A German Verbalization Template
for ORM conceptual models and rules

Mustafa Jarrar, Vrije Universiteit Brussel, Belgium. (Contact Author)
Maria Keet, Free University of Bozen-Bolzano, Italy.
Paolo Dongilli, 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 German verbalization template file. Given an ORM schema (or an ORM-ML file), and given the verbalization template, a German 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 German verbalization of the ORM models and rules.* The verbalization of several other languages (including, but not limited to: Italian, Arabic, Russian, Spanish, Dutch, 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

[1] For Citation use: *Jarrar, M., Keet, C.M., Dongilli, P.: A German 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].*

logics and reasoning services are hidden from the user. Our approach with the provided templates can be 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 German 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.

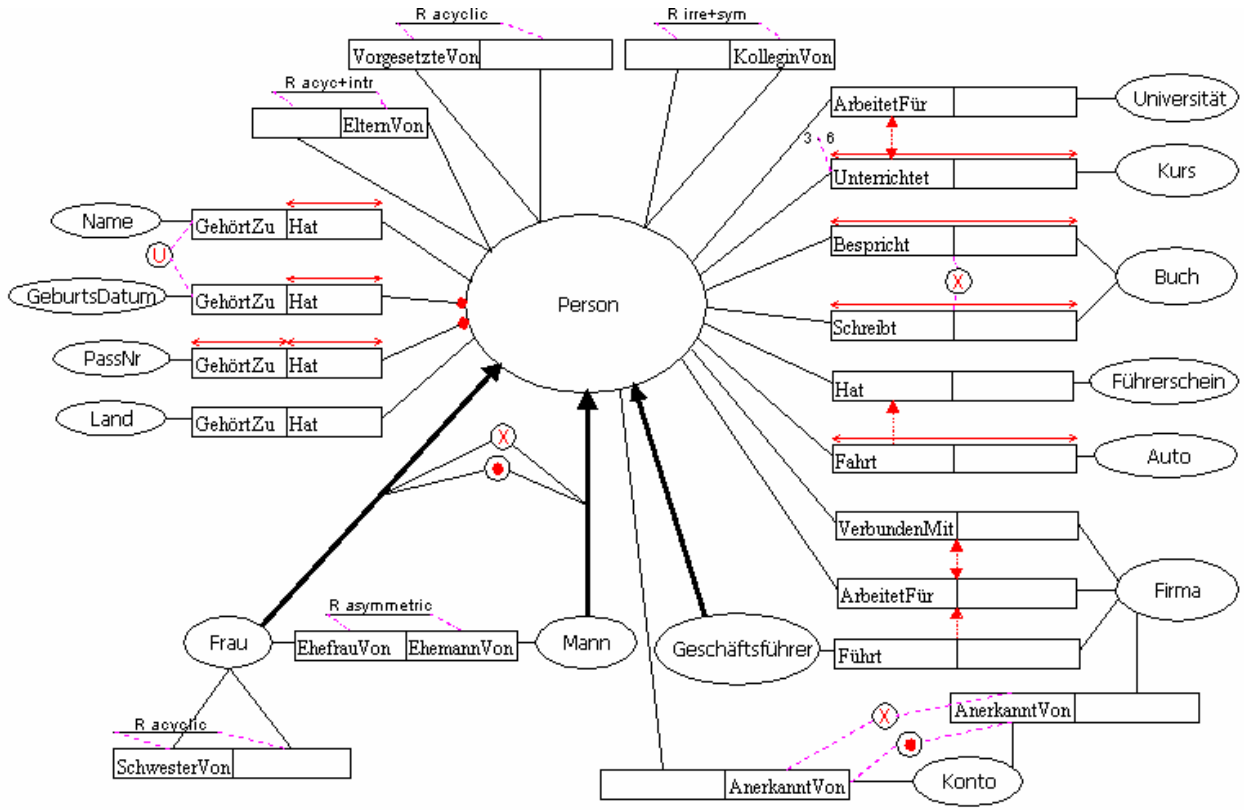

Fig. 1. Example of ORM rules, in German.

