VO-URP:
on data modeling, UTYPEs and more

Gerard Lemson

Laurent Bourges

Aim 1: overview of VO-URP

- Request Omar
- VO-URP = Virtual Observatory UML Representation Pipeline
- Used in SimDM to facilitate DMing effort
- Defines a syntax for UTYPE-s that is used in UTYPE doc (Mireille)

Aim 2: discuss UTYPE-s

- Not: What are they and what are they for?
  - See also Norman‘s questions http://nxg.me.uk/note/2009/utype-questions/
- Yes: **Producing UTYPE-s**
- Not much: Using UTYPE-s
  - *Tiger team*

Context: data modeling in IVOA

DaM	Photometry DM	1.0
	Simulation Data Model	1.0
	Space-Time Coordinate Metadata for the Virtual Observatory (STC)	1.33
	Data Model for Astronomical DataSet Characterisation	1.13
	Simple Spectral Lines Data Model	1.0
	IVOA Spectrum Data Model	1.1
	Observation Data Model Core Components and its Implementation in the Table Access Protocol	1.0

  - Some in pipe-line
  - Also: VOResource family of models
- How do we make them interoperable?
  - Standard answer seems to be: "UTYPE-s !!”
  - Mechanism is still missing.
  - Very incomplete interoperability.

Instantiation

- What does one do with models?
  - Create instances
  - Needs a serialisation/representation mechanism
- What we do with instances depends on representation
  - XML: write them, send them, upload them, transform them (XSLT)
  - RDB: insert them in DB, query DB

Transformation

- Potentially we want to do more with a model:
  - Mapping: Create alternative representations of the model, as faithfully as possible (Java, C#,…)
  - Transformation: in queries or views, produces *ad hoc* model
  - Use: import in other model, either original of transformed version
- Generally:
  - Produce a view which is a partial representation of a transformed alternative version of some *ad hoc* model that used part of original model in some representation.

Problem

How do we keep track of source and meaning of information

Questions on UTYPE-s

- How can UTYPE-s help interoperability between models/representations/views?
  - In what contexts?
- How far can one use them; fields only, classes, relations?
- Meaning of query results can not be represented by simple mappings in general.
- Usefulness of UTYPEs depends very much on the model
  - E.g. in SimDM UFIs would be more useful.

Data models in IVOA

MUST/SHOULD provide

- UML
- XSD
- UTYPE-s
- Documentation

No formal constraints on any of these representations!

IVOA models rather non-uniform.

- Some UML
  - mainly as illustration, not rigorous
- Most XSD (incl. VOResource etc)
  - but with different “style”
  - in general not equivalent to UML
- Some RDB (ObsCore, SimDM)
- Some(most?) UTYPE-s
  - Separately defined

Interoperability of data models

- Non-uniformity of modeling language makes this a technically *and conceptually* non-trivial process
  - Queries, implying transformations of models, complicate matters even more.
- Idea seems to be that UTYPE-s will help
  - I doubt that very much, unless for trivial use cases.
- It is not a natural solution in any case.
- Natural would be if different data models would be expressed in common language
  - Using common **meta-model**

