Transforming Between UML Conceptual Models And OWL 2 Ontologies

Jesper Zedlitz[1] and Norbert Luttenberger[2]

[1] German National Library for Economics
[2] CAU Kiel

Abstract. The ISO 19103 standard—defining rules and guidelines for conceptual modeling in the geographic domain—has deliberately chosen the Unified Modeling Language (UML) as "conceptual schema language" for geographic information systems. From today's perspective—i.e. when taking into account today's mature semantic web technology—another language might also be envisioned as language for specifying application-oriented conceptual models, namely the Web Ontology Language OWL 2. Both language definitions refer to comparable meta-models laid down in terms of OMG's Meta Object Facility, but in contrast to UML, OWL 2 is fully built upon formal logic which allows logical reasoning on OWL 2 ontologies. In this paper, we investigate language similarities and differences by specifying and implementing the transformation on the meta-model level using the QVT transformation language.

Keywords: OWL 2, UML, conceptual modeling, ontology, model transformation, GML, Semantic Web, QVT, meta-modeling

1 Introduction

In its introduction, ISO Standard 19103 states: "Standardization of geographic information requires the use of a formal CSL [(conceptual schema language)] to specify unambiguous schemes that can serve as a basis for data interchange and the definition of interoperable services." In focusing on "the combination of the Unified Modeling Language (UML) static structure diagram with its associated Object Constraint Language (OCL)"—a combination, which is probably the most often used CSL—ISO standard 19103:2005 follows mainstream. To illustrate its use, Fig. 1 shows a UML class diagram taken from the Geography Markup Language (GML) standard, where it serves as conceptual model for some application-specific purpose. Advantages are obvious: UML's graphical syntax lets also non-computer scientists easily comprehend the intention of such diagrams. Also in favor of UML is the rich tool support for UML class diagrams which recommends UML as a good starting point for software development.

Unfortunately, UML class models are not completely backed up by formal logic, and we do not enjoy reasoning support as we do for ontologies. The OWL 2

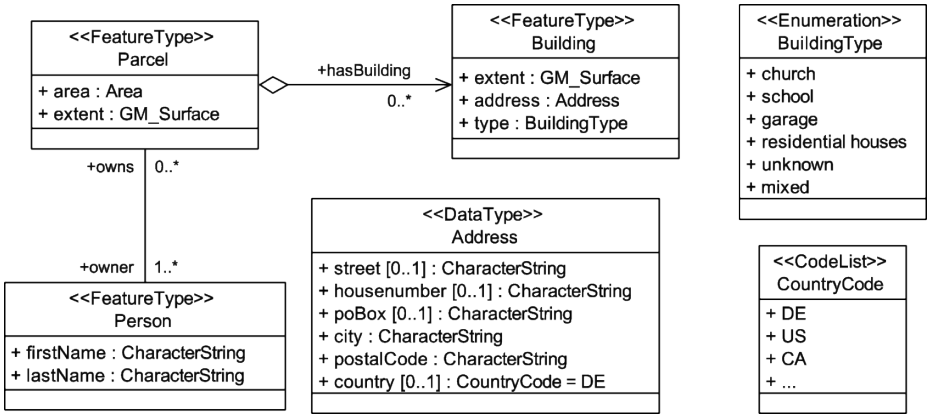

Fig. 1. Example for a UML conceptual model taken from the GML specification [6].

Web Ontology Language[3] or suitable subsets thereof in contrast are completely backed up with formal logical and there is out-of-the-box reasoning support of OWL ontologies. Reasoning over ontologies can be used to discover inferences not detected by programmers, among them subsumption relations between classes and properties in the ontology schema, which helps to determine where a concept can be located in a class hierarchy. Reasoning also helps to assert the consistency of the conceptual model (e.g. validity of intentional definitions or in other words: class satisfiability), and it allows us to inspect the conceptual knowledge encoded in the model. (Bucella et al. [1] follow the same line of arguments when giving their outline for an integration tool for geographical schemas.)

