International
Virtual
Observatory
Alliance

IVOA N-Dimensional Cube Model

Version 1.0

IVOA Working Draft 20190919

This version:

WD-CubeDM-1.0-20190919

Previous version(s):

Editor(s):

Mark Cresitello-Dittmar

Authors:

Mark Cresitello-Dittmar, Doug Tody, Francois Bonnarel, Omar Laurino, Gerard Lemson, Markus Demleitner, and the IVOA Data Model Working Group.

Abstract

Status of This Document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

Acknowledgements

This document has been developed with support from NSF and NASA under the Virtual Astronomical Observatory (VAO) project, the National Science Foundation's http://www.nsf.gov Information Technology Research Program under Cooperative Agreement AST0122449 with The Johns Hopkins University, from the UK Particle Physics and Astronomy Research Council (PPARC) http://www.pparc.ac.uk, and from the European Commission's Sixth Framework Program http://fp6.cordis.lu/fp6/home.cfm via the Optical Infrared Coordination Network (OPTICON), http://www.astro-opticon.org.

Change Log:

2013 May 05: Initial release of ImageDM draft document.
2013 Aug 16: Rev 16 updates.
2013 Nov 15: Second draft of ImageDM release.
Updated to reflect initial reviewer comments. Architecture section extensively revised to provide compatibility with ObsCore/Observation and Char2
2014 Sep 10: Model restructured as extension of Observation Dataset
Extensive revision, basing Cube model as extension of dependent models: Dataset, and STC
Define new components defining generic Cube datasets (image and sparse) which can be utilized for Spectral and other models
Baseline to STC-2.0 prototype model.
2015 Mar 20: Updated STC-2.0 prototype model spec.
2016 Nov 09: Model revisions from implementation
- Generalized to NOT be directly dependent on STC, which should be considered to implement elements in this model (eg: CoordSys)
- relation changes due to type change on coordinate elements.
2017 Feb 03: Model revisions from implementation
- Again modifying STC relation, showing as aggregated relations which will allow the appropriate relations in this model, while not confining STC usage.
- Updated representation of STC sub-models (coordsys, coords, trans)
- A bit of restructuring to pull Dataset-DataProduct relation to forefront, should improve readability.
2017 Oct 10: Updates from TimeSeries prototyping experience.
- loosens relation between Dataset and DataProduct.
- generalized NDPoint and Voxel as sub-classes of a DataElement type organizing data axes.
2018 Mar 21:
- restored relation between Dataset and DataProduct, but as a simple reference.
- added more extensibility to support various extensions of products like TimeSeries
+ removed composition of DataAxis in DataElement
+ subclasses of DataElement compose subclasses of DataAxis. This enables reuse of various kinds of DataAxis (eg: MeasuremenAxis) in constructing different kinds of products.
2018 May 16:
- add support for Virtual data; VirtualMeasure, VirtualImageAxis
- removed extranious abstract classes; DataElement, PointDataProduct, PixelatedDataProduct
2019 Sep 19:
- update model to PR versions of Meas/Coords models (url-s etc); and Transform WD.
- added Mappings object, no longer in Transform model.

Contents

1 Introduction

1.1 Motivation

The concept of the *astronomical image* goes back decades, most notably to the introduction of the flexible image transport system (FITS) in the late 1970s by Wells and Greisen. Later papers by Greisen, Calabretta, and others added the capability to define a *world coordinate system* (WCS) that can be associated with an image to define the physical coordinates in an M-dimensional space of each data sample (observable) in the N-dimensional array comprising the data segment of the image. Other metadata elements (FITS keywords) are defined to describe the origins and content of the particular image dataset. The astronomical multi-dimensional image concept is a type of data hypercube, and is related to the volumetric cubes used in medical and geological applications, and to commercial technologies such as the OLAP cube, which projects a RDBMS relation onto the axes of a hypercube.

An important aspect of the astronomical image is *abstraction*. The image model hides how the data are physically stored; this is especially important for large images or image cubes, which may be Gigabytes or Terabytes in size for a single dataset. While logically the data portion of the image may be a simple N-d array, physically the data may be represented or stored in many different ways. Large cubes may be physically stored in multiple smaller segments, or data may be stored in N-d blocks to provide uniform access along any dimension or axis of the image. Sparse cubes may be stored as multiple segments, each at a given location within the larger logical cube. Data may be stored in a compressed form, or may be encoded, e.g., via a multiresolution technique such as a wavelet transform (JPEG2000). Each such representation offers certain advantages and disadvantages; by separating the logical view of the data from the details of how it is physically represented, the optimum choice may be made for each application, transparently to higher-level analysis software.

This model presents an abstracted representation of N-Dimensional cube datasets and serves as a framework on which to construct models for more specialized Astronomical datasets.

1.2 Use Cases

A use case study of multidimensional/cube data performed by the US VAO and community partners in early 2013 contributed to early development of this data model. A special session on multidimensional data held by the IVOA in May 2013 helped to further refine community use cases for the data model.

A comprehensive analysis of use cases for the Image data model in the context of multidimensional astronomical data is beyond the scope of this document, but is available in [TBD: *reference to VAO use case analysis*].

Here we summarize major use cases from this study:

- Simple 2-D image, the most common astronomical image type.
- A single cube image is often sparse in either the spectral or spatial or temporal plane. The JCMT cubes are a good example of a spectral data cube that is sparse in the spatial plane.
- Multiple-subarray images are common for instrumental observations composed of multiple detectors. For example, a spectral data cube with multiple spectral bands, or a

CCD mosaic where the detectors may not be exactly aligned, requiring different WCS calibrations.

- Very large cubes are increasingly common with modern instruments. It becomes physically impossible to store these as a single cube dataset (file); large cubes must be stored as multiple files, often with sophisticated voxel encoding, compression, tiling, or other data representation algorithms. The logical image abstraction needs to hide this complexity from client applications to simplify and unify data access.
- A key use case for using image/cube data is interactive image visualization and analysis, particularly for very large cubes. In this case, the image dataset is initially rendered in 2-D, greatly reduced in size, then the user interactively slices and dices the cube, computes 2-D projections, extracts spectra, computes moments, etc., to interactively view the contents of the cube. For very large cubes, this must be done via remote access to the cube data, which is staged to a parallel cluster providing the parallel computing capability required to quickly render the large cube, that may be tens to hundreds of GB in size or larger.