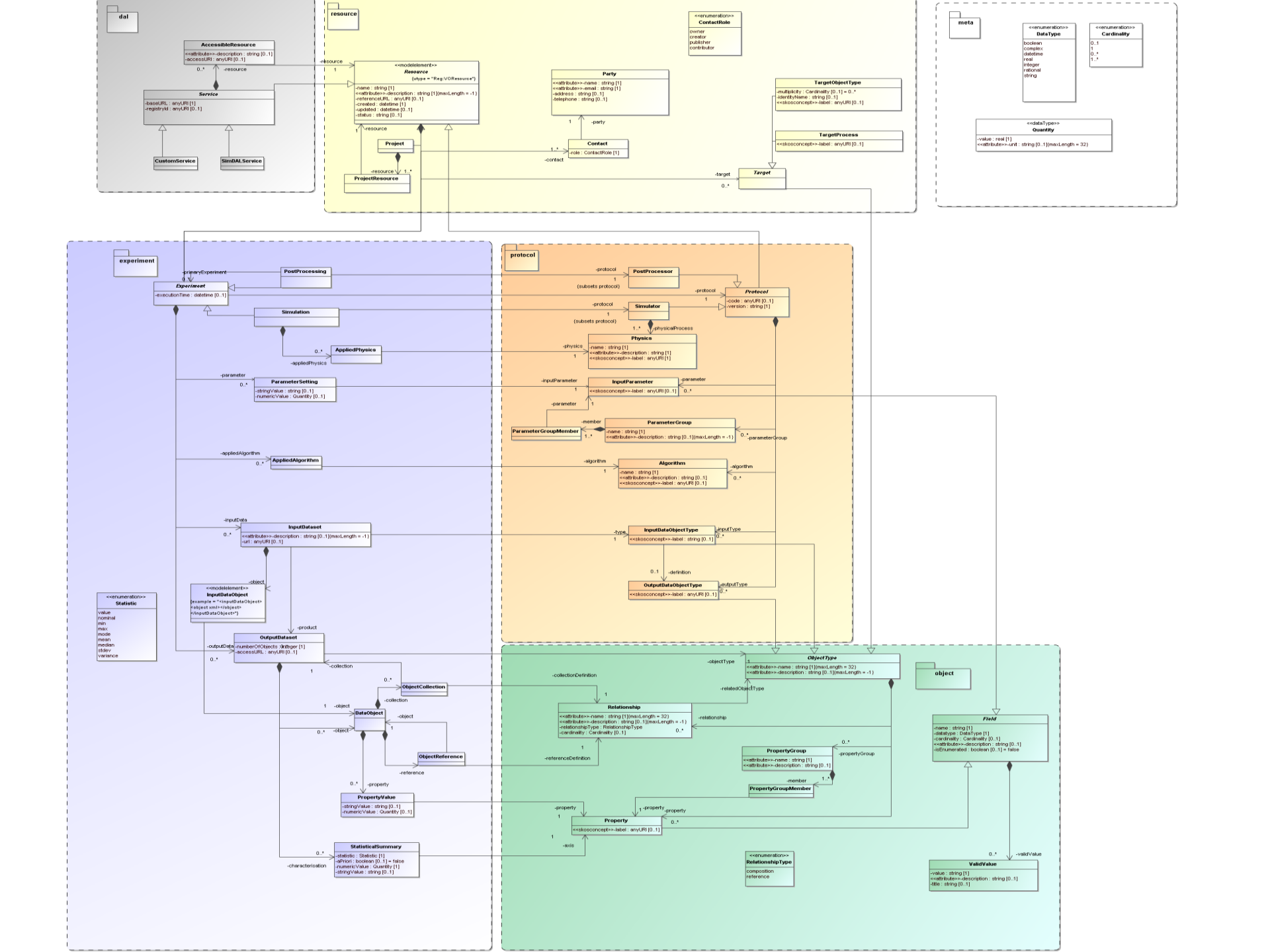
An example what this could look like:

VO-URP

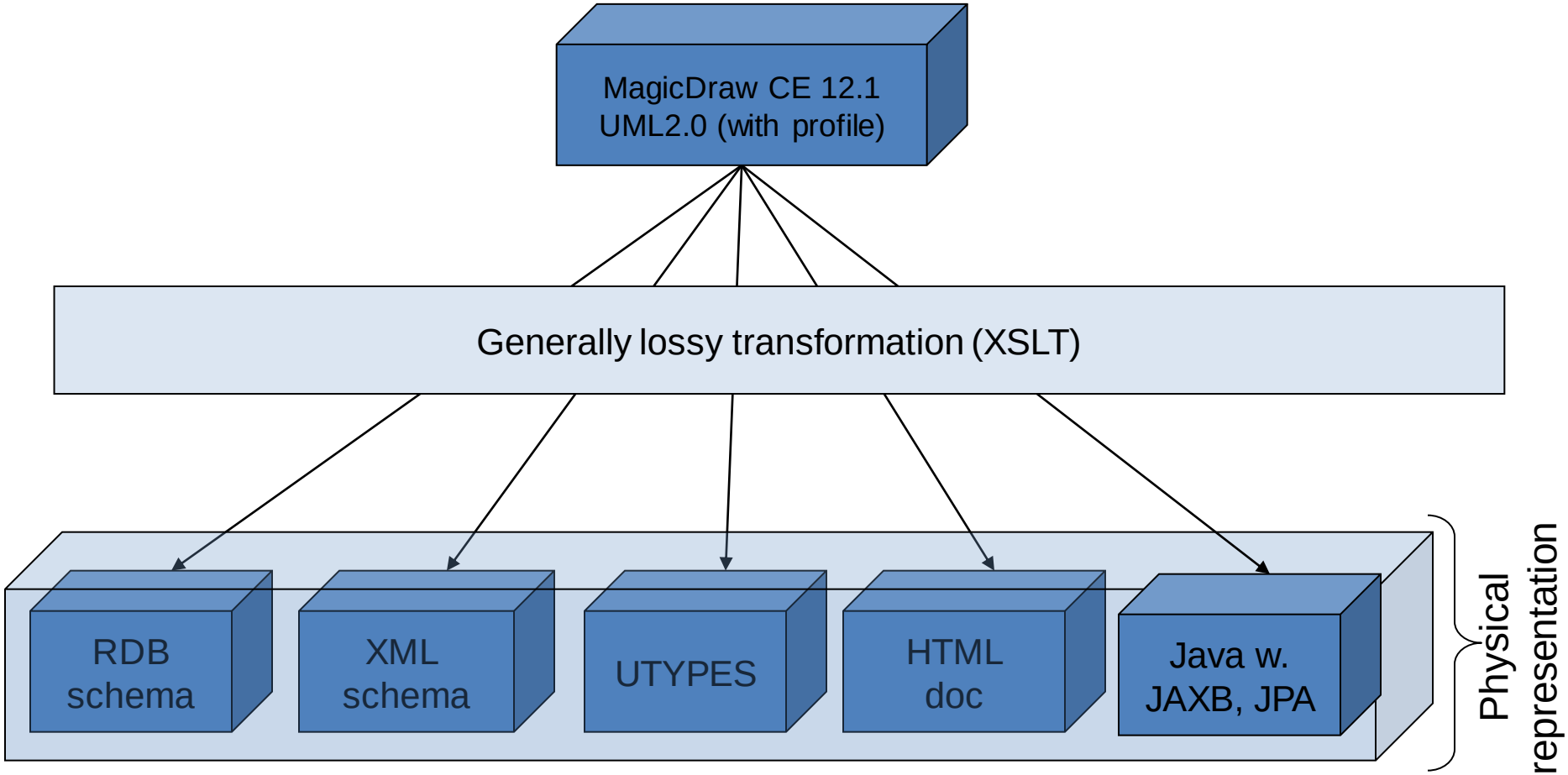
http://vo-urp.googlecode.com

Spin-off from Simulation Database (SimDB)

- Use UML for complete data model
  - Including documentation text
- Do NOT write other representations by hand
  - Automate generation using XSLT
- From XMI ⇨
  - XSD + template XML docs
  - HTML+UTYPE
  - DDL+TAP_SCHEMA
  - Java +JAXB/JPA


