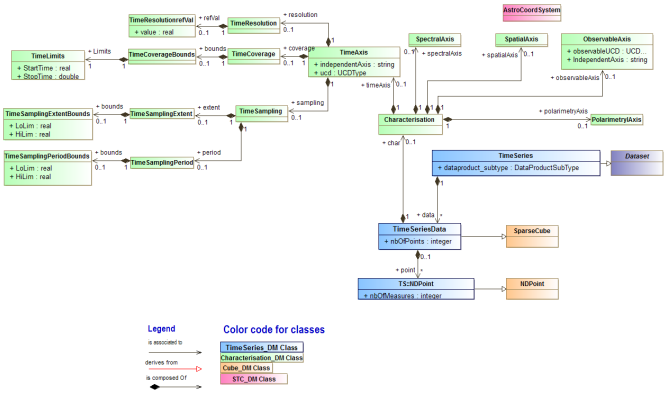

Figure 8: Time series main components: Timeseries data are represented by the TimeSeriesData object which is a collection of NDpoints as a SparseCube. How the data are spanned along the various axes is summarized in the main properties of the Characterization class. The TimeAxis class especially is described for Coverage , resolution and sampling.

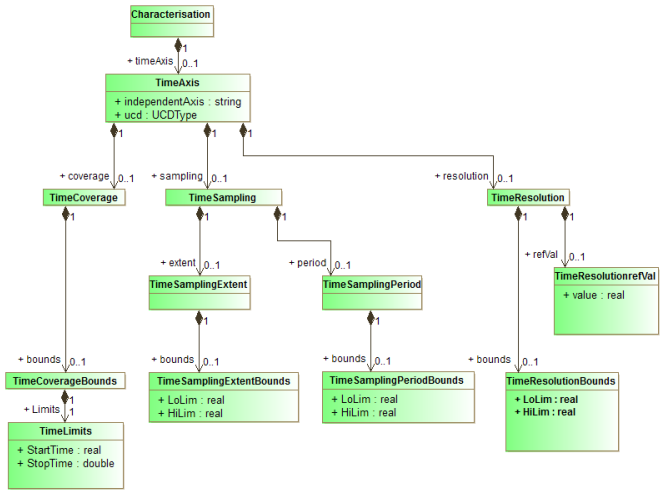

Figure 9: Time Axis main properties. Coverage, sampling, time sampling step as well as time period between two samples are described. The time resolution can be present either as a typical resolution value in TimeResolution Refval or by a set of minimal and maximal values representing the time resolution (or also considered as accuracy along the time axis)

How the data set can be localised in spatial, spectral, polarisation axis, flux axis or other observable is also described via the Characterization DM using spatial, spectral, polarimetry, etc. coverage description. The flux axis if present is bound to a PhotCal object present in the PhotDM . This allows to bind a flux measure to its spectral coverage, filter and photometric system

easily , as proposed in the Spectral and Photometry DM. This is needed in the use case of a multiband lightCurve for instance.

The TimeSeries data only needs two concepts : a time stamp representation and the corresponding measure taken at this time instant. These are two Observable objects , and derived as TimeObservable for the timestamp and a specialized Coordinate measure as defined in the meas package of the STC v2.0 data model.

The following diagram in fig. **??** highlights how the measurements classes from STC can be reused, but leaves the details to further discussions until the STCV2.0 DM is finalized completely.

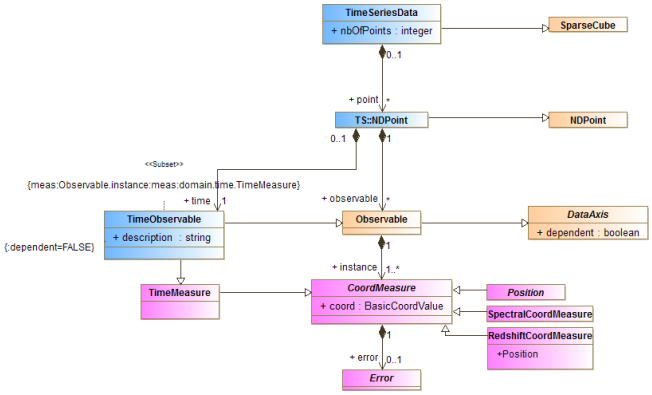

Figure 10: Time Axis main properties. The Coverage , the sampling , time sampling step as well as time period between two samples are described. The time resolution can be present either as a typical resolution value in TimeResolutionRefval or by a set of minimal and maximal values representting the time resolution (or also considered as accuracy along the time axis)

More than one single measure corresponding to a time stamp, we can also gather time series of dataproducts like spectra, images or cubes. This is the use case described in class diagram of Fig.11.

