

ORM Markup Language, version 3

Mustafa Jarrar
STAR Lab, Vrije Universiteit Brussel, Belgium,
mjarrar@vub.ac.be

Published as: Mustafa Jarrar: ORM Markup Language, version 3 (Technical Report). Vrije Universiteit Brussel, Belgium. 2007

Abstract. In this technical report, we define a conceptual markup language (ORM-ML) for the ORM graphical notation. It is an extension of the ORM-ML version 2 that has been published in [J05]. The first version ORM-ML appeared in [JDM03] and [DJM03]. This version provides an improved version of the previous versions, allows representing modular schemes, and introduces a decent set of metadata elements. Section 1 provides a brief introduction and discuss our motives for constructing the ORM markup language, the ORM markup language itself is presented in section 2. To end, section 3 draws some conclusions and summarizes the main advantages and usage of ORM-ML in a general sense.

Contents

1 Introduction and motivation	2
1.1 Why ORM.....	2
2 ORM-Markup Language	3
2.1 ORM-ML metadata.....	4
2.2 ORM-ML Body.....	5
Object Types	5
Subtypes	6
Predicates	6
Predicate Objects.....	6
Constraints.....	7
Modular schemes.....	10
3 Discussion and conclusions	11
Appendix A: ORM Markup Language	13
Appendix A1 (tree view of the ORM-ML XML-Schema).....	13
Appendix A2 (ORM-ML XML-Schema)	14
Appendix A3: Complete Example	21
Appendix A4: ORM-ML Metadata.....	24
References:	27

1 Introduction and motivation

Indeed, successful conceptual data modeling approaches, such as ORM or EER, became well known because of their methodological guidance in building conceptual models of information systems. They are semantically rich disciplines and support quality checks at a high level of abstraction [V82] and they provide modeling constructs like integrity, taxonomy, and derivation rules [H01] [F02]. Merely, conceptual data schemes -also called semantic data models - were developed to capture the meaning of an application domain as perceived by its developers [WSW99] [M99a]. This meaning is being represented in diagram formats (which are proprietary and therefore are limited to use inside specific CASE tools), and typically used in an *off-time mode*, i.e. used during the design phases. Nowadays, the Internet and the open connectivity environments create a strong demand for sharing and exchanging not only data but also data semantics. By defining a conceptual markup language (ORM-ML) that allows for the representation of ORM conceptual diagrams in an open, textual syntax, we enable ORM schemes to be shared, exchanged, and processed at run-time.

1.1 Why ORM

ORM (Object-Role Modeling) [H01] is a conceptual modeling approach that was developed in the early 70's. It is a successor of the NIAM (Natural-language Information Analysis Method) [VB82]. Based on ORM, several conceptual modeling tools exist, such as Microsoft's VisioModeler™ and the older InfoModeler. This has the functionality of modeling a certain Universe of Discourse (UoD) in ORM while supporting the automatic generation of a consistent and normalized relational database schema.

ORM schemas can be translated into pseudo natural language statements. The graphical representation and the translation into pseudo natural language make it a lot easier, also for non-computer scientists, to create, check and adapt the knowledge about the UoD needed in an information system.

The ORM conceptual schema methodology is fairly comprehensive in its treatment of many "practical" or "standard" business rules and constraint types. Its detailed formal description, (we shall take ours from [H01][H89]) makes it an interesting candidate to non-trivially illustrate our XML based ORM-markup language as an exchange protocol for representing ORM conceptual models.

Of course, similar to ORM-ML, a markup language could be defined for any other conceptual modeling method.

ORM is fairly comprehensive in its treatment of many "practical" and "standard" rules, (e.g. identity, mandatory, uniqueness, subtyping, subset, equality, exclusion, frequency, transitive, acyclic, etc.). Furthermore,

ORM has an expressive and stable graphical notation since it captures many rules graphically and it minimizes the impact of change on the models¹. ORM has well-defined formal semantics (see e.g. [H89] [BHW91] [HPW93] [T96] [TM95] [HP95]). In addition, it is perhaps worthwhile to note that ORM derives from NIAM (Natural Language Information Analysis Method), which was explicitly designed to play the role of a stepwise methodology, to arrive at the "semantics" of a business application's data based on natural language communication.

2 ORM-Markup Language

This section presents the ORM markup language (ORM-ML). ORM-ML is based on the XML syntax, and is defined in an XML-Schema (provided in Appendix A) that acts as its complete and formal grammar. Hence, any ORM-ML document should be valid according to this XML-Schema.

ORM-ML is not meant to be written by hand or interpreted by humans. It is meant to be implemented for example, as a "save as" or "export to" functionality in ORM tools.

In what follows, we describe the main elements of the ORM-ML grammar and demonstrate it using a few elementary examples. A more complete example is provided in Appendix A3. We chose to respect the ORM structure as much as possible by not "collapsing" it through the usual relational transformer that comes with most ORM-based tools. ORM-ML allows the representation of any ORM schema without a loss of information or a change in semantics, except for the geometry and topology (graphical layout) of the schema (e.g. location and shapes of the symbols) We include this in a separate graphical style sheet from that of the ORM Schema (see Appendix B2).

We represent the ORM document as a one node element called the ORMSchema, which consists itself of two nodes: ORMMeta and ORMBody. Fig. 1 shows an "empty" instance of this schema.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ORMSchema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://starlab.vub.ac.be/ORMML/orxml.xsd"
xmlns:dc="http://purl.org/dc/elements/1.1/">
  <ORMMeta>
    ...
  </ORMMeta>
  <ORMBody>
    ...
  </ORMBody>
</ORMSchema>
```

Fig. 1. An empty instance of the ORMSchema, as an example of ORM-ML document.

¹ In comparison with other approaches (e.g. ER, UML), ORM models are attribute-free; so they are immune from changes that cause attributes to be remodeled as entity types or relationships.

2.1 ORM-ML metadata

As a header to an ORM-ML document, an ORMMeta node includes metadata elements about the ORM document, such as ‘Title’, ‘URI’, ‘Creator’, ‘Version’, etc. A ORMMeta node consists of a set of Meta elements. Each Meta element has two attributes: name and content. The main idea of this *elementary* structure is to enable the flexibility of adopting existing metadata standards. For example, one may use the 15 well-known Dublin Core Meta elements² - an example of their use appears in fig. 2.

```
....  
<ORMMeta>  
  <Meta name="DC.Title" content="Complaint Problems"/>  
  <Meta name="DC.Creator" content="Mustafa Jarrar"/>  
  <Meta name="DC.Language" content="English"/>  
  <Meta name="DC.Description" content="An axiomatization of  
complaint problems categories..."/>  
  ...  
</ORMMeta>  
....
```

Fig. 2. An example of an ORMMeta node, using Dublin Core metadata elements.

To enable the foundation of libraries of ORM models, we have developed a decent set of 25 metadata elements that better suit the description of a conceptual model, in a general sense, such as ontologies. These elements are a specialization and extension of the Dublin Core elements. An example of this metadata appears in fig. 3. Appendix A4 presents a definition of these metadata elements³.

² The Dublin Core Metadata Initiative (<http://www.dublincore.org> , June 2004) is a cross-disciplinary international effort to develop mechanisms for the discovery-oriented description of diverse resources in an electronic environment. The Dublin Core Element Set comprises 15-elements which together capture a representation of essential aspects related to the description of resources. These 15-elements are namely: title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage and rights.

³ It is perhaps worthwhile to note that our metadata elements (and their definitions) are adopted in the KnowledgeWeb Network of excellence project (KWEB EU-IST-2004-507482), and will be proposed as a standard for Ontology Metadata (or also called Ontology Registries). For more details, see [SGG+05].