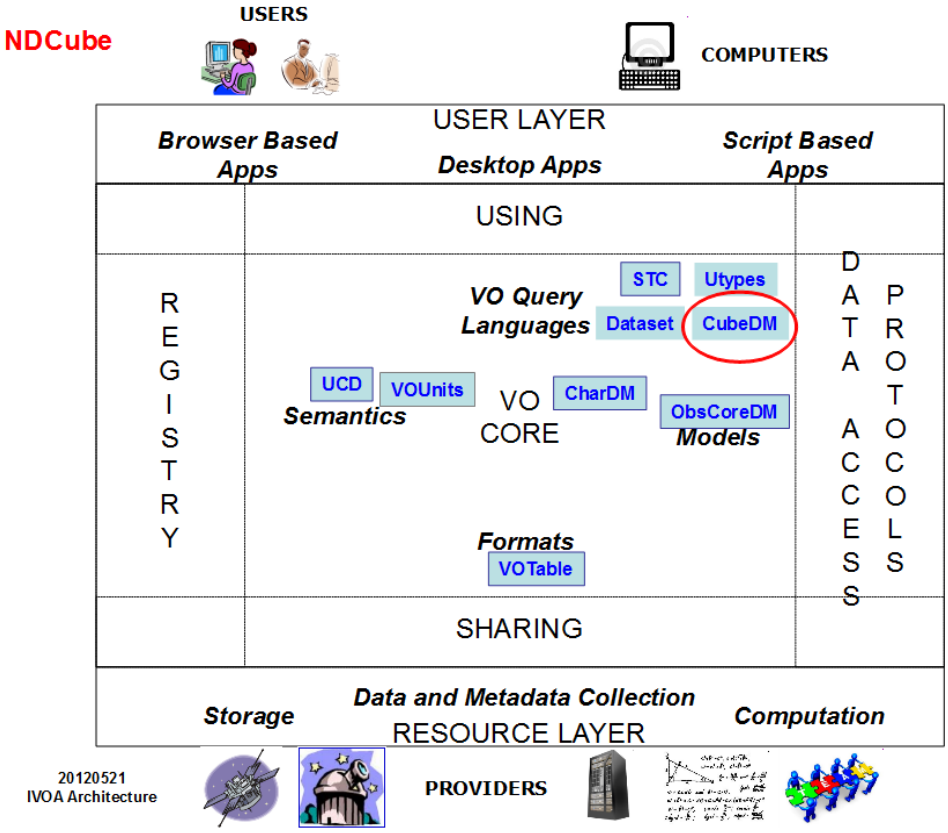
1.3 Cube Classifications

From the primary use cases, we can define these classifications of cube data to be addressed by this model:

1. **Simple image**. A single filled or mostly filled N-Dimensional image array with associated metadata describing the dataset, its WCS coordinate systems, etc.
2. **Simple sparse image.** A single N-Dimensional array as in case #1, however, large portions of the image may be sparse (have no data samples).
3. **Multiple subarray image.** An image dataset containing multiple subarrays, i.e., N-Dimensional image data arrays within the coverage of the overall image dataset. The subarrays may differ in size, resolution, coverage, or other characteristics. The overall image dataset containing the subarrays may be sparse. The overall image dataset does not have an explicit image geometry or sampling, only coverage. There are two sub-cases here:
   a) the subarrays are all part of the same observation and share common metadata. For example, a multi-band image observation.
   b) the subarrays may differ arbitrarily and the overall image dataset is essentially an aggregation of (possibly sparse) simple images as in case #1 or #2. For example, a complex image aggregating data from multiple observations.
4. **Large cube.** Very large cubes may be represented as either a simple image or multi-subarray type. The cube model abstraction does not impose any size limitations. The important factor here, is that the chosen view of the cube is independent of the physical storage structure.
5. **Wide-field survey.** This is essentially the same as #3, except that no subarrays (no explicitly pixelated data) or individual image datasets may be externally visible. The survey has coverage, but only automated virtual data generation techniques may be used to access the data, with sub-regions being computed on-the-fly and returned to the client. Alternatively, a survey might expose a collection of discrete cubes, in which case #1 or #3 may be used.
6. **Sparse data cube.** Sparse data are commonly used for higher-dimensional cubes, and are frequently sparse along one or more axes. For example, a multi-band image has

data at only a few given spectral coordinates, (each corresponding to a spectral bandpass). A spectral (or velocity) data cube may contain data for a number of widely spaced spectral bands, each of which may differ in the spectral resolution and number of channels. A time cube likewise may contain data, either individual points, or time series, arbitrarily spaced along the time axis with time regions where no data was taken. A multi-object spectral data cube may be sparse in the spatial plane. Event data can be considered a data cube which is sparse in all measurement axes.

