The University of Manchester
Jodrell Bank
Observatory

VO-DML 1.1 Candidate Extensions

Paul Harrison (JBO)
IVOA Interop Autumn 2024

Introduction

- VO-DML Tooling update introduced in previous Interop talks now quite mature.
  - Refined by the needs of ProposalDM, the generated code for which is used as the serialisation basis for Polaris, a proposal submission toolkit (In fact the ProposalDM is the internal data model of Polaris).
  - Has already introduced some extensions to VO-DML that have not yet been included in the standard document.
- This talk
  - Updates on the VO-DML tooling (since last Interop)
  - Suggestions for VO-DML 1.1 WD - invitation for comment
  - Thoughts on the using VO-DML in the P3T process.

VO-DML Tooling

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

- Tools to create models and derive "products" from them
  - Based on VO-URP by Lemson and Bourgès
  - most of the business logic is in XSLT 3.0 (using functions)
    - packaged as a gradle plugin
  - If you don't like writing models in XML then there is VOSDL - 10 years old!
    - language agnostic
    - much more 'human-readable'

ProposalDM: 451 lines VODSL $\Longrightarrow$ 2158 lines VO-DML

- Products
  - XML & JSON schema
  - Various forms of documentation
  - Java and Python code to instantiate models and be an ORM for RDBs

The University of Manchester
Jodrell Bank
Observatory

MANCHESTER
1824

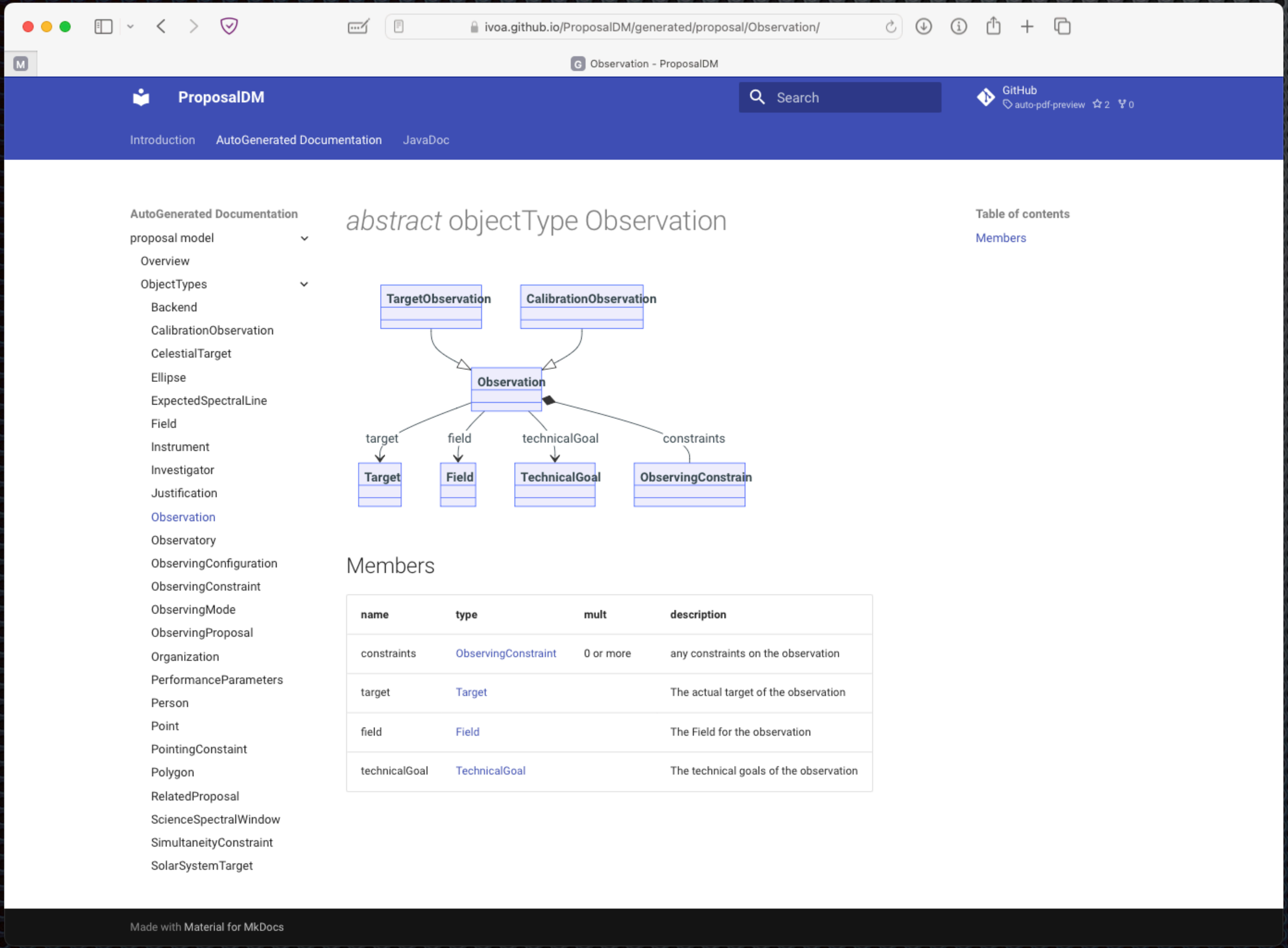
OPTICON RadioNet Pilot

VO-DML Tooling Updates

- Updates since a last Interop, v0.5.1 when last reported - now v0.5.10
  - Added support for validation against IVOA vocabularies (Semantic Concept in VO-DML)
  - Added support (in Java generated code) for serialising attributes with multiplicity > 1 of primitive types as colon separated string database column
  - Improved generated model documentation
  - Improved contained references support in Java.