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

-[Mandatory] Jeder/s Person Hat mindestens 1 PassNr.
-[Mandatory] Jeder/s Person Hat mindestens 1 GeburtsDatum.
-[Mandatory] Jeder/s Konto entweder AnerkanntVon ein Person oder AnerkanntVon ein Firma.
-[Uniqueness] Jeder/s Person Hat höchstens 1 GeburtsDatum.
-[Uniqueness] Jeder/s Person Hat höchstens 1 Name.
-[Uniqueness] Jeder/s Person Hat höchstens 1 PassNr.
-[Uniqueness] Jeder/s PassNr GehörtZu höchstens 1 Person.
-[Uniqueness] Es ist möglich das ein Person mehr als 1 Kurs Unterrichtet , und umgekehrt .
-[Uniqueness] Es ist möglich das ein Person mehr als 1 Buch Bespricht , und umgekehrt .
-[Uniqueness] Es ist möglich das ein Person mehr als 1 Buch Schreibt , und umgekehrt .
-[Uniqueness] Es ist möglich das ein Person mehr als 1 Auto Fahrt , und umgekehrt .
-[Uniqueness] Jeder Kombination von { GeburtsDatum und Name } ist verbunden mit nur 1 Person.
-[Exclusive] Jeder/s Person kann entweder ein Mann oder ein Frau sein.
-[Totality] Jeder/s Person ist mindestens ein Frau oder ein Mann.
-[Subset] Wenn ein Person Fahrt ein Auto , dann soll diese Person auch Hat ein Führerschein.
-[Subset] Wenn ein Geschäftsführer ein Firma Führt , dann soll diese Person auch ArbeitetFür der/den/dem Firma.
-[Equality] Ein Person ArbeitetFür ein Universität nur wenn diese Person ein Kurs Unterrichtet.
-[Equality] Ein Person VerbundenMit Firma nur wenn diese Person ArbeitetFür der/dem/den Firma.
-[Exclusion] Kein Konto ist AnerkanntVon ein Person und auch AnerkanntVon ein Firma.
-[Exclusion] Kein Person Bespricht ein Buch und auch Schreibt das/der/dieselbe Buch.
-[Value] Die mögliche Instanzen von Land sind : {Belgien, Frankreich, Deutschland}.
-[Irreflexive] Kein einzige Person KolleginVon sich selbst.
-[Symmetric] Als Person X KolleginVon Person Y, dann auch umgekehrt.
-[Acyclic] Person kann nicht direkt (oder indirekt via eine Verkettung) VorgesetzteVon sich selbst sein
-[Acyclic] Frau kann nicht direkt (oder indirekt via eine Verkettung) SchwesterVon sich selbst sein .
-[Asymmetric] Als Frau X EhefrauVon Frau Y, dann ist umgekehrt unmöglich .
-[Intransitive] Als Person X ElternVon Person Y, und Y ElternVon Z, dann ist es nicht möglich das X ElternVon Z.
-[Frequency] Wenn ein Person Unterrichtet ein Kurs, dann diese Person Unterrichtet mindestens 3 und höchstens 6 Kurs.

3 The German Verbalization Template

The template is presented in an XML syntax, it is being implemented in the DogmaModeler tool to support the German 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/staff/mustafa/orm/verbaliz
ation/'>

<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="C. Maria Keet"/>
<Meta name="DC.Language" content="German"/>
```

```
</ORMNLMeta>

<ORMNLBody>

<FactType xsi:type="FactType" >
<Text>Ein(e)</Text>
<Object index="0" />
<Role index="0" />
<Text>/</Text>
<Role index="1" />
<Text> ein(e)</Text>
<Object index="1" />
</FactType>

<Constraint xsi:type="Mandatory" >
<Text> -[Mandatory] Jeder/s</Text>
<Object index="0" />
<Role index="0" />
<Text> mindestens 1</Text>
<Object index="1" />
</Constraint>

<Constraint xsi:type="Backward Mandatory" >
<Text> -[Mandatory] Für jeder/s</Text>
<Object index="0" />
<Text>gibt es mindestens 1</Text>
<Object index="1" />
<Text> das</Text>
<Role index="1" />
<Text> diesen/m</Text>
<Object index="0" />
</Constraint>

<Constraint xsi:type="Disjunctive Mandatory" >
<Text> -[Mandatory] Jeder/s</Text>
<Object index="0" />
<Text>entweder</Text>
<Role index="0" />
<Text>ein</Text>
<Object index="1" />
<Loop index="1" >
<Text>oder </Text>
<Role index="n" />
<Text>ein</Text>
<Object index="1" />
</Loop>
</Constraint>

<Constraint xsi:type="Uniqueness" >
<Text> -[Uniqueness] Jeder/s </Text>
<Object index="0" />
<Role index="0" />
<Text> höchstens 1 </Text>
<Object index="1" />
</Constraint>

<Constraint xsi:type="Backward Uniqueness" >
<Text> -[Uniqueness] Für jeder/s </Text>
<Object index="0" />
<Text> gibt es höchstens ein </Text>
<Object index="1" />
<Text> das/wer </Text>
<Role index="1" />
```

```
<Text> diese/r/m </Text>
<Object index="0" />
</Constraint>

<Constraint xsi:type="External Uniqueness" >
<Text> -[Uniqueness] Jeder Kombination von { </Text>
<Object index="1" />
<Loop index="1">
<Text>und</Text>
<Object index="n" />
</Loop>
<Text>} ist verbunden mit nur 1</Text>
<Object index="0" />
</Constraint>

<Constraint xsi:type="Many Uniqueness" >
<Text> -[Uniqueness] Es ist möglich das ein </Text>
<Object index="0" />
<Text>mehr als 1</Text>
<Object index="1" />
<Role index="0"></Role>
<Text>, und umgekehrt </Text>
</Constraint>
<Constraint xsi:type="Subtype" >
<Text> -[Subtype] Jeder/s</Text>
<Object index="child" />
<Text>ist auch ein </Text>
<Object index="parent" />
</Constraint>

