A Dutch Verbalization Template
for ORM conceptual models and rules

Mustafa Jarrar, Vrije Universiteit Brussel, Belgium. (Contact Author)
Maria Keet, Free University of Bozen-Bolzano, Italy.

A technical report of the article[1]: *Jarrar, M., Keet, C.M., Dongilli, P.* ***Multilingual verbalization of ORM conceptual models and axiomatized ontologies****. [Submitted].*
URL: http://www.starlab.vub.ac.be/staff/mustafa/orm/verbalization/

Abstract. In the above-mentioned article we describe a novel approach to support *multilingual* verbalization of logical theories, axiomatizations, and other specifications such as business rules. This engineering solution is demonstrated with the Object Role Modeling language, although its underlying principles can be reused with other conceptual models and formal languages, such as Description Logics, to improve its understandability and usability by the domain expert. The engineering solution for multilingual verbalization is characterized by its flexibility, extensibility and maintainability of the verbalization templates, which allow for easy augmentation with other languages than the 11 currently supported.

This report presents the Dutch verbalization template file. Given an ORM schema (or an ORM-ML file), and given the verbalization template, a Dutch verbalization of the rules and fact types (in the schema) is generated automatically. A comprehensive example of an ORM schema and its corresponding verbalization is generated and given in this report.

1 Introduction

In the above-mentioned article, we present a novel approach to support multilingual verbalization of logical theories, formal axiomatizations, and other specifications such as business rules, ontologies, etc. We demonstrate our approach by providing a flexible and extensible verbalization template for the Object Role Modeling language. This template can be easily customized and translated into other human languages. *This technical report provides the Dutch verbalization of the ORM models and rules.* The verbalization of several other languages (including, but not limited to: German, Italian, Arabic, Russian, Spanish, French, and Lithuanian) can be found at the above-mentioned URL.

The underlying principles of our approach can be reused for other conceptual models and formal languages, such as Description Logics. The objective was to define a template parameterized over a given set of rules, models, or axioms, with as output fixed-syntax pseudo natural language sentences. A simple example is the following: the formal rule

$$\forall x\,(\mathrm{Book}(x) \rightarrow \exists y\,(\mathrm{ISBN}(y) \wedge \mathrm{Has}(x,y)))$$

can be translated into

It is mandatory that each **Book Has** an **ISBN**.

In this way, we enable domain experts themselves to build and/or validate the formal specifications of their domains, without having to know that these sentences are formal axioms; i.e. the underpinning logics and reasoning services are hidden from the user. Our approach with the provided templates can be

[1] For Citation use: *Jarrar, M., Keet, C.M.: A Dutch Verbalization Template for ORM conceptual models and rules. A technical report of the article: Jarrar, M., Keet, C.M., Dongilli, P. Multilingual verbalization of ORM conceptual models and axiomatized ontologies. [Submitted]*.

reused in modeling business rules, ontologies, knowledge bases, etc. See [H04] for a similar approach to ORM business rules verbalization.

In the following section, we present an ORM example followed by verbalization of all rules in this ORM schema. These verbalizations are generated *automatically*, according to the Dutch verbalization template presented in section 3. This approach is fully implemented and supported in the DogmaModeler ontology modeling tool [J05]. It is worth noting that DogmaModeler's automated verbalization has been used by tens of lawyers to build the Customer Complaint Ontology [J05][JVM03].

Remark on Modality: Our verbalization template can be adapted easily according to the application/reasoning scenario, whether it is used as integrity constraints, derivation/inference rules, business rules, etc. For example, the above mandatory constraint can be verbalized in different ways, such as: 1) Each **Book** must **Has** at least one **ISBN**. 2) Each **Book Has** some **ISBN** values. 3) If there is a **Book** then it **Has** an **ISBN** value. 4) A **Book** that does not **Has** an **ISBN** is not allowed. 5) If a **Book** does not **Has** an **ISBN** value then….

2 Example of an ORM Schema

We illustrate in one diagram most types of rules supported in ORM. Our article [JKD06] describes technical details on *how* our verbalization approach is implemented, see [H01] to know more about ORM, and [J05] to know about the DogmaModeler tool that we use to build and automatically verbalize ORM models.

