VO-DML Modeling: Beginner's Guide

Version: 0.1
IVOA Note: 20160916

This version:
0.1-20160916

Authors:
Mark Cresitello-Dittmar, Thomas Donaldson, Gerard Lemson, Omar Laurino, Laurent Michel, Arnold Rots.

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.

Contents

Introduction

The VO-DML specification[1] defines the concepts and relations to be used when generating IVOA data models. This document, provides support information for IVOA data modelers to assist them in getting started generating VO-DML compliant models. We will cover:

- VO-DML XML
- Model generating tools
    - Resources available to get started
- Model Validation
- Utilities for generating other products from VO-DML XML
    - Interactive HTML model description

This document does NOT address serializing instances of a data model using the “VO-DML: Mapping Data Models to VOTable” specification.

As work progresses in generating VO-DML compliant models, the resources available are expected to grow and evolve fairly quickly. Our goal is to capture the procedures and resources available at the time of the writing, and update the document as needed.

Volute Repository

Volute is a svn repository currently being used to warehouse resources available in support of vo-dml data modeling (among other projects in the IVOA).
Important URLs

- https://volute.g-vo.org/ - launch page for the repository
- https://volute.g-vo.org/viewvc/volute/trunk/projects/dm/ - Data Model group base.. viewer
- https://volute.g-vo.org/svn/trunk/projects/dm/ - Data Model base.. Direct browse

For generating new IVOA data models, it is recommended to use this resource for iterating on models and documents in preparation for upload to the IVOA Documents and Standards page. See the primary launch page for relevant information about access privileges.

The general structure of the dm project is fairly loose for each model, an effort is under way to make the area more consistent using the following general framework.

.../dm	Top Level
.../dm/<DM>	Data model top (eg: CubeDM-1.0)
.../dm/<DM>/doc	Data model document (doc, ivoatex, pdf)
.../dm/<DM>/model	Data model itself (UML, diags, etc)
.../dm/vo-dml	VO-DML resource top
.../dm/vo-dml/xslt	VO-DML model processing scripts
.../dm/vo-dml/models	VO-DML model product base
.../dm/vo-dml/models/<model>	VO-DML products (XML, HTML) for model (eg: cube, stc2)

VO-DML XML

What is it?

VO-DML XML is a serialization of a data model in XML format conforming to the syntax defined by the VO-DML XML schema[2]. The file must also validate to against the VO-DML modeling rules, which are enforced by an associated schematron file[3].

Why do we need it?

A standardized serialization of the data model facilitates the generation of resources which automatically generate various products based on the model specification. Examples include:

- Validators - which compare the XML against the schema and schematron specifications, ensuring that the model follows the specification and is, therefore, compatible with other VO-DML compliant models.
- HTML - automatically generate HTML pages of the model description. Optionally, may include a UML-like image of model objects and relations which is dynamically linked to the HTML content.
- Code - automatically produce code which implements the model components. This code may not be useful for all purposes, but can serve prototypes, instance validators, and other generic purposes.

With the approval of the VO-DML specification in the IVOA, it becomes a requirement that data models pass VO-DML compliance verification.

How do I generate it?

Much of the rest of this document will be devoted to answering this question. In short, it does not matter HOW one generates the XML file. There are several options available, including:

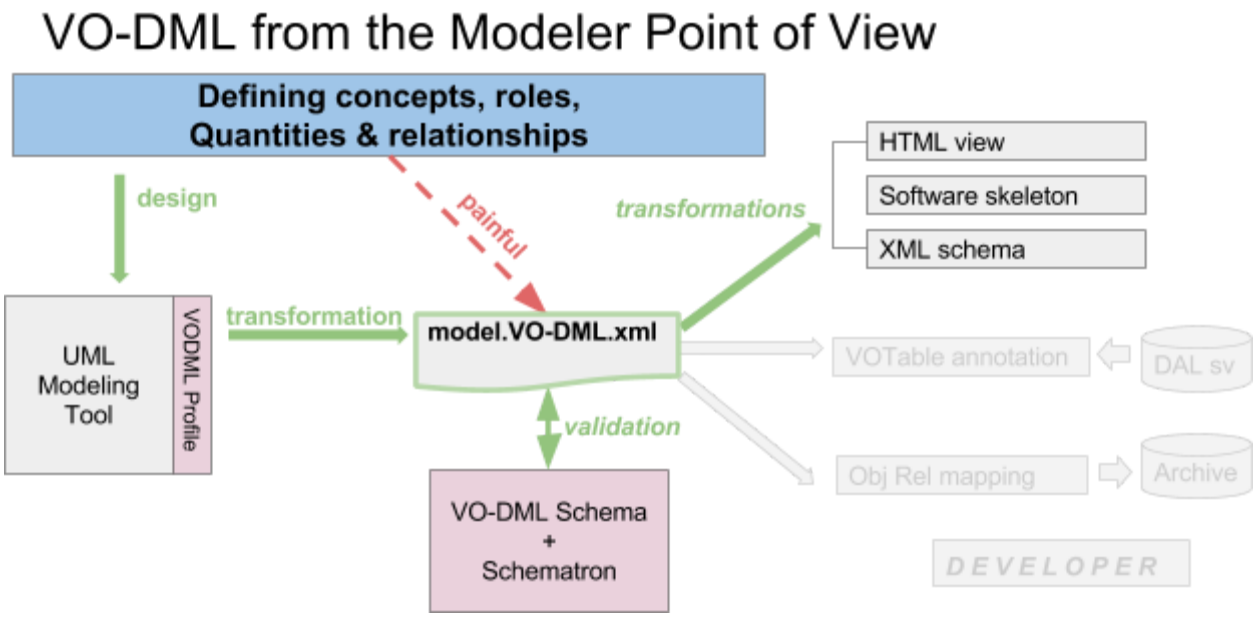
- Text editor - most basic, generate the XML by hand.. Not recommended.
- XML editor - could import the VO-DML schema to assist in using the proper components and relations. The modeler would need to be mindful of the rules enforced by the schematron while producing the model.
- UML tools - Since most IVOA data models include UML diagrams of model components, this approach serves both the model and document generation. However, these tools typically export to some version of UML/xmi file. There is a good deal of variation in this format between modeling tools, even for the same version of xmi. For this reason, it is difficult to share models between tools. To facilitate the translation of xmi files to VO-DML XML format, a set of xslt scripts have been written for 3 modeling tools currently being used within the IVOA. See below for details.
- Domain Specific Language (DSL) - Simplified model aware user interface which hides the complexity of the specific output format and/or facilitates output in multiple formats.
- Custom GUI - An excellent project for someone! A VO-DML aware gui which can facilitate the production of UML-style diagrams, enforce the schema and schematron rules, and export to VO-DML XML format.

Modeling with UML Tools

At the time of this writing, there is active modeling work being done using 3 modeling tools:

- MagicDraw CE 12.1: <URL?>
- Modelio 3.0: https://www.modelio.org/downloads/download-modelio.html
- Altova Umodel 2016 release 2 sp1: http://www.altova.com/download/umodel.html

As mentioned above, the primary export format for UML models (xmi) has a great deal of variation between modeling tools, even for the same version of xmi format. This makes it very difficult to share models between modeling tools. It also makes this format a poor choice for the machine readable serialization of data models, and is a primary driver for the specification of VO-DML XML.



Do I have to use a specific modeling tool?

Modelers are not required to use any specific modeling tool or version thereof. However, keep in mind that the xslt processing scripts to generate the VO-DML XML only exist for the above tools. To use another tool will involve extra work to define the proper stereotypes, and imported base types package as well as the xslt translation script.

Template Models

For each of these tools, a Template model has been generated which is pre-loaded with

- IVOA UML profile - defining VO-DML stereotypes for use in the models
- IVOA base types - package of primitive data types and Quantities. The package is pre-tagged with the <modelimport> stereotype.

These templates should provide the framework necessary to start modeling with one of these tools. Templates are stored in volute at

https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/models/templates/

- Template_MD_CE_12.1.xml
- Template_Modelio_3.0.zip
- Template_AltovaUModel_2016r2sp1.ump

and may be imported into the respective tool. The exact steps going from importing the template to generating a new model project will vary be tool. Please see the appropriate Appendix for detailed instructions.

What if I want to use another modeling tool?

There is no required modeling tool for the VO. However, since xmi output is so tool dependent, using another modeling tool would require creating a new translation xslt script. To facilitate this, we have a sample model which exercises the full set of VO-DML objects and relations. This makes it a good testbed for the xslt translation scripts.

