An Italian Verbalization Template
for ORM conceptual models and rules

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

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

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., Dongilli, P. Keet, C.M.: An Italian 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 Italian 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.

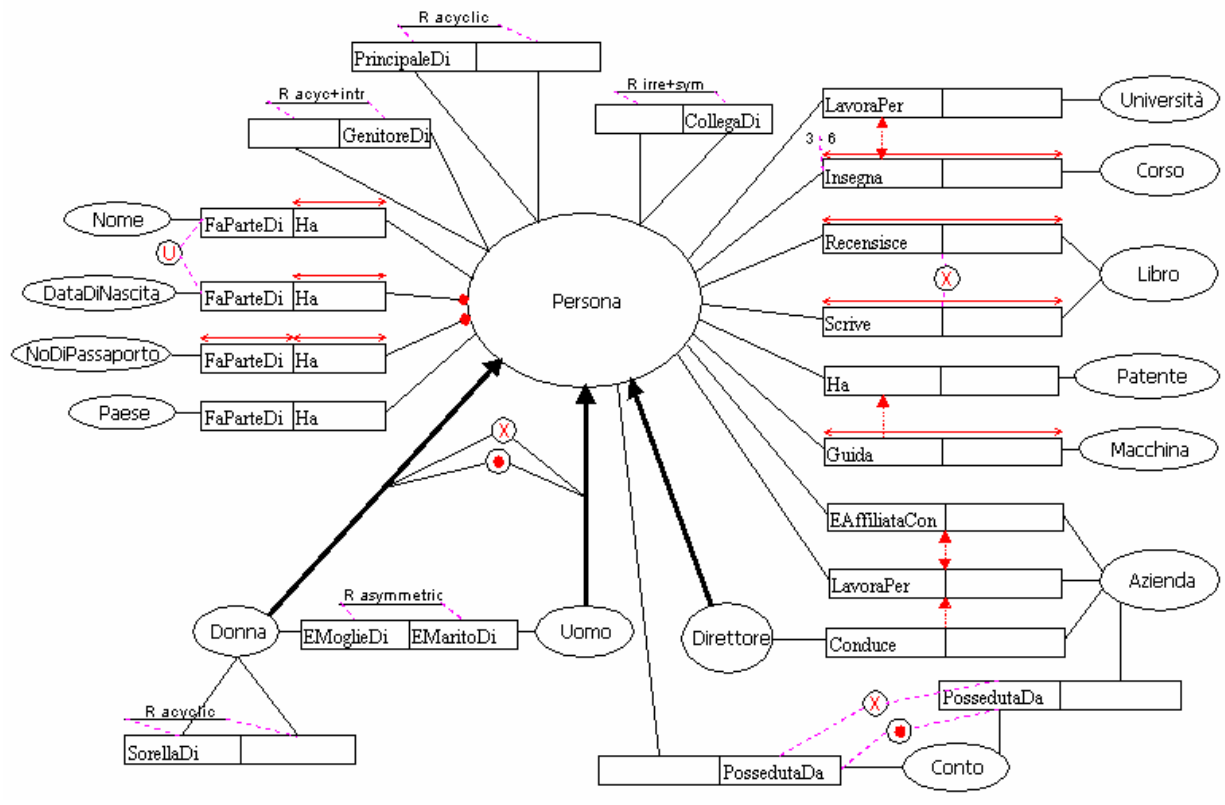

Fig. 1. Example of ORM rules, in Italian.