dal
AccessibleResource
<<attribute>>-description : string [0..1]
-accessURI : anyURI [0..1]
-resource
0..*
Service
-baseURL : anyURI [1]
-registryId : anyURI [0..1]
CustomService
SimDALService
resource
-resource
1
<<modelelement>>
Resource
{utype = "Reg:VOResource"}
-name : string [1]
<<attribute>>-description : string [1][maxLength = -1]
-referenceURL : anyURI [0..1]
-created : datetime [1]
-updated : datetime [0..1]
-status : string [0..1]
-resource
1
Project
ProjectResource
-resource
1..*
<<enumeration>>
ContactRole
owner
creator
publisher
contributor
Party
<<attribute>>-name : string [1]
<<attribute>>-email : string [1]
-address : string [0..1]
-telephone : string [0..1]
1
-party
Contact
-role : ContactRole [1]
1..*
-contact
TargetObjectType
-multiplicity : Cardinality [0..1] = 0..*
-identifyName : string [0..1]
<<skosconcept>>-label : anyURI [0..1]
TargetProcess
<<skosconcept>>-label : anyURI [0..1]
-target
Target
0..*
meta
<<enumeration>>
DataType
boolean
complex
datetime
real
integer
rational
string
<<enumeration>>
Cardinality
0..1
1
0..*
1..*
<<dataType>>
Quantity
-value : real [1]
<<attribute>>-unit : string [0..1][maxLength = 32]
experiment
-primaryExperiment
PostProcessing
Experiment
-executionTime : datetime [0..1]
Simulation
0..*
AppliedPhysics
-appliedPhysics
-parameter
0..*
ParameterSetting
-stringValue : string [0..1]
-numericValue : Quantity [0..1]
-appliedAlgorithm
0..*
AppliedAlgorithm
-inputData
0..*
InputDataset
<<attribute>>-description : string [0..1][maxLength = -1]
-url : anyURI [0..1]
-object
<<modelelement>>
InputDataObject
{example = "<inputDataObject>
<object xmlI='</object>
</inputDataObject>"}
<<enumeration>>
Statistic
value
nominal
min
max
mode
mean
median
stdev
variance
-product
OutputDataset
-outputData
-numberOfObjects : 0nBiger [1]
-accessURL : anyURI [0..1]
0..*
1
-collection
0..*
ObjectCollection
-collection
1
-object
DataObject
-object
0..*
-object
1
ObjectReference
-reference
0..*
-property
PropertyValue
-stringValue : string [0..1]
-numericValue : Quantity [0..1]
0..*
StatisticalSummary
-statistic : Statistic [1]
-aPriori : boolean [0..1] = false
-numericValue : Quantity [1]
-stringValue : string [0..1]
-characterisation
protocol
-protocol
1
PostProcessor
(subsets protocol)
-protocol
1
Protocol
-code : anyURI [0..1]
-version : string [1]
-protocol
1
Simulator
(subsets protocol)
1..*
-physicalProcess
Physics
-physics
1
-name : string [1]
<<attribute>>-description : string [1]
<<skosconcept>>-label : anyURI [1]
-inputParameter
1
InputParameter
<<skosconcept>>-label : anyURI [0..1]
-parameter
0..*
-parameter
1
-member
ParameterGroupMember
1..*
ParameterGroup
-name : string [1]
<<attribute>>-description : string [0..1][maxLength = -1]
0..*
-parameterGroup
-algorithm
1
Algorithm
-name : string [1]
<<attribute>>-description : string [0..1]
<<skosconcept>>-label : anyURI [0..1]
-algorithm
0..*
-type
1
InputDataObjectType
<<skosconcept>>-label : string [0..1]
-inputType
0..*
0..1
-definition
OutputDataObjectType
<<skosconcept>>-label : anyURI [0..1]
-outputType
0..*
-objectType
1
ObjectType
<<attribute>>-name : string [1][maxLength = 32]
<<attribute>>-description : string [0..1][maxLength = -1]
-collectionDefinition
1
-relatedObjectType
Relationship
<<attribute>>-name : string [1][maxLength = 32]
<<attribute>>-description : string [0..1][maxLength = -1]
-relationshipType : RelationshipType
-cardinality : Cardinality [0..1]
-relationship
0..*
1
-referenceDefinition
0..*
PropertyGroup
<<attribute>>-name : string [1]
<<attribute>>-description : string [0..1]
-propertyGroup
1..*
-member
PropertyGroupMember
-property
1
-property
-property
Property
<<skosconcept>>-label : anyURI [0..1]
0..*
1
-axis
<<enumeration>>
RelationshipType
composition
reference
object
Field
-name : string [1]
-datatype : DataType [1]
-cardinality : Cardinality [0..1]
<<attribute>>-description : string [0..1]
-isEnumerated : boolean [0..1] = false
0..*
-validValue
ValidValue
-value : string [1]
<<attribute>>-description : string [0..1]
-title : string [0..1]