This datamodel has been designed with Modelio 3.6. It includes the Modelio exportfiles from Cube and STCv2.0 DM in their current version. The complete integration phase of those datamodel parts is still in progress with imports beeing unwieldy . Once solved the VODML description of the TimeseriesDM should be easy to generate and the HTML docs as well.

5.1 datamodel fields

```
t\_min TimeAxis.Coverage.Bounds.Limits.LoLim
t\_max TimeAxis.Coverage.Bounds.Limits.LoLim
t\_exp\_min TimeAxis.SamplingExtent.Bounds.Limits.LoLim
t\_exp\_max TimeAxis.SamplingExtent.Bounds.Limits.HiLim
t\_delta\_min TimeAxis.SamplingPeriod.Bounds.Limits.LoLim
```

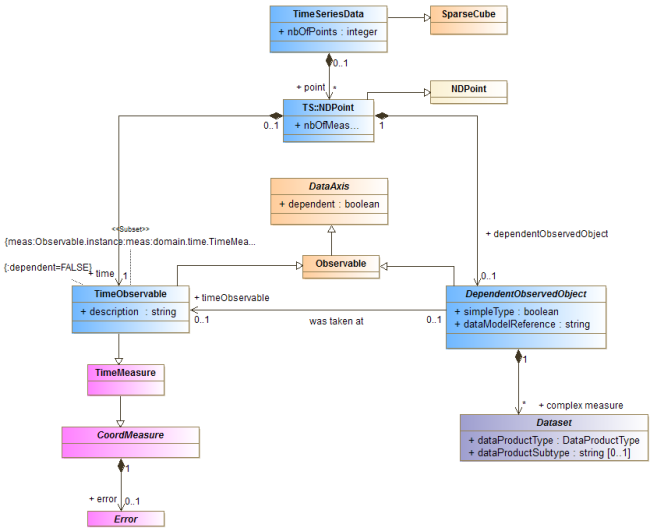

Figure 11: When the measures are datasets by themselves.

```
t\_delta\_max TimeAxis.SamplingPeriod.Bounds.Limits.HiLim
t\_sampling type of sampling TimeAxis.   irregular, regular .. if regular period if computed
```

5.2 Mark

For the purposes of this exercise, we generated a toy timeseries model based off current models (Dataset, Cube, Coords, Meas). The model extends Cube model elements to facilitate the modeling of a wide variety of Time-Series flavors.

There are multiple ways the TimeSeries model could be derived from Cube. The details will depend on a thorough review of the full set of use cases and requirements. This toy model focuses on representing the Simple-TimeSeries as a fairly direct extension of SparseCube... with the addition of requiring a 'timeAxis'. The cube component elements are designed to facilitate the definition of other TimeSeries described in the CSPTimeseries document, but are not heavily considered here at this point.

The Simple Time Series model shown in Fig. 12, is defined as an extension of the Cube data model with the addition of only 3 objects. This approach maximizes the interoperability of the data and reinforces the concept that these data can be considered a slice through a generalized sparse cube. By creating a VO-DML compliant model, instances of a Simple Time Series can be annotated using their assigned VODML-IDs. Applications which understand Cube model instances will **automatically** understand a very large percentage of the Time Series content.

Having separate models for different areas of the 'Universe of Discourse' maximizes interoperability, but can take some practice to interpret. Below