1.4 IVOA Architecture Context



1.5 Model Dependencies

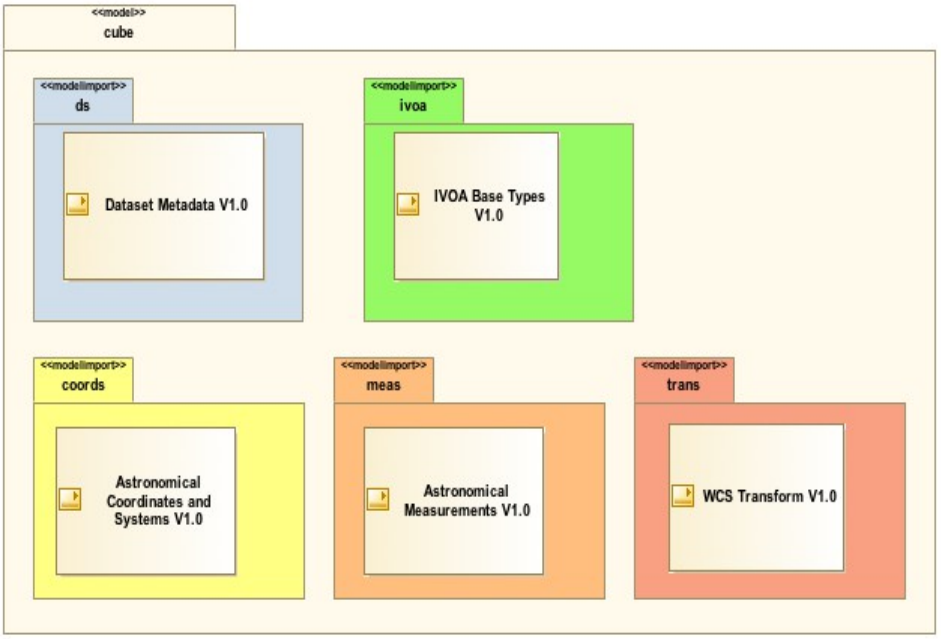


Figure 1: Model dependency diagram

The ND-Cube model is built on other IVOA data models as indicated in Figure 1. The <<model>> and <<modelimport>> stereotypes provide information identifying the model, it's version, any dependencies, and URLs to find more information about the model definitions including HTML and schema documentation. See Appendix B for more information about the content of these stereotypes and how they are used in serializations.

1.6 Structure of this Documentation

+ Major sections for each model area (Dataset, Observation, etc.).
+ First subsection in each section is the primary element within that model
+ Subsequent subsections for secondary elements, in alphabetical order.
+ Each subsection has sub-subsections for each attribute/relation
    - attributes show the full definition including datatype and usage.
    - relations describe the usage of the object in that context, the type of the target of the relation, and a reference to the full definition of that type.

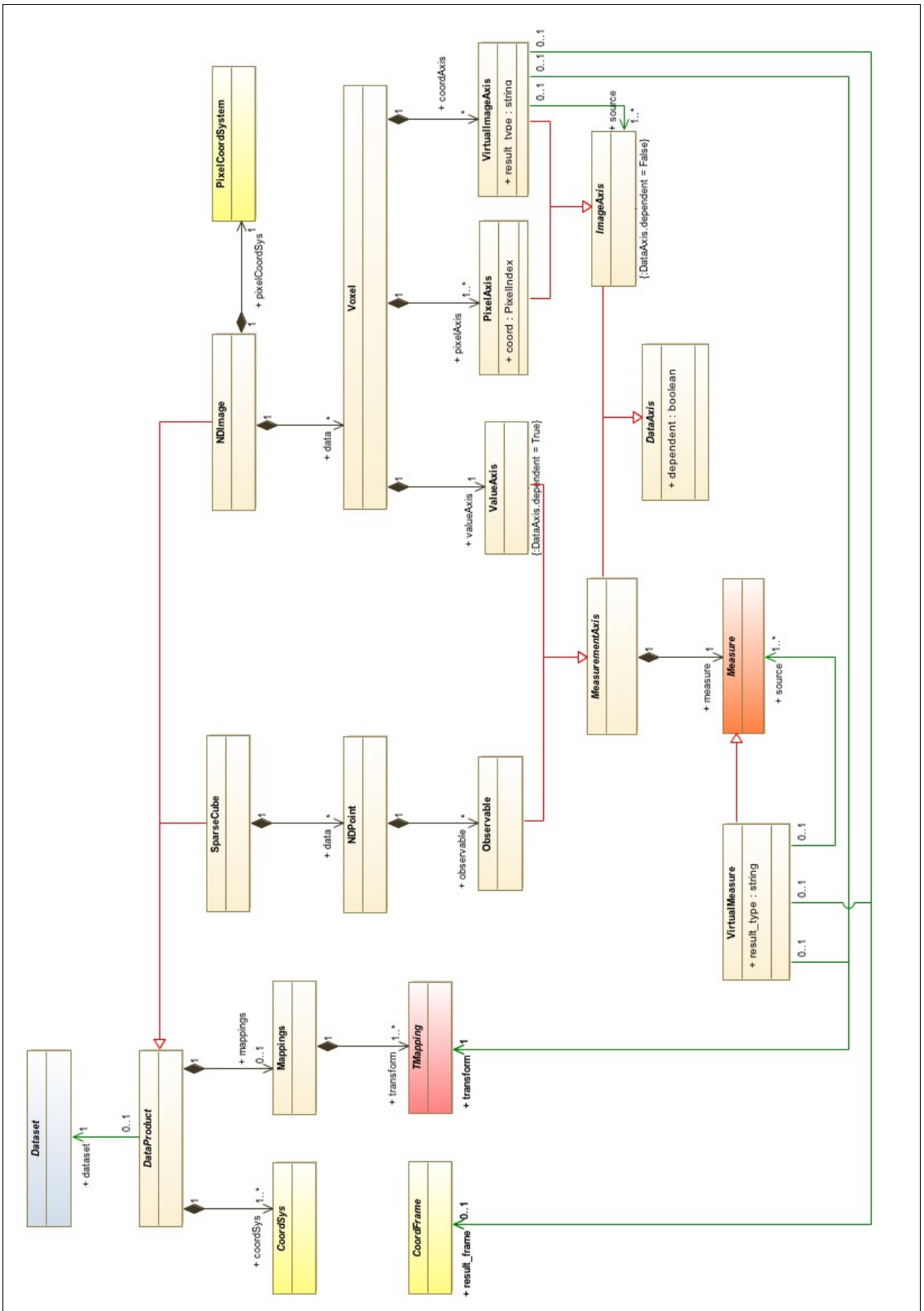
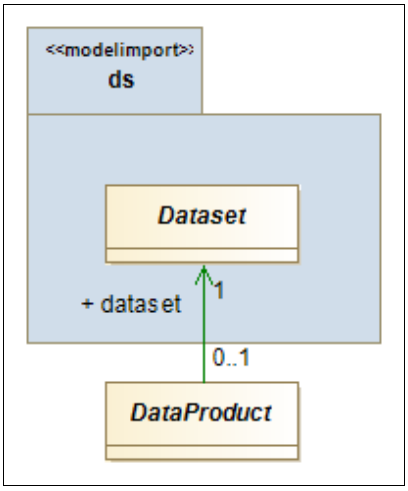


Figure 2: Overview diagram of NDCube model

2 Dataset/DataProduct Relation



To define a complete Dataset instance requires two components. A Dataset metadata instance, providing descriptive, identification, and provenance metadata related to how the dataset was generated, and one or more DataProduct-s with the data content. Each DataProduct refers to the appropriate Dataset metadata instance. This pattern facilitates the description of particular data products generated by different means, as well as complex datasets composed of multiple data products.