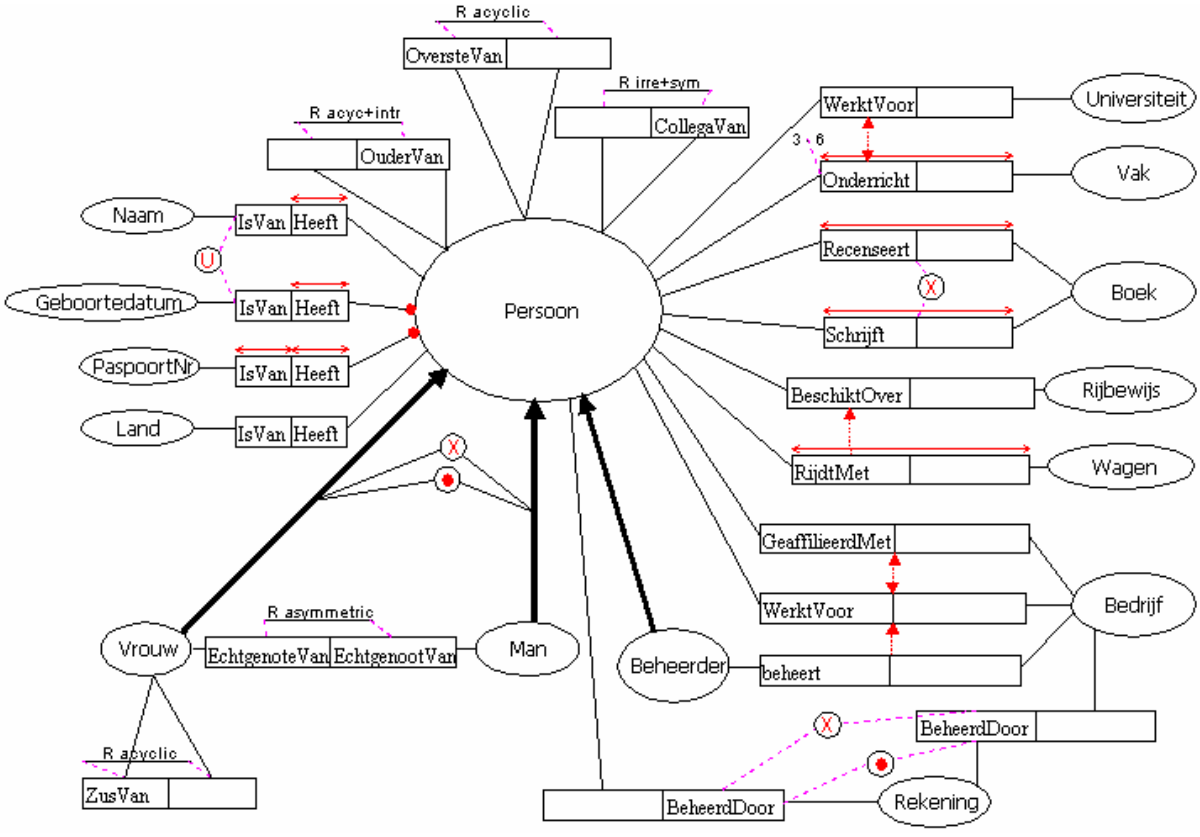

Fig. 1. Example of ORM rules, in Dutch.

The constraints/rules in the above ORM example are verbalized automatically (according to the verbalization template presented in section 2):

