semantic arts

Extending an Upper-Level Ontology

Michael Sullivan, Cloud Solution Architect, Oracle (2023)

If you have been following my blogs over the past year or so, then you will know I am a big fan of adopting an upper-level ontology to help bootstrap your own bespoke ontology project. Of the available upper-level ontologies I happen to like gist as it embraces a “less is more” philosophy.

Given that this is 3rd party software with its own lifecycle, how does one “merge” such an upper ontology with your own? Like most things in life, there are two primary ways.

CLONE MODEL

This approach is straightforward: simply clone the upper ontology and then modify/extend it directly as if it were your own (being sure to retain any copyright notice). The assumption here is that you will change the “gist” domain into something else like “mydomain”. The benefit is that you don’t have to risk any 3rd party updates affecting your project down the road. The downside is that you lose out on the latest enhancements/improvements over time, which if you wish to adopt, would require you to manually re-factor into your own ontology.

As the inventors of gist have many dozens of man-years of hands-on experience with developing and implementing ontologies for dozens of enterprise customers, this is not an approach I would recommend for most projects.

EXTEND MODEL

Just as when you extend any 3rd party software library you do so in your own namespace, you should also extend an upper-level ontology in your own namespace. This involves just a couple of simple steps:

First, declare your own namespace as an owl ontology, then import the 3rd party upper-level ontology (e.g. gist) into that ontology. Something along the lines of this:

```
<https://ont.mydomain.com/core>
    a owl:Ontology ;
    owl:imports <https://ontologies.semanticarts.com/o/gistCore11.0.0> ;
    .
```

Second, define your “extended” classes and properties, referencing appropriate gist subclasses, subproperties, domains, and/or range assertions as needed. A few samples shown below (where “my” is the prefix for your ontology domain):

```
my:isFriendOf
    a owl:ObjectProperty ;
    rdfs:domain gist:Person ;
    .
my:Parent
   a owl:Class ;
   rdfs:subClassOf gist:Person ;
   .
my:firstName
   a owl:DatatypeProperty ;
   rdfs:subPropertyOf gist:name ;
   .
```

The above definitions would allow you to update to new versions of the upper-level ontology* without losing any of your extensions. Simple right?

**When a 3rd party upgrades the upper-level ontology to a new major version — defined as non-backward compatible — you may find changes that need to be made to your extension ontology; as a hypothetical example, if Semantic Arts decided to remove the class gist:Person, the assertions made above would no longer be compatible. Fortunately, when it comes to major updates Semantic Arts has consistently provided a set of migration scripts which assist with updating your extended ontology as well as your instance data. Other 3rd parties may or may not follow suit.*