Fig. 1

Fig. 2

Fig. 3

1. What's new in Provenance

(cf. Fig. 1)

Markus Demleitner, Kristin Riebe
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

Experiments with translating actual provenances into formalised data structures made clear to us:

We have to make a big decision. Plato or Locke?

Help us by considering what we have to say…

(cf. Fig. 3)

2. Plato: Prov-with-Prototypes

In the SimDM-inspired model, there's a prototype for every process, parameter, etc:

(cf. Fig. 4)

The Plato part here is that "real things" (e.g., a reduction step) as something like a prototype that defines what it is, what its parameters are, and so forth. The same goes of the parameters that go into such an activity.

That's nice because in a database, metadata common to all similar activities only needs to be stored once – it's plain normalisation.

It's not so nice because in a typical provenance that comes with a data item, you'd have lots of definitions, and essentially single instances pointing to them ("doubling the number of entities"). We're discounting the possibility of having a public library of activity types as we don't believe we're in a position to reliably pull this off right now.

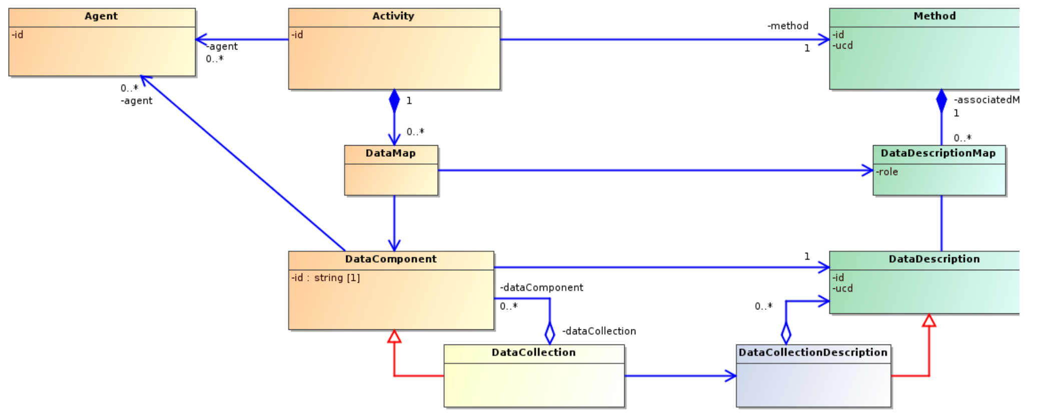

Fig. 4

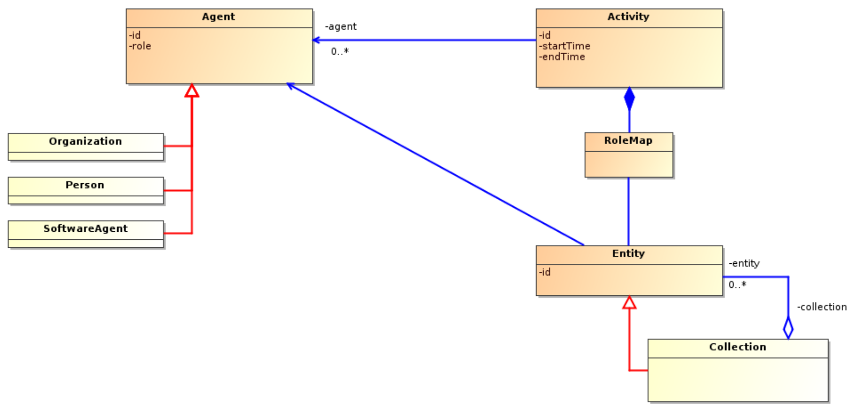

Fig. 5

3. Locke: Instances Only

This is a fairly flat translation of W3C provenance:

(cf. Fig. 5)

This is nice because it's essentially W3C (with slight adjustments for the VO case) and because it nicely serialises into single provenances that accompany data products in what we envisage as the plain VO-DML instance serialisation .

It would be clearly ugly if someone were to generate VO-DML's default database schema from this – it would be severely denormalised.

4. Meet Plato

As an example for how this works out in serialisation, here's some examples in PROV-N notation. First, for Plato (the SimDM-inspired approach), we define a parameter that's a FITS file, then we create two instances of it, and then we say one such instance was used as source image to a stacking operation; to do that, we first introduce the notion of a source image in an image stacking operation.

```
dataDescription(fitsfile_id, [
    type = "fits_file",
    description = "A file written in the standard image...
    documentation-url = "http://fits.gsfc.nasa.gov/fits...
  ])

data(img500_id, [
    dataDescription = fitsfile_id,
    accessReference = "http://foo.bar/ta220500_1OFCU2als"
  ])
data(img501_id, ...)

dataDescriptionMap(sourceimage_descmap_id,
  fits_file_id, image_stacking_method_id, [
    role = "role:source image",
    roleFlag = "input"])

dataMap(sourceimage_descmap_id,
  img501_id,
  image_stacking_2014_08_07_id)
```

5. Meet Locke

Here's about the same thing in the W3C-inspired DM:

```
entity(e_img500_id, [
    type = "file:fits",
    accessReference = "http://foo.bar/ta220500_1OFCU2als"])

entity(e_img501_id, [
    type = "file:fits",
    accessReference = "http://foo.bar/ta220501_1OFCU2als"])

used(a_image_stacking_2014_08_07_id,
  e_img500_id, [role = "role:source image"])
```

Just looking at the much shorter serialisation, Locke seems to be a no-brainer. But it's not – designing a data model based on serialisation length rather than on it being normalised is something we should think hard about.

6. Be heard, help out

To get anywhere with Provenance, we need to be use-case driven. It's therefore important that you're on board the effort if you need to represent provenance.

Join our mailing list:

http://lists.g-vo.org/cgi-bin/mailman/listinfo/prov-adhoc

Check out our stuff from volute:

https://volute.googlecode.com/svn/trunk/projects/dm/provenance