Having given this background, we feel that the following "What-if" question suggests itself: What if OWL 2 were taken as CSL for geographic information systems? Three further arguments back up the validity of our question:

A closer look at conceptual models reveals the systematic use of the object-property model that "has been the basis of the GML encoding model since the first version was adopted by OGC" [6]. It is deeply elaborated in [9]. Needless to be mentioned here, the object-property modeling pattern is at heart of RDF[4], RDF Schema[5], OWL[6], and OWL 2.[7]

Secondly, OWL is one of the building blocks of the semantic web and the Linked Open Data (LoD) Cloud, even if Jain, Hitzler et al. argue that currently the LoD cloud is missing conceptual descriptions [10]. The integration of geographical information systems with the semantic web is an obvious necessity—and might profit much from being built upon common concepts and languages.

[3] http://www.w3.org/TR/owl2-syntax/
[4] http://www.w3.org/TR/rdf-concepts/
[5] http://www.w3.org/TR/rdf-schema/
[6] http:/www.w3.org/TR/owl-ref/
[7] GML even contains a references to RDF: "[...], GML follows RDF (W3C, 1999) terminology [...]"[6, p. 20]

Thirdly, both UML and OWL 2 refer to comparable meta-models laid down in terms of OMG's Meta Object Facility. Thus replacing UML by OWL 2 seems to be a feasible task.

We have chosen a special approach to examine the question if OWL 2 can be used for conceptual modeling. Instead of looking at a bunch of examples (which is always problematic because you cannot be sure to cover all relevant cases with your examples) we approach the question by trying to transform between UML class models and OWL ontologies automatically (in both directions). This systematic approach will show what is possible and what is not.

This paper shows the transformation between a UML model and a OWL 2 ontology with special care for restrictions and extensions GML applies to UML models. We specify a transformation using OMG's Query/View/Transformation (QVT) transformation language and the meta-models of UML and OWL 2.

This paper is organized as follow: Section 2 presents the relation of UML and GML and the restrictions resp. extensions it applies. In Section 3 we show some existing work on the transformation of UML and OWL. Section 4 explains our approach in general. Section 5 shows general differences between UML and OWL 2. In section 6 we present some of our transformations en detail. Section 7 gives a short summary of the paper.

2 UML and GML

The ISO 19109 standard [9] defines rules how to create "UML Application Schema" in a common way. Basis for these application schemas is the General Feature Model (GFM). However, the GFM only defines the semantics of the meta-model but does not provide a concrete syntax how to write the schemas. In ISO 19103 [8] UML is chosen as "conceptual schema language". By defining rules for the usage of UML a so called "UML profile" is defined.

The restrictions made by ISO 19103 limit the number of UML model elements and their use. Also defined are extensions—particularly noteworthy are the stereotypes ≪CodeList≫ and ≪Union≫. However, these are not without controversy, as can be seen below. The ISO 19136 standard—GML [6] picks up the restrictions and amplifies them to some extent. A complete list of restrictions can be found in the GML specification [6].

- All UML elements have the visibility "public". [6, E.2.1.1.1]
- Class names within a class diagram are unique. [6, E.2.1.1.2].
- Operations are ignored. [6, E.2.1.1.2]
- A class can either be a FeatureType if it is marked with the stereotype ≪FeatureType≫, a DataType if it is marked with the stereotype ≪DataType≫ or an ObjectType—classes without any stereotype. [6, E.2.1.1.2]
- A generalization between two classes is only allowed if both classes are FeatureTypes, both classes are ObjectTypes or if both classes are DataTypes. [6, E.2.1.1.2]
- A generalization between classes must not be marked with a stereotype. [6, E.2.1.1.2]

– Multiple inheritance is not allowed. [6, E.2.1.1.2]
– Every association must have exactly two ends which links to a FeatureType, ObjectType or DataType. [6, E.2.1.1.3]
– Associations must not be marked with stereotypes and must not contain attributes. [6, E.2.1.1.3]

As [4, 2.4.3] observed, the UML profile described in the ISO 19103 standard is not a profile within the meaning of the definition of UML profiles given by the OMG. One reason is that the profile defines two stereotypes for data types that are applied to classes. The two stereotypes ≪CodeList≫ and ≪Union≫ are no semantics conserving specializations. For the transformation of classes marked with the disputed stereotypes this observation, however, plays no role. We will use the newly defined semantic of the marked classes.

3 Related Work

Several publications deal with general transformation of UML models into ontologies. Most of them work on XML serializations using XSLT. [2], [5], [3], and [11] fall into this category.