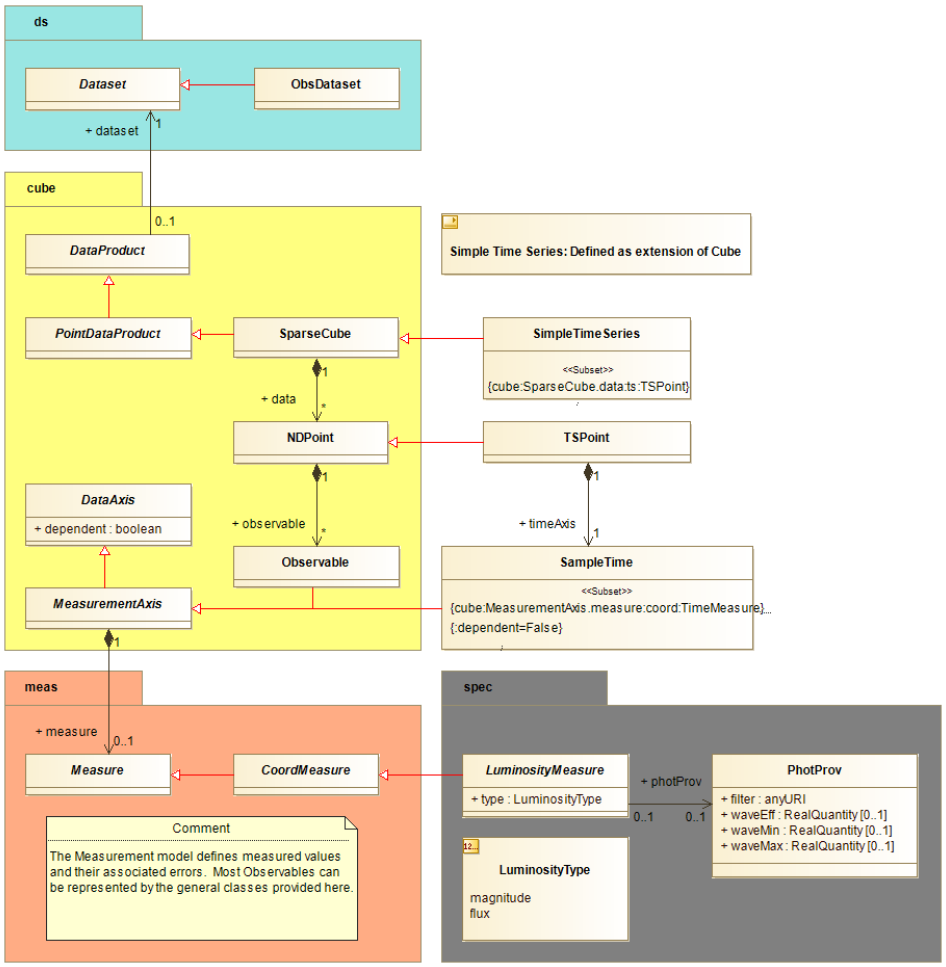

Figure 12: Simple Time Series as extension of Cube.

is a descriptive outline of the model structure.

ds:Dataset Metadata

- Dataset - provides generic identification and curation metadata for the dataset (eg: Curation, DataID, etc).
- ObsDataset - adds metadata related to the Observation (eg: Target, Instrument, Configuration, etc)

cube:N-Dimensional Cube - This model defines generic cube data products, either as SparseCube-s (eg: Event lists) or Image-s (eg: nD Pixelated image). The Simple Time Series model will extend from SparseCube, inheriting the following:

- A reference to the relevant Dataset Metadata instance which describes it.
- A collection of NDPoint-s, each being a collection of associated Observables. (eg: time, position, mag)
- Observable - identifies the item as 'dependent' or 'independent', and provides the measured data value (value + error) which is described in the Measurement model.

meas:Measurement - This model defines measured data and their associated errors. It is a generic model, so provides the basis for many types of measured data. In some cases, additional metadata is required to fully describe the measurement and how it was obtained. These are expected to be defined in models which specialize in that content.

- CoordMeasure - class of Measure given by a Coordinate 'value' plus one or more Error-s also defined in that model.

spec:Spectral Model - This is a dummy Spectral model for illustrating how a specialized LuminosityMeasure could be defined which provides additional metadata related to how the Luminosity value was determined.. eg: which photometry filter was used.

- LuminosityMeasure - Extends CoordMeasure, so has a Coordinate value + Errors, and adds a reference to associated Photometry metadata.
- NOTE: being an extension of CoordMeasure, this type is automatically usable in a Cube instance, and therefore, SimpleTimeSeries.

ts:Simple Time Series - The Simple Time Series is modeled as a 'view' through a sparse cube. Time becomes the independent axis, with other observable axes providing the measured data.