- Implement sample model using your favorite modeling tool.
    - https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/models/sample/sample/Sample.html
- Export the model in xmi format
- Basing off existing xslt scripts, or starting from scratch, generate script to translate your model xmi file to VO-DML XML format.
- When your script output matches the baseline VO-DML XML file, the script properly translates all VO-DML elements and relations.
    - https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/models/sample/sample/Sample.vo-dml.xml
- Add your xmi file to the sample model directory as reference for others.
- Add to xslt script to volute repository.
    - https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/xslt/

Validating your model

Once you have generated your model, you will need to generate a VO-DML XML representation of it, and process it against a validation tool. At present, this involves the following steps:

- Export model in xmi format. Currently the xslt scripts handle version UML-2.4.1
- Install xmi file in volute working copy
- Process xmi file with appropriate xslt script to produce VO-DML XML file
- Run VO-DML XML validator and review results

See ‘XSLT Processing scripts’ section for more details.

XSLT Processing scripts

This process is set up as an 'ant' project in the 'vo-dml' subdirectory of the volute repository. These instructions assume that the reader has checked out a working copy of the repository. We use the following notations:

- <BASE> = path to 'dm' directory of your svn working copy.
- <your model name> = destination for vo-dml output; typically is the same as the model name (prefix).

Relevant files: files and spaces involved in the procedure

- <BASE>/vo-dml/build.xml
  Ant build file with targets to execute various utilities
  + run_xmi2vo-dml - convert xmi to VO-DML XML format
  + run_validate_vo-dml - validate VO-DML XML
  + run_vo-dml2html - generate HTML documentation

- <BASE>/vo-dml/build.properties
  Control file for ant project. This file specifies which areas are to be processed and various support information for the scripts.

- <BASE>/vo-dml/models/<your model name>
  Destination for script output.

- <BASE>/vo-dml/models/templates/vo-dml.properties
  Provides specific information for processing a particular model.

Destination Setup

Initial setup of the xslt script output destination. The following are provides as a unix command thread. Users on other platforms should adjust to equivalent actions.

- Create destination directory
  - cd <BASE>/vo-dml
  - mkdir ./models/<your model name>
- Transfer your data model xmi file to this location
  - cp myModel_uml2p4p1.xmi ./models/<your model name>/.
- Transfer properties file to destination area
  - cp ./models/templates/vo-dml.properties ./models/<your model name>/.
- Edit properties file
  - replace Sample values with those appropriate for your model.
  - Choose profile according to modeling tool used.
- Edit build.properties control file
  - Set 'models' value. eg: models=./models/<your model name>

- Other variables should be set as needed to run the various targets.

XMI to VO-DML Conversion: run_xmi2vo-dml

Once the framework is set as described above, the xmi to VO-DML XML conversion script is invoked by executing the ant build target 'run_xmi2vo-dml'.

- cd <BASE>/vo-dml
- ant run_xmi2vo-dml

The build will send informative text to the screen describing the various packages found and whether it is including them or setting up as import. (eg: 'ivoa' package should always be imported).

If all goes well, a final "BUILD SUCCESSFUL" message will appear.
The destination directory will contain a new file named according to the properties file..
<prefix>.vo-dml.xml

Validate VO-DML XML: run_validate_vo-dml

This target invokes verification utilities which validate the VO-DML XML file against the schema and against additional VO-DML ruled prescribed in a Schematron.

- cd <BASE>/vo-dml
- ant run_validate_vo-dml

The build will send informative text to the screen regarding the files generated and results of validation. Schema issues will be reported to the screen while schematron issues are reported to a validation log in the destination directory as indicated on the screen:
Eg: "Schematron errors found, see <prefix>.vo-dml.xml.validation-report.txt"

Review reported issue, correct as necessary and repeat until a successful validation results.

- Screen output indicating
  - No schema validation issues
  - "No errors found, VO-DML document passes Schematron validation"
- Validation report indicates "No errors"

Generate HTML: run_vo-dml2html

Once one has a successfully validated their VO-DML model, an HTML version should be generated. This provides a browsable version of the model which can be reviewed and used as reference by other modelers. Optionally, an interactive graphical image of the model may be generated. This currently uses GraphViz dot utility to create a UML-like diagram showing object and their relations.