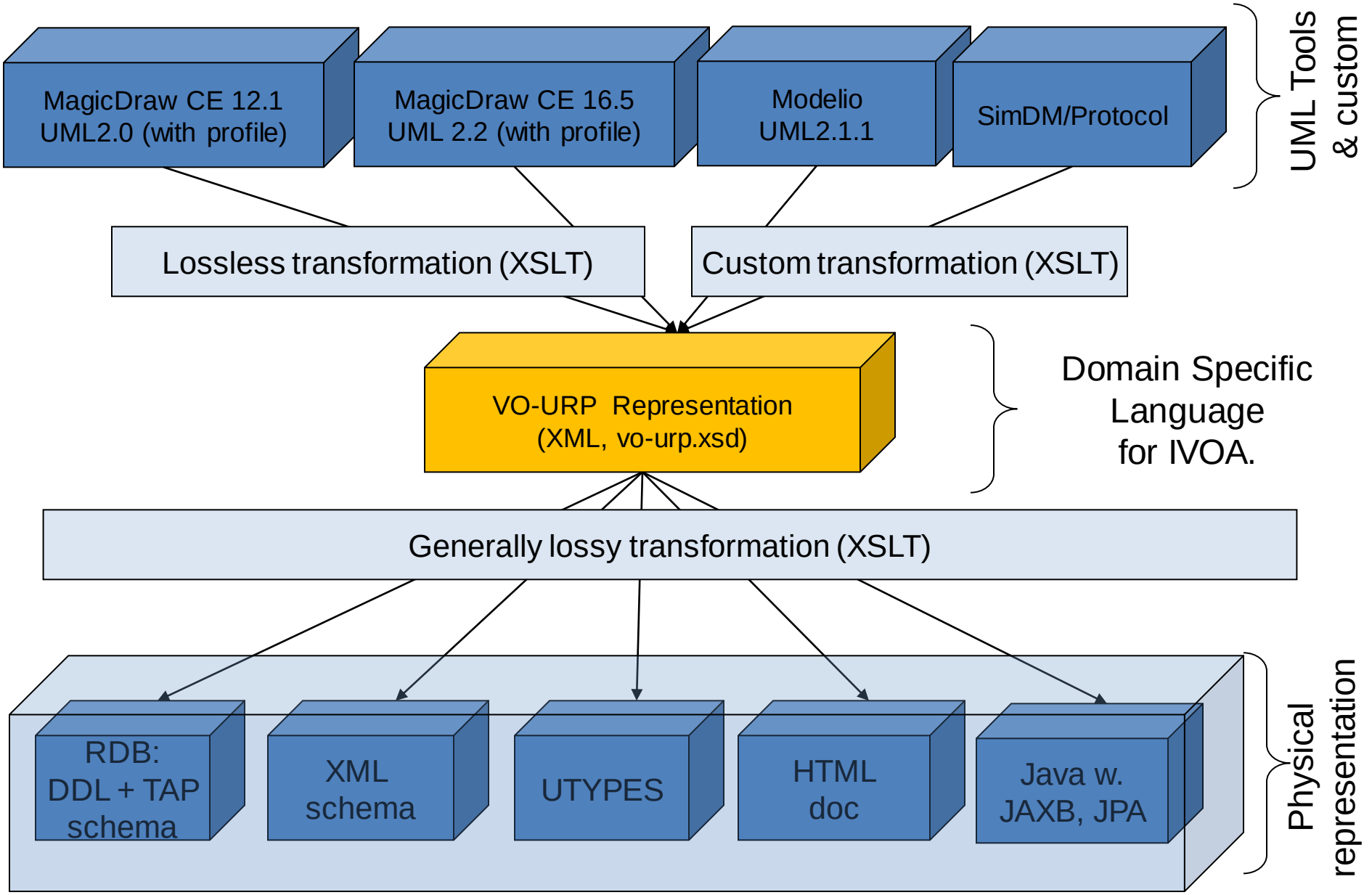
SimDB, Trieste 2008



VO-URP representation: .vo-urp

- Many different XMI dialects
- XMI complex for XSLT
- Need for simpler representation, also for use in web app (Laurent)
- Representation of UML Profile in XML Schema
  - vo-urp.xsd (intermediateModel.xsd)
- Domain Specific Language in sense of MOF
  - A MOF level 2 *meta-model*

VO-URP Data model representations



.vo-urp as domain specific language

- XSD schema representing UML2 profile
- Profile
  - subset of modeling elements, extension through stereotypes/tags, predefined types
  - See Appendix B SimDM REC
- Used explicitly in VO-URP browser web application
- TBD
  - Make conventions/rules explicit
  - Expand types
  - Add mapping configuration

VO-URP: Reference Web Application

http://localhost:8080/SimDM-browser/ — Simulation Databas...

File Edit View Favorites Tools Help

Simulation Database - Browser - 1.0-20110514 - Browser

Simulation Database - Browser - 1.0-20110514 :

Browse Model

- All Resources
  - All Projects
  - All Experiments
    - All Simulations
    - All PostProcessings
  - All Protocols
    - All Simulators
    - All PostProcessors
  - All Services
    - All SimDALServices
    - All CustomServices
- All Partys

Documentation

Browse documentation of the data model.