-[Mandatory] Elk(e) Persoon Heeft tenminste 1 PaspoortNr.
-[Mandatory] Elk(e) Persoon Heeft tenminste 1 Geboortedatum.
-[Mandatory] Elk(e) Rekening ofwel BeheerdDoor een Bedrijf ofwel BeheerdDoor een Persoon.
-[Uniqueness] Elk(e) Persoon Heeft ten hoogste 1 Geboortedatum.
-[Uniqueness] Elk(e) Persoon Heeft ten hoogste 1 Naam.
-[Uniqueness] Elk(e) Persoon Heeft ten hoogste 1 PaspoortNr.
-[Uniqueness] Elk(e) PaspoortNr IsVan ten hoogste 1 Persoon.
-[Uniqueness] Het is mogelijk dat een Persoon Onderricht meer dan 1 Vak , en omgekeerd .
-[Uniqueness] Het is mogelijk dat een Persoon Recenseert meer dan 1 Boek , en omgekeerd .
-[Uniqueness] Het is mogelijk dat een Persoon Schrijft meer dan 1 Boek , en omgekeerd .
-[Uniqueness] Het is mogelijk dat een Persoon RijdtMet meer dan 1 Wagen , en omgekeerd .
-[Uniqueness] Elke combinatie van { Naam en Geboortedatum } is gerelateerd met slechts 1 Persoon.
-[Exclusive] Elk(e) Persoon kan ofwel een Man ofwel een Vrouw zijn.
-[Totality] Elk(e) Persoon is tenminste een Man of een Vrouw.
-[Subset] Als een Persoon RijdtMet een Wagen dan moet ook dit/deze Persoon BeschiktOver een Rijbewijs.
-[Subset] Als een Beheerder beheert een Bedrijf dan moet ook dit/deze Persoon WerktVoor dat Bedrijf.
-[Equality] Een Persoon WerktVoor een Universiteit dan en slechts dan als dit/deze Persoon Onderricht een Vak.
-[Equality] Een Persoon GeaffilieerdMet een Bedrijf dan en slechts dan als dit/deze Persoon WerktVoor dat/die Bedrijf.
-[Exclusion] Geen enkel(e) Rekening BeheerdDoor een Bedrijf en ook BeheerdDoor een Persoon .
-[Exclusion] Geen Persoon Schrijft een Boek en ook Recenceert datzelfde Boek .
-[Value] De mogelijke instanties van Land zijn : {Belgie, Frankrijk, Duitsland}
-[Irreflexive] Geen enkel(e) Persoon CollegaVan zich/hem/haarzelf .
-[Symmetric] Indien Persoon X CollegaVan Persoon Y , dan ook vice versa .
-[Acyclic] Persoon kan niet rechtstreeks (of indirekt door een aaneenschakeling) OversteVan zichzelf/haarzelf zijn .
-[Acyclic] Vrouw kan niet rechtstreeks (of indirekt door een aaneenschakeling) ZusVan zichzelf/haarzelf zijn .
-[Asymmetric] Indien Vrouw X EchtgenoteVan Vrouw Y, dan kan het niet vice-versa .
-[Intransitive] Indien Persoon X OuderVan Persoon Y, en Y OuderVan Z , dan is het niet mogelijk dat X OuderVan Z.
-[Frequency] Indien Persoon Onderricht een Vak, dan deze/dit Persoon Onderricht tenminste 3 en ten hoogste 6 Vak.