- cd <BASE>/vo-dml
- ant run_vo-dml2html

The build will send informative text to the screen indicating steps taken with a final "BUILD SUCCESSFUL" message.

The destination directory will contain new file(s) named according to the properties file..
<prefix>.html == HTML page
<prefix>.png == if graphvis.path is specified in build.properties, interactive image.
You can then direct your browser to the resulting HTML page to review.

Adding to Volute

You will want to upload the VO-DML model specification to volute in order to:

- Make results visible to other modelers
- Make model available for reference by other models. In order for a model to be imported to another model, the VO-DML XML file must be available on the web. The URL is specified in the modelimport stereotype. At present, all VO-DML models are installed to the Volute repository and imported from there.

The above process generates some intermediate files which do not need to be added to volute. The following should be included in the repository:

- <prefix>.html
- <prefix>.png
- <prefix>.vo-dml.xml
- <prefix>.vo-dml.xmi.validation-report.txt
- xmi file
- vo-dml.properties

Commands to add the new model results to volute:

- svn add models/<your model name>
- svn commit -m “<check-in comment>” models/<your model name>

References

1. “VO-DML: a consistent modeling language for IVOA data models”: Version 1.0, 30 March 2016. http://www.ivoa.net/documents/VODML/index.html
2. VO-DML/XML schema: Version 1.0, https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/xsd/vo-dml-v1.0.xsd
3. VO-DML/XML Schematron file: Version 1.0, https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/xsd/vo-dml.sch.xml

Appendix A: Working with Modelio Template

Getting started with Modelio Template

We provide template files to assist modelers with starting new data modeling projects. These templates are pre-loaded with:

- stereotype definitions
- primary packages with assigned stereotypes
- the base 'ivoa' package of base data types with modelimport stereotype assigned.
- Model dependency diagram
- Initial model overview diagram with ivoa diagram style

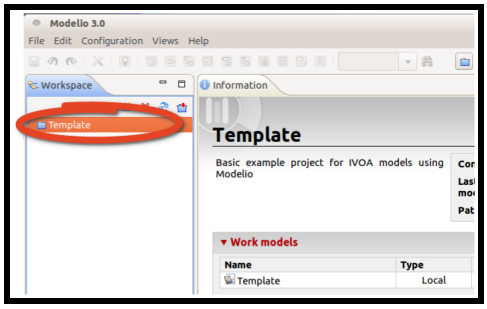
Note for Eclipse users on Linux: Using Modelio might alter the Eclipse workspace and vice versa. That can be prevented by using a shell wrapper making Modelio run from a base directory other than $HOME:

```
#!/bin/bash
STARTDIR=$HOME/Modelio3.0
if [ ! -d "$STARTDIR" ]; then
    mkdir "$STARTDIR"
fi
pushd "$STARTDIR"
$HOME/Modelio3.0/modelio
popd
```

For Modelio, we provide a full project export file which can be downloaded from the volute repository at:
https://volute.g-vo.org/svn/trunk/projects/dm/vo-dml/models/templates/Template_Modelio_3.0.zip
We have found that with Modelio, if we import a Template model xmi file, the common UML base types (real, boolean, integer, etc) are removed from the 'ivoa' package which defeats the primary purpose. We, therefore, must convert the template model to a new model specification with the following steps.

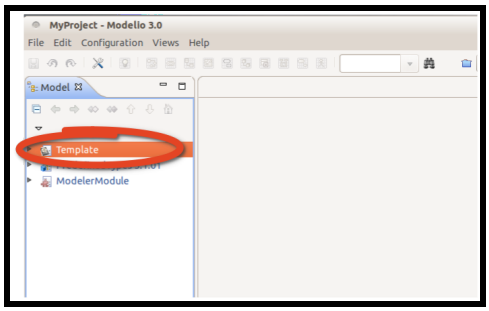
Step1: Import Template project

+ From modelio main window, select **File>Import a project**
+ Find and select Template model in pop-up window, click **Open**
  The new "Template" project should appear in the 'Workspace' panel of the Modelio GUI.



Step2: Rename project

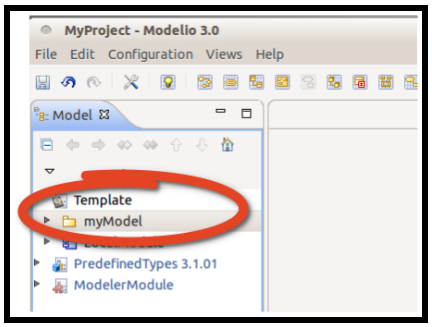
+ Select Template project in Workspace panel.
+ Right-click; select **Rename project**
+ Enter new project name in pop-up window.
  This will set the workspace directory tree to use the new project name.