Milanović [12] describes the transformation of a UML model into a OWL ontology using the Atlas Transformation Language, Höglund [7] uses MOFScript for a transformation to OWL 2. However, the goal of his work is validation of models—therefore additional elements needed for the validation are inserted into the ontology that hinder further use in an information system.

Tschirner et al. [13] describe conversion rules from UML-data models to OWL. They specify four main rules to map UML classes and attributes to OWL-classes and properties. However, the constraints specific model elements (e.g. a Union) impose on the model are not mapped.

4 Basic idea

Commonly model driven architecture uses a four-layer architecture: meta-meta-model (M3), meta-model (M2), model (M1) and instance (M0) layer. OMG's MOF is a standard M3-system with a well-developed suite of software tools.

Instead of transforming elements of a M1-model directly we describe the transformation using elements of the M2-meta-models. By describing the transformation on a higher meta-level the transformation does not depend on the models that are going to be transformed. It only depends on the involved meta-models. This enables an elegant description of the transformation—for example compared to a XSLT-based transformation that works with the concrete syntax of M1-models.

It is very common that additional to one or more concrete syntaxes for a language an abstract syntax exists. For example for OWL 2 has various concrete syntaxes: Functional-Style Syntax, Turtle Syntax, OWL/XML Syntax, Manchester Syntax, etc. By working with the abstract syntax our transformation becomes independent of any particular representation.

We choose OMG's QVT Relations Language for our transformations because it is declarative and works with MOF-based meta-models. The support by the OMG consortium and several independent implementations makes it future-proof.

5 Differences of UML and OWL 2

To assess the usage of OWL 2 as CSL we first take a look at some fundamental differences of UML and OWL 2 and point out ways to circumvent some of them.

5.1 Open-World vs. Closed-World Assumption

In UML class models we work under a Closed-World Assumption (CWA): All statements that have not been mentioned explicitly are false. In contrast OWL 2 uses an Open-World Assumption (OWA) where missing information is treated as undecided. These different semantics make it necessary to add various restrictions to the ontology during the transformation process from a UML model to an OWL 2 ontology to preserve the original semantics of the model.

5.2 Profiles

UML has the concept of "profiles" which allow extensions of meta-model elements. There is no corresponding construct in OWL 2. In most cases UML profiles are used to define stereotypes to extend classes. The information of these stereotypes can be mapped to OWL 2 by clever creation of some new classes and generalization assertions. However a large part of an UML profile is too specific and would require transformation rules adapted for the particular profile.

5.3 Abstract Classes

Abstract classes can not be transformed into OWL 2. If a class is defined as abstract in UML no instances of this class (objects) can be created. In contrast OWL 2 has no language feature to specify that a class must not directly contain any individual. An approach to preserve most of the semantics of an abstract class is the usage of a `DisjointUnion`. This would ensure that any individual belonging to a subclass would also belong to the abstract superclass. However, it does not prohibit to create direct members of the abstract superclass.

5.4 Access Control and Operations

In UML the visibility of model elements can be reduced by marking them as "public", "private", etc. It is also possible to declare UML model elements as read only. OWL 2 does not have this kind of control mechanism to restrict the access to model elements. OWL 2 ontologies also do not contain any operations. However, in the list of restriction show in section 2 we have seen that both access control and operations are ignored.

5.5 Global Properties

In OWL 2 it is possible to define (object) properties at ontology level. Connections to classes (in the form of domain and range definitions) are optional. The following listing shows both cases:

```
1 Declaration( ObjectProperty( :belongsTo ) )
2 Declaration( ObjectProperty( :owns ) )
3 ObjectPropertyDomain( :owns :Person )
4 ObjectPropertyRange( :owns :Parcel )
```

`owns` is a connection between individuals belonging to the classes `Person` and `Parcel`. No domain or range has been specified for `belongsTo`, therefore the default value of `owl:Thing` is used for domain and range. The `belongsTo` object property can be used to connect any two individuals because every individual belongs to the class `owl:Thing`.

UML offers two ways to connect classes: class-dependent attributes and associations. As the name states, class-depended attribute belong to a class and connect it with an other class or data type. Associations are package level elements themselves. However, they need (at least) two so called members-ends which require classes as types. Therefore UML associations are not completely suitable to represent (generic) object properties.