3 The Dutch Verbalization Template

The template is presented in an XML syntax, it is being implemented in the DogmaModeler tool to support the Dutch verbalization of ORM models. More details about this approach can be found [JKD06], and refer to [J05] about DogmaModeler.

```
<?xml version='1.0' encoding='UTF-8'?>
<ORMSchema xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be'>

<ORMNLMeta>
 <Meta name="DC.Title" content="Dutch verbalization template (Ver0.2)"/>
 <Meta name="DC.Version" content="0.3"/>
 <Meta name="DC.Creator" content="Mustafa Jarrar"/>
```

```
   <Meta name="DC.Contributor" content="Maria Keet"/>
 <Meta name="DC.Language" content="Dutch"/>
 </ORMNLMeta>

<ORMNLBody>

<FactType xsi:type="FactType" >
<Text>Een</Text>
<Object index="0" />
<Role index="0"  />
<Text>/</Text>
<Role index="1"  />
<Text> een</Text>
<Object index="1" />
</FactType>

<Constraint xsi:type="Mandatory"  >
 <Text> -[Mandatory] Elk(e)</Text>
 <Object index="0"  />
 <Role index="0"  />
 <Text> tenminste 1</Text>
 <Object index="1"  />
</Constraint>

<Constraint xsi:type="Backward Mandatory"  >
 <Text> -[Mandatory] Voor elk(e)</Text>
 <Object index="0"  />
<Text>is er tenminste 1</Text>
<Object index="1"  />
 <Text> dat</Text>
<Role index="1"  />
 <Text> dit/deze</Text>
 <Object index="0"  />
</Constraint>

<Constraint xsi:type="Disjunctive Mandatory" >
<Text> -[Mandatory] Elk(e)</Text>
<Object index="0" />
<Text>ofwel</Text>
<Role index="0"   />
<Text>een</Text>
<Object index="1" />
<Loop index="1" >
  <Text>ofwel </Text>
  <Role index="n"   />
  <Text>een</Text>
  <Object index="1" />
</Loop>
</Constraint>

<Constraint xsi:type="Uniqueness" >
<Text> -[Uniqueness] Elk(e)</Text>
<Object index="0" />
<Role index="0"  />
<Text> ten hoogste 1 </Text>
<Object index="1" />
</Constraint>

<Constraint xsi:type="Backward Uniqueness"  >
 <Text> -[Uniqueness] Voor elke </Text>
<Object index="0" />
<Text>is er ten hoogste een </Text>
<Object index="1"  />
```

```
<Text> dat/die </Text>
<Role index="1"  />
<Text> dit/deze </Text>
<Object index="0"  />
</Constraint>

<Constraint xsi:type="External Uniqueness" >
<Text> -[Uniqueness] Elke combinatie van {</Text>
<Object index="1" />
<Loop index="1">
  <Text>en</Text>
  <Object index="n" />
</Loop>
<Text>} is gerelateerd met slechts 1</Text>
<Object index="0" />
</Constraint>

<Constraint xsi:type="Many Uniqueness" >
<Text> -[Uniqueness] Het is mogelijk dat een </Text>
<Object index="0" />
<Role index="0"></Role>
<Text>meer dan 1</Text>
<Object index="1" />
<Text>, en omgekeerd </Text>
</Constraint>

<Constraint xsi:type="Subtype" >
<Text> -[Subtype] Elk(e)</Text>
<Object index="child" />
<Text>is ook een</Text>
<Object index="parent"  />
</Constraint>

<Constraint xsi:type="Value">
 <Text> -[Value]  De mogelijke instanties van </Text>
 <Object index="0"/>
 <Text> zijn :{</Text>
 <Value index="0"/>
 <Loop index="1">
  <Text>,</Text>
  <Value index="n"/>
 </Loop>
 <Text> }</Text>
 </Constraint>

<Constraint xsi:type="Exclusive" >
<Text> -[Exclusive] Elk(e)</Text>
<Object index="0"/>
<Text> kan ofwel een</Text>
<Object index="1"/>
<Loop index="1">
  <Text>ofwel een</Text>
  <Object index="n" />
</Loop>
<Text>zijn</Text>
</Constraint>

<Constraint xsi:type="Total" >
<Text> -[Totality] Elk(e)</Text>
<Object index="0" />
<Text>is tenminste een</Text>
<Object index="1"  />
<Loop index="1" >
```

```
  <Text>of een</Text>
  <Object index="n" />
</Loop>
</Constraint>

<Constraint xsi:type="Subset" >
<Text> -[Subset] Als een</Text>
<Object  index="0" />
<Role index="child"  />
<Text>een</Text>
<Object index="child" />
<Text>dan moet ook  dit/deze</Text>
<Object index="1"  />
<Role index="parent"  />
<Text>een</Text>
<Object index="parent" />
</Constraint>

<Constraint xsi:type="Subset FactType" >
<Text> -[Subset] Als een  </Text>
<Object  index="0" />
<Role index="child"  />
<Text>een</Text>
<Object index="child" />
<Text>dan moet ook dit/deze </Text>
<Object index="1"  />
<Role index="parent"  />
<Text> dat </Text>
<Object index="parent" />
</Constraint>

<Constraint xsi:type="Equality" >
<Text> -[Equality] Een </Text>
<Object index="0"  />
<Role index="first"  />
<Text>een </Text>
<Object index="first"  />
<Text>dan en slechts dan als</Text>
<Text>dit/deze </Text>
<Object index="0" />
<Role index="second"  />
<Text>een </Text>
<Object index="second"  />
</Constraint>

<Constraint xsi:type="Equality FactType" >
<Text> -[Equality] Een</Text>
<Object index="0"  />
<Role index="first"  />
<Object index="first"  />
<Text>dan en slechts dan als</Text>
<Text>dit/deze </Text>
<Object index="1"  />
<Role index="second"  />
<Text>dat/die</Text>
<Object index="second"  />
</Constraint>

<Constraint xsi:type="Exclusion" >
<Text> -[Exclusion] Geen enkel(e) </Text>
<Object index="0"  />
<Role index="first"  />
<Text>een </Text>
```

```
<Object index="first"  />
<Text>en ook</Text>
<Role index="second"  />
<Text>een </Text>
<Object index="second"  />
</Constraint>

<Constraint xsi:type="Exclusion FactType" >
<Text> -[Exclusion] Geen </Text>
<Object index="0"  />
<Role index="first"  />
<Text>een </Text>
<Object index="first"  />
<Text>en ook</Text>
<Role index="second"  />
<Text>datzelfde </Text>
<Object index="second"  />
</Constraint>

<Constraint xsi:type="Frequency">
 <Text> -[Frequency] If </Text>
 <Object index="0"/>
 <Role index="0"/>
 <Object index="1"/>
 <Role index="0"/>
 <Text>, then this </Text>
 <Object index="0"/>
 <Role index="0"/>
 <Text>at least </Text>
 <Minimum/>
 <Text> and most most </Text>
 <Maximum/>
 <Role index="0"/>
 <Text>(s)</Text>
</Constraint>

<Constraint xsi:type="Irreflexive">
 <Text> -[Irreflexive] Geen enkel(e) </Text>
 <Object index="0"/>
 <Role index="0"/>
 <Text> zich/hem/haarzelf</Text>
</Constraint>

<Constraint xsi:type="Symmetric"  >
<Text>-[Symmetric] Indien</Text>
<Object index="0"/>
<Text> X</Text>
<Role index="0"/>
<Object index="0"/>
<Text> Y</Text>
<Text> , dan ook vice-versa</Text>
</Constraint>

<Constraint xsi:type="Asymmetric">
 <Text> -[Asymmetric] Indien </Text>
 <Object index="0"/>
 <Text> X</Text>
 <Role index="0"/>
 <Text> </Text>
 <Object index="parent"/>
<Text> Y, dan kan het niet vice-versa</Text>
</Constraint>
```

```
<Constraint xsi:type="Acyclic">
 <Text> -[Acyclic]</Text>
 <Object index="0"/>
 <Text> kan niet rechtstreeks (of indirekt door een aaneenschakeling)</Text>
 <Role index="0"/>
 <Text> zichzelf/haarzelf zijn</Text>
</Constraint>

<Constraint xsi:type="Transitve">
 <Text> -[Intransitve] Indien</Text>
 <Object index="0"/>
 <Text>X</Text>
 <Role index="0"/>
 <Object index="0"/>
 <Text>Y, en Y</Text>
 <Role index="0"/>
 <Text> Z, dan is het niet mogelijk dat X</Text>
 <Role index="0"/>
 <Text>Z</Text>
</Constraint>

</ORMNLBody>
</ORMSchema>
```