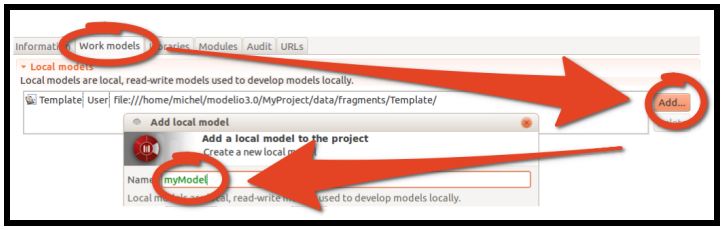
Step3: Convert template to your model

There is no straight-forward means of renaming the template model. The following is the simplest procedure for re-defining the template as another model. This retains all stereotype assignments etc from the template.

+ Open project by double-clicking on project.
  The Workspace panel will be replaced by a Model panel containing the template model content.
+ Expand Template model in Model panel.
+ Right click template package. Select **Rename**; enter new model name.



+ From toolbar select **Configuration>Work models…**
+ Select ‘Work models’ tab
+ Click ‘**Add**’ to add your new model; enter model name in pop-up window.



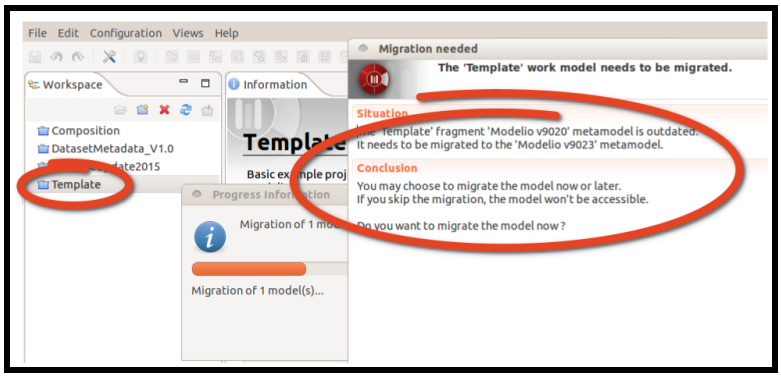
+ Now we must replace the new model workspace content with the Template model content.
    + Note the workspace location from the work models window
    + Outside of modelio, use file explorer or “cd” and browse to workspace fragments location.
    + Manually delete new model directory and rename Template to new model name.
    + This effectively replaces the content of the new workspace fragment with the original Template content.
+ Back in “Project Configuration” window,
    + Select “Template” model; click “**Delete**”
    + Click “**Close**”
+ From Modelio main window; Select **File>Close project**
+ Re-open project; double click on project
+ Expand new project, rename primary package by right click - **Rename**

The new project is now ready for development! Open the “overview diagram” and begin generating the new model specification.

Migrating to Modelio 3.x

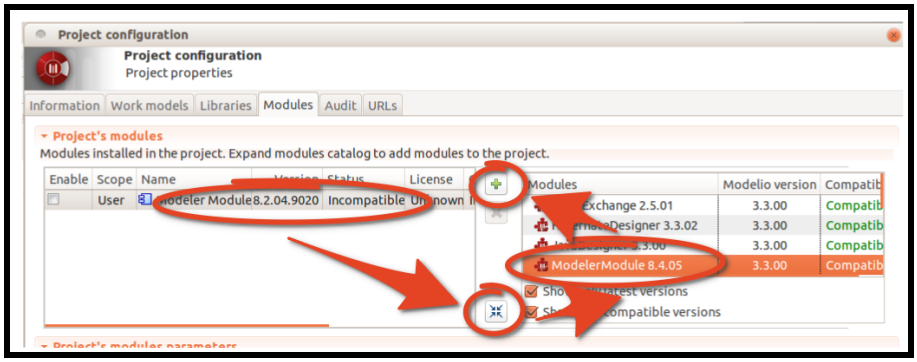
Although the VO-DML template has been designed for Modelio 3.0, it is possible to work with a newer version (tested up to 3.5). This requires upgrading the Template modules.