5.6 Complement

In many places OWL 2 allows you to work with the complement of classes and data types. In UML, this is not generally possible.

6 Transformations

Several transformation rules for the transformation direction UML → OWL 2 can be found in our article [14] where we have first presented our idea to use a declarative transformation language on meta-model level for transforming generic UML class models into OWL 2 ontologies. However, to answer the question whether OWL 2 could be used as CSL for geographic information systems special challenges of ISO19100/OGC conceptual models have to be taken into account (e.g. available top-level-classes and stereotypes with extended semantics). Therefore we would like to highlight these areas:

6.1 Global Properties

One way to map object properties with `owl:Thing` as domain and range to UML is the definition of a single top-level class that is super-class C_{super} of all other classes in the model (like the `AbstractGMLType`) that represents the `owl:Thing` class. In that case a generic object property can be mapped onto a UML association with two members ends of type C_{super}.

6.2 Complement

As already mentioned the definition of a complement which is possible in OWL 2 is difficult. Only if you define a single top-level class, a *GeneralizationSet* marked *disjoint* can be used to model a class C and its complement ¬C.

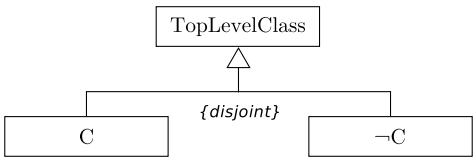

Fig. 2. UML diagram showing how the complement of a class C can be modelled.

In GML such a single top-level class exists: `AbstractGMLType`. Since each object can only be instances of either `FeatureType`, `DataType` or `ObjectType` it would also be possible to use these three classes as *TopLevelClass* when dealing with complements.

For conceptual models following GML's restrictions it is not even necessary to mark the generalization set as *disjoint*. GML does not allow an object to be instance of more than one element type.[6, E.2.1.1.2]

6.3 Associations and Class-Dependent Attributes

In UML two ways to connect classes exist: associations and class-dependent attributes. In the UML meta-model both kinds are represented by the model element *Property* as shown in Fig. 6.3. In general an association can connect two or more objects (cardinality 2..*). However, GML restricts associations to have exactly two ends. That means both class-dependent attributes and associations are connections between two element types. Therefore it stands to reason that the transformation of both associations and attributes can be handled together.

Since the model element *Association* is a subclass of *Classifier* all associations in a UML class diagram are direct members of a package. Therefore an OWL 2 concept that is similar to a association is an object property which is also direct members of an ontology. Associations can be directed or bi-directional. A directed association can be transformed into one object property. For a bi-directional association two object properties will be created—one for each direction. To preserve the information that both resulting object properties were part of one association a `InverseObjectProperties` axiom is added to the ontology.

The transformation of class-dependent attributes is more complex. There are no directly corresponding concepts in OWL 2 that allow a simple transformation. The main problem is that classes in OWL 2 do not contain other model elements which would be necessary for a direct transformation. The most similar concepts in OWL 2 for class-dependent attributes are object properties and data properties.

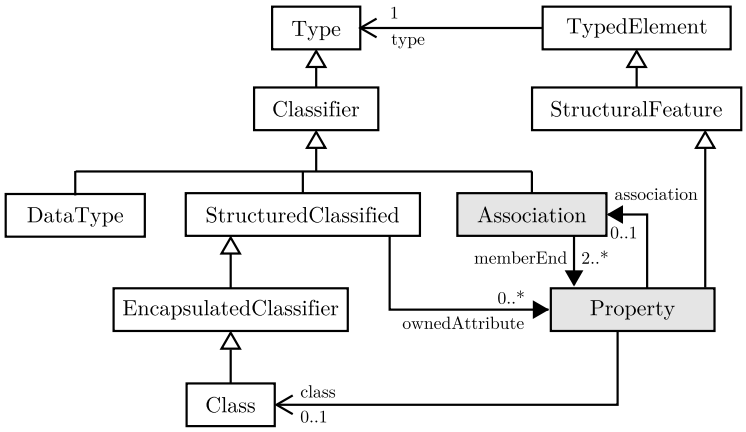

Fig. 3. Excerpt from the UML meta-model showing both possibilities to connect classes.