Acknowledgments

We are in debt to Andriy Lisovoy who helped in the implementation of DogmaModeler, and to Hai Nguyen Hoang who helped in the first implementation of the verbalization component during his Master thesis. This work is partially supported by the EU Knowledge Web NoE project (IST-2004-507482).

References

[JKD06] Jarrar, M., Keet, C.M., Dongilli, P. Multilingual verbalization of ORM conceptual models and axiomatized ontologies. [Submitted].

[J05] Jarrar, M.: Towards Methodological Principles for Ontology Engineering. PhD thesis, Vrije Universiteit Brussel, 2005.

[JVM03] Jarrar, M., Verlinden, R., Meersman, R.: Ontology-based Customer Complaint Management. In: Jarrar M., Salaun A., (eds.): Proceedings of the workshop on regulatory ontologies and the modeling of complaint regulations, Catania, Sicily, Italy. Springer Verlag LNCS. Vol. 2889. November (2003) pp. 594–606

[H01] Halpin, T.: Information Modeling and Relational Databases. 3rd ed. Morgan-Kaufmann. (2001)

[H04] Halpin, T.: Business Rule Verbalization. In Doroshenko, A., Halpin, T., Liddle, S., Mayr H. (eds): Information Systems Technology and its Applications, 3rd International Conference (ISTA'2004), LNI 48 GI ISBN 3-88579-377-6, (2004) pp:39-52.