<Constraint xsi:type="Value">
 <Text> -[Value]  Die mögliche Instanzen von </Text>
 <Object index="0"/>
 <Text> sind :{</Text>
 <Value index="0"/>
 <Loop index="1">
  <Text>,</Text>
  <Value index="n"/>
 </Loop>
 <Text> }</Text>
 </Constraint>

<Constraint xsi:type="Exclusive" >
<Text> -[Exclusive] Jeder/s</Text>
<Object index="0"/>
<Text> kann entweder ein </Text>
<Object index="1"/>
<Loop index="1">
<Text>oder ein</Text>
<Object index="n" />
</Loop>
<Text>sein</Text>
</Constraint>

<Constraint xsi:type="Total" >
<Text> -[Total] Jeder/s</Text>
<Object index="0" />
<Text>ist mindestens ein</Text>
<Object index="1" />
<Loop index="1" >
<Text>oder ein</Text>
<Object index="n" />
</Loop>
```

```
</Constraint>

<Constraint xsi:type="Subset" >
<Text> -[Subset] Wenn ein</Text>
<Object index="0" />
<Role index="child" />
<Text>ein</Text>
<Object index="child" />
<Text>, dann soll diese </Text>
<Object index="1" />
<Text>auch</Text>
<Role index="parent" />
<Text>ein</Text>
<Object index="parent" />
</Constraint>

<Constraint xsi:type="Subset FactType" >
<Text> -[Subset] Wenn ein </Text>
<Object index="0" />
<Text>ein</Text>
<Object index="child" />
<Role index="child" />
<Text>, dann soll diese </Text>
<Object index="1" />
<Text>auch</Text>
<Role index="parent" />
<Text> der/den/dem </Text>
<Object index="parent" />
</Constraint>

<Constraint xsi:type="Equality" >
<Text> -[Equality] Ein </Text>
<Object index="0" />
<Role index="first" />
<Text>ein </Text>
<Object index="first" />
<Text>nur wenn </Text>
<Text>diese </Text>
<Object index="0" />
<Text>ein </Text>
<Object index="second" />
<Role index="second" />
</Constraint>

<Constraint xsi:type="Equality FactType" >
<Text> -[Equality] Ein</Text>
<Object index="0" />
<Role index="First" />
<Object index="First" />
<Text>nur wenn</Text>
<Text>diese </Text>
<Object index="1" />
<Role index="Second" />
<Text>der/dem/den</Text>
<Object index="Second" />
</Constraint>

<Constraint xsi:type="Exclusion" >
<Text> -[Exclusion] Kein </Text>
<Object index="0" />
<Text> ist </Text>
<Role index="first" />
<Text> ein </Text>
```

```
<Object index="first" />
<Text>und auch</Text>
<Role index="second" />
<Text>ein </Text>
<Object index="second" />
</Constraint>

<Constraint xsi:type="Exclusion FactType" >
<Text> -[Exclusion] Kein </Text>
<Object index="0" />
<Role index="first" />
<Text> ein </Text>
<Object index="first" />
<Text>und auch</Text>
<Role index="second" />
<Text>das/der/dieselbe </Text>
<Object index="second" />
</Constraint>

<Constraint xsi:type="Frequency">
<Text> -[Frequency] Wenn ein </Text>
<Object index="0"/>
<Role index="0"/>
<Object index="1"/>
<Role index="0"/>
<Text>, dann diese </Text>
<Object index="0"/>
<Role index="0"/>
<Text>mindestens </Text>
<Minimum/>
<Text> und höchstens </Text>
<Maximum/>
<Role index="0"/>
<Text>(s)</Text>
</Constraint>

<Constraint xsi:type="Irreflexive">
<Text> -[Irreflexive] Kein einzige</Text>
<Object index="0"/>
<Role index="0"/>
<Text> sich selbst</Text>
</Constraint>

<Constraint xsi:type="Symmetric" >
<Text>-[Symmetric] Als</Text>
<Object index="0"/>
<Text> X</Text>
<Role index="0"/>
<Object index="0"/>
<Text> Y</Text>
<Text> , dann auch umgekehrt</Text>
</Constraint>

<Constraint xsi:type="Asymmetric">
<Text> -[Asymmetric] Als</Text>
<Object index="0"/>
<Text> X</Text>
<Role index="0"/>
<Text> </Text>
<Object index="0"/>
<Text> Y, dann ist umgekehrt unmöglich</Text>
</Constraint>
```

```
<Constraint xsi:type="Acyclic">
<Text> -[Acyclic]</Text>
<Object index="0"/>
<Text> kann nicht direkt (oder indirekt via eine Verkettung)</Text>
<Role index="0"/>
<Text> sich selbst sein</Text>
</Constraint>

<Constraint xsi:type="Transitve">
<Text> -[Intransitve] Als</Text>
<Object index="0"/>
<Text>X</Text>
<Role index="0"/>
<Object index="0"/>
<Text>Y, und Y</Text>
<Role index="0"/>
<Text> Z, dann ist es nicht möglich das 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.