Model Site Documentation

e.g. ProposalDM


ivoa.github.io/ProposalDM/generated/proposal/Observation/

Observation - ProposalDM

ProposalDM

Search

GitHub
auto-pdf-preview ☆2 ⑂0

Introduction AutoGenerated Documentation JavaDoc

AutoGenerated Documentation
- proposal model
  - Overview
  - ObjectTypes
    - Backend
    - CalibrationObservation
    - CelestialTarget
    - Ellipse
    - ExpectedSpectralLine
    - Field
    - Instrument
    - Investigator
    - Justification
    - Observation
    - Observatory
    - ObservingConfiguration
    - ObservingConstraint
    - ObservingMode
    - ObservingProposal
    - Organization
    - PerformanceParameters
    - Person
    - Point
    - PointingConstaint
    - Polygon
    - RelatedProposal
    - ScienceSpectralWindow
    - SimultaneityConstraint
    - SolarSystemTarget

abstract objectType Observation

Members

name	type	mult	description
constraints	ObservingConstraint	0 or more	any constraints on the observation
target	Target		The actual target of the observation
field	Field		The Field for the observation
technicalGoal	TechnicalGoal		The technical goals of the observation

Table of contents
Members

Made with Material for MkDocs

- individual pages for each model element
- neighbourhood diagram
- uses mkdocs

VO-DML 1.1 WD

- Backwards compatible extensions (as required)
  - already tested in the deployed tools gradle plugin
- Managed via GitHub milestones with PR for each feature
- Main update for 1.1 on the 20-update-vo-dml-standard-document branch
- 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

VODML-ID syntax made normative

- In the VO-DML meta-model XML schema VODML-ID is simply a string, rather than an ID/IDREF structure, so having arbitrary form would be potentially problematic as there would be no validation via the schema - although the standard says that they should be unique.
  - Data models that were created via the original tooling have the (proposed) normative form anyway as the UML to VO-DML conversion generated such elements.
- Originally the textual syntax of the VODML-ID for each model element was only specified in an appendix - moved to main body to <u>become normative</u>
  - essentially the VODML-ID is derived from the location in the model
- Tooling now checks that VODML-ID is correct via a schematron rule, however
  - tooling never “reads” that element value - it always “calculates” it, so the element could be removed from VO-DML schema entirely.

VO-DML extension - Natural Keys

- Object Relational Mapping uses surrogate keys widely - however, in the model it is sometimes better to use a “natural key” i.e. an existing attribute - often the case for the target of “references”.