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

-[Mandatory] Ogni Persona deve Ha almeno un NoDiPassaporto.
-[Mandatory] Ogni Persona deve Ha almeno un DataDiNascita.
-[Mandatory] Ogni Conto deve essere PossedutaDa Azienda o PossedutaDa Persona.
-[Uniqueness] Ogni Persona deve Ha al massimo un(a) DataDiNascita.
-[Uniqueness] Ogni Persona deve Ha al massimo un(a) Nome.
-[Uniqueness] Ogni Persona deve Ha al massimo un(a) NoDiPassaporto.
-[Uniqueness] Ogni NoDiPassaporto deve FaParteDi al massimo un(a) Persona.
-[Uniqueness] E' possibile che un(a) Persona Insegna più di un(a) Corso , e viceversa.
-[Uniqueness] E' possibile che un(a) Persona Recensisce più di un(a) Libro , e viceversa.
-[Uniqueness] E' possibile che un(a) Persona Scrive più di un(a) Libro , e viceversa.
-[Uniqueness] E' possibile che un(a) Persona Guida più di un(a) Macchina , e viceversa.
-[Uniqueness] La combinazione di { Nome e DataDiNascita } deve essere riferito a al massimo un(a) Persona.
-[Exclusive] Ogni Persona deve essere o Uomo o Donna.
-[Totality] Ogni Persona deve essere almeno o Donna o Uomo.
-[Subset] Se un(a) Persona Guida un(a) Macchina, in questo caso il/lo/la Persona in questione Ha un(a) Patente.
-[Subset] Se un(a) Direttore Conduce un(a) Azienda, in questo caso il/lo/la Direttore in questione LavoraPer questo(a) Azienda.
-[Equality] Un(a) Persona LavoraPer un(a) Università se e solo se questo(a) Persona Insegna un(a) Corso.
-[Equality] Un(a) Persona EAffiliataCon un(a) Azienda se e solo se questo(a) Persona LavoraPer questo(a) Azienda.
-[Exclusion] Nessun(a) Conto PossedutaDa un(a) Azienda ed anche PossedutaDa Persona.
-[Exclusion] Nessun(a) Persona Scrive un(a) Libro ed anche Recensisce lo/la stesso(a) Libro..
-[Value] Gli oggetti possibili di Paese sono: {Belgio, Francia, Germania }.
-[Irreflexive] Nessun Persona CollegaDi se stesso.
-[Symmetric] Se Persona X CollegaDi Persona Y, deve essere viceversa.
-[Acyclic] Un(a) Persona non può essere diretto (o indiretto per una catena) PrincipaleDi se stesso .
-[Acyclic] Un(a) Donna non può essere diretto (o indiretto per una catena) SorellaDi se stesso .
-[Asymmetric] Se Donna X EMoglieDi Donna Y, viceversa è impossibile .
-[Intransitive] Se Persona X GenitoreDi Persona Y, e Y GenitoreDi Z, in quel caso non è possibile che X GenitoreDi Z.
-[Frequency] Se Persona Insegna un(a) Corso, in questo caso il/lo/la Persona in questione Insegna almeno 3 ed al massimo 6 Corso.

3 The Italian Verbalization Template

The template is presented in an XML syntax, it is being implemented in the DogmaModeler tool to support the Italian 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="Italian verbalization template (Ver0.1)"/>
```

```
 <Meta name="DC.Version" content="0.1"/>
 <Meta name="DC.Creator" content="Mustafa Jarrar"/>
 <Meta name="DC.Contributor" content="Maria Keet"/>
 <Meta name="DC.Language" content="Italian"/>
 </ORMNLMeta>

<ORMNLBody>

<FactType xsi:type="FactType" >
<Text>Un(a)</Text>
<Object index="0" />
<Role index="0" />
<Text>/</Text>
<Role index="1" />
<Text> un(a)</Text>
<Object index="1" />
</FactType>

<Constraint xsi:type="Mandatory">
 <Text> -[Mandatory]  Ogni</Text>
 <Object index="0"/>
 <Text>deve</Text>
 <Role index="0"/>
 <Text>almeno un</Text>
 <Object index="1"/>
</Constraint>

<Constraint xsi:type="Backward Mandatory">
 <Text> -[M]  Per ogni</Text>
 <Object index="0"/>
 <Text>c'è almeno un</Text>
 <Object index="1"/>
 <Text>che</Text>
 <Role index="1"/>
 <Text>questo(a)</Text>
 <Object index="0"/>
</Constraint>