+ Import the Template model as described in the previous section.
+ An alert will warn you on the need to upgrade Template: **accept**



+ The VO-DML template model may contain modules which must be upgraded.
    + Click on ***Configuration>WorkModels***
    + Select the ***Module*** tab and look at the modules to update.

+ Click on the Expand button and add all modules tagged as *incompatible*.

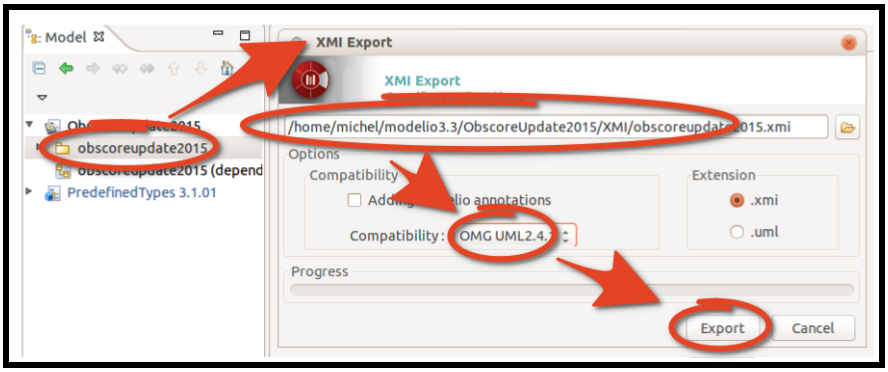


You can now work on your project as shown in the previous section.

Making former models VO-DML compliant

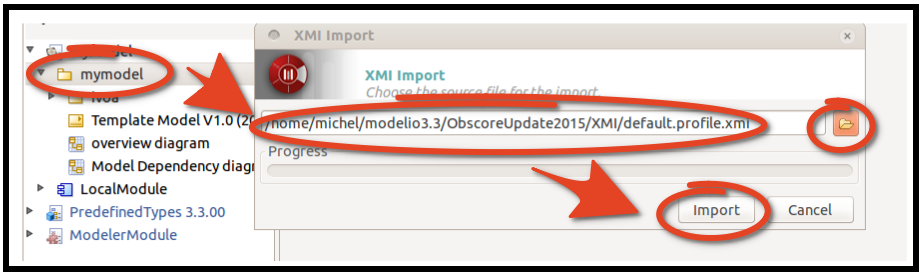
VO-DML enforces using some specific stereotypes and data types. In this section we show how to move a model developed out the VO-DML framework into a VO-DML project. This operation is only possible for models designed with Modelio.

+ Create a new model from the VO-DML template as shown before
+ Open the project containing the model to be upgraded (ObsCore in the current example)
    + Right click on it
    + Select **XMI…>export.** Take care with using the UML 2.4 compatibility

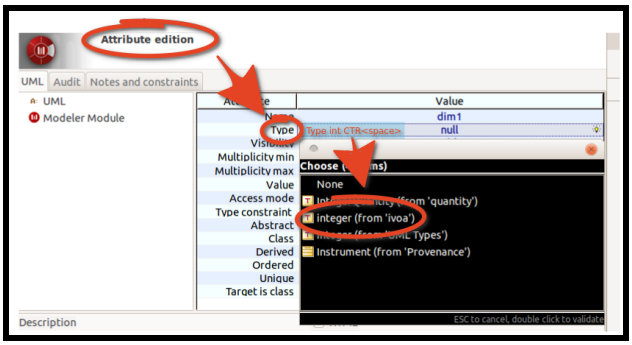


    + Save and close the project. XMI files are usually stored in the XMI project subdirectory of the workspace.
+ Open the VO-DML project
    + Right click on your new model
    + Select **XMI…>import**

+ Select the XMI file and import



+ Set IVOA types for all attributes
  + Expend the Obscore model
  + Double click on each attribute
  + Edit the Type



  + Type the first letter (i for integer e.g.) and then CTRL <SPACE> (or MAJ RET on MacOS). A popup opens and proposed all available matching types.
  + Select the ivoa type.

Once this operation is complete, your model is compliant with the VO-DML stereotypes. That does not means that it is compliant with the VO-DML scheme. It can contain prohibited features (bad formatted labels or aggregation relationships e.g.) which are checked at the Schematron validation step.