```

....
<ORMMeta>
  <Meta name="DM.Title" content="Complaint Problems"/>
  <Meta name="DM.Version" content="v1.0"/>
  <Meta name="DM.Creator" content="Mustafa Jarrar"/>
  <Meta name="DM.Genericity" content="Application"/>
  <Meta name="DM.Language" content="English"/>
  <Meta name="DM.DevelopmentStatus" content="Final"/>
  <Meta name="DM.DomainSubject" content="e-business, customer complaint"/>
  <Meta name="DM.CreationDate" content="5/3/2003"/>
  <Meta name="DM.pecificationLanguage" content="ORM-ML V1.2"/>
  <Meta name="DM.Tool" content="DogmaModeler"/>
  <Meta name="DM.Application" content="CCform"/>
  <Meta name="DM.Description" content="An axiomatization of complaint
problems categories..."/>
  ...
</ORMMeta>
....

```

Fig. 3. An example of an ORMMeta Node, using DogmaModeler metadata elements.

2.2 ORM-ML Body

The ORMBody node consists of these five different (meta-ORM) elements: Object, Subtype, Predicate, Predicate_Object and Constraint.

Object Types

Object elements are abstract XML elements that are used to represent Object Types. They are identified by an attribute ‘Name’, which is the name of the Object Type in the ORM Schema, see fig. 4. Objects are implemented by two XML elements: LOT (Lexical Object Type, called Value Types in [H01]) and NOLOT (Non-Lexical Object Type, called Entity Types in [H01])⁴. LOT elements may have a *numeric* attribute, which is a boolean and indicates whether we deal with a numeric Lexical Object Type. NOLOT elements have a boolean attribute called *independent*, which indicates whether the Non Lexical Object Type is independent. NOLOT elements may also have a *reference* element. A reference element would indicate how this NOLOT is identified by LOTs and other NOLOTs in a given application environment. A reference element has two attributes: *ref_name* (the name of the reference and numeric) and a boolean (to indicate whether it is a numeric reference).

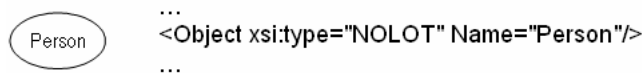


Fig. 4. ORM-ML representation of an Object Type.

⁴ *Informally speaking*, the idea of LOT and NOLOT in ORM, is similar the idea of ValueProperty and ObjectProperty in OWL. LOT represents ValueProperty, and NOLOT represents ObjectProperty.

Subtypes

Subtype elements are used to represent subtype relationships between (non-lexical) object types. A subset element is required to have two elements: parent and child, where both refer to predefined object type elements. See fig. 5.

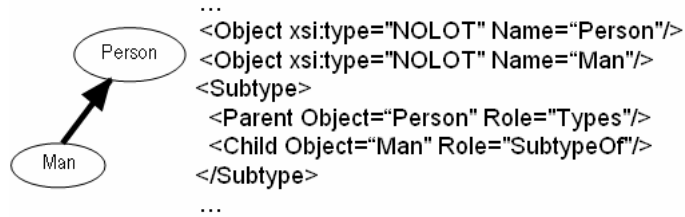


Fig. 5. ORM-ML representation of subtypes.

Predicates

Predicates consist of at least one Object_Role element. Such an element contains a reference to an object and may contain a role. They actually represent the rectangles in an ORM schema. Every Object_Role element needs a generated attribute 'ID' which identifies the Object_Role (see fig. 6). By using this ID attribute, we can refer to a particular Object_Role element in the rest of the XML document, which for example, we will need to do when we define constraints.

Predicates can have one or more rule elements. These elements can contain extra rules that are defined for the predicate.

Predicates also have two boolean attributes that are optional: *'Derived'* and *'Derived_Stored'* which indicate whether a predicate respectively is derived, or derived and stored, or not.

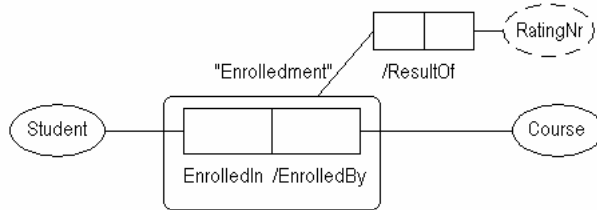


```
...
<Object xsi:type="NOLOT" Name="Book"/>
<Object xsi:type="NOLOT" Name="Author"/>
<Predicate>
  <Object_Role ID="ID1" Object="Book"
    Role="WrittenBy"/>
  <Object_Role ID="ID2" Object="Author"
    Role="Writes"/>
</Predicate>
...
```

Fig. 6. A simple binary predicate and its representation in ORM-ML.

Predicate Objects

Predicate_Objects are actually objectified predicates, which are used in nested fact types. They contain a predicate element and have an attribute called 'Predicate_Name'. So in fact, they are merely predicates that have received new object type names. In building Object_Roles, the Predicate_Name can be referenced. In this way we build predicates that contain objectified predicates instead of object types. See fig. 7.