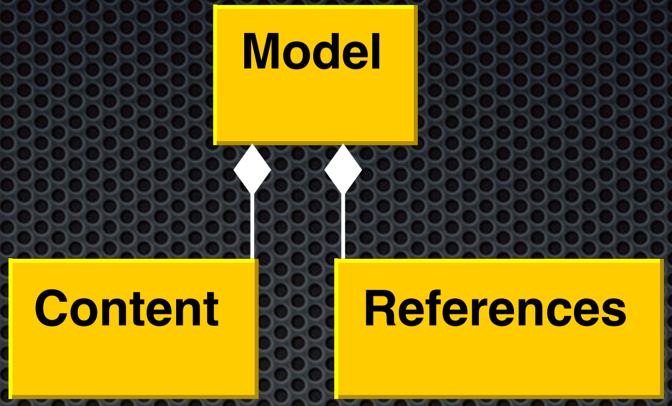
```
<xsd:complexType name="NaturalKey">
    <xsd:annotation>
      <xsd:documentation>
        This constraint is used to indicate that an attribute is a natural key for its owning ObjectType, meaning that the
        attribute value should be globally unique. This may be applied multiple times to indicate that only a composition
        of several attributes make the globally unique key.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="Constraint">
        <xsd:sequence>
          <xsd:element name="Position" type="xsd:positiveInteger">
                <xsd:annotation>
                  <xsd:documentation>In the case where multiple attribute values make up the natural key, this
                  value indicates the ordinal number of this particular key in the compound key.</xsd:documentation>
                </xsd:annotation>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

VO-DML Metamodel XML Schema updates

- Aforementioned natural key extension
- make <name> and <documentationURL> optional (and deprecated) in the <import> as they merely repeat information that is in the imported document
- replace grouping of Attributes, Composition and References with xsd:choice so the the definitions can be in a “natural order”
- the suggestions above have already happened on the main branch - non-breaking - following the XML schema versioning endorsed note.
- Should VO-DML 1.1 metamodel have its own namespace? or an attribute to mark that it is the 1.1 version.

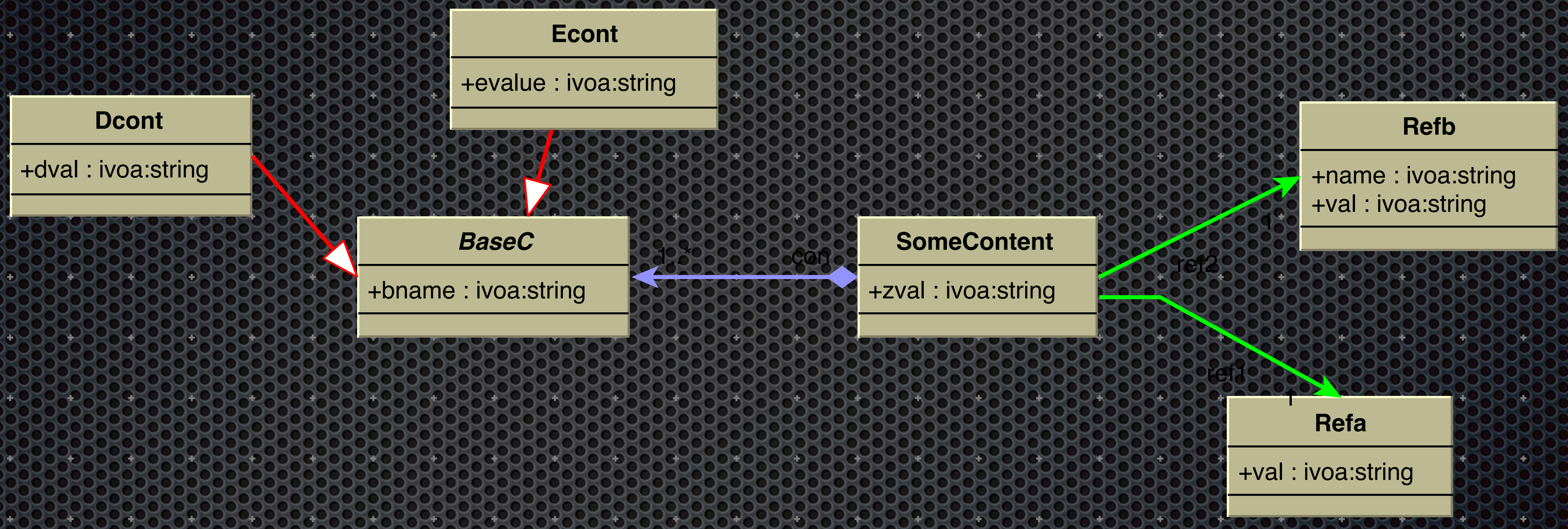
The University of Manchester
Jodrell Bank
Observatory

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” (see later) are emitted in the references section
  - tooling creates both XML and JSON schema which can be used to validate model instances.
- Proposal is to rewrite Appendix B to make clear that new serialisation is 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”.

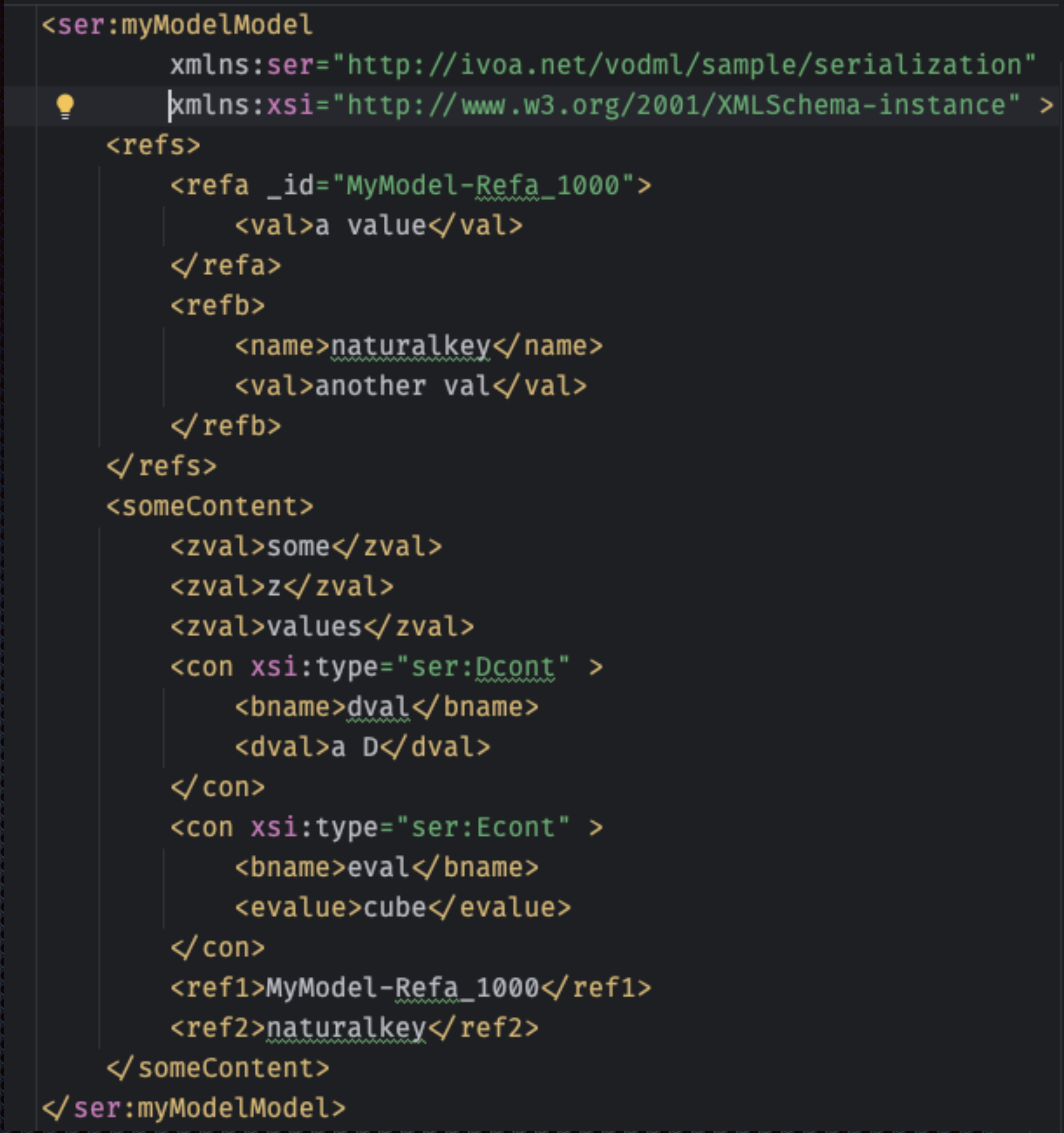
Serialization 2 - Example Model



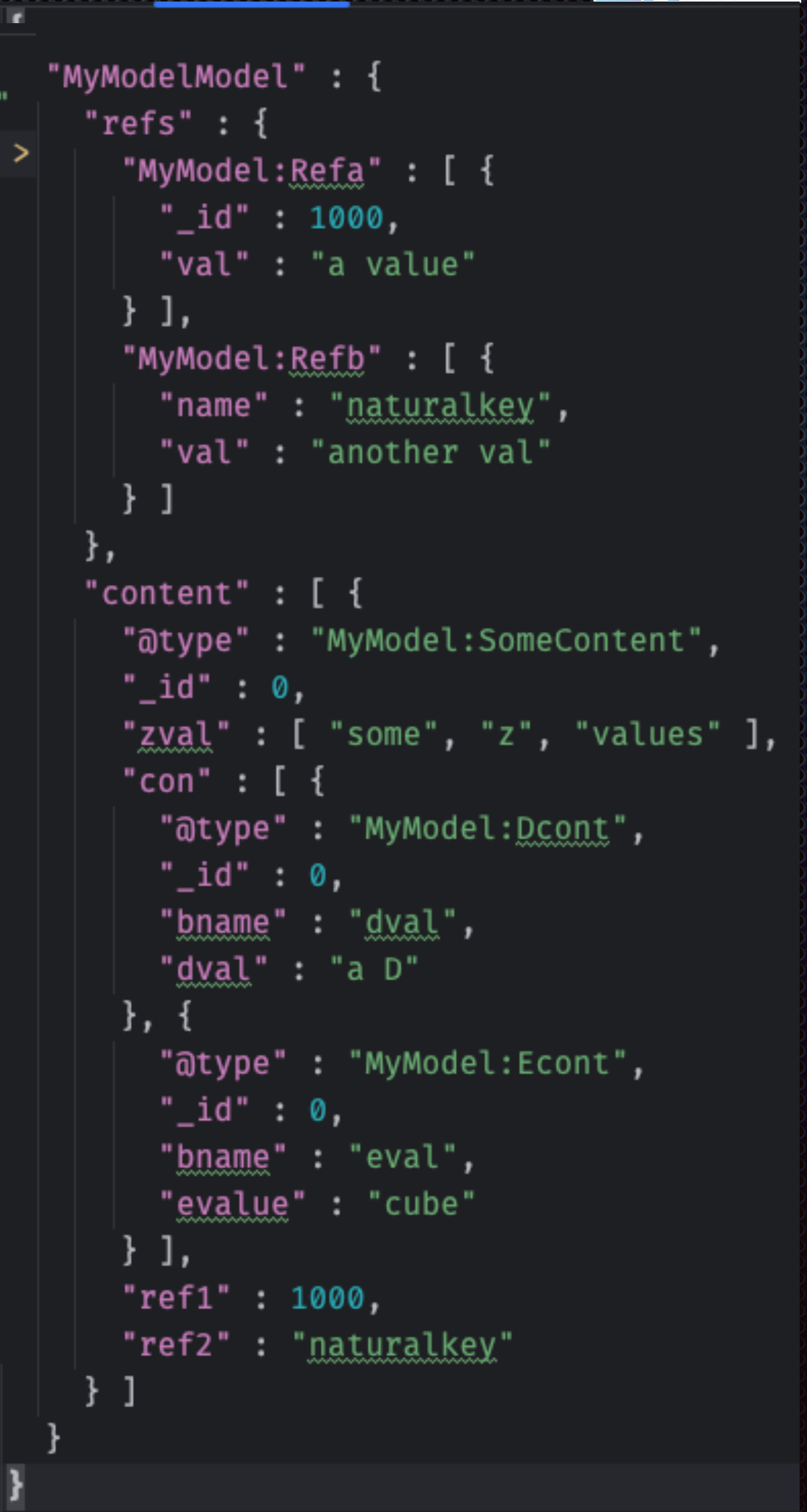
- https://ivoa.github.io/vo-dml/Serialization/
- note that tooling includes automated round-trip serialisation unit tests against generated schema.

Serialization 3 - Comparison

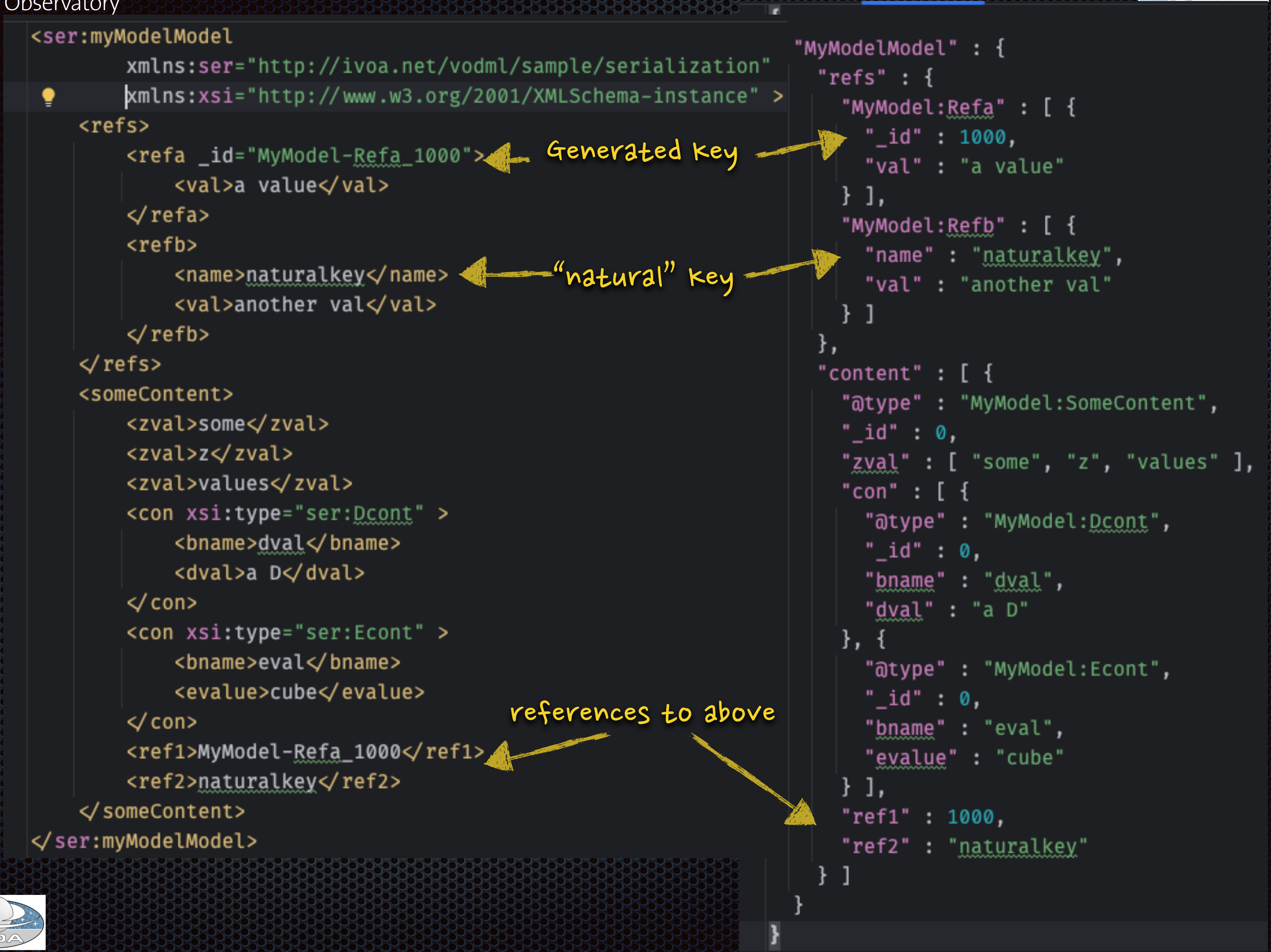
XML vs JSON



