Design issues of the Ontology for ODRL 2.0

Víctor Rodríguez-Doncel, Mari Carmen Suárez-Figueroa
Ontology Engineering Group - Universidad Politécnica de Madrid

1. Introduction

This document analyzes several design issues about the ontology version of ODRL2.0. The reference for making the mapping is the ODRL specification given in April 2012:

- **[model]** ODRL Version 2.0 Core Model http://www.w3.org/community/odrl/two/model/
- **[vocab]** ODRL Version 2.0 Common Vocabulary - http://www.w3.org/community/odrl/two/vocab/

During this analysis, the following implementations will be referred:

- **[Draft]** The draft version. http://www.w3.org/community/odrl/wiki/SemanticWeb
- **[Koblenz]** The version made in the framework of ROX at the University of Koblenz-Landau. http://userpages.uni-koblenz.de/~aggrimm/rox/index.html
- **[Barcelona]** The version of ODRL1.1, at Universitat Politècnica de Catalunya. http://dmag.ac.upc.edu/ontologies/odrlonto/index.html

2. Design Issues List

#1. Lack of reuse / mappings to other vocabularies

Comment. The [Draft] ontology does not reuse existing terms and it is hardly connected with other vocabularies. Based on the guidelines for developing and publishing Linke Data [1], the new vocabulary should reuse as many terms as possible from those existing in the vocabularies already published. As a second step, mappings should be made.
We believe that the lack of connectivity with other pieces of the Linked Data is a critical drawback today: it will hinder the ODRL OWL spread, making the ontology possibly isolated and unused. The ontology should not merely describe the model but match the elements already present in other vocabularies who have gained spread.
Recommended action: Reuse existing terms. Match (subclassing, same as, etc.) the classes and properties to existing elements in other famous vocabularies. [this is a work that cannot be dispatched in a single comment]

#2. Namespaces and scope of the files

In the week of June, 24th, we thought this was an important issue. After the discussion in the list, it has become more clear that a united namespace has to be defined for the JSON/OWL/XML serialization whenever is possible.

~~**Comment**: [Draft] proposes two separate files (core, vocab), as it was done in [Barcelona], while [Koblenz] uses up to 5 different namespaces. The scope is not complete, though.~~
~~Having two files reflects the two namespaces defined in [model][vocab]. However, the current specification misses the Informative Section 5 in [model]. Also, some elements defined in the "model" are expressed as "vocab". For example, "assignee" is defined in [Draft] to belong to~~

~~the "vocab" namespace, while it is defined in "model" and not even mentioned in the [vocab] document.~~

~~**Proposed action**: Include elements from Section 5 in the same [model] namespace, properly annotated. Move terms to their correct namespace.~~

#3. uid as the instance URI

Comment: Having "uid" as the instance URI is the intuitive meaning of an instance URI. [Draft, Koblenz]. Alternatively, it might have been a datatype property, respecting the "uid" term of ODRL. We believe the current state of the affairs is satisfactory as it simplifies the model, but it should be documented in the ontology.
Proposed action: Document that "uids" are understood to be the URI of the instances.

#4. Relation as an object property

Comment. The logical meaning of "Relation" is equal to that of an object property, although he standard defines them as association *classes*. As we read in [model] "*The Relation entity is an association class and can be used to link to an Asset from either Permission , Duty or Prohibition*".
Having Relation as an object property looks a sound option, being the different values for its attribute "relation" (currently target and output) modeled as having subproperties of Relation.
The Relation in [Draft] is an object property and it has two subproperties: target (where the action takes place) and output (newly Asset produced after the Action has been executed).
The objectProperty name "hasAsset" used in [Koblenz] is more clear perhaps but abandons the ODRL specification terminology. However, the extended relations in Section 5.1 in [model] have not been included (AND,OR,XOR).
Proposed action: Stay with Relation as object property. Include the extended relations of Section 5.1

#5. Asset as superclass of Policy / Asset not superclass of Policy. Asset same as Thing.

Comment: All the policies are certainly assets, although perhaps it is not needed that much to stress that all the policies are assets (they are not disjoint classes, either)
Policy can be sublclass of Asset. But please note that by making Asset same as Thing, we say that all the assets are things –what is fine–, and that all the things are assets, what may be considered wrong: Asset is defined as "*Content that is subject to an ODRL policy*". Not all the assets are subject to ODRL policies.
Proposed action: We strongly recommend **not** making Asset the same as Thing.

#6. Duty / Permission / Prohibition as subclasses of Rule

Comment: It is defined so by the specification [model]. Please see Section 5.2. [Draft] does not declare this superclass. A Rule superclass of Duty/Permission/Prohibition should be declared. Note that [Koblenz] also models this Rule class.
Proposed action: Create a Rule superclass.

#7. Handling of conflicts (valid also for handling undefined, etc.)

Comment: *Conflict* is defined as an attribute of Policy, which can take one of three pre-defined values. While [Draft] defines a class for these three elements, [Koblenz] makes use of oneOf intelligently avoiding the use of an artificial class. However, this oneOf permits choosing among three strings, while ODRL2.0 specifies that URIs should be preserved.

Proposed action. We recommend not declaring an artificial class to contain these three elements, but declaring them as individuals with the correct URI.

#8. Definition of Constraint

Comment: In [Draft], Constraint is both defined in model and vocab.

Proposed action: We recommend defining Constraint only in the model, as declared in 2.8 in [model]

#9. Direct import of SKOS

Comment: In [Draft], SKOS is directly imported. skos:definition is used only once in [Draft], in the vocab.rdf. Besides this, SKOS is only used in three occasions with skos:scopeNote.
It is arguable that SKOS should be imported, their URIs might be directly used. Alternatively, SKOS might be avoided at all.
Recommended action: Remove skos completely, use rdfs:comment instead.

#10. Example of Figure 3.1

Comment. Example of [Draft] for Figure 3.1 relates an Asset and a Policy with a dcterms:license element, omitting the ODRL term for this: a relation "target". Easy reading, but not ODRL. Alternatively, "target" might be made a subproperty of dct:license (what might be discussed, too).
Recommended action: Use the object property target instead of an alien property.

#11. Actions as classes or actions as individuals.

Comment. Actions in the [vocab] are represented as individuals in [Draft], while they are represented as subclasses in [Koblenz]. The first approach prevents from future extensions, but the second approach needs from instances not always meaningful. In order to reflect the ODRL specification, a "name" property should exist, but it can be assumed to be its URI.
Recommended action: Further dicuss this policy, as it is a critical piece. If they are to remain class instances, additionally include the name as a datatype property.

#12. Relation / Role as top object properties

Comment. Relation is defined in [Draft] as an object property, parent of target and output. This is equivalent to the Role, logically parent of function. While "Relation" is defined, Role it is not.
Recommended action: Check consistency and either supress relation, or include Role as a super property of "Function".

3. References

[1] Heath, T., Bizer, C.: Linked data: Evolving the Web into a global data space (1st edition). Morgan & Claypool (2011)