<Constraint xsi:type="Disjunctive Mandatory">
 <Text> -[Mandatory]  Ogni</Text>
 <Object index="0"/>
 <Text>deve essere</Text>
 <Role index="0"/>
 <Object index="1"/>
 <Loop index="1" >
  <Text>o</Text>
  <Role index="n"/>
  <Object index="n"/>
 </Loop>
</Constraint>

<Constraint xsi:type="Uniqueness">
 <Text> -[Uniqueness]  Ogni</Text>
 <Object index="0"/>
 <Text>deve</Text>
 <Role index="0"/>
 <Text>al massimo un(a)</Text>
 <Object index="1"/>
</Constraint>

<Constraint xsi:type="Backward Uniqueness">
 <Text> -[Uniqueness]   Per ogni</Text>
 <Object index="0"/>
```

```
 <Text>deve esssere al massimo un(a)</Text>
 <Object index="1"/>
 <Text>che</Text>
 <Role index="1"/>
 <Text>questo</Text>
 <Object index="0"/>
</Constraint>

<Constraint xsi:type="Many Uniqueness">
 <Text> -[Uniqueness]  E' possibile che un(a)</Text>
 <Object index="0"/>
<Role index="0"></Role>
 <Text> più di un(a) </Text>
 <Object index="1"/>
 <Text>, e viceversa</Text>
</Constraint>

<Constraint xsi:type="External Uniqueness">
 <Text> -[Uniqueness]  La combinazione di {</Text>
 <Object index="1"/>
 <Loop index="1">
  <Text>e</Text>
  <Object index="n"/>
 </Loop>
 <Text>} deve essere riferito a al massimo un(a) </Text>
 <Object index="0"/>
</Constraint>

<Constraint xsi:type="Subtype">
 <Text> -[Subtype]  Ogni oggetto </Text>
 <Object index="child"/>
 <Text>è anche un oggetto di </Text>
 <Object index="parent"/>
</Constraint>

<Constraint xsi:type="Value">
 <Text> -[Value]  Gli oggetti possibili di </Text>
 <Object index="0"/>
 <Text> sono :{</Text>
 <Value index="0"/>
 <Loop index="1">
  <Text>,</Text>
  <Value index="n"/>
 </Loop>
 <Text> }</Text>
 </Constraint>

<Constraint xsi:type="Exclusive">
<Text> -[Exclusive]  Ogni</Text>
<Object index="0"/>
<Text>deve essere o </Text>
<Object index="1"/>
<Loop index="1">
  <Text>o</Text>
  <Object index="n"/>
</Loop>
</Constraint>

<Constraint xsi:type="Total">
 <Text> -[Totality]  Ogni</Text>
 <Object index="0"/>
 <Text>deve essere almeno o </Text>
 <Object index="1"/>
```

```
 <Loop index="1">
  <Text>o</Text>
  <Object index="n"/>
 </Loop>
</Constraint>

<Constraint xsi:type="Partition">
 <Text> -[Partition]  Ogni</Text>
 <Object index="0"/>
 <Text>e' almeno uno di </Text>
 <Object index="1"/>
 <Loop index="1">
  <Text>o</Text>
  <Object index="n"/>
 </Loop>
 <Text>, pero non tutti</Text>
</Constraint>

<Constraint xsi:type="Subset">
 <Text> -[Subset] Se un(a)</Text>
 <Object index="0"/>
 <Role index="child"/>
 <Text>un(a)</Text>
 <Object index="child"/>
 <Text>, in questo caso il/lo/la</Text>
 <Object index="0"/>
 <Text>in questione</Text>
 <Role index="parent"/>
 <Text>un(a)</Text>
 <Object index="parent"/>
</Constraint>