```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>
```



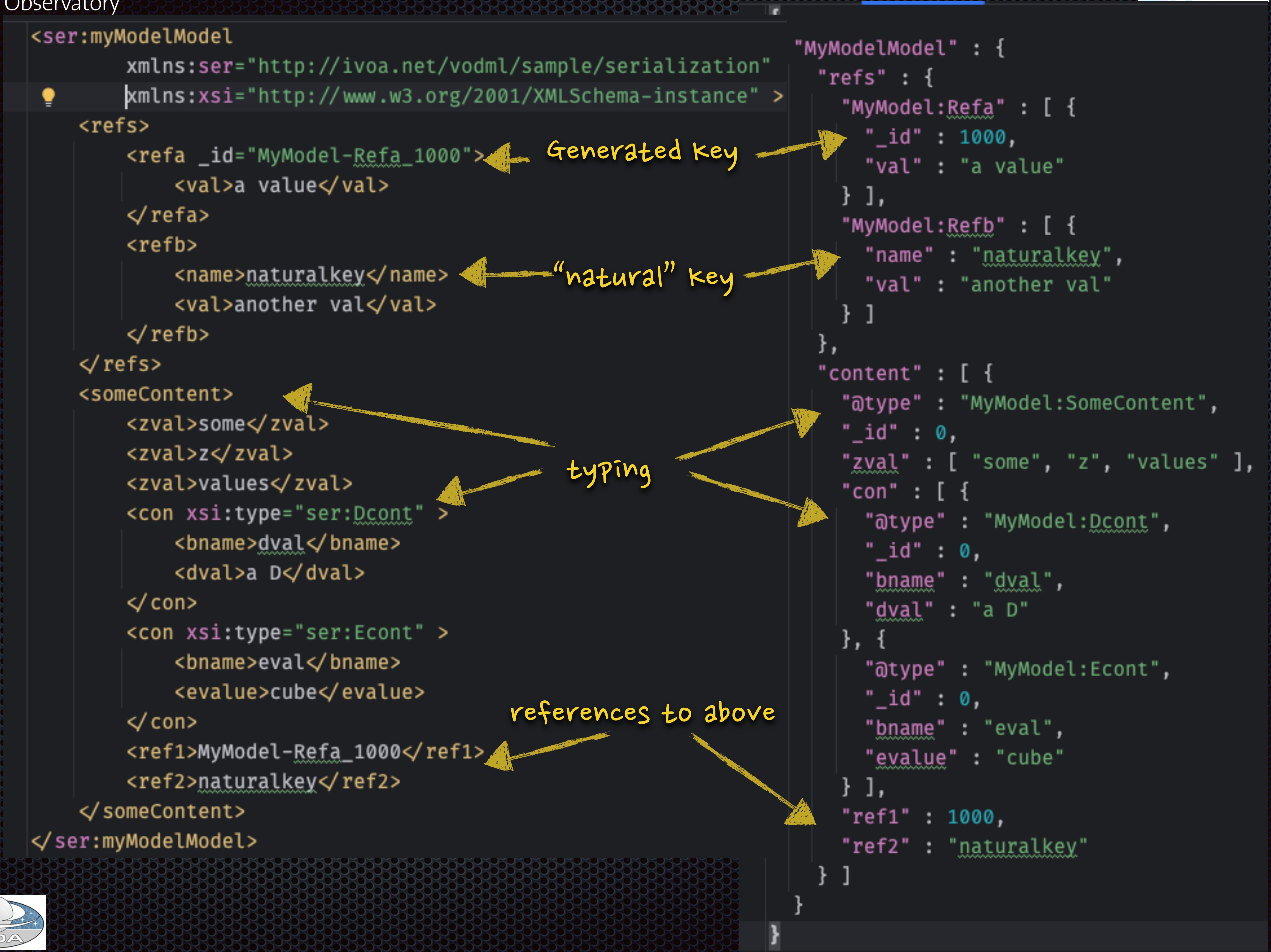
```json
"MyModelModel" : {
  "refs" : {
    "MyModel:Refa" : [ {
      "_id" : 1000,
      "val" : "a value"
    } ],
    "MyModel:Refb" : [ {
      "name" : "naturalkey",
      "val" : "another val"
    } ]
  },
  "content" : [ {
    "@type" : "MyModel:SomeContent",
    "_id" : 0,
    "zval" : [ "some", "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"
  } ]
}
}
```