In this model, we define elements for describing NDCube datasets.

N-Dimensional Image Dataset:

The N-Dimensional Image Dataset, should be considered a collection of one or more NDImage objects, each of which defines a particular image instance.

In terms of our image cube classifications:

- **Simple Image** - a single Dataset with identifying metadata, and a single NDImage instance containing the data array definitions, coordinate systems, and (WCS) mapping definitions.
- **Single Sparse Image** - is the same as Simple Image, except in the mapping definitions used to transform the pixel space to world coordinates.
- **Multiple Subarray Image** - Use case 3a consists of a single Dataset with multiple subarrays, each of which is a separate NDImage instance. The dataset instance contains metadata for the entire image dataset. Each NDImage instance contains an N-Dimensional image array with associated metadata, including coordinate mappings, related to that instance. Use case 3b is an aggregation of related but otherwise independent Image instances. This is a form of *complex dataset* wherein standard datasets are combined to model more complex data. How the individual datasets are related is defined by the application and is not within the scope of this model.

Sparse Cube Dataset:

The Sparse Cube Dataset encapsulates N-Dimensional data which are sparse in nature, and do not have a set of regularly sampled (pixel) axes. For example, an event list.

3 DataProduct

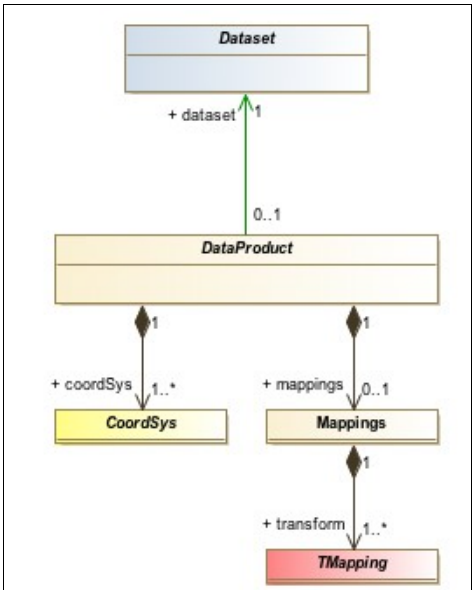


Figure 3: Data Product detail

Data Products represent a single instance of a particular type of data. It contains the minimum set of metadata required to define the data product type, all other metadata is left to the Dataset. The Data product definition is constant, regardless of what entity or process is generating it. For example, a Spectrum data product has the same structure and content regardless of whether it is a result of an observation, simulation, or fit to a theoretical model.

3.1 DataProduct

Abstract head of all data product types. All astronomical data products are expected to define the set of coordinate systems associated with the data, and any mappings which define the relations between coordinate frames. Typically mappings are used to define transformations between a pixelated coordinate frame and their corresponding world coordinate frame, but this is not always the case. For example, in sparse cube data such as event lists, mappings may be defined between physical frames (chip or sky) to world coordinate frames (csc or eqpos).

3.1.1 DataProduct.dataset

type: Dataset **type-detail: ds:dataset.Dataset**
multiplicity: 1

Reference to the Dataset Metadata instance associated with this data product.

3.1.2 DataProduct.coordSys

type: CoordSys **type-detail: coords:CoordSys**
multiplicity: 1..*

One or more coordinate systems associated with a particular data product. CoordSys is defined in the STC Coords model.

3.1.3 DataProduct.mappings

type: Mappings **type-detail: Section 3.2**
multiplicity: 0..1

The Mappings object holds a collection of frame transform definitions which facilitate the conversion of coordinates and other objects from one frame representation to another. For example, convert pixel coordinates from the pixel space to world coordinates in a 2D space frame. The Mappings object is defined in the STC Transform model.

3.2 Mappings

Simple container object to hold the collection of Mappings associated with the data product.

3.2.1 Mappings.transform

type: TMapping **type-detail: trans:TMapping**
multiplicity: 1..*

4 DataAxis

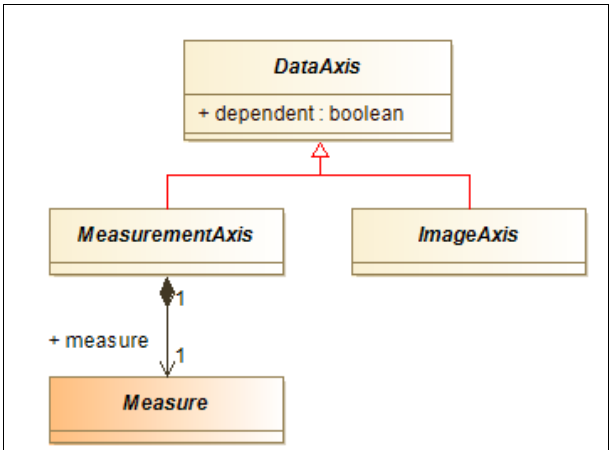


Figure 4: DataAxis primary subclasses

The DataAxis object provides a means of organizing data of various kinds and complexity, and identifying them as a single entity in the data product. Here we also identify the DataAxis as being dependent or independent of the other DataAxis instances in the data product. This element provides a great deal of flexibility in defining a wide variety of data products, from a simple collection of measured values, to a time series of spectra.

NOTE: We acknowledge that the use of the term 'Axis' here can be somewhat confusing. It is important to keep in mind that this 'Axis' does not necessarily map 1-1 to data along a single Coordinate Axis.

4.1 DataAxis

Abstract head of the DataAxis types.

4.1.1 DataAxis.dependent

type: boolean **type-detail: Section 8.1**
multiplicity: 1

Specifies whether or not the values associated with this DataAxis are dependent on the other data 'axes'. For example, an event list would comprise an NDPoint with n independent axes since all values were obtained together and none is a function of the other. In a time series, the 'time' axis is identified as independent, and all others dependent.

4.2 MeasurementAxis

MeasurementAxis is a sub-class of DataAxis whose data values are obtained through some measurement or determination process. These data values are instances of Measure from the Measurment model.

