BRGM notes

2017/11/13

S.Grellet / A.Feliachi

## BRGM use case considered

* Generating ontologies for a subset: GeoSciML Basic, Borehole and Lite,
* Testing populating instances for Boreholes, Geologic Units pointing to vocabs when available (FR, EU, CGI GTWG) and testing some representations (maps, graph).

## ISO 19150-2 general notes

* The rules of ISO 19150-2 are restrictive on many aspects if we respect them all. Respecting all of them means we don’t take into account the open world assumption when working with ontologies
* Properties naming when translating attributes: dots in properties identifiers could be interpreted somehow that they are still scoped to classes, while in ontologies, properties are scoped to a name space instead. Properties are independent entities that may or may not have a specific class as a domain. This is one major structural difference between UML and OWL.
* The transformation rules are consistent but limits the resulting ontologies to the UML paradigm. Some additional work may be done on the resulting ontologies to add semantics between classes (disjunctions, subsumption, equivalence, etc) and within or between properties (functional properties, transitive properties, symmetric properties, inverse of, etc).
* The rules for transforming UML constraints into OWL constraint are weak. The norm recommends translating them as automatically as a descriptive text that is related to the corresponding element (ontology, class or property) via an annotation property. A manual work can be done to translate some (if not possibly all) the UML constraints into OWL.
* No specific indications about association classes are mentioned in the norm. It is obvious that an association class is translated as an OWL class. No rule for linking this class to the related class(es) appear

## ShapeChange: GeoSciML Basic and GeoSciML Borehole

* ShapeChange “Map entries” provide a flexible way to choose recommended names for properties and classes. This would enable one to reuse existing specialized classes and properties from external ontologies.
* The requirements of the model cannot be all respected in the ontology representation (eg. “QuantityRange properties that must report a single value SHALL assign both lower and upper value as equal to that single value.”). This should be checked and translated manually as restrictions (when possible) afterwards.
* Basic types (from SWE types for example) must be modified if needed by specialized Classes from other ontologies or by defining new ones
* CodeLists take skos:concept for the moment. Check afterwards if a specific controlled vocabulary is recommended. In that case, two possible solutions can be applied: the first one is to leave that recommendation as a comment on the property that refers to the codelist, the second one is to type the range of that property with the top concept of the recommended controlled vocabulary.
* Association classes must be handled differently: ShapeChange transforms an association class into separate class and properties. Thus, no link is created between the association class and the classes that are initially related by it in the UML. No rule is found in ShapeChange to handle that.

🡪 this must be defined afterwards with two properties: associationSource and associationTarget (exactly as in passing from conceptual model to a logical schema). As a solution, this could be locally defined as [association name]+”Source” and [association name]+”Target”. These two properties must have the right domain and range. The direct property between the source and the target automatically created by Shapechange must be deleted.

* ShapeChange does not follow ISO 19150-2 rule for transforming a Union of classes (see 19150-2 section 6.8): a union of UML classes is presented in OWL a union of two classes. On example use case is GSMLitem.

The ShapeChange rule for transforming a union of UML classes (rule-owl-cls-union) uses cardinality restriction on the different properties that are members of the union to force the instantiation of maximum one property at a time. It means that this transformation will add a property to refer to each class (member) of the union.

It is specified in ISO-19150-2 that a union of UML classes should be represented by a union of OWL classes (and not properties) in the ontology, which make total sense. The issue with this rule is that it does not follow the recommendation of ISO-19118 which states that the union of UML classes provides an exclusive choice, while it is not the case for an OWL union (using the property owl:unionOf).

We decided to use the OWL2 disjoint union (using the property owl:disjointUnionOf) to respect the ISO-19118 recommendation.

* Use general (non-scoped to class) property names when the name of the attribute or association is unique. Thus, leave the domain of the properties open (or typed as owl:Thing). The restrictions on the properties valueswhich class use the property can be
* Allow scoped names for properties (class.Property) then verify whether automatically created properties can be merged into one (eg. GeologicFeature.purpose and EarthMaterial.purpose)
* Use GSML\_QuantityRange instead of swe:QuantityRange as recommended
* Rename swe:Category to skos:Concept or mdl:Lineage (depending on the case) and swe:Quantity to the relevant class in the context (ts:TimePosition, mdq:PositionalAccuracy, etc.)
* In the GeoSciML Borehole ontology, only the class BoreholeDetails and its related properties are edited for the moment to be used in GeoSciML Lite ontology. The rest is to be done.
* For the properties of GeoSciML Basic and Borehole to be reused in GeoSciML Lite, we activate the ShapeChange rule “rule-owl-prop-globalScopeByUniquePropertyName” that scopes unique name property to global use, and thus not specify the domain of these properties. The scoping of the properties to their classes in Basic and Borehole is done using restrictions on the values that these properties can take for their corresponding classes. This can be done thanks to the ShapeChange rule “rule-owl-prop-range-local-withUniversalQuantification”
* …

## ShapeChange: GeoSciML Lite

* The placeholder attribute “any” becomes useless property in owl 🡪 delete it.
* Choice made to replace the “character string” data properties by object properties from GeoSciML Basic, borehole and other ontologies when possible (using the XPath mapping detailed in GeoSciML specification).
* GeologicUnitView contains mixed information from Both GeologicUnit and MappedFeature. A decision must be made to which entity the view must be associated (using the same URI as the GeologicUnit or MappedFeature )
* …

## Extend the GeoSciML Lite with an application ontology to represent French RGF (‘Reférentiel Géologique de la France’) data

* Add GeologicFeatureRelation to the application ontology to enable navigation through GeologicUnit:rank (group -> subgroup -> formation ->…)

## General Best practices to respect while defining an ontology

* Reuse existing ontologies. Map to existing ontologies.
* Define clear classes and predicates.
* Separate the TBox from the ABox.
* Start with a core ontology, build the ontology incrementally
* Build modular ontologies.
* Use properties restrictions and disjoint classes.
* Use annotation properties.
* Naming: noun for concepts, verbs for properties, use mixed case
* Label everything
* Document the ontology
* Version the ontology: use owl:priorVersion and owl:versionInfo properties to describe the ontology, and owl:deprecatedClass and owl:deprecatedProperty annotation properties to specify the version status of a class or a property when deprecated.