```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"
    } ]
  }
}
```

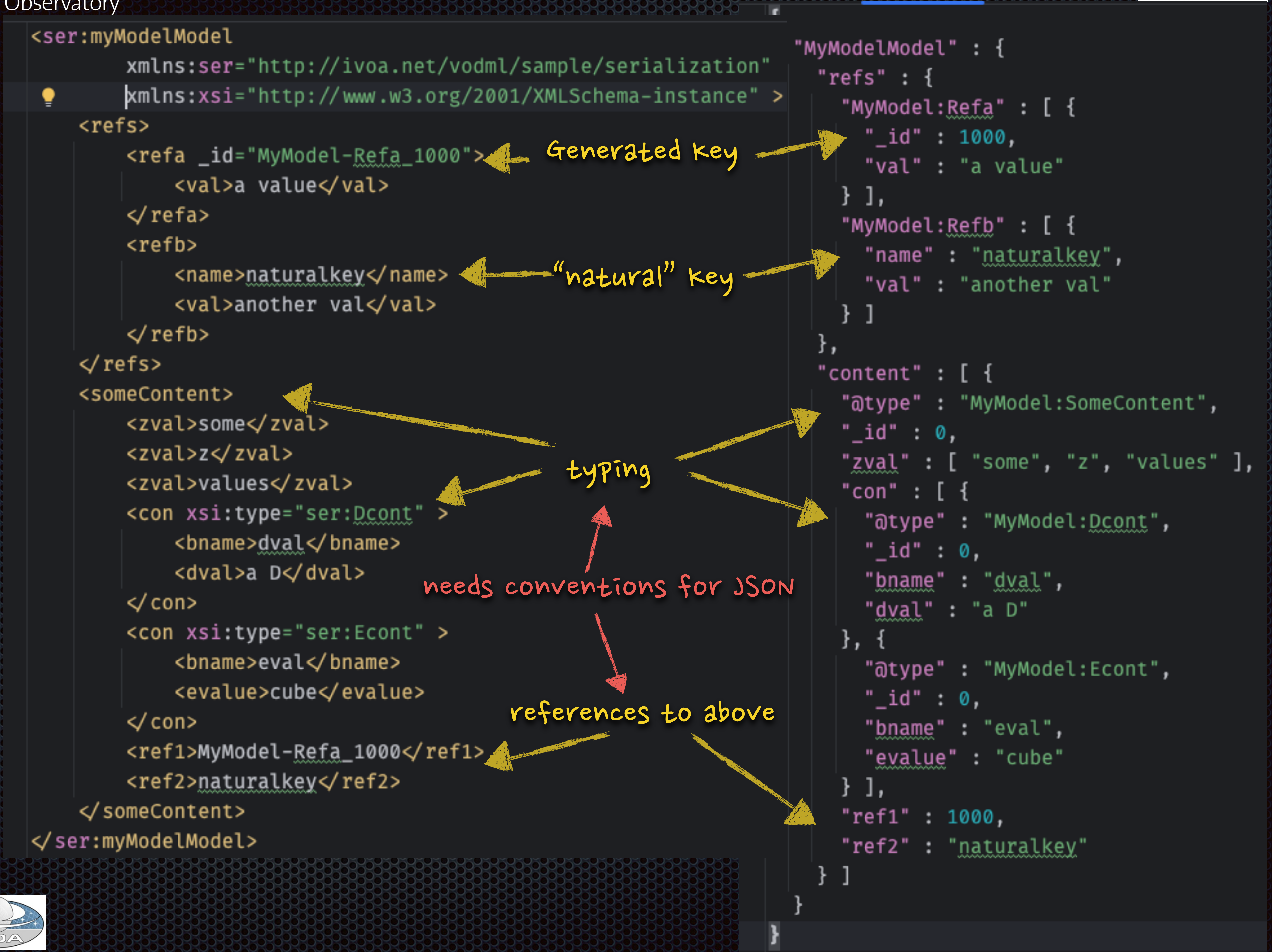
The University of Manchester
Jodrell Bank Observatory