4.2.1 MeasurementAxis.measure

type: Measure **type-detail: meas:Measure**
multiplicity: 0..1

An instance of a data value obtained through experimental measure or detection as defined by the Measurement model.

4.3 ImageAxis

Extension of DataAxis for describing image axis data. Image axes differ from measurement axes in that they are typically defined either in pixel space or are virtual in nature.

5 N-Dimensional Image

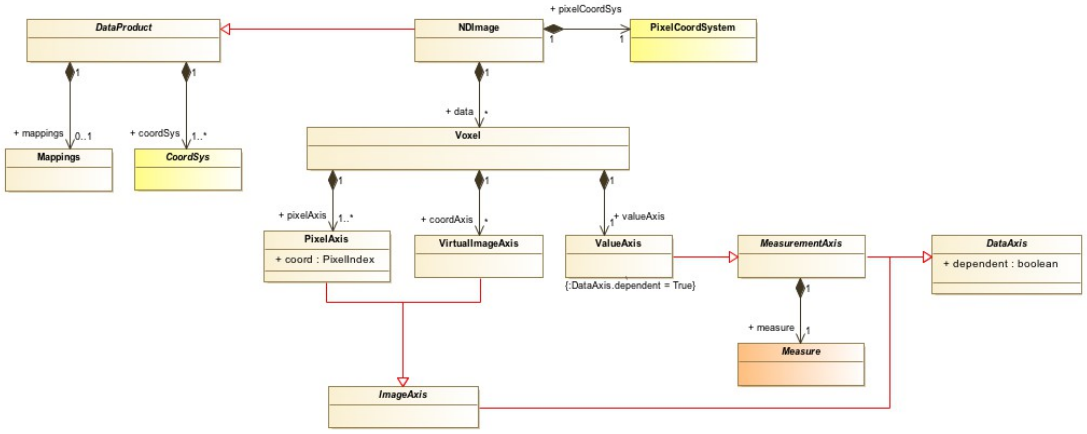


Figure 5: NDImage detail

5.1 NDImage

The essential characteristic of an "image" dataset is the presence of a multidimensional, regularly sampled numerical array, with associated metadata describing the image instance. While the concept of an image as a multidimensional array is fully general, astronomical datasets have some combination of specific "domain" related axes (spatial, spectral, time, polarization, etc), with flux or some other derived value, as the array element (voxel) value. The mapping of image pixel axes to physical coordinate frames is described by a set of transformation definitions associated with the image.

Unless dimensionality is otherwise indicated, the terms *image*, *cube*, and *hypercube* are interchangeable and refer to image (array-valued) data of arbitrary dimension. Image is a specialized case of general *hypercube* or *n-cube* data where the value at a given point in the hypercube is restricted to a simple numerical value. The data samples of an image are referred to as *pixels* (picture elements) or as *voxels* (volume elements), pixels being the preferred term for 2D images.

5.1.1 NDImage.pixelCoordSys

type: PixelCoordSystem **type-detail: coords:domain.pixel.PixelCoordSystem**
multiplicity: 1

The PixelCoordSys object provides the pixel domain specification. Typically, this will include a description of the pixel space (the number and span of each pixel axis).

5.1.2 NDImage.data

type: Voxel **type-detail: Section 6.2**
multiplicity: 0..*

The NDImage contains a set of zero or more voxels, holding the pixel and corresponding physical coordinates in N-D space, and the associated value at that location.

5.2 Voxel

The 'voxel' or volume element, is a fundamental atom on an N-Dimensional image. It is defined by a pixel coordinate, locating the volume in pixel space, corresponding physical coordinates, and the value at that location.

5.2.1 Voxel.coordAxis

type: VirtualImageAxis **type-detail: Section 7.1**
multiplicity: 0..*
Provide the physical coordinate location for the voxel. These are defined virtually, as transforms of either the pixel coordinates or of other physical coordinates.

5.2.2 Voxel.pixelAxis

type: PixelAxis **type-detail: Section 5.3**
multiplicity: 1..*

One or more PixelAxis objects which, as a set, locate the voxel in N-D pixel space.

5.2.3 Voxel.valueAxis

type: ValueAxis **type-detail: Section 5.4**
multiplicity: 1
Provides the value at the voxel pixel coordinates. The value axis is always dependent on the pixel axes.

5.3 PixelAxis

The PixelAxis identifies a pixel coordinate location associated with the voxel.

5.3.1 PixelAxis.coord

type: coords:PixelIndex **type-detail: coords:domain.pixel.PixelIndex**
multiplicity: 1

Pixel coordinate value for the Voxel along a particular pixel axis.

5.4 ValueAxis

Concrete subclass of MeasurementAxis providing the measured value (with errors) at that location in pixel space.

6 Sparse Cube

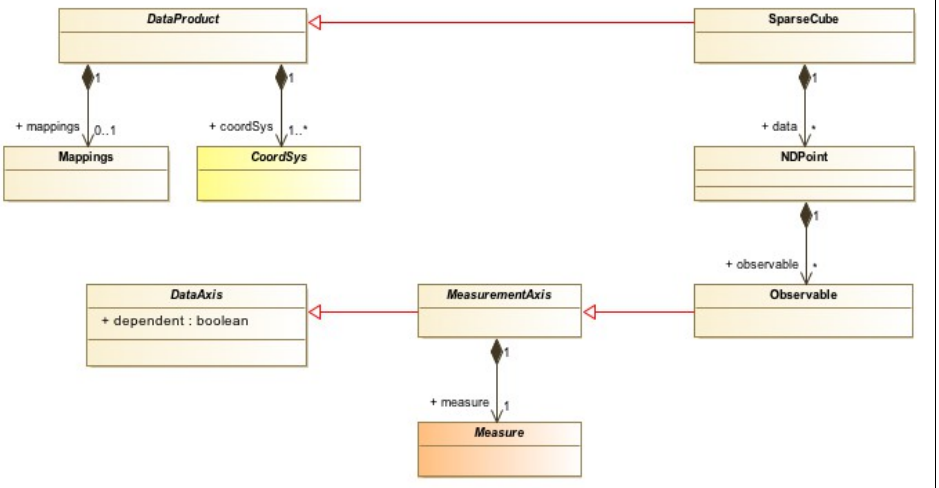


Figure 6: Sparse Cube detail

6.1 SparseCube

An extension of PointDataProduct, the SparseCube is a collection of N-Dimensional points, each of which provide a value for the point in each represented domain.