```

...
<Object xsi:type="NOLOT" Name="Student"/>
<Object xsi:type="NOLOT" Name="Course"/>
<Object xsi:type="LOT" Name="RatingNr"/>
<Predicate_Object Predicate_Name="Enrollment">
  <Predicate>
    <Object_Role ID="ID18" Object="Student" Role="EnrolledIn"/>
    <Object_Role ID="ID19" Object="Course" Role="EnrolledBy"/>
  </Predicate>
</Predicate_Object>
<Predicate>
  <Object_Role ID="ID20" Object="Enrollment" Role=""/>
  <Object_Role ID="ID21" Object="RatingNr" Role="ResultOf"/>
</Predicate>
...

```

Fig. 7. ORM-ML representation of nested fact types (Objectified predicates).

Constraints

Constraint elements represent the ORM constraints. The Constraint element itself is abstract, but it is implemented by different types of constraints, *viz.* Mandatory, Uniqueness, Subset, Equality, Exclusion, Value, Frequency, and Ring constraints. As mentioned above, we use the IDs of the Object_Role elements to define constraints.

Uniqueness and mandatory constraint elements possess only Object_Role elements. These elements are the object_roles in the ORM diagram on which the constraint is placed. In this way, there is no need to make a distinction between the ORM-ML syntax of "external" and "internal" uniqueness constraints (see [H01]), or between mandatory and disjunctive mandatory constraints, see fig. 8.

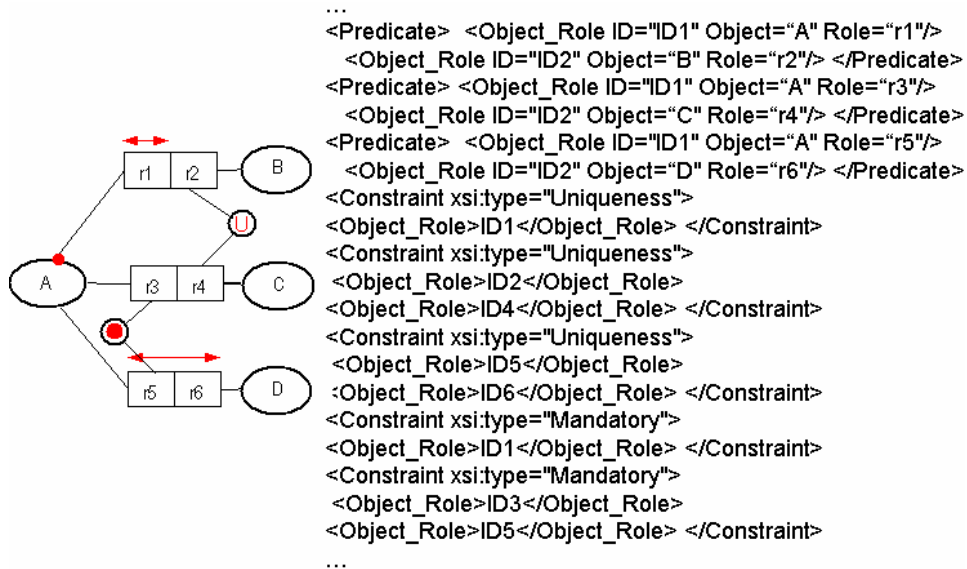


Fig. 8. ORM-ML representation of Uniqueness and Mandatory constraints.

The representation for *subset*, *equality*, and *exclusion* constraints is analogous, so we will only discuss them in general terms. Each of these constraints has references to (combinations of) object_role elements. For instance, to represent a subset constraint between two roles, we create a Subset element, containing two elements, Parent and Child. In the Parent element, we put references to the subsumed object_role, and in the Child element, we put references to the subsuming object_role. For equality and exclusion, we use First and Second elements instead of Parent and Child elements. Fig. 9., fig. 10, and fig. 11 show the ORM-ML representation of subset, equality, and exclusion constraints respectively.

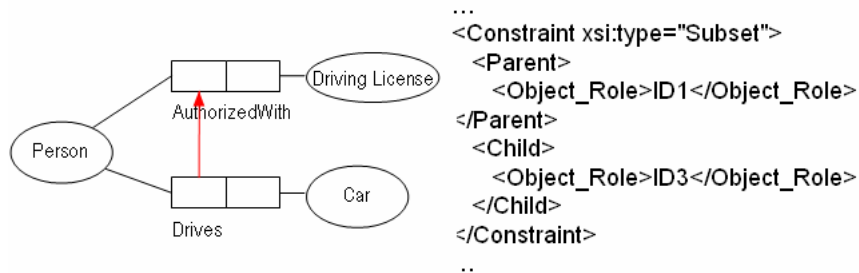


Fig. 9. ORM-ML representation of the Subset constraint.

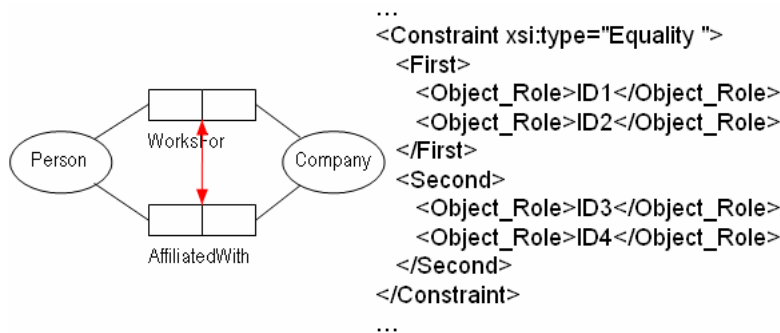


Fig. 10. ORM-ML representation of the Equality constraint.

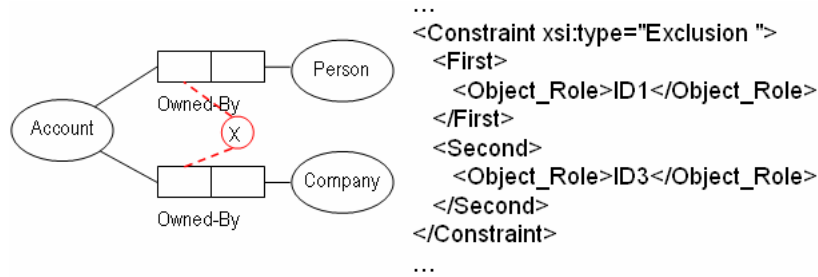


Fig. 11. ORM-ML representation of the Exclusion constraint.

The representation for *Exclusive* and *Totality* constraints is analogous, and very simple. Each constrain has one supertype elements and (at least two) subtypes elements. See fig. 12.

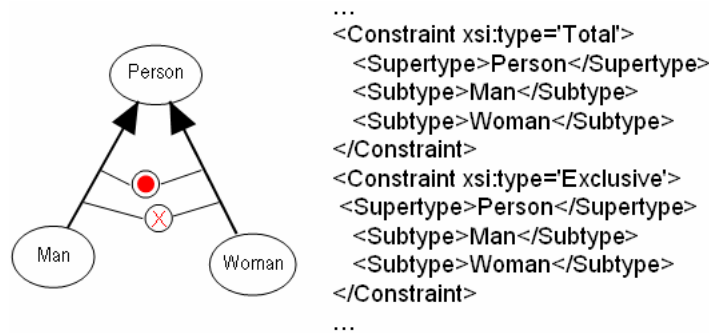


Fig. 12. ORM-ML representation of the Exclusive and Totality constraint.

The *Value* constraint is represented in ORM-ML using the Value and ValueRange elements. The ValueRange element has two attributes: *begin* and *end*, with obvious meanings. Each of the Value and ValueRange elements have an additional attribute called “datatype” to indicate the datatype of the value. See fig. 13.

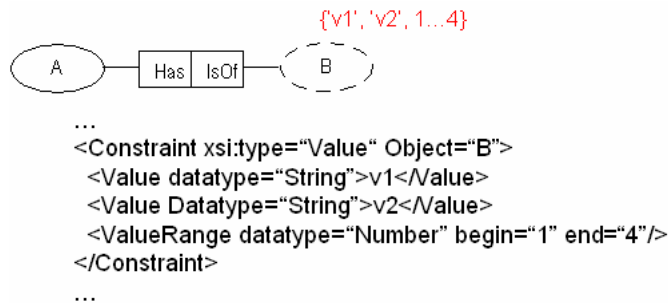


Fig. 13. ORM-ML representation of the value constraint.

The *Frequency* constraint is represented in ORM-ML by two attributes: Minimum and Maximum, which can defined on Object_Roles. See fig. 14.

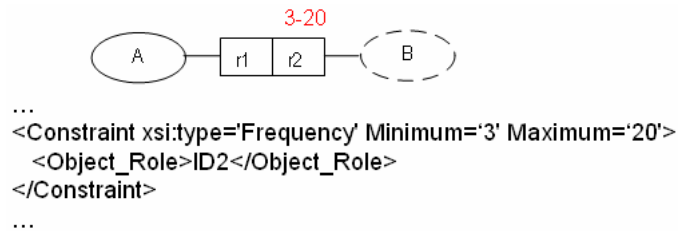


Fig. 14. ORM-ML representation of the Frequency constraint.

Finally, ring constraint elements are: antisymmetric (ans), asymmetric (as), acyclic (ac), irreflexive (ir), intransitive (it), symmetric (sym), acyclic+intransitive (ac+it), asymmetric+intransitive (as+it), intransitive+symmetric (it+sym), and irreflexive+symmetric (ir+sym). Ring constraint elements contain references to the object_roles they are put on. See Fig 15.

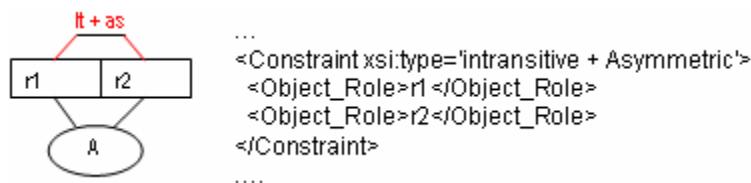


Fig. 15. ORM-ML representation of the Ring constraints.

Modular schemes

ORM-ML also supports modular ORM schemes, which allows the representation of sub ORM schemes (seen as composed modules), see e.g. [J05a].

We allow representing modular ORM schemes by either 1) referring to the composed schemes by their URIs, or 2) including the content of these composed schemes inside the ORM-ML document. Fig. 16 illustrates the ORM-ML representation of a modular schema using RUIs as references to the composed modules. In this way, each of the composed modules will be fetched when opening (or using) the modular schema. Notice that the main disadvantage of using this method is that any changes to the modules may influence the composition.