```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"
  } ]
}
}
```

The University of Manchester
Jodrell Bank
Observatory

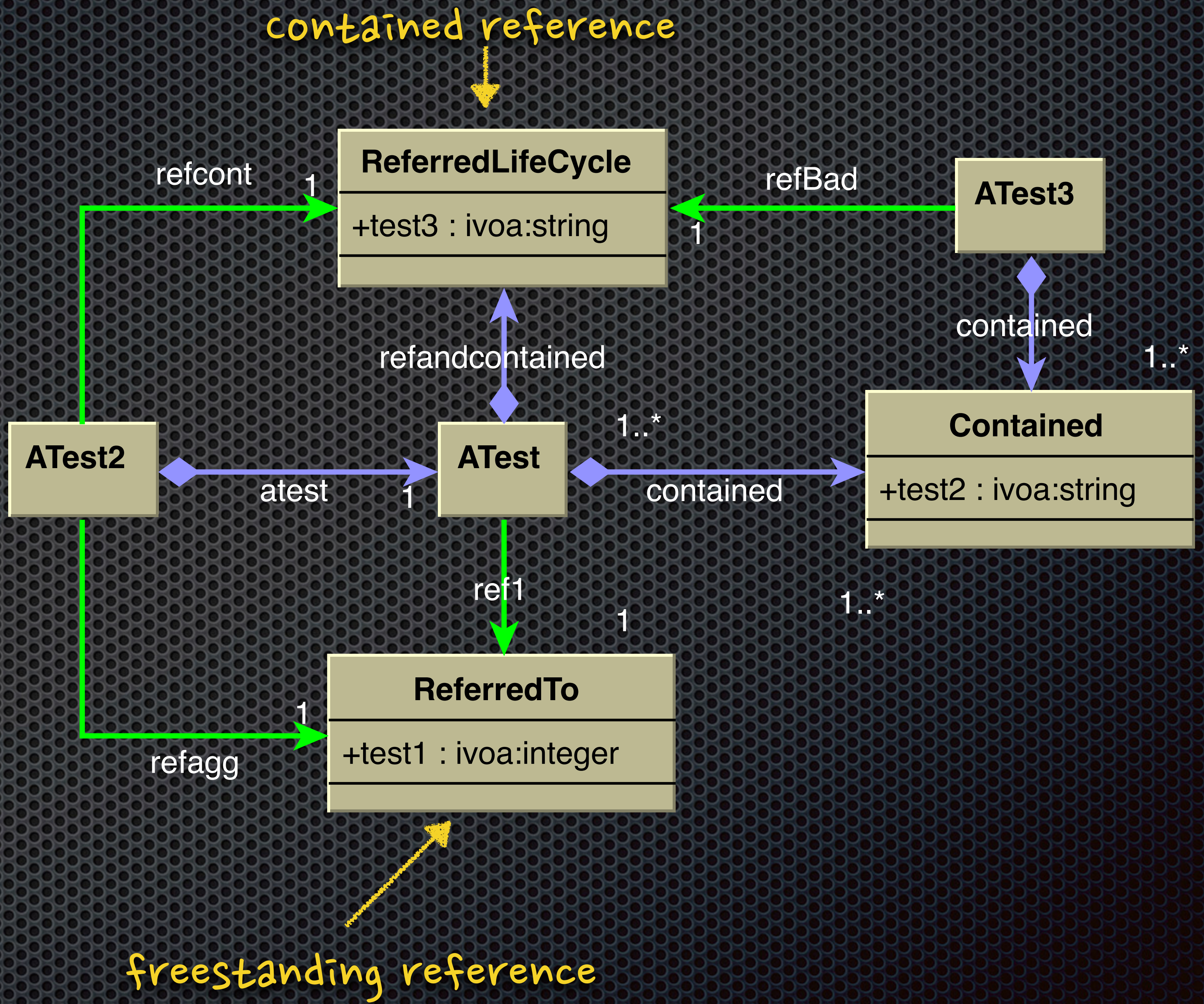