<Constraint xsi:type="Subset FactType">
 <Text> -[Subset]  Se un(a)</Text>
 <Object index="0"/>
 <Role index="child"/>
 <Text>un(a)</Text>
 <Object index="child"/>
 <Text>, in questo caso il/lo/la</Text>
 <Object index="0"  />
 <Text>in questione</Text>
 <Role index="parent"/>
 <Text>questo(a)</Text>
 <Object index="parent"/>
</Constraint>

<Constraint xsi:type="Equality">
 <Text> -[Equality] Un(a) </Text>
 <Object index="0"/>
 <Role index="first"/>
 <Text> un(a) </Text>
 <Object index="first"/>
 <Text>se e solo se</Text>
 <Text>questo(a) </Text>
 <Object index="0"/>
 <Role index="second"/>
 <Text> un(a) </Text>
 <Object index="second"/>
</Constraint>

<Constraint xsi:type="Equality FactType">
 <Text> -[Equality] Un(a) </Text>
 <Object index="0"/>
```

```
 <Role index="First"/>
 <Text>un(a)</Text>
 <Object index="First"/>
 <Text>se e solo se</Text>
 <Text>questo(a)</Text>
 <Object index="1"/>
 <Role index="Second"/>
 <Text>questo(a)</Text>
 <Object index="Second"/>
</Constraint>

<Constraint xsi:type="Exclusion">
 <Text> -[Exclusion]  Nessun(a) </Text>
 <Object index="0"/>
 <Role index="first"/>
 <Text>un(a)</Text>
 <Text> </Text>
 <Object index="first"/>
 <Text> ed anche </Text>
 <Role index="second"/>
 <Text> </Text>
 <Object index="second"/>
</Constraint>

<Constraint xsi:type="Exclusion FactType">
 <Text> -[Exclusion]  Nessun(a)</Text>
 <Object index="0"/>
 <Role index="first"/>
 <Text>un(a)</Text>
 <Object index="first"/>
 <Text>ed anche</Text>
 <Role index="second"/>
 <Text>lo/la stesso(a)</Text>
 <Object index="second"/>
</Constraint>

<Constraint xsi:type="Frequency">
 <Text> -[Frequency] Se </Text>
 <Object index="0"/>
 <Role index="0"/>
 <Text>un(a)</Text>
 <Object index="1"/>
 <Role index="0"/>
 <Text>, in quel caso il/lo/la</Text>
 <Object index="0"/>
 <Text>in questione</Text>
 <Role index="0"/>
 <Text>almeno </Text>
 <Minimum/>
 <Text> ed al massimo </Text>
 <Maximum/>
 <Role index="0"/>
 <Text>i/e</Text>
</Constraint>

<Constraint xsi:type="Irreflexive">
 <Text> -[Irreflexive]  Nessun</Text>
 <Object index="0"/>
 <Role index="0"/>
 <Text> se stesso</Text>
</Constraint>

<Constraint xsi:type="Symmetric">
```

```
 <Text> -[Symmetric]  Se </Text>
 <Object index="0"/>
 <Text>X</Text>
 <Role index="0"/>
 <Object index="0"/>
 <Text>Y</Text>
 <Text>, deve essere viceversa</Text>
</Constraint>

<Constraint xsi:type="Asymmetric">
 <Text> -[Asymmetric]  Se</Text>
 <Object index="0"/>
 <Text> X</Text>
 <Role index="0"/>
 <Text> </Text>
 <Object index="0"/>
<Text> Y, viceversa è impossibile</Text>
</Constraint>

<Constraint xsi:type="Acyclic">
 <Text> -[Acyclic] Un(a)</Text>
 <Object index="0"/>
 <Text> non può essere diretto (o indiretto per una catena)</Text>
 <Role index="0"/>
 <Text> se stesso</Text>
</Constraint>

<Constraint xsi:type="Transitve">
 <Text> -[Intransitve]  Se</Text>
 <Object index="0"/>
 <Text>X</Text>
 <Role index="0"/>
 <Object index="0"/>
 <Text>Y, and Y</Text>
 <Role index="0"/>
 <Text> Z, in quel caso non e' possibile che 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.