6.1.1 SparseCube.data

type: NDPoint **type-detail: Section 6.2**
multiplicity: 0..*

Zero or more NDPoints, providing the sparsely sampled data values in each represented domain.

6.2 NDPoint

The NDPoint object is defined by collection of associated values on each data axis. As a unit, these completely define a Point in N-Dimensional coordinate space.

6.2.1 NDPoint.observable

type: Observable **type-detail: Section 6.3**
multiplicity: 0..*

Zero or more DataAxis objects, each identifying the data sample for a particular axis of the Point.

6.3 Observable

Concrete subclass of MeasurementAxis. Each element of the NDPoint is an Observable composed of some sort of measured value with any associated errors.

7 Virtual Data

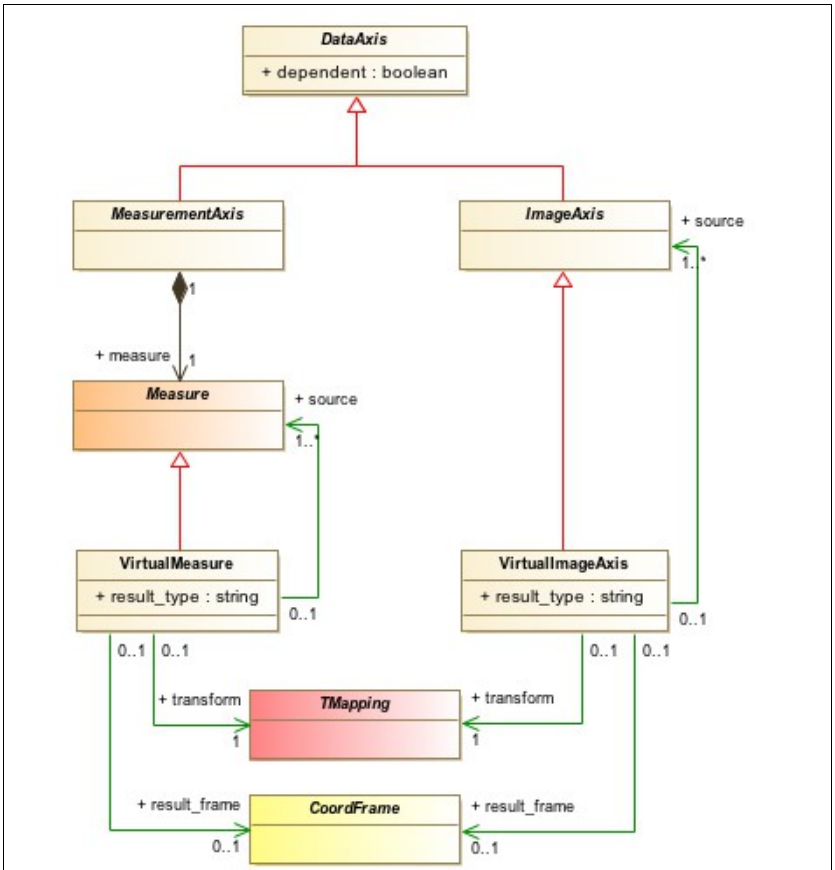


Figure 7: Virtual Data Objects

Virtual data is derived or calculated as a transform of some other data. Data products often define certain elements as a function of other elements. This is not simply a serialization strategy, but makes an important statement about the relation between the two elements. We define two types of virtual data in this model.

7.1 VirtualImageAxis

In an NDImage, the physical coordinate data is defined as a function of either the pixel coordinates (pixel to sky), or of other physical coordinates (sky to equatorial). This object encapsulates the various components needed to define the resulting data in terms of the source elements.

7.1.1 VirtualImageAxis.result_frame

type: CoordFrame **type-detail: coords:CoordFrame**
multiplicity: 0..1

Reference to the resulting Frame of the virtual coordinate data.

7.1.2 VirtualImageAxis.result_type

type: string **type-detail: 8.1**
multiplicity: 1

vo-dml ID of the resulting type created by applying the transform to the given source.

7.1.3 VirtualImageAxis.source

type: ImageAxis **type-detail: 4.3**
multiplicity: 1..*

Reference(s) to the source data. In this case, these will be either the PixelAxis data, or some other ImageAxis data.

7.1.4 VirtualImageAxis.transform

type: Transform **type-detail: trans:Transform**
multiplicity: 1

Reference to the Transform to apply.

7.2 VirtualMeasure

VirtualMeasure defines data which are determined as a transform of some other Measure.

7.2.1 VirtualMeasure.result_frame

type: CoordFrame **type-detail: coords:CoordFrame**
multiplicity: 0..1

Reference to the resulting Frame of the virtual data.

7.2.2 VirtualMeasure.result_type

type: string **type-detail: 8.1**
multiplicity: 1

vo-dml ID of the resulting type created by applying the transform to the given source.

7.2.3 VirtualMeausre.source

type: Measure **type-detail: meas:Measure**
multiplicity: 1..*

Reference(s) to the source Measure data.

7.2.4 VirtualMeasure.transform

type: Transform **type-detail: trans:Transform**
multiplicity: 1

Reference to the Transform to apply.

8 Data Types

8.1 IVOA Data Types

The ivoa model provides a set of standardized primitive data types as well as types for representing quantities (values with associated units and ucd). We provide a diagram of the model here, and refer the reader to Appendix F of the VO-DML modeling specification document[8] for more information.