- SimpleTimeSeries - defined as a SparseCube which MUST contain TSPoint-s.
- TSPoint - adds a single 'independent' timeAxis to the generic NDPoint.
- SampleTime - provides the TimeMeasure for the point. TimeMeasure is defined in the Measurment model, giving the Coordinate value in JD, MJD, ISO, etc.. plus any associated errors. The Coordinate is associated with a TimeFrame identifying the timescale and reference position. **NOTE:** This element includes basically all the information described in Table 2.

With a validated VO-DML compliant model, we can generate serializations which are annotated using the element VODML-ids. In Section 6.3, we describe the serialized sample files generated for this study based on this model, using the current VO-DML/Mapping syntax proposal.

6 Serialisations

This section presents various attempts of serialisation using different strategies.

Subsection 1 is using VOTable GROUPs for separating axes in the serialisation and is using utypes from different models on GROUPS and FIELDS. It has been written by Jiri Nadvornik. Subsection 2 "Full utype serialisation" subsection has been written by F.Bonnarel. Subsection 3 is using the full VO-DML mapping serialisation and has been written by Mark Cresitello. Subsection 3 is a proposal for a "Lighter" VO-DML mapping serialisation and has been written by Laurent Michel.

6.1 Jiri's approach

6.2 Full utype serialisation

In this approach, relational views of the TimeSeries described above are provided. Hierarchies of classes showing 1 to 1 relationships are grouped in one single table. Table x shows the list of tables used to map the model with the full set of their columns metadata (utype, datatype, ucd, unit and xtype), in the case of example 1. The VODataservice "tableset" representation of this set of tables is described in Appendix.

Each TimeSeries project as its own subset of attributes of the generic model. This leads to a specific partial relational view of the model. For each of these a dedicated table like table x could be provided. See the tableset representations in Appendix.

The structure is made of 4 "tables" number which seems to be a compromise to separate very differents groups of classes without having too much of them.

The main "table" is the data table where the actual values of the independant variable (Time stamps) and of the "one to many" varying variables are stored.

The three other tables, which we can call "generic TimeSeries metadata" tables gathers respectively :

- Dataset, curation and dataid classes attributes. This is directly derived from similar concepts in ObsCore and Dataset metadata model.

- Characterization describes where and how the dataset is situated in the physical parameter space; most of the columns there should be redondant with what is in Obscore Char.

- Coordsys table. This table mainly serializes stc and gathers the coordinate frames (time, space, spectral) used in the TimeSeries as well as the different photometric filter descriptions, assimilated to "flux frames".

Utypes are built by concatenating class and attributes names following a path from a top element to the actual considered leave. They don't designate a class atrribute per se but a role of the attribute value with respet to the full model.

Table x and its vodaitaservice tableset expressions are an absolute reference for the utypes definition and should be reproduced in the timeSeries model specification.

"Tables" mentionned here and described in the Appendix are not required to be serialized as TABLE elements in VOTable. In the case there is only one single TimeSeries in the VOTable document, "generic metadata tables" have only one single instance. They can be serialized as groups and their columns as PARAMS instead of using TABLES with one single row.

The Appendix illustrates the background of the examples the serialization of which is provided at the URL included here, starting with the simplest example :
http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/
/francois/SDSS_J080434.20+510349.2_VizieR_complete_utypes.xml
This TimeSeries is a light curve for a single SDSS object and is provided as full Catalog in VizieR. It has one single dependant variable : V magnitude.

Obviously most of the use cases will be more complex, as in the two following examples

Example 2 is an excerpt of table 3 of the VizieR catalog provided by Shenavrin and coworkers in 2011 (Shenavrin et al, Astronomicheskii Zhurnal, 2011, Vol. 88, No. 1, pp. 34-85.) where the rows present a basic time with a Time Shift for each photometric band. It presents a main time as independant variable and several magnitudes in different bands for the same individual exposure. The exact measurement time is slightly varying according to the band (due to a shift of time for changing the filter) in such a way that several secondary times are given for each main TimeStamp as well.
(http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/
francois/BetaLyr_Vizier_complete_utypes.xml)

In example 3 where exoplanet data are provided (GAPS project) some columns provide flux or radial velocity measurements which have a specific utype in an IVOA model (stc or phot). Some other don't have because the definition of the quantity is outside the scope of STC or

photometric datamodel. In that case a generic "measurement" utype (ts:TimeSeriesData.NDPoint.dependantObservedObject.CoordMeasure.coord) is given and the flavor can be precised with help of the ucd. In some other cases the column contains an error on some of these measurements. In that case the utype is set to : "

(http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/francois/GAPS_kp7_complete_utypes.xml)

In example 4, extracted from Gaia DR2 catalog, the TimeSeries is a concatenation of three set of NDpoints, one per each photomteric band

(http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/francois/GaiaSample3Tables_complete_utypes.vot)

In example 5, the Gaia DR2 TimeSeries with its three bands is organized in a single table.

