Reasoning with RDF: Utypes

Norman Gray
VOTech/AstroGrid
University of Leicester, UK
(and University of Glasgow, UK)

IVOA Interop, Moscow, 2006 September 18–20

why do we standardise?

It’s so we can interoperate – it’s not an end in itself.

But standardisation is expensive, in time, effort and documentation.

Thus if we can interoperate with minimal standardisation, that’s a win.

standardisation has contradictory goals

A standard must be as small as possible, so that it’s possible to agree on, and possible to read the documentation. But it must also be as large as possible, so that it covers enough of what people want to communicate.

Because if you go beyond the standard, you have nothing. Going beyond the standard is very costly.

i’m not saying. . .

I am *not* saying any of:

- Abandon DM effort
- Don’t have central/IVOA standard
- . . .

utypes

What are utypes? JCM has characterised them as a way of (de-)serialising structured information into VOTables. This conceives utypes as ‘pointers into data models’.

I claim I can show a way of using utypes that potentially *decreases* the amount that needs to be standardised, and simultaneously *increases* the amount of interoperability.

step 1: view the utype as a url

No need for syntactic change, just enhanced interpretation:

```
xmlns="http://www.ivoa.net/ut#"
utype="characterization.characterizationAxis"
```

or

```
xmlns:cha="http://www.ivoa.net/ut#"
utype="cha:characterization.characterizationAxis"
```

or

```
utype="http://www.ivoa.net/ut#characterization.characteriz
```

Cf, CURIEs:

`http://www.w3.org/2001/sw/BestPractices/HTML/CURIE`

step 2: define your own utypes

- Create a web page `http://example.org/utypes/1.0` with `<a name="myCharAx">...</a>`.
- Implies utypes `http://example.org/utypes/1.0#myCharAx`
- Immediate documentation win for humans, but surely we've lost interoperabilty! No, because. . .

step 3: explain what these are formally

```
% curl --header accept:text/rdf+n3 \
    http://example.org/utypes/1.0
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
<http://example.org/utypes/1.0#myCharAx> a rdfs:Class;
  rdfs:subClassOf
    <http://www.ivoa.net/ut#characterization.characterizationA
%
```

(ie, subtyping, in O-O terms)

what does this give you?

- Permits complete labelling of object.
- Allows you to promptly deduce, when you see this unknown utype, that it’s a more specific version of something you already know about.
- Can be multiple levels, from your specific utype, through community consenses, to ivoa standards.
- [demo]: external reasoner aggregates assertions and serves inferences

workable

- Reasoning is simple=fast.
- Utype definitions will be stable – they won't change – so they can be aggressively cached (inferences are one-off in principle, but would include bugfixes and updates in practice).
- This demo uses a generic reasoner. It would be very simple to wrap this in a thin layer to produce a service which specialises in resolving utypes:
- `http://localhost/resolver?q=http://example.org/utypes/1.0%23myCharAx`

benefits (i)

- It allows data producers to say *exactly* what a column (etc) is. So if you happen to recognise that utype, you win immediately.
- But if you don’t recognise it, you can promptly find out what it’s most like.
- Versioning is very easy (eg, characterization.characterizationAxis version 2 is a subclass of characterization.characterizationAxis version 1). Plus deprecation, replacement, and so on.

benefits (ii)

- Greatly lowers the cost of going beyond the standard, and of mutating the standard, making it feasible to define smaller, or even prototype, standard sets of utypes.
- Supports community-specified layers of utypes for special cases and extensions
- . . . without losing interoperability.

the characterization data model

- In several places, the model document defers things to a future version
- . . . or documents ambiguities, such as spatial location (pointing, or astrometric solution, or. . .)
- . . . or overloads terms, describing patterns for describing features.

All of these are necessary, but the costs can be lowered.

possibilities

- Let the utype attribute take multiple utypes. Might avoid the lookup, but ugly.
- UCDs can *potentially* be seen, in this picture, as a very generic utype

summary

Standardisation is pushed towards large standards and long processes, because going beyond the standard is costly, and because the standard must have broad coverage.

(i) So if extension and versioning of a data model can be made easy and cheap, then it becomes feasible to make data models smaller, and thus faster to develop, without frustrating interoperability. Characterization is largely sorted out now, but there's still provenance, sources, . . . , still to go.

(ii) Serve a detailed view of the data without sacrificing interop.