```

<ORMSchema xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be/staff/mustafa/orxml.xsd'
xmlns:dc='http://purl.org/dc/elements/1.1/' OntologyBase="Dogma OntologyBase" Context="e-commerce">
  <ORMMeta> <Meta name="title" content="BookShoping"/> ... </ORMMeta>
  <ORMBody>
    <Subcommitment URI="http://www.starlab.ac.be/staff/mustafa/bookorder.ormml"/>
    <Subcommitment URI="http://www.starlab.ac.be/staff/mustafa/e-payment.ormml"/>
  </ORMBody>
</ORMSchema>

```

Fig. 16. An example of the ORM-ML representation of a modular schema, using URIs.

In the second choice, users can choose to include “a copy” of each module as a subpart of the ORM-ML document (see fig. 17.). In this way, several problematical issues are prevented, such as the influence of module changes and broken links. However, the main disadvantage of this method is that some useful changes, to the original modules, will not be captured.

We allow users to decide on the most appropriate method, given their application scenario, the steadiness of their module evolution and whether their usage is on or off-line etc.

```

<ORMSchema xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be/staff/mustafa/ormml.xsd'
xmlns:dc='http://purl.org/dc/elements/1.1/' OntologyBase="Dogma OntologyBase" Context="e-
commerce">
<ORMMeta> <Meta name="title" content="BookShopping"/> .... </ORMMeta>
<ORMBody>
  <Subcommitment >
    <ORMSchema xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be/staff/mustafa/ormml.xs
d' xmlns:dc='http://purl.org/dc/elements/1.1/' OntologyBase=" Dogma OntologyBase "
Context="e-commerce">
      <ORMMeta><Meta name="title" content="BookOrder"/></ORMMeta>
      <ORMBody> .... </ORMBody>
    </ORMSchema>
  </Subcommitment>
  <Subcommitment >
    <ORMSchema xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be/staff/mustafa/ormml.xs
d' xmlns:dc='http://purl.org/dc/elements/1.1/' OntologyBase=" Dogma OntologyBase "
Context="e-commerce">
      <ORMMeta> <Meta name="title" content="e-Payment"/></ORMMeta>
      <ORMBody> .... </ORMBody>
    </ORMSchema>
  </Subcommitment>
</ORMBody>
</ORMSchema>

```

Fig. 17. An example of an ORM-ML representation of a modular schemes, where the content of a module is included as a sub-schema.

3 Discussion and conclusions

In this document, we have presented the ORM markup language that represents ORM conceptual diagrams in an XML-based syntax. Our main goals of doing this are:

Enable the ORM conceptual diagrams to be shared, exchanged, and processed at run-time. ORM-ML as a standardized syntax for ORM models may assist interoperation tools to exchange, parse or understand the ORM schemas. Like ORM-ML, any conceptual modeling approach (e.g. EER, UML, etc.) could have a markup language.

Enable conceptual data modeling methods to be (re)used for ontology engineering purposes. Indeed, as have been discussed in [J05][JDM03] conceptual data modeling methods suit many (or maybe most) application scenarios and usability perspectives. In addition, the large set of existing conceptual modeling methods, graphical notations, and tools can make ontologies better understandable, and easier to adopt, construct, visualize

and verbalize. Legacy conceptual schemes can be mined and/or “ontologized”.

Interoperability for exchanging and sharing conceptual data models over the Internet. Facilities are needed to share and exchange ORM conceptual models in terms of a networked, distributed computing-driven, and collaborative environment, and to allow users to browse and edit shared knowledge over the Internet, intranets and other channels. A conceptual schema markup language provides a standardizable method to achieve interoperability among CASE tools that use the conceptual modeling technique.

Implementing a conceptual query language over the Web. In open and distributed environments, the building of queries should be possible regardless of the internal representation of the data. Query languages based on ontologies (seen as shared conceptual models) help users not only to build queries, but also make them easier, more expressive, and more understandable than corresponding queries in a language like SQL. Exchanging, reusing, or sharing such queries efficiently between agents over the web is substantially facilitated by a standardized markup language. Consequently, ORM-based query languages (e.g. RIDL [VB82] [M81], ConQuer [BH96]) would gain from ORM-ML by representing queries in such an exchangeable representation.

Building transformation style sheets. Building transformation style sheets for a given usage or need, for example, to transform the XML-based representation into another XML-based representation languages, such as DLR[], DIG[], etc, Another important and strategic issue is that one could write a style sheet to generate the given ORM model instance into a given business rule-engine’s syntax, to allow for run-time interpretation by that rule engine. It could for instance, perform instance validation and integrity checks, etc.

