The University of Manchester
Jodrell Bank
Observatory

VO-DML 1.1 and VO-DML Tool Updates

Paul Harrison (JBO/UKSRC)
IVOA Interop Northern Autumn 2025

uk | SRC
SKAO Regional Centre United Kingdom

The University of Manchester
Jodrell Bank Observatory

uk | SRC
SKAO Regional Centre United Kingdom

Introduction

https://ivoa.github.io/vo-dml/

- VO-DML Tools introduced in previous Interop talks now quite mature.
  - Has already introduced some extensions to VO-DML that have are being included in VO-DML 1.1
  - New draft (Endorsed?) Note on VO-DML tools
- This presentation
  - Updates on the VO-DML tooling (since last Interop)
  - Structure of VO-DML 1.1 WD - invitation for comment
  - Why VO-DML tools are essential for creating a standard DM
  - We need to talk about UTypes

The University of Manchester
Jodrell Bank Observatory

MANCHESTER 1824

uk | SRC
SKAO Regional Centre United Kingdom

VO-DML Tooling Updates

- Updates v0.5.10 when last reported - now v0.5.28 - highlights include
  - TAPSchema generation
  - Support for local vocabularies
    - useful when developing a new vocabulary
  - Alternative prototype python packaging of the tools
    - still only a proof of concept as much restricted functionality compared to the gradle tooling.

TAPSchema

- VO-DML tools can now produce a TAPSchema for a VO-DML model as part of the output of `./gradlew vodmlSchema`
- Final part of the serialization triumvirate - much of the original design of VO-DML and the constraints that were put on structures in the meta-model was aimed at storing instances in relational databases.
- Because there was no obvious way to serialise a TAPSchema in a vendor neutral way, a TAPSchemaDM was created in VO-DML so that the serialization *was* “obvious”. The TAPSchema is serialised as XML.

The University of Manchester
Jodrell Bank
Observatory

uk | SRC
SKAO Regional Centre United Kingdom

VO-DML 1.1 WD

- Backwards compatible extensions (as required)
  - already tested in the deployed VO-DML Tools
- Managed via GitHub milestones with PR for each feature
- Main update for 1.1 on the 20-update-vo-dml-standard-document branch
- Current “cumulative” WD (i.e. effect of accepting all the pull requests) is also published on GitHub
- Original 1.0 REC was written in Word - the 1.1 WD is in markdown (via an automated conversion with pandoc)
  - might even produce yet another publishing option via pandoc transforming markdown to LaTeX

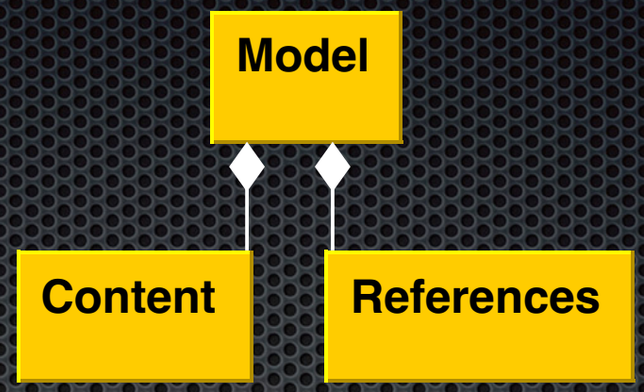
VO-DML 1.1 pull requests

- Each pull request deals with a specific topic on its own branch - can see in the pull request which branch and what has been changed and discuss there.
  - VODML-ID normative
  - Updates to the base model
  - Update the SemanticConcept to explicitly refer to IVOA Vocabularies
  - Natural Key constraint
  - Description of VO-DML Tools
  - Miscellaneous typographical updates