In both cases the decision whether a *Property* is transformed into an object property or a data property depends on the *type*-association of the *Property*: If it is associated with an instance of *Class* an object property is needed. If it is associated with an instance of *DataType* a data property is needed.

The OWA would allow that two properties that have been transformed from distinct UML properties are interpreted as one. To avoid that and to map UML's CWA best we mark all properties that are not in a generalization relationship (i.e. a `SubPropertyOf` axiom exists for them) as disjoint. To do this we add `DisjointObjectProperties` and `DisjointDataProperties` axioms to the ontology: For all pairs of UML *Property* elements we check if they were transformed into a OWL 2 property, are not identical, no generalization relationship exists between them, and they have not been marked disjoint before.

6.4 Codelist

A *Codelist* is a special kind of enumeration defined by the ISO 19103 standard. A class that is a *Codelist* is marked with the stereotype ≪Codelist≫. In addition to the fixed values of a normal *Enumeration* a *Codelist* might contain other values, too. The GML standard specifies the lexical form of these additional entries.

Similar to the mapping of an *Enumeration* a *Codelist* can be transformed to OWL 2 by using `DataUnionOf` to add the additional values of the *Codelist* to the `DataOneOf` element that has been created for a normal *Enumeration*. The `DataTypeRestriction` element allows an elegant way to add the restrictions for additional values to the data-type—we can use the same syntax from XML Schema[8] that is used in the GML standard.

[8] http://www.w3.org/TR/xmlschema-2/

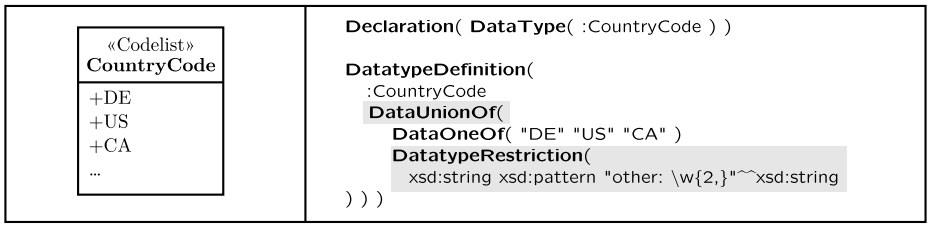

Fig. 4. Transformation of a GML Codelist.

6.5 Union

Another GML specific stereotype is *Union*. The semantics of a *Union* is that only one element of a set of properties may be present at any time. In UML a *Union* is modelled as a class annotated with the ≪Union≫ stereotype. The set of properties is the collection of class-dependent attributes.

We have developed two different mappings to transform a *Union* to OWL 2. The first solution works if the type of all attributes are either data-types or classes. In that case the transformation of the attributes results in `ObjectProperty` or `DataProperty` elements, not a mixture of both.

Let C be a class representing a *Union* with properties $p_1 \dots p_n$. To assure that only one property $p_x \in p_1 \dots p_n$ is specified for an individual we insert a helper property p_{Union} with the domain C and $p_i \sqsubseteq p_{\text{Union}} \forall i \in 1..n$

Now we can add a `DataExactCardinality` axiom to the ontology which limits the use of our helper property p_{Union} to exactly one per individual of class C. That prohibits the use of two or more different properties. However, due to the OWA we cannot make sure that at least one property is present—there might be an individual that is simply not listed in the ontology.

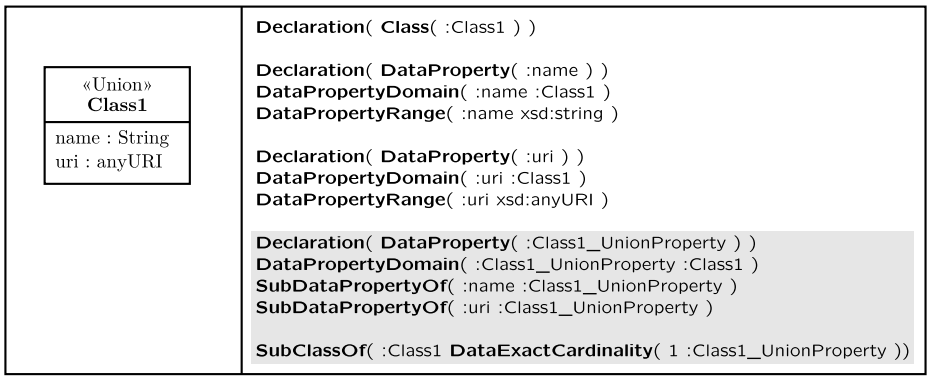

Fig. 5. First solution for a transformation of a GML Union.