Generating Verbalizations. The verbalization of a conceptual model is the process of writing its facts and constraints in pseudo natural language sentences. This assumedly allows non-experts to check, validate, or even build conceptual schemas. [] shows how to generate the verbalization of ORM models by building a verbalization template (built as separate XML-based style sheets) parameterized over ORM-ML documents.

Appendix A: ORM Markup Language

This appendix presents the *XML-Schema* for the ORM Markup Language, as the grammar reference of ORM-ML documents. This schema (Ver. 3) is an improved version of the ORM-ML XML-schema (Ver.2) that have been published in [J05]. Ver.1 is the earlier version of ORM-ML which appears in [DJM02a][DJM02b] and [JDM03]. In appendix A1 we present a tree view of the ORM-ML XML-schema, and in appendix A2 we present the ORM-ML XML-schema. Appendix A3 presents a complete example, as an instance of this schema. Appendix A4 presents the metadata elements.

Appendix A1 (tree view of the ORM-ML XML-Schema)

A tree view of the elements in the XML Schema is given in Appendix A2. Please note the attributes of the elements are omitted here for clarity of presentation.

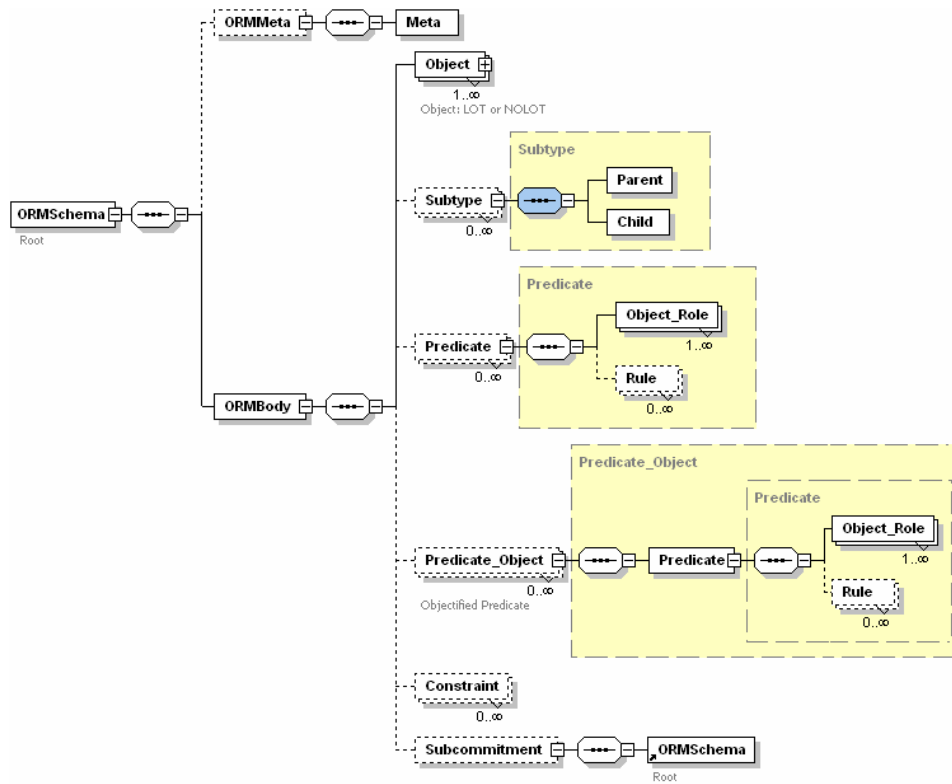


Fig. A.1. A tree view of the elements in the ORM-ML XML Schema.

Appendix A2 (ORM-ML XML-Schema)

The XML-schema below can also found at :

<http://www.starlab.vub.ac.be/staff/mustafa/orxml.v1.3.xsd>

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by rth77 (rth77)
-->
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:dc="http://purl.org/dc/elements/1.1/"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="http://purl.org/dc/elements/1.1/"
schemaLocation="http://www.ukoln.ac.uk/metadata/dcmi/dcxml/xmils/dc.xsd" />
- <xs:element name="ORMSchema">
- <xs:annotation>
  <xs:documentation>Root</xs:documentation>
</xs:annotation>
- <xs:complexType>
- <xs:complexContent>
- <xs:extension base="ORMType">
- <xs:sequence>
- <xs:element name="ORMMeta" minOccurs="0">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="Meta">
- <xs:complexType>
  <xs:attribute name="Name" type="xs:string" use="required" />
  <xs:attribute name="Content" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
- <xs:element name="ORMBody">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="Object" type="Object" minOccurs="0" maxOccurs="unbounded">
- <xs:annotation>
  <xs:documentation>Object: LOT or NOLOT</xs:documentation>
</xs:annotation>
</xs:element>
  <xs:element name="Subtype" type="Subtype" minOccurs="0" maxOccurs="unbounded" />
  <xs:element name="Predicate" type="Predicate" minOccurs="0" maxOccurs="unbounded" />
- <xs:element name="Predicate_Object" type="Predicate_Object" minOccurs="0"
maxOccurs="unbounded">
- <xs:annotation>
  <xs:documentation>Objectified Predicate</xs:documentation>
</xs:annotation>
</xs:element>
  <xs:element name="Constraint" type="Constraint" minOccurs="0" maxOccurs="unbounded" />
- <xs:element name="Subcommitment" minOccurs="0">
- <xs:complexType>
- <xs:sequence>
  <xs:element ref="ORMSchema" />
</xs:sequence>
  <xs:attribute name="order" type="xs:integer" use="optional" />
  <xs:attribute name="URI" type="xs:string" use="optional" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```

```

</xs:sequence>
<xs:attribute name="OntologyBase" type="xs:string" use="required" />
<xs:attribute name="Context" type="xs:string" use="required" />
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
- <xs:complexType name="Object" abstract="true">
- <xs:annotation>
  <xs:documentation>Object: LOT or NOLOT</xs:documentation>
</xs:annotation>
- <xs:sequence>
- <xs:element name="Translation" minOccurs="0" maxOccurs="unbounded">
- <xs:complexType>
  <xs:attribute name="Language" type="xs:string" use="required" />
  <xs:attribute name="Name" type="xs:string" use="required" />
  <xs:attribute name="Description" type="xs:string" use="required" />
  <xs:attribute name="Reference" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Name" type="xs:ID" use="required" />
<xs:attribute name="Gloss" type="xs:string" use="optional" />
<xs:attribute name="Datatype" type="xs:string" use="optional" />
<xs:attribute name="TermUpperForm" type="xs:string" use="optional" />
<xs:attribute name="NameSpace" type="xs:string" use="optional" />
</xs:complexType>
- <xs:complexType name="LOT">
- <xs:annotation>
  <xs:documentation>Lexical Object Type</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Object">
  <xs:attribute name="numeric" type="xs:boolean" use="optional" default="false" />
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="NOLOT">
- <xs:annotation>
  <xs:documentation>Non Lexical Object Type</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Object">
- <xs:sequence>
- <xs:element name="Reference" minOccurs="0">
- <xs:complexType>
  <xs:attribute name="Ref_Name" use="required" />
  <xs:attribute name="numeric" type="xs:boolean" use="optional" default="false" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="Independent" type="xs:boolean" use="optional" default="false" />
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Object_Role">
- <xs:annotation>
  <xs:documentation>Object + Role</xs:documentation>
</xs:annotation>
  <xs:attribute name="ID" type="xs:ID" use="required" />
  <xs:attribute name="Object" type="xs:IDREF" use="required" />
  <xs:attribute name="Role" type="xs:string" use="optional" />

```