(http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/francois/GaiaDR2Example-time-series.xml)

6.3 Marks's approach

For this study, we generated annotation for the 2 sample files according to the current VO-DML Mapping Syntax proposal **?**. Annotation was generated to both the current Cube model and the toy Time Series model described in Section 5.2. The files themselves may be found at the following URL: http://volute.g-vo.org/svn/trunk/projects/time-domain/time-series/standardized_votables/MarkCD/

6.3.1 Sample Files

In the text below we will focus on the annotation to the Time Series model as that is the focus of this study.

1. **BetaLyr_Vizier** Example file annotated as multiple SimpleTimeSeries data product instances, one per photometric band.

   The structure is:

   - 1-instance of ObsDataset
   - 5-instances of SimpleTimeSeries data products, one for each band (J,K,L,M,N)
   - 5-TSPoint templates, each with 1-Time + 1-Observable (Mag with Error)

The use of FIELD elements to store metadata (eg: ObsDataset metadata) presents a challenge for proper annotation. Being in FIELDs, they must be annotated within a TEMPLATE element, which may have multiple instances. This forces the INSTANCES to identify a particular instance via the ORM PRIMARYKEY/FOREIGNKEY elements.

For a model which specifies that there should only be 1 instance of that type, this is quite a lot of overhead. This annotation shows that the syntax can accommodate the structure, but in practical application, it is probably easier for both client and provider to re-cast the VOTable segment to provide these singular elements as PARAMs. This results in very straight-forward annotation and VOTable.

2. **GaiaSample3Tables** Example file annotated to the toy TimeSeries model, using the current Mapping syntax proposal, as multiple SimpleTimeSeries data products.

   - 0-instances of ObsDataset
   - 3-instances of SimpleTimeSeries data product
   - 3-instances of TSPoint with 3 Observables each (1-Time + 1-Flux_w_error + 1-Mag)

   This appears to be a straight forward representation of 3 SimpleTimeSeries data products, each serialized in their own TABLE. In this example, the filter metadata is constant for each table, so we've annotated them as individual instances which are referred to by the LuminosityMeasures within the TEMPLATE.

   There is no DatasetMetadata included.

   For annotation purposes, I needed to make the following changes.

   - Changed VOTABLE tag to version 1.4 to validate against vo-dml schema
   - Added IDs to TABLE elements to reference in vo-dml annotation
   - Added IDs to PARAM elements for Filter metadata

6.3.2 Mapping Syntax

The syntax used is fully documented in the VO-DML Mapping document **?**. It is important to note that this syntax was designed to satisfy a broad set of use cases and essential requirements identified by the community. Any syntax being considered by the community for application, should be evaluated against these requirements to ensure that it will meet the needs of the community.

The document fully describes the syntax, so we will not go into detail here. The key structural components are:

- <VODML> - primary element, all annotation is isolated from the VOTable serialization.
- <MODEL> - identifies which vo-dml compliant models are represented in the serialization
- <GLOBALS> - instances which are global in nature. (eg: Frames, Dataset Metadata)
- <TEMPLATES> - defines multiple instances by iterating over entries in the template target (rows of a TABLE)
- <INSTANCE> - defines a particular instance of a structured object (TimeFrame, SimpleTimeSeries, etc) identified by the vodml-id of the object type (coords:domain.time.TimeFrame, ts:SimpleTimeSeries)

This syntax provides severval benefits over the earlier utype convention, or any utype based annotation syntax, such as:

- Isolating the annotation from the VOTable elements allows data providers to retain their native serialization.
  - is an additive change to current serialization thread
  - allows native code to continue functioning
- Protects against model changes
  - if a model changes, say elements move from one group to another (which has happened), data providers do not need to change the VOTable GROUP content, just update the annotation section.
- Allows reuse of elements
  - a single PARAM value=TOPOCENTER can be used for all Frame.refPosition roles.
  - a single FIELD name='time', can provide the time measure for multiple SimpleTimeSeries instances.
- Clear association of an instance to a role
  - The syntax directly associates an instance to the role it plays in the parent object; ds:Target.position is instance of meas:Position
  - The same INSTANCE may serve multiple roles. eg: a single Organization instance may serve as both ds:DataID.curator and ds:Curation.publisher

- Allows annotation to multiple models, or versions thereof.
  - Providers can annotate the same VOTable content as a SparseCube AND SimpleTimeSeries (for example).
  - Providers can annotate the same VOTable content to multiple versions of the same model, allowing for smoother transitions by clients.
- Allows clients to 'discover' content it understands.
  - since any modeled instance ALWAYS has the same vodml-id, client applications can find and work with content that it understands. The common example is a generic plotting package can find instances containing the roles 'meas:CoordMeasure.coord' and 'meas:CoordMeasure.error' to plot these data without knowing that they reside in a SparseCube, TimeSeries, or other data product.

These limitations of the utype annotation approach were the primary drivers for the VO-DML Mapping project.

6.4 Laurent's approach

The VO-DML workflow is 2 folds :

1) The model serialization in a *model.vo-dml.xml* file (REC process currently close to complete)

2) the data mapping. This last step consists in an XML bloc on the top of the VOTAble acting as a bridge between the model and the data so that a client can easily build model instances by exploring that mapping bloc (see previous section). The VO-DML concepts are endorsed by the community but the mapping syntax may appear as too complex making both data annotation process and VOTable parsing difficult.

- VO-DML workflow ++
  - Enable to map any modelling feature
  - Mapping block independent from the VOTable content
    - No dependency with the VOTable schema
    - Easiness to join data taken out from different tables
    - Easy to skip for clients not model aware
- VO-DML workflow –
  - Obscure and chatty mapping syntax

– Coupling between the mapping leaves and the VOTable data structure. For instance, the mapping has to be changed when a value is moved from a PARAM to a FIELD

The present section explores a possible simplification of the mapping syntax relying on the strengths of the VO-DML workflow while targeting an easier job for both data providers and data consumers. The syntax baseline is designed to provide general-purpose clients (e.g. Aladin, JS widget ...) with nothing more than what they need, considering that more advanced clients could retrieve in any case any model features within the model itself. This simplification might also improve the reliability of the data annotation process. This is not a complete proposal, but a proof of concept validated against TD data.

6.4.1 Mapping Syntax

Basics The design of this mapping syntax being not complete; we just describe here the outlines of this simplification and the features we need to map the present data sample.

As describe in the previous section, the <VODML> block is split in 3 sections.

Table 4: Main mapping elements.

<MODELS>	References to all referenced models (models + imports)
<GLOBALS>	Not implemented here
<TEMPLATES>	One template per table

Each template refers to a particular table and contains the mapping of the classes hosted by that table. The mapping is based on the 3 concepts necessary to describe a hierarchy of classes:

Table 5: Mapping element roles.

<VALUE>	Model leaf. Either points to a <PARAM> or a <FIELD> or contains its own value for literals
<INSTANCE>	Denotes a class. An <INSTANCE> is a tuple of elements which can be either <VALUE>, <INSTANCE> or <COLLECTION>
<COLLECTION>	Denotes a list or an array. A collection can contain either <VALUE>, <INSTANCE> or <COLLECTION>

Each one of these elements supports some attributes wich can be mandatory (mand.) or optional (opt). It has be be noted that some attributes such as units or formats are not described here as well as the foreign key mechanism.

Table 6: Mapping element attributes.

Attribute:	**@dmrole**	**@dmtype**	**@ref**	**@tableref**	**@size**	**@value**
<VALUE>	mand.	none	mand. if no @value	none	none	mand. if no @ref
<INSTANCE>	mand.	opt.	none	opt.	none	none
<COLLECTION>	none	none	none	none	opt	none

<VALUE> element The @dmrole attribute is the one set in the VO-DML model. Other VO-DML attributes (e.g. @dmtype) can be retrieved in the VO-DML file from the location of the <VALUE> element and from its @dmrole.