```xml
<ser:myModelModel
        xmlns:ser="http://ivoa.net/vodml/sample/serialization"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
    <refs>
        <refa _id="MyModel-Refa_1000">
            <val>a value</val>
        </refa>
        <refb>
            <name>naturalkey</name>
            <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"
  } ]
}
}
```

Reference Lifecycle/Containment

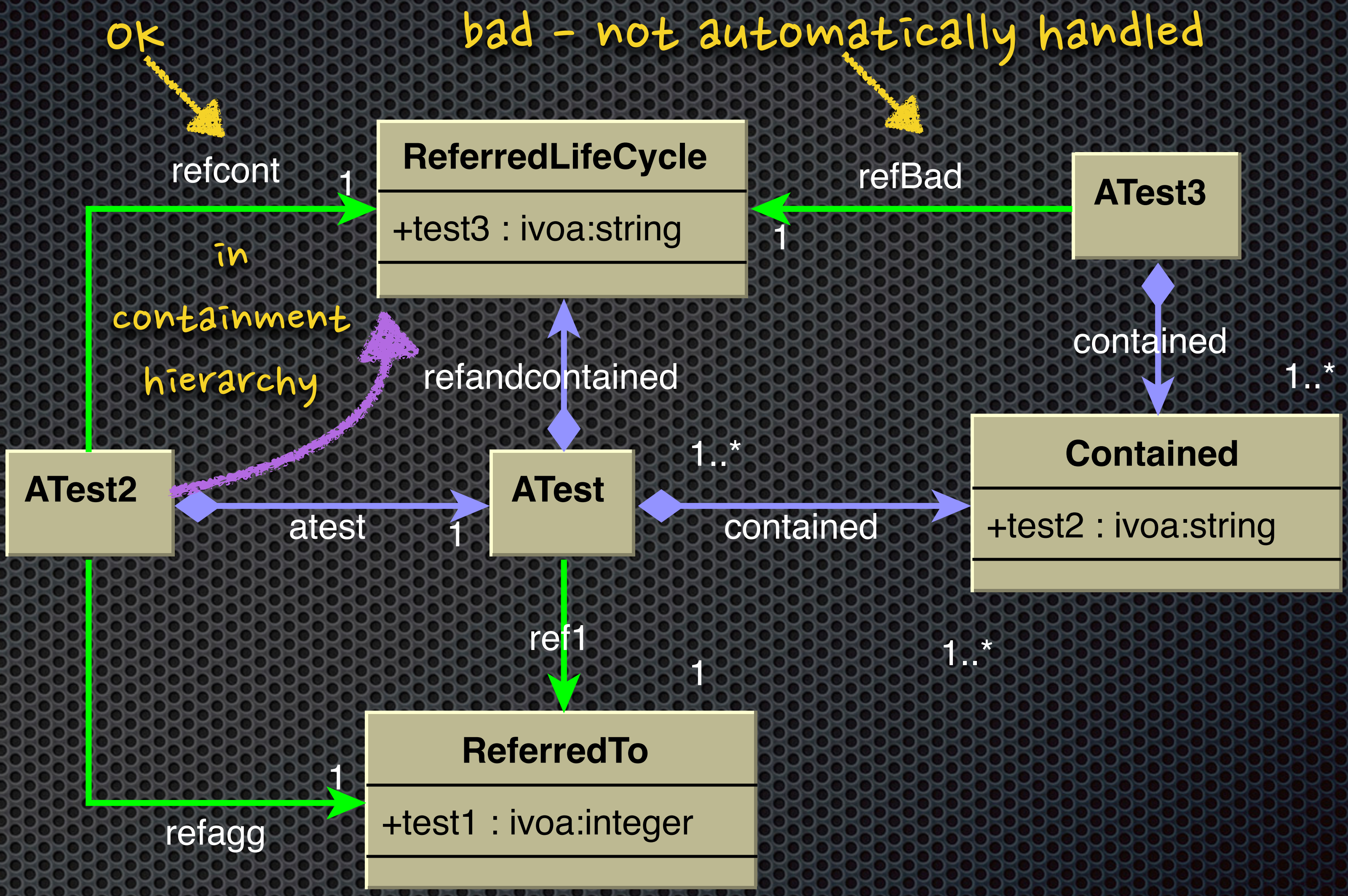
- Original tooling/std assumed that references "freestanding" - i.e. lifecycles independent of any particular model instance
- In latest tooling references can be "contained" i.e. referenced element can exist as a composition within some parent.



The University of Manchester
Jodrell Bank
Observatory

Reference Lifecycle/Containment 2

- tooling will generate Java code that will deal properly with contained references
- schematron rules warn of "dangerous" contained reference use

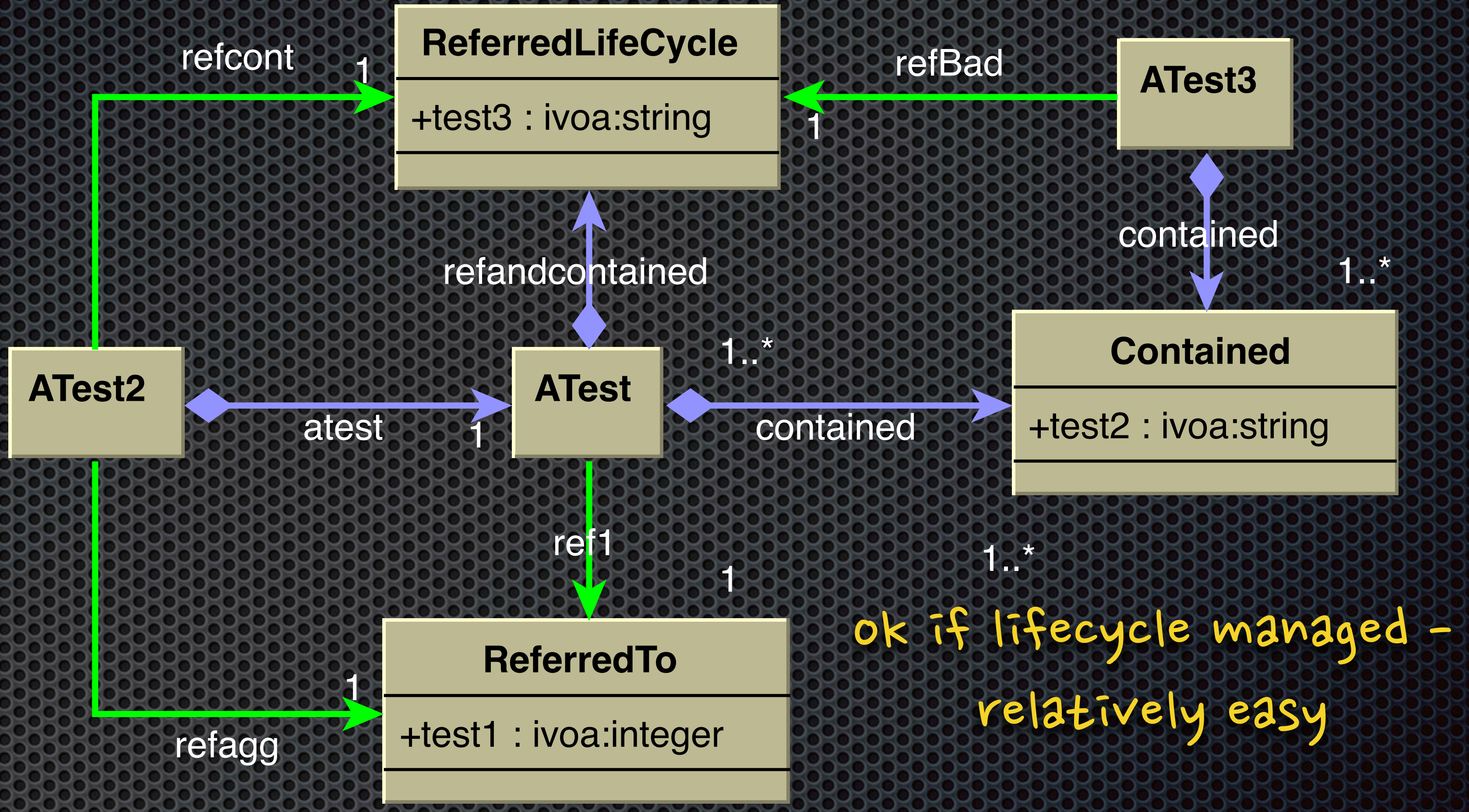