```

</xs:complexType>
<xs:complexType name="ORMType" />
- <xs:complexType name="Predicate">
- <xs:sequence>
  <xs:element name="Object_Role" type="Object_Role" maxOccurs="unbounded" />
  <xs:element name="Rule" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
  <xs:attribute name="Derived" type="xs:boolean" default="false" />
  <xs:attribute name="Derived_Stored" type="xs:boolean" default="false" />
</xs:complexType>
- <xs:complexType name="Constraint" abstract="true">
- <xs:annotation>
  <xs:documentation>Abstract element for constraints</xs:documentation>
</xs:annotation>
</xs:complexType>
- <xs:complexType name="Predicate_Object">
- <xs:annotation>
  <xs:documentation>Objectified Predicate</xs:documentation>
</xs:annotation>
- <xs:sequence>
  <xs:element name="Predicate" type="Predicate" />
</xs:sequence>
  <xs:attribute name="Predicate_Name" type="xs:ID" use="required" />
</xs:complexType>
- <xs:complexType name="Subtype">
- <xs:annotation>
  <xs:documentation>SubType</xs:documentation>
</xs:annotation>
- <xs:sequence>
- <xs:element name="Parent">
- <xs:complexType>
  <xs:attribute name="Object" type="xs:IDREF" />
  <xs:attribute name="Role" type="xs:string" />
</xs:complexType>
</xs:element>
- <xs:element name="Child">
- <xs:complexType>
  <xs:attribute name="Object" type="xs:IDREF" />
  <xs:attribute name="Role" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
- <xs:complexType name="Mandatory">
- <xs:annotation>
  <xs:documentation>Mandatory Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Uniqueness">
- <xs:annotation>
  <xs:documentation>Uniqueness Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>

```

```

<xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Subset">
- <xs:annotation>
  <xs:documentation>SubSet Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
- <xs:element name="Parent">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
</xs:element>
- <xs:element name="Child">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Equality">
- <xs:annotation>
  <xs:documentation>Equality Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
- <xs:element name="First">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
</xs:element>
- <xs:element name="Second">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Exclusion">
- <xs:annotation>
  <xs:documentation>Exclusion Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>

```

```

- <xs:element name="First">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
- <xs:element name="Second">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Frequency">
- <xs:annotation>
  <xs:documentation>Frequency Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" maxOccurs="unbounded" />
</xs:sequence>
  <xs:attribute name="Minimum" type="xs:integer" />
  <xs:attribute name="Maximum" type="xs:integer" />
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Irreflexive">
- <xs:annotation>
  <xs:documentation>Irreflexive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Intransitive">
- <xs:annotation>
  <xs:documentation>Intransitive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Transitive">
- <xs:annotation>
  <xs:documentation>Transitive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">

```

```

- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Acyclic">
- <xs:annotation>
  <xs:documentation>Acyclic Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Type" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Asymmetric">
- <xs:annotation>
  <xs:documentation>Assymmetric Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Antisymmetric">
- <xs:annotation>
  <xs:documentation>Antisymmetric Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Symmetric">
- <xs:annotation>
  <xs:documentation>Symmetric Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Reflexive">
- <xs:annotation>
  <xs:documentation>Reflexive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
  <xs:extension base="Constraint" />
</xs:complexContent>
</xs:complexType>

```

```

- <xs:complexType name="Total">
- <xs:annotation>
  <xs:documentation>Total constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Supertype" />
  <xs:element name="Subtype" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Exclusive">
- <xs:annotation>
  <xs:documentation>Exclusive constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Supertype" />
  <xs:element name="Subtype" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Value">
- <xs:annotation>
  <xs:documentation>Exclusive constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Value" maxOccurs="unbounded">
- <xs:complexType>
  <xs:attribute name="datatype" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
- <xs:element name="ValueRange" maxOccurs="unbounded">
- <xs:complexType>
  <xs:attribute name="datatype" type="xs:string" use="required" />
  <xs:attribute name="begin" type="xs:string" use="required" />
  <xs:attribute name="end" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Partition">
- <xs:annotation>
  <xs:documentation>Partition constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Subtype" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
  <xs:attribute name="Supertype" type="xs:IDREF" use="required" />
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

```

- <xs:complexType name="Intransitive_symmetric">
- <xs:annotation>
  <xs:documentation>Intransitive + symmetric Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Acyclic_intransitive">
- <xs:annotation>
  <xs:documentation>Acyclic+intransitive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Asymmetric_intransitive">
- <xs:annotation>
  <xs:documentation>Asymmetric+intransitive Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
- <xs:complexType name="Irreflexive_symmetric">
- <xs:annotation>
  <xs:documentation>Irreflexive + symmetric Ring Constraint</xs:documentation>
</xs:annotation>
- <xs:complexContent>
- <xs:extension base="Constraint">
- <xs:sequence>
  <xs:element name="Object_Role" type="xs:IDREF" maxOccurs="unbounded" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>

```

Appendix A3: Complete Example

A complete example of an ORM schema diagram with the associated ORM-ML document generated by the DogmaModeler.