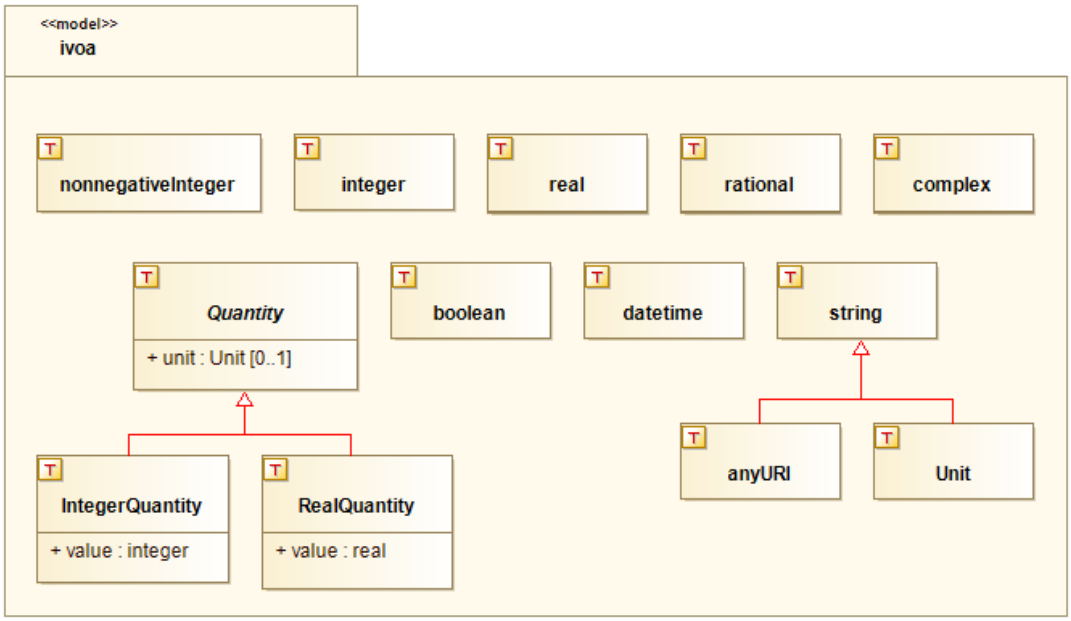


Figure 8: IVOA Base Data Types

8.1.1 Units

This model requires the use of the IVOA VOUnits Standard[9] for representing units of physical quantities. This standard reconciles common practices and current standards for use within the IVOA community.

8.1.2 UCDs

This model requires the ucd field to comply with syntax defined in ”An IVOA Standard for Unified Content Descriptors”[11].

8.1.3 Dates

The 'datetime' datatype is for expressing date-time values. The string representation of a datetime value should follow the FITS convention for representing dates. The FITS standard is effectively ISO8601 format without the "Z" tag to indicate UTC (YYYY-MM-DDThh:mm:ss). Values are nominally expressed in UTC.

8.2 Cube Model DataTypes

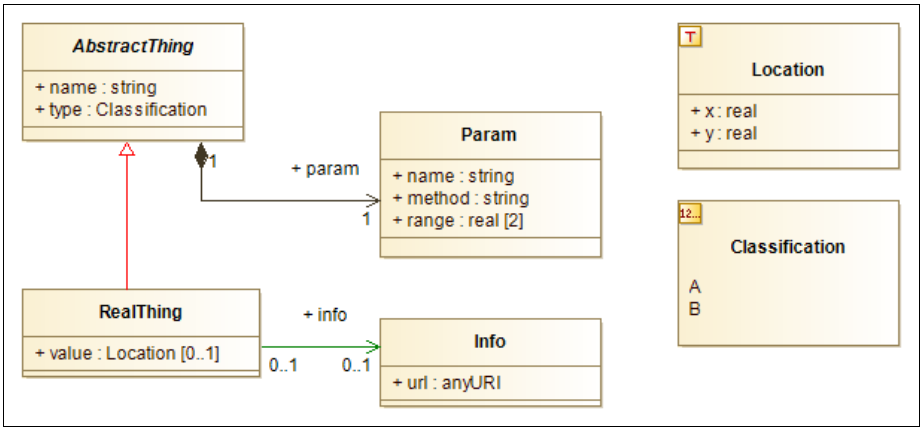
This model defines no new data types or enumerations.

Appendix A: Modeling Conventions

This model adopts the same modeling conventions as outlined in Appendix A of the Dataset Metadata model. Here, we provide a brief summary of these statements.

1 Diagram notation

This model follows the VO-DML modeling practices, however, UML representations may vary depending on the tool used. Below, we describe the graphical representation of the modeling concepts and relations.



1.1 ObjectType

ObjectType-s are represented by a plain box. The class name is annotated in the top window, abstract classes use italic typeface. Attributes, if any, are listed in the lower panel. Attributes may only be of primitive type (real, string, etc), a defined DataType, or an Enumeration type. Relationships to other objects are defined via the composition and reference relation arrows.

1.2 DataType

DataTypes are represented by a box shape similar to ObjectType, but annotated with a "T" symbol in the top left corner.

1.3 Enumerations

Enumerations are represented by a box shape similar to ObjectType, but annotated with a "1,2.." symbol in the top left corner. Enumeration Literals (possible values) are listed below the enumeration class name.

1.4 Generalization

Generalizations are represented by a red line, with open triangle at the end of the source, or more general, object.

1.5 Composition

The composition relation is indicated by a black line with a solid diamond attached to the containing object, and an arrow pointing to the object being contained. The composition relation is very tight, where the container is responsible for the creation and existence of the target. Any object may be in no more than one composition relation with any container. The attribute name for the composition relation is annotated at the destination of the relation (e.g. "+ dataID"). This is typically a lower-cased version of the destination class name, but this is not required.

1.6 Reference

The reference relation is indicated by a green line, with an arrow pointing to the object being referenced. The reference relation is much looser than composition, the container has no ownership of the target, but merely holds a pointer, or other indirect connection to it. The attribute name is annotated at the destination of the relation (e.g. "+ proposal"). This is typically a lower-cased version of the destination class name, but may be another name indicating the role that the class is playing in this context.

1.7 Multiplicity

All attributes and relations have a multiplicity associated with them. For attributes, the multiplicity is contained within brackets just after the attribute name. If no bracket is displayed, this is equivalent to '[1]'.

+ 1 = one and only one value must be provided.
+ 0..1 = zero or one value may be provided.
+ * = zero or more values may be provided (open ended).

2 Model Identification metadata

We defined stereotypes on elements of this model to provide metadata identifying this model, and other models on which it is dependent.

2.1 Model stereotype

The Model stereotype (<<Model>>) consists of a set of Model properties which identify a particular model and its dependencies.