SKOS Concepts

Browse SKOS vocabularies used in the model.

SQL Interface

Query the SimDM database using ADQL (well, SQL).

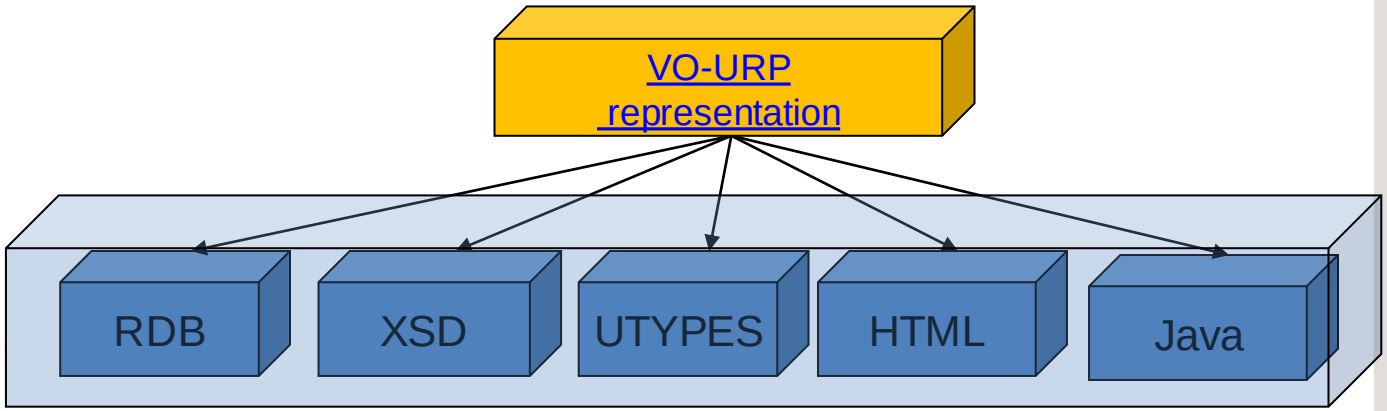
XML Validator

Validate an XML document against the data model's XML schemas.

XML Loader

Load an XML document into the database.
Requires a login.

Log management (requires a login!)



http://galformod.mpa-garching.mpg.de/dev/SimDM-browser

100%

VO-URP and UTYPE-s

- Our assumptions
  - A UTYPE is a term that in the context of a data model identifies an element of that data model uniquely.
- To ensure uniqueness following syntax sufficient.
  - NB based on concepts in .vo-urp representation!!

VO-URP's UTYPE BNF

utype	:=	[model-utype | package-utype | class-utype | attribute-utype | collection-utype | reference-utype | container-utype
model-utype	:=	<model-name>
package-utype	:=	model-utype ":/" package-hierch
package-hierch	:=	<package-name> ["/" <package-name>]*
class-utype	:=	package-utype "/" <class-name>
attribute-utype	:=	class-utype "." attribute
attribute	:=	[primitive-attr | struct-attr]
primitive-attr	:=	<attribute-name>
struct-attr	:=	<attribute-name> "." attribute
collection-utype	:=	class-utype "." <collection-name>
reference-utype	:=	class-utype "." <reference-name>
container-utype	:=	class-utype "." "CONTAINER"a
identifier-utype	:=	class-utype "." "ID"

Use of UTYPE-s in VO-URP/SimDM

- For identification of model elements
  - a HTML document (part of spec) has anchors at each position where a definition of an element is made. <base-url>#<utype> will bring you to that element.
  - The .vo-urp representation has for each modeling element a <utype> giving explicitly the UTYPE for that element, plus full definition.
  - Less essential: utypes can be parsed and one could walk from the root of the .vo-urp representation down to the element.
    - Precise syntax somewhat arbitrary

Our suggestion

(possibly) independent of UTYPE discussion

- Data modeling is not easy.
  Especially not in a distributed world.
  - It would be useful if the DM WG were to define a common meta-model for data modeling efforts in IVOA
  - Allows compliance rules and validation
  - Mappings can also be standardised
  - Best practices can be enforced easily

- VO-URP has a reference implementation for this and we're happy to provide it as start for further development.