The second solution also works with a mixture of `ObjectProperty` and `DataProperty` elements. However, the resulting ontology becomes a bit more complex.

For each property $p_i \in p_1 \dots p_n$ of the union we create a helper class C_i. With a disjoint classes axiom we state that all of these classes are pairwise disjoint: `DisjointClasses(`C_1 ... C_n`)`. Each class is stated as equivalent to a set that contains all those individuals connected by p_i with exactly one individual/literal: `EquivalentClasses(` C_i `DataExactCardinality( 1` p_i `) )` resp. `EquivalentClasses(` C_i `ObjectExactCardinality( 1` p_i `) )`

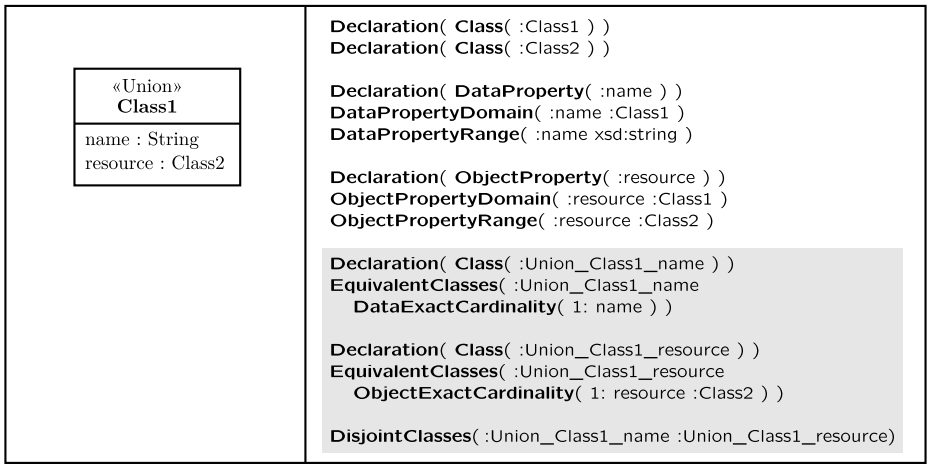

Fig. 6. Second solution for a transformation of a GML Union.

While the first solution only adds $(n + 3)$ axioms per UML property of the union to the ontology the second solution requires $(2n + 1)$ additional axioms per property. Therefore it is clever to choose the first solution if all attributes of a union only link to data-types resp. classes and only choose the second solution if it is a mixture of both.

6.6 Stereotypes

In UML stereotypes can be applied to classes (and other model elements). A few stereotypes are defined in the UML specification. A user can define his own stereotypes in UML profiles. One of the advantages of QVT is the possibility to access profiles and stereotypes.

As mentioned earlier, some of ISO 19103's stereotypes modify the semantics of the model element they are applied to. In that case a modified transformation is necessary. We have shown these specialized transformation above for the stereotypes ≪CodeList≫ and ≪Union≫.

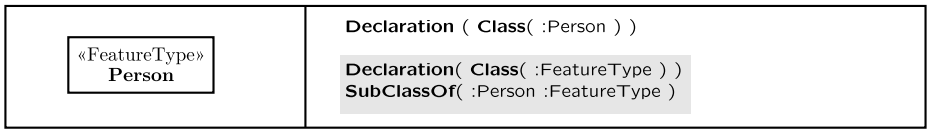

Fig. 7. Transformation of the stereotype ≪FeatureType≫.

For the other three stereotypes defined by ISO 19103 (≪FeatureType≫, ≪ObjectType≫, and ≪DataType≫) OWL 2 classes are defined in the ontology. Classes to which these stereotypes have been applied to become sub-classes of those classes in the ontology. Fig. 7 shows an example of such a transformation.