```
failed-assert /Q{http://www.ivoa.net/xml/VODML/v1}model[1]/Q{}
objectType[6]/Q{}reference[1]


Reference lifecycleTest:ReferredLifeCycle used in
ATest3.refBad is already use in unrelated composition ATest
which has lifecycle implications (i.e. the reference could
disappear unless code is aware of relationship)
```

Reference Lifecycle/Containment 3

- Schematron complains with “unique composition rule”
- however, this is just a warning
- Wording in Standard probably OK



```
failed-assert /Q{http://www.ivoa.net/xml/VODML/v1}
model[1]/Q{}objectType[6]/Q{}composition[1]/Q{}
datatype[1]/Q{}vodml-ref[1]


objecttype lifecycleTest:Contained is used more than
once, as target of composition relation. In this case for
containing objectType lifecycleTest:ATest3


        ** (this message will repeat itself 2 times!,
once for each different container) **
```

IVOA Base Model Additions

- This is being done on the base_update branch

- Following on from the serialisation and reference containment discussions it is useful to be able to mark in a model where the intention is to point to an external entity (which cannot be done with references as they are internal)

```
primitive intIdentifier -> integer "an integer identifier that can be used as a key for lookup of an entity that is *outside this datamodel*"
primitive stringIdentifier -> string "a string identifier that can be used as a key for lookup of an entity that is *outside this datamodel*"
primitive ivorn -> anyURI "an identifier that can be used as a key to look up in an IVOA registry - see https://www.ivoa.net/documents/IVOAIdentifiers/"
```

- also add?
  - a Period (cf DateTime) - reasonably obvious
  - Shape - still needs clarification

- Base model, so perhaps have to be conservative….

VO-DML 1.2 and beyond

- Lots of potential ideas/improvements, but have left them out of 1.1 in the hope of speeding up approval of this document.
  - specifying UCDs
    - could then automatically create TAP schema/services
  - concept of Choice/OneOf
  - some specific simple constraints
    - e.g greaterThan

VO-DML 1.2 and beyond

- Lots of potential ideas/improvements, but have left them out of 1.1 in the hope of speeding up approval of this document.
  - specifying UCDs
    should bump to 1.1 - only functional
    reason why ruben felis need exist
    - could then automatically create TAP schema/services
  - concept of Choice/OneOf
  - some specific simple constraints
    - e.g greaterThan

Distribution/Publishing models

- It would be nice to be able to publish the generated code libraries
  - difficulty with using Maven Central, PyPI etc. is authentication in CI
  - Could use GitHub Packages
    - there do seem to be some quirks/limited functionality
    - No Python….
  - Could run a Sonatype Nexus repository server on IVOA web site
    - could put credentials into GitHub secret for CI publishing
- Also publish the "site-style" documentation
  - more than just a single file.

Importance of VO-DML

- Provides rigour in the DM design
    - Created around 10 years ago as a response to approximately 10 years of trying to create interoperable data models ***without*** a machine-readable expression of the data model.
    - Allows real re-use (not just “my diagram looks like your diagram”)
        - Machine readable single source of truth
        - Makes factoring out common parts possible
- Provides a framework for validating instances.
    - serializations in different format need conventions to be interoperable.
- Can be used to generate the “schema” part of OpenAPI in a uniform way
    - The exact form of the generated serialization code is fixed cf. if you use a 3rd party OpenAPI generator.
    - help deal with the vagaries of the $ref rules modularity