ORM Schema diagram

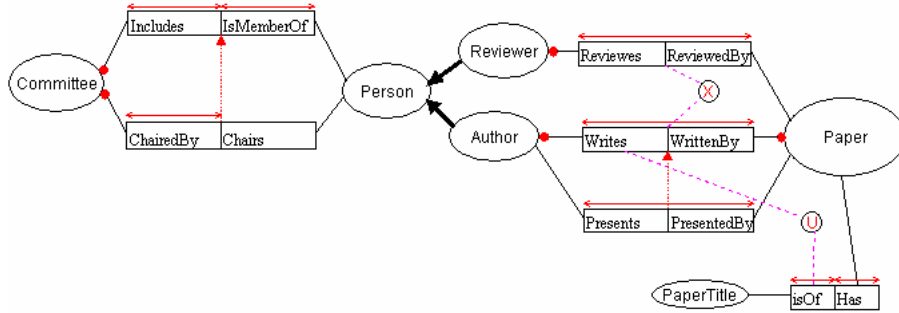


Fig. A.2. ORM schema diagram example

Corresponding ORM-ML

```

<?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/orddl.v1.3.xsd'
xmlns:dc='http://purl.org/dc/elements/1.1/' OntologyBase="Publishing" Context="Scientific Conference">

<ORMMeta>
<Meta name="DC.title" content="ORM ML example"/>
<Meta name="DC.creator" content="Mustafa Jarrar"/>
<Meta name="DC.description" content="A complete example of an ORM ML file"/>
</ORMMeta>
<ORMBody>
<Object xsi:type='NOLOT' Name='Committee'/>

<Object xsi:type='NOLOT' Name='Person'/>
<Object xsi:type='NOLOT' Name='Author'/>
<Object xsi:type='NOLOT' Name='Reviewer'/>
<Object xsi:type='NOLOT' Name='Paper'/>
<Object xsi:type='NOLOT' Name='PaperTitle' />
<Subtype>
<Parent Object="Person" Role="Types"/>
<Child Object="Author" Role="IsA"/>
</Subtype>
<Subtype>
<Parent Object="Person" Role="Types"/>
<Child Object="Reviewer" Role="IsA"/>
</Subtype>
<Predicate>
<Object_Role ID='ORM ML example:42' Object='Committee' Role='Includes'/>
<Object_Role ID='ORM ML example:43' Object='Person' Role='IsMemberOf'/>
</Predicate>
<Predicate>
<Object_Role ID='ORM ML example:44' Object='Committee' Role='ChairedBy'/>
<Object_Role ID='ORM ML example:45' Object='Person' Role='Chairs'/>
</Predicate>
<Predicate>
<Object_Role ID='ORM ML example:46' Object='Reviewer' Role='Reviews'/>
<Object_Role ID='ORM ML example:47' Object='Paper' Role='ReviewedBy'/>
</Predicate>
<Predicate>
<Object_Role ID='ORM ML example:48' Object='Author' Role='Writes'/>
<Object_Role ID='ORM ML example:49' Object='Paper' Role='WrittenBy'/>
</Predicate>
<Predicate>
<Object_Role ID='ORM ML example:50' Object='Author' Role='Presents'/>
<Object_Role ID='ORM ML example:51' Object='Paper' Role='PresentedBy'/>

```

```
</Predicate>
<Predicate>
  <Object_Role ID='ORM ML example:52' Object='PaperTitle' Role='isOf'/>
  <Object_Role ID='ORM ML example:53' Object='Paper' Role='Has'/>
</Predicate>
```

```
<Constraint xsi:type='Mandatory'>
  <Object_Role>ORM ML example:42</Object_Role>
</Constraint>
<Constraint xsi:type='Mandatory'>
  <Object_Role>ORM ML example:44</Object_Role>
</Constraint>
<Constraint xsi:type='Mandatory'>
  <Object_Role>ORM ML example:46</Object_Role>
</Constraint>
<Constraint xsi:type='Mandatory'>
  <Object_Role>ORM ML example:49</Object_Role>
</Constraint>
<Constraint xsi:type='Mandatory'>
  <Object_Role>ORM ML example:48</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:42</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:44</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:43</Object_Role>
</Constraint>
<Constraint xsi:type='Subset'>
  <Parent>
    <Object_Role>ORM ML example:42</Object_Role>
    <Object_Role>ORM ML example:43</Object_Role>
  </Parent>
  <Child>
    <Object_Role>ORM ML example:44</Object_Role>
    <Object_Role>ORM ML example:45</Object_Role>
  </Child>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:50</Object_Role>
  <Object_Role>ORM ML example:51</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:48</Object_Role>
  <Object_Role>ORM ML example:49</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:46</Object_Role>
  <Object_Role>ORM ML example:47</Object_Role>
</Constraint>
<Constraint xsi:type='Exclusion'>
  <First>
    <Object_Role>ORM ML example:48</Object_Role>
    <Object_Role>ORM ML example:49</Object_Role>
  </First>
  <Second>
    <Object_Role>ORM ML example:46</Object_Role>
    <Object_Role>ORM ML example:47</Object_Role>
  </Second>
</Constraint>
```

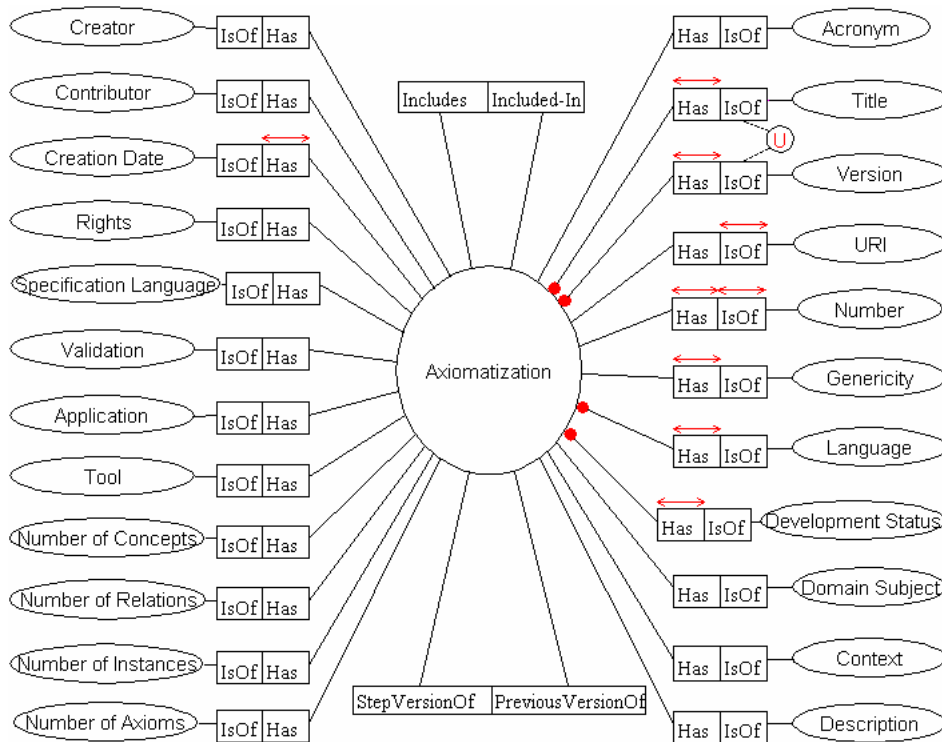
```

<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:48</Object_Role>
  <Object_Role>ORM ML example:52</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:53</Object_Role>
</Constraint>
<Constraint xsi:type='Uniqueness'>
  <Object_Role>ORM ML example:52</Object_Role>
</Constraint>
<Constraint xsi:type='Subset'>
  <Parent>
    <Object_Role>ORM ML example:48</Object_Role>
    <Object_Role>ORM ML example:49</Object_Role>
  </Parent>
  <Child>
    <Object_Role>ORM ML example:50</Object_Role>
    <Object_Role>ORM ML example:51</Object_Role>
  </Child>
</Constraint>
</ORMBody>
</ORMSchema>

```

Appendix A4: ORM-ML Metadata

The following diagram presents the metadata elements and the relationships between them as an ORM schema. The idea of these element is be the foundation for ORM schema library. As ORM schemes are being made not for only database modeling, but also for other purposes such as ontologies, business rules, etc. we call an ORM schema as an axiomatization. each of the element in this diagram is described by a gloss in the following table.



