UTypes as URIs

Norman Gray
VO-TECH / Uni. Leicester / Uni. Glasgow, UK

IVOA Interop, Beijing, 2007 May 16

what is the problem?

An application is asked to process data which refers to a UType it doesn't understand.

What's it supposed to do?

the short version of the answer

Making UTypes be (dereferenceable) URIs is cheap, and causes Good Things to happen.

the votable definition of utypes

The VOTable specification says:

> In some contexts, it can be important that FIELDs or PARAMeters are explicitly designed as being the parameter performing some well-defined role in some external data model. [...]. None of the existing name, ID or ucd attributes can fill this role, and the utype (usage-specific or unique type) attribute has been added in VOTable 1.1 to fill this gap.

other uses of utypes

The SSA spec introduces its UTypes with:

> The word [UTYPE] is now used generally to mean a standard identifier for a data model field. They are [...] of the form a.b.c.d where the dots indicate a 'has-a' hierarchy [...]

And the Characterisation spec says:

> A Utype is elaborated for each VOTable item in the serialisation as a string based on instance variable paths in our object-oriented datamodel.

votable utype syntax

The VOTable spec goes on to say:

> In order to avoid name collisions, the data model identification should be introduced following the XML namespace conventions, as `utype="datamodel_identifier:role_identifier"`. The mapping of `datamodel_identifier` to an xml-type attribute is recommended, but not required.

That looks like a URI to me. . .

utypes as uris

So: `http://www.ivoa.net/Documents/latest/utype-uri.html`

- Regard the `datamodel_identifier` prefix above as an XML namespace (syntax), and interpret the UType as a URI name.
- Require that each UType URI be resolvable to human-readable documentation for the concept thus named.
- Persistently!
- Require that each UType URI be *additionally* resolvable to a formal (RDF) expression of its semantics, aimed at software.

documentation

Consider UType ‘sharpBounds’ in the namespace
`http://example.org/my-utypes#`.

Retrieve `http://example.org/my-utypes#sharpBounds` and get:

```
[...]
<h2><a name='sharpBounds'>Accurate bounds</a></h2>
<p>In our data, <code>#sharpBounds</code> are the
bounds on a bandpass where the transmission goes from 0% to 100%
within 10nm.  This is more specific than the Char'n
<code>coverage-bounds</code> type, because...
[...]
```

That’s nice and friendly, but doesn’t help the machines.

explanation

Retrieve `http://example.org/my-utypes` with `Accept:text/rdf+n3` and get:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix myns: <http://example.org/my-utypes#>.
@prefix u: <http://www.ivoa.net/utype-spec#>.
@prefix ivoa: <http://www.ivoa.net/ut/characterization#>.

myns:sharpBounds a u:UType;
    rdfs:subClassOf
        ivoa:coverage-bounds .
```

But how are we to process this?

resolving

```
% curl 'http://thor.roe.ac.uk/utype-resolver/superclasses?
    http://example.org/my-utypes%23sharpBounds'
http://www.ivoa.net/ut/characterization#coverage-bounds
%
```

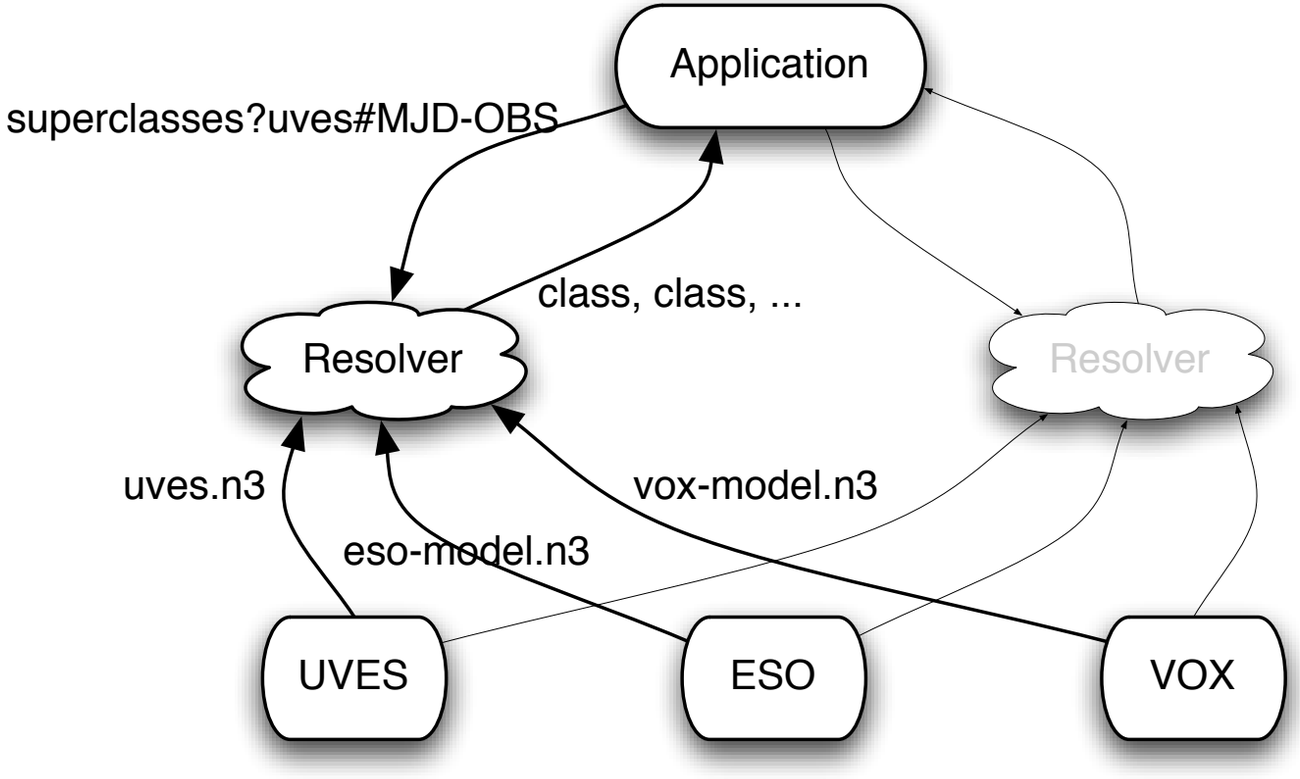
or

```
new java.net.URL("http://thor.../superclasses?"+ut).getContent();
```

- The first superclass you recognise, you use.
- The resolver goes and retrieves the explanation if it needs to
- . . . and then caches it; so no preloading; so can be local.

DEMO

architecture



summary

- Simple access to simple reasoning.
- Highly extensible (and possibly portable to UCDs).
- Versioning is easy.
- Makes UTypes cheaper to define, *without sacrificing interoperability* (though we don't want it to be trivial).
- Makes UTypes easy for applications.

links

IVOA Note: http://www.ivoa.net/Documents/latest/utype-uri.html

Resolver demo: http://thor.roe.ac.uk/utype-resolver

Try retrieving (for example)

http://thor.roe.ac.uk/utype-resolver/superclasses?

http://www.astro.gla.ac.uk/users/norman/demo/uves%23GRAT2

rdf quick intro (again)

- All the world is *triples*, consisting of *resources* named by URIs (`ivo:...` or `http://nxg.me.uk#norman`)
- . . . which have *properties* whose *values* are resources or literals.
- RDF/RDFS/OWL describe these using `rdf:type`, `rdfs:subClassOf`, `owl:Property`, and so on.

There is an analogy with XML Schemas, *but it is a loose one* – they're not addressing the same problem.