The University of Manchester
Jodrell Bank Observatory

uk | SRC
SKAO Regional Centre United Kingdom

Serialisation



- Appendix B in the 1.0 document describes how the model *might* be serialised
- Current tooling attempts to produce a **standard** serialisation for XML and JSON
  - based on the UML above so that a single model instance serialisation will contain both the content and references
    - references that are not otherwise “contained” are emitted in the references section
  - tooling creates both XML and JSON schema which can be used to validate model instances.
    - needs conventions (e.g. using @type in the json serialization)
    - there are design choices to be made to create the schema too.
  - TAP schema too as explained earlier
- Proposal is to rewrite Appendix B to make clear that XML, JSON and TAPschema serialisations are intended for interoperability, and thus “standard”.
- Note that this form of serialisation is more suitable for writing REST web service interfaces for the models than MIVOT - however, MIVOT has other use cases and is thus complementary and not a “competitor”.

```xml
<ser:myModelModel
        xmlns:ser="http://ivoa.net/vodml/sample/serialization"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <refs>
        <refa _id="MyModel-Refa_1000">
            <val>a value</val>
        </refa>
        <refb>
            <name>naturalkey</name>
            <val>another val</val>
        </refb>
    </refs>
    <someContent>
        <zval>some</zval>
        <zval>z</zval>
        <zval>values</zval>
        <con xsi:type="ser:Dcont" >
            <bname>dval</bname>
            <dval>a D</dval>
        </con>
        <con xsi:type="ser:Econt" >
            <bname>eval</bname>
            <evalue>cube</evalue>
        </con>
        <ref1>MyModel-Refa_1000</ref1>
        <ref2>naturalkey</ref2>
    </someContent>
</ser:myModelModel>
```

```json
"MyModelModel" : {
  "refs" : {
    "MyModel:Refa" : [ {
      "_id" : 1000,
      "val" : "a value"
    } ],
    "MyModel:Refb" : [ {
      "name" : "naturalkey",
      "val" : "another val"
    } ]
  },
  "content" : [ {
    "@type" : "MyModel:SomeContent",
    "_id" : 0,
    "zval" : [ "some", "z", "values" ],
    "con" : [ {
      "@type" : "MyModel:Dcont",
      "_id" : 0,
      "bname" : "dval",
      "dval" : "a D"
    }, {
      "@type" : "MyModel:Econt",
      "_id" : 0,
      "bname" : "eval",
      "evalue" : "cube"
    } ],
    "ref1" : 1000,
    "ref2" : "naturalkey"
  } ]
}
```

Generated Key

"natural" Key

typing

needs conventions for JSON

references to above

The University of Manchester
Jodrell Bank Observatory

uk | SRC
SKAO Regional Centre United Kingdom

Standardization

needed for interoperability

VO-DML Model + Binding

Only available in VO-DML Tools

- XML Schema
- JSON Schema
- TAPSchema

- We can write a formal document that defines all the binding rules, but it will take a while - they are all encoded in the VO-DML Tools. Anyway the necessary binding constructs are still a work in progress (and are documented on-line)
- Think of VO-DML Tools in the same way as ivoatex - it is possible to obey all the standards documentation rules (*except that they have not all been documented*) without it, but using it produces more more uniformity, and is less work.

UTypes

- Link a FIELD or PARAM with a data model element (VOTable)
- Still not formally defined!
  - usage note (2013) illustrates inconsistent practice
    - Norman Gray's analysis
    - attempt at formal definition stalled at WD 0.7 (2012)
    - VO-DML 1.1 only mentions UType once in main text
- *However* VO-DML tools offer mechanised way to define UTypes - Indeed it is needed for the TAPSchema
  - VO-DML 1.0 says manually specifying should only be used for compatibility with existing UTypes

UTypes Standardization

- Open Issues (there are probably more!)
  - Done as part of VO-DML document or stand-alone?
  - Should it have parts (like UCDs)?
    - e.g. a model has luminosity (say photom:std.luminosity)- which is a DataType (measure:measuredValue) with three attributes (say, Value, Error and Unit) - then to mark up say the “luminosity_error” column of a database with a UType it is desirable to have both pieces of information represented.
      - photom:std.luminosity;measure:measuredValue.Error
        - something like this is required, especially if vodml-ID is allowed to be arbitrary
  - It is unlikely to be compatible with all existing use.

Conclusion

- VO-DML 1.1 REC is still a work in progress
  - though all changes have been thoroughly road-tested
    - code-first rather than document-first.
- VO-DML tools should be used for developing data models
  - provides the concrete extra conventions that are needed e.g. to “fill-in” the missing information needed to fully interpret json.
- UTypes - still more work required….