2.2 Import Stereotype

The <<import>> stereotype is attached to Packages representing imported models. It identifies the model by name, and provides URLs from which the full description may be obtained.

3 Extensibility

User-defined content would be modeled as an extension of the IVOA standard.

3.1 Scope

We permit any object modeled in this document to be extended with user-defined content, with the following restrictions:

- Follow VO-DML modeling practices.
- Values of extended content must be consistent with the content of modeled data. That is, using the IVOA base primitive types, Quantity, and STC Coordinates as appropriate.
- Since extended content, by definition, does not follow the corresponding model, it is not possible for general applications to interpret complex structures within that content. It is, therefore, recommended that users define extended content in such a way as to avoid ambiguity between its components.

3.2 Support

Applications should, but are not required to, provide the following support for extended content:

- Retain existence of extended content, including namespace and UTypes.
- Retain association with modeled component.
- Provide access to extended content by users.

Appendix B: N-Dimensional Cube Model Summary

Cube Model Identification				
Model Element	**Datatype**	**Mult.**	**Meaning**	**value**
Model identification				
Model				
Model.name	string	1	Data model name	"N-Dimensional cube"
Model.version	string	1	Data model version	"1.0"
Model.prefix	string	1	Data model prefix tag	"cube"
Model.url	anyURI	1	Reference URL for model	<TBD>
Imported Model				
Import.name	string	1	Imported model name	"ds"
Import.version	string	1	Imported model version	"1.0"
Import.url	anyURI	1	Reference URL for imported model	https://volute.g-vo.org/svn/trunk/projects/dm/DatasetMetadata/vo-dml/DatasetMetadata-1.0.vo-dml.xml
Imported Model				
Import.name	string	1	Imported model name	"coords"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	http://volute.g-vo.org/svn/trunk/projects/dm/STC/vo-dml/STC_coords-v2.0.vo-dml.xml
Imported Model				
Import.name	string	1	Imported model name	"meas"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	http://volute.g-vo.org/svn/trunk/projects/dm/STC/vo-dml/STC_meas-v2.0.vo-dml.xml
Imported Model				
Import.name	string	1	Imported model name	"trans"
Import.version	string	1	Imported model version	"2.0"
Import.url	anyURI	1	Reference URL for imported model	http://volute.g-vo.org/svn/trunk/projects/dm/STC/vo-dml/STC_trans-v2.0.vo-dml.xml
Imported Model				
Import.name	string	1	Imported model name	"ivoa"
Import.version	string	1	Imported model version	"1.0"
Import.url	anyURI	1	Reference URL for imported model	http://www.ivoa.net/xml/VODML/IVOA-v1.vo-dml.xml

Cube Model Summary				
Model Element	**Datatype**	**Mult.**	**Meaning**	**UCD1+**
Cube Model Elements				
DataAxis			Provides access to 'value', and context within the product.	
DataAxis.dependent	ivoa:boolean	1	specifies if data axis is dependent on other axes	
DataProduct			Instance of data product (abstract)	
DataProduct.coordSys	coords:CoordSys	*	Coordinate frames associated with a particular data product	
DataProduct.mappings	Mappings	0..1	Transforms associated with this data product	
Mappings			Container of Coordinate frame transformations	
Mappings.transform	trans:TMapping	1..*	Coordinate frame transforms	
MeasurementAxis			Data values obtained through measurement process	
MeasurementAxis.measure	meas:Measure	1	Instance of measured value associated with data axis	
NDImage			N-Dimensional Image data	
NDImage.data	Voxel	*	N-Dimensional Image data array	
NDImage.pixelCoordSys	coords:PixelCoordSystem	1	Pixel coordinate system defintion.	
NDPoint			N-Dimensional data point	
NDPoint.observable	Observable	*	Data axes of NDPoint	
Observable			Observable axes of NDPoint (extends MeasurementAxis)	
PixelAxis			Pixel coordinate location associated with a voxel	
PixelAxis.coord	coords:PixelIndex	1	Pixel coordinate	
SparseCube			Instance of sparse cube data	
SparseCube.data	NDPoint	*	N-Dimensional sparse cube points	
ValueAxis			Value axis of Voxel (extends MeasurementAxis)	
VirtualImageAxis			Virtual image axis data	
VirtualImageAxis.result_frame	coords:CoordFrame	0..1	Frame of resulting image axis data	
VirtualImageAxis.result_type	ivoa:string	1	vo-dml ID of resulting image axis data type	
VirtualImageAxis.source	ImageAxis	1..*	Source data to transform	
VirtualImageAxis.transform	trans:TMapping	1	Transform to apply	
VirtualMeasure			Virtual image axis data	
VirtualMeasure.result_frame	coords:CoordFrame	0..1	Frame of resulting image axis data	
VirtualMeasure.result_type	ivoa:string	1	vo-dml ID of resulting Measure data type	
VirtualMeasure.source	meas:Measure	1..*	Source data to transform	
VirtualMeasure.transform	trans:TMapping	1	Transform to apply	
Voxel			Volume pixel	
Voxel.coordAxis	VirtualImageAxis	*	Voxel physical image axis data	
Voxel.pixelAxis	PixelAxis	1..*	Voxel pixel dimension axis	
Voxel.valueAxis	ValueAxis	1	Voxel value axis	

References

[1] "IVOA Dataset Metadata Model": Version 1.0, 10 Sept. 2014
http://www.ivoa.net/Documents/...
[9] ”Units in the VO”: Version 1.0
http://www.ivoa.net/Documents/VOUnits/20120820/PR-VOUnits-1.0-20120820.pdf
[10] ”The UCD1+ controlled vocabulary”: Version 1.23
http://www.ivoa.net/Documents/REC/UCD/UCDlist-20070402.pdf
http://cdsweb.u-strasbg.fr/UCD/ucd1p-words.txt
[11] ”An IVOA Standard for Unified Content Descriptors”: Version 1.10
http://www.ivoa.net/Documents/REC/UCD/UCD-20050812.pdf
http://www.ivoa.net/Documents/latest/UCD.html
[8] ”VO-DML a consistent modeling language for IVOA data models”: Version 1.0
http://www.ivoa.net/documents/VODML/20180307/VO-DML-PR-v1.0.pdf