Element Name	Gloss
Axiomatization	A specification of a knowledge (i.e. a conceptual model) about a certain subject-matter written as a set of axioms.
Acronym	An abbreviation formed from the initial letter or letters of words in the axiomatization title. E.g. ‘CCOntology’, or ‘DOLCE’.
Title	The full and official heading or name of the model. It gives a brief summary of the matters it deals with. E.g. ‘Customer Complaint Ontology’, or ‘Descriptive Ontology for Linguistic and Cognitive Engineering’.
Version	Information about the edition of this model. Typically, it includes the version number, label, and date. Whenever the model is enhanced, updated or improved, it is often assigned a new version. Although versions represent the different states of an model during its life cycle, different versions are seen as different models.
Number	A unique code assigned to the model for identification. This number is usually assigned by a registration entity.
URI	Uniform Resource Identifier, the W3C's codification of the address syntax of an ontology. In its most basic form, a URI consists of a scheme name (such as file, http, ftp) followed by a colon, followed by a path whose nature is determined by the scheme that precedes it (see RFC 1630). URI is the umbrella term for URNs, URLs, and all other Uniform Resource Identifiers.
Genericity	The level of generalization of an the model. The genericity level of a model is typically one of the { ‘Application’, ‘task’, ‘Domain’, ‘Core’, ‘Foundational’, ‘Linguistic’, ‘Metamodel’ }. Examples: The CCOntology is a ‘core’ ontology; DOLCE is a ‘foundational’ ontology; Dublin Core is ‘metamodel’ etc.
Language	The human language in which the model terms (i.e. labels of concepts, roles, etc) is expressed. In case this terminology is expressed in more than one language, the value of this attribute is ‘Multilingual’. The best practice recommended is the use of RFC 3066 [RFC3066] which, in conjunction with ISO639 [ISO639]), defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English, "akk" for Akkadian", and "en-GB" for English as it is used in the United Kingdom.
DevelopmentStatus	The completion status or condition of this ontology, typically one of {Draft, Final, Revised, Unavailable}.
DomainSubject	A heading descriptor indicating the subject matter and the domain of the model. For example, e-business, sport, book-shopping and car-rental. Typically, domain subjects are expressed as keywords, key phrases, or classification codes. The recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.
Context	Information about of the scope of the model, in which the

	interpretation (i.e. the intended meaning) of the terminology is bounded. For example: the context of the WordNet ontology could be the English language, the context of the “CCOntology” is the EU complaint regulations, The context of the data models of Microsoft is Microsoft enterprise, etc.
Description	Further information about the model. It may include but is not limited to: an abstract, reference to a graphical representation, a free-text account of the content, the methodology used to build this ontology, documentation, etc.
Creator	An entity primarily responsible for creating the model. Examples of creators include persons, organizations and services. Typically, the name of a creator should be used to indicate the entity.
Contributor	An entity responsible for making contributions to the ontology content. Examples of a Contributor include a person, an organization, and a service. Typically, the name of a contributor should be used to indicate the entity.
CreationDate	The date that is associated with the creation of the model. In other words, the first date in the model lifecycle. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and includes (among others) dates of the form YYYY-MM-DD.
Rights	Information about rights held in and over the model. Typically, rights will contain a copyrights statement and other restriction for the model, and the cost description in case the use of this model requires payment. If the Rights element is absent, no assumptions may be made about any rights held in or over the resource.
SpecificationLanguage	The formal language in which the model is being specified; for example, OWL, DAML-OIL, ORM-ML, UML, KIF, etc. For ORM models, this can be ORM or ORM2.
Validation	An evidence about the testing activities of the model. Such tests might be conceptual or ontological quality, syntax validation, etc. Typically, one should indicate the validation methodology and comments about the results.
Tool	The name of the tool by which the ontology has been developed, e.g. NORMA, VisoModeler, DogmaModeler, etc.
Application	Citation to the application(s) using/has used this model. Typically, one should provide the name, URL, and some description about the application.
NumberOfConcepts	Statistics about the number of concepts in the model.
NumberOfRelations	Statistics about the number of relations in the model.
NumberOfAxioms	Statistics about the number of constraints/rules in the model - an axiom is typically a formal definition/expression.

NumberOfInstances	Statistics about the number instances in the model.
IncludesOntology/ IncludedInOntology	A reference to another model, which is supposed to be included as part of this model. Examples of such relations between models include “Imports” in OWL, “inclusion” in Ontolingua and “Compose” in DogmaModeler. The formal semantics of such relationships are necessarily the same.
StepVersionOf/ PreviousVersionOf	A reference to the step/previous version of this model.

References:

- [BCMNP03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. The Description Logic Handbook. *Cambridge University Press*, 2003.
- [BGH99] Bird, L., Goodchild, A., Halpin, T.A.: Object Role Modelling and XML-Schema. In: Laender, A., Liddle, S., Storey, V. (eds.): *Proc. of the 19th International Conference on Conceptual Modeling (ER'00)*. LNCS, Springer, 1999.
- [BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object role models. *Information Systems*, 16(5), pp. 471-495, 1991.
- [BvHHMP04] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. <http://www.w3.org/TR/owl-ref/>, 2004.
- [CDLNR98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. Information integration : Conceptual Modeling and reasoning support. In *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS'98)*, pp. 280-291, 1998.
- [DJM02a]: Demey, J., Jarrar, M., Meersman, R.: A Conceptual Markup Language that supports interoperability between Business Rule modeling systems. *Proc. of the Tenth International Conference on Cooperative Information Systems (CoopIS 02)*. Springer LNCS 2519, pp. 19-35, 2002.
- [DMV] O. De Troyer, R. Meersman, and P. Verlinden. RIDL^{*} on the CRIS case: A Workbench for NIAM. *Technical report*. INFOLAB, Tilburg University, The Netherlands.
- [H] T. Halpin. Object-Role Modeling: an overview. White paper, <http://www.orm.net>.
- [H01] T. Halpin. Information Modeling and Relational Databases. 3rd edn. Morgan-Kaufmann, 2001.
- [H89] T. Halpin. A logical analysis of information systems: static aspects of the data-oriented perspective. *PhD thesis*, University of Queensland, Brisbane, Australia, 1989.
- [H97] Halpin, T.: An Interview- Modeling for Data and Business Rules. In: *Ross, R. (eds.): Database Newsletter*. vol. 25, no. 5. (Sep/Oct 1997). -This newsletter has since been renamed Business Rules Journal and is published by Business Rules Solutions, Inc.
- [J06]: Jarrar, M.: Towards the notion of gloss, and the adoption of linguistic resources in formal ontology engineering. Proceeding of the 15th International World Wide Web Conference, WWW2006. Edinburgh, Scotland. May 2006. ACM, 2006.
- [J05] M. Jarrar. Towards Methodological Principles for Ontology Engineering. *PhD thesis*, Vrije Universiteit Brussel, 2005.
- [JF05] Jarrar, M., Franconi, E.: Mapping ORM into the DLR description logic. Technical Report, August 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
- [N99] North, K.: Modeling, Data Semantics, and Natural Language. In: *New Architect maga-zine*, 1999.
- [T96] O. De Troyer. A formalization of the binary Object-role Model based on Logic. *Data & Knowledge Engineering* 19, North-Holland, Elsevier, pp. 1-37, 1996.
- [TM95] O. De Troyer and R. Meersman. A Logic Framework for a Semantics of Object-Oriented Data Modelling. In *Proc. Of 14th International Conference Object-Orientation and Entity-Relationship Modelling (OO-ER'95)*, LNCS 1021, pp. 238-249, Springer, 1995.
- [VB82] G. Verheijen and P. van Bekkum. NIAM, an Information Analysis Method. In *Proc. Of the IFIP Conference on Comparative Review of Information Systems Methodologies*, North-Holland, 537-590, 1982