- If a <VALUE> element has a @ref attribute but no @value, its value is taken out from the refered element of the table refered by the current <TEMPLATES>. The client must first search in the <FIELD>, and then in the <PARAM>. The value of a <VALUE> element located out of a <COLLECTION> block and pointing onto a <FIELD> is set with the values read the first table row.
- If a <VALUE> element has a @value attribute but no @ref, the value of @value is taken.
- If a <VALUE> element has both @value and @ref, the client must first resolve @ref and then @value.

<INSTANCE> element The @dmrole attribute is the one set in the VO-DML model.

- An <INSTANCE> without @dmtype attribute matches the type set in the VO-DML file. Otherwise, @dmtype can specify a subtype of the dmtype read in the model. This feature is used to support abstract classes.
- An <INSTANCE> with a @tableref is mapped in the <TEMPLATES> designated by @tableref and having the same @dmrole.

<COLLECTION> element The @dmrole attribute is the one set in the VO-DML model. All <COLLECTION> sub-elements must have the same @dmtype. The number of elements in a collection is specified the @size attribute.

- A <COLLECTION> without @size attribute contains one instance of each element mapped (see example below)

```
<collection dmrole='Axes'>
    <instance dmrole='Axis'>
        <value dmrole='name' valu='time' />
        .....
    </instance>
    <instance dmrole='Axis'>
        <value dmrole='name' value='observable' />
        .....
    </instance>
</collection>
```

- A <COLLECTION> with a @size attribute contains as many instances of the mapped elements as requested by @size. To be consistant, when @size is set, <COLLECTION> must have one unique child containing only one mapped entity.

```
<collection dmrole='Points' size='-1'>
    <instance dmrole='NDPoint'>
        .....
    </instance>
</collection>
```

  If @size equals to * or -1, the collection is populated as long as data are available. For instance, this is the case when the <COLLECTION> maps photometric points which are read row per row in the table.

6.4.2 Workflow (to be updated)

The model used for these serializations, designed with Modelio 3.5, has been presented at (ref ivoa). This serialization has also been tested with SparseCube but the outcomes are less advanced. This demonstrator used for the serialization of the data sample relies on 2 specific Python tools developed on this occasion but possibly reusable as seeds for further developments.

- *transform.py* reads a VO-DML model and genererates a single <VODML> block with unresolved references. The tool supports directives specifying which classes must be used in replacement of the abstract ones. The <VODML> block must be inserted by hand in the VOtable to be annotated. It must be split in multiple <TEMPLATES> if data are spread out on muliple tables. The references to actual <FIELD> and <PARAM> must also be set by hand.

- *transform.py* is a basic client which plots the annotated timeseries. The key point is that *transform.py* is able, to some extent, to read any VOTable mapped with the model.

6.4.3 Conclusions and Prospects

This approach confirms the power and versatility of the VO-DML workflow since it doesn't use any feature external to the VO-DML ecosystem. The main outcome is that clients knowing a model are able by construction to read VOTables annotated with that model. There is no need to adapt the client code to specific data collection as long as they are are not too much

exotic. The added value of the light syntax sketched here is that the same outcomes can be reached with more compact and more readable annotations. This opens the way for light clients such as JavaScript widgets which could have difficulties to process hundreds of XML lines each time they have to parse a VOTable.

7 View from Data Providers

The purpose of this section is two folded:

- Pros & cons of each solution
- Difficulty of implementation

7.1 GAVO

GAVO requirements at a glance

7.2 vizier

VizieR requirements at a glance by Sebastien, Gilles, Thomas

8 View from Applications

The purpose of this section is two folded:

- Pros & cons of each solution
- Difficulty of implementation

8.1 Aladin

Aladin requirements at a glance by PF and TB

8.2 Topcat

Topcat requirements at a glance by MT

8.3 Javascript widget

Vizier Javascript widget requirements at a glance by GL

Table 7: Model dependances

Model	Function	ML	MCD	Rec	VO-DMLized
TS Characterization	Physical description (mag, flux, position, ...)	✓	✗	✓	✗
Cube data model	N-Dim observable	✓	✓	✗	✓
PhotoDM	Phot. system	✓	✗	✓	✗
SparseCube	Event lists	✓	✓	✗	✓
Meas	Def. of measurement	✓	✓	✗	✓
spec:Spectral Model	Phot. system	✗	✓	✗	✗
ts:Simple Time Series	Event lists with time axis	✗	✓	✗	✓
Dataset Metadata	Non-physical description (telescope, pubDID,...)	✓	✓	✗	✓