The transformation of stereotypes into classes in OWL 2 is reasonable since we can connect additional axioms with these classes. That allows us to write down some of the semantics of GML/ISO 19103 in a machine interpretable way:

[6, E.2.1.1.2] states that each element type must be either a *FeatureType*, a *DataType* or an *ObjectType*. There are exclusively these three groups of element types. This can be expressed with a `DisjoinUnion` axiom:

```
1 Declaration( Class( gml:DataType ) )
2 Declaration( Class( gml:FeatureType ) )
3 Declaration( Class( gml:ObjectType ) )
4 DisjointClasses( owl:Thing gml:DataType gml:FeatureType gml:
    ObjectType )
```

Both *FeatureType* and *ObjectType* have a unique identifier[9]. In constract, *DataType* must not have such an identifier. This can be expressed in OWL by defining a `DataProperty` like this:

```
1 Declaration( DataProperty( gml:id ) )
2 DataPropertyDomain( gml:id ObjectUnionOf( gml:FeatureType gml
    :ObjectType ) )
3 DataPropertyRange( gml:id xsd:string )
4 FunctionalDataProperty( gml:id )
```

Since all classes to which the stereotype ≪FeatureType≫ or ≪ObjectType≫ had been applied to in the UML class diagram become sub-classes of either `FeatureType` `ObjectType` in the ontology the data property `gml:id` can be used for them. Instances of classes which had the stereotype ≪DataType≫ applied must not use the key: The domain of the property is `FeatureType` or `ObjectType` and these classes are disjoint with the class `DataType`.

[9] "Object types are types where the instances shall have an identity, [...]" [6, E.2.1.1.2]

7 Summary

We have shown differences and similarities between UML conceptual models following the ISO19100/OGC guidelines and OWL 2 ontologies. The use of QVT Relations Language enables us to describe the transformations between both technology spaces declaratively and to use model elements of the meta-models.

In further work the ideas presented here could be used for a real-world geographic information system that makes use of semantic web technology. Using our transformations the implementation could be based on either an existing UML conceptual model or a newly created OWL 2 ontology or even using alternate editing in UML and OWL 2.

References

1. Buccella, A., Gendarmi, D., Lanubile, F., Cechich, A., Colagrossi, A.: Ontology-Driven Generation of a Federated Schema for GIS. Semantic Web Applications and Perspectives p. 31 (2007)
2. Cranefield, S.: Networked knowledge representation and exchange using UML and RDF. Journal of Digital information 1(8) (2001)
3. Djurić, D.: MDA-based ontology infrastructure. Computer Science and Information Systems 1(1), 91–116 (2004)
4. Eisenhut, C., Kutzner, T.: Vergleichende Untersuchungen zur Modellierung und Modelltransformation in der Region Bodensee im Kontext von INSPIRE (Sep 2010)
5. Gašević, D., Djuric, D., Devedzic, V., Damjanovi, V.: Converting UML to OWL ontologies. In: Proceeding WWW Alt. '04. pp. 488–489. ACM (2004)
6. GML 3.2.1: Geography Markup Language (GML) Encoding Standard 3.2. 1 (2007)
7. Höglund, S., Khan, A., Liu, Y., Porres, I.: Representing and Validating Metamodels using the Web Ontology Language OWL 2. Tech. rep., D. of Information Technologies, Åbo Akademi University (2010)
8. ISO 19103: Norm ISO/TS 19103:2005 Geographic information – Conceptual schema language. ISO, Geneva, Switzerland (2005)
9. ISO 19109: Norm ISO 19109 Geographic information – Rules for application schema. ISO, Geneva, Switzerland (2005)
10. Jain, P., Hitzler, P., Yeh, P., Verma, K., Sheth, A.: Linked data is merely more data. Linked Data Meets Artificial Intelligence pp. 82–86 (2010)
11. Leinhos, S.: OWL Ontologieextraktion und -modellierung auf der Basis von UML Klassendiagrammen (2006)
12. Milanović, M., Gašević, D., Guirca, A., Wagner, G., Devedžić, V.: On Interchanging Between OWL/SWRL and UML/OCL. In: Proceedings of 6th Workshop on OCL for (Meta-) Models in Multiple Application Domains (OCLApps). pp. 81–95 (2006)
13. Tschirner, S., Scherp, A., Staab, S.: Semantic access to INSPIRE. Terra Cognita Workshop (2011)
14. Zedlitz, J., Jörke, J., Luttenberger, N.: From UML to OWL 2. In: Proceedings of Knowledge Technology Week 2011. Springer (2012)