Table 8: Pros and Cons of each proposed model

	models		**serializations**		
Type of data	**ML**	**MCD**	**u-type**	**VODML**	**VODML-light**
RV/position/Light-curves	✓	✓	✓	✓	✓
Radial-velocity curves	✓	✓	✓	✓	✓
Series of dataproduct	✓	✗	✓	✓	✓

9 Discussion

Table 8 highlighs and summarizes pros and cons of each of the proposed serialisations.

10 Conclusions

Conclusions by the TDIG

A Recognized time scales

References

Louys, M., Bonnarel, F., Schade, D., Dowler, P., Micol, A., Durand, D., Tody, D., Michel, L., Salgado, J., Chilingarian, I., Rino, B., de Dios Santander, J. and Skoda, P. (2011), 'Observation data model core components

Table 9: Recognized time scale values from **?** and Rots (2007).

Parameter	**Explanation**
TAI	(International Atomic Time) atomic time standard maintained on the rotating geoid
TT	(Terrestrial Time; IAU standard) defined on the rotating geoid, usually derived as TAI + 32.184 s
TDT	(Terrestrial Dynamical Time) synonym for TT (deprecated)
ET	(Ephemeris Time) continuous with TT; should not be used for data taken after 1984-01-01
IAT	synonym for TAI (deprecated)
UT1	(Universal Time) Earth rotation time
UTC	(Universal Time, Coordinated; default) runs synchronously with TAI, except for the occasional insertion of leap seconds intended to keep UTC within 0.9 s of UT1; as of 2012-07-01 UTC = TAI - 35 s
GMT	(Greenwich Mean Time) continuous with UTC; its use is deprecated for dates after 1972-01-01
UT()	(Universal Time, with qualifier) for high-precision use of radio signal distributions between 1955 and 1972; see Sect. A.9
GPS	(Global Positioning System) runs (approximately) synchronously with TAI; GPS ≈ TAI - 19 s
TCG	(Geocentric Coordinate Time) TT reduced to the geocenter, corrected for the relativistic effects of the Earth's rotation and gravitational potential; TCG runs faster than TT at a constant rate.
TCB	(Barycentric Coordinate Time) derived from TCG by a 4-dimensional transformation, taking into account the relativistic effects of the gravitational potential at the barycenter (relative to that on the rotating geoid) as well as velocity time dilation variations due to the eccentricity of the Earth's orbit, thus ensuring consistency with fundamental physical constants; Irwin & Fukushima (1999) provide a time ephemeris.
TDB	(Barycentric Dynamical Time) runs slower than TCB at a constant rate so as to remain approximately in step with TT; runs therefore quasi-synchronously with TT, except for the relativistic effects introduced by variations in the Earth's velocity relative to the barycenter; when referring to celestial observations, a pathlength correction to the barycenter may be needed which requires the Time Reference Direction used in calculating the pathlength correction.
LOCAL	for simulation data and for free-running clocks.

Table 10: Recognized time reference positions.

Parameter	**Explanation**
TOPOCENTER	Topocenter: the location from where the observation was made (default)
GEOCENTER	Geocenter
BARYCENTER	Barycenter of the Solar System
RELOCATABLE	Relocatable: to be used for simulation data only
CUSTOM	A position specified by coordinates that is not the observatory location
HELIOCENTER	Heliocenter
GALACTIC	Galactic center
EMBARYCENTER	Earth-Moon barycenter
MERCURY	Center of Mercury
VENUS	Center of Venus
MARS	Center of Mars
JUPITER	Barycenter of the Jupiter system
SATURN	Barycenter of the Saturn system
URANUS	Barycenter of the Uranus system
NEPTUNE	Barycenter of the Neptune system

and its implementation in the Table Access Protocol, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/ObsCore/20111028/REC-ObsCore-v1.0-20111028.pdf

Rots, A. (2007), 'Space-time coordinate metadata for the virtual observatory', IVOA Recommendation.
http://www.ivoa.net/documents/latest/STC.html