[JDM03] Jarrar, M., Demey, J., Meersman, R.: On Using Conceptual Data Modeling for Ontology Engineering. In Spaccapietra, S., March, S., Aberer, K., (Eds.): Journal on Data Semantics (Special issue on Best papers from the ER, ODBASE, and COOPIS 2002 Conferences). LNCS, Vol. 2800, Springer, pp.:185-207. October 2003.

On Using Conceptual Data Modeling for Ontology Engineering

Mustafa Jarrar, Jan Demey, Robert Meersman

STARLab Vrije Universiteit Brussel, Pleinlaan 2,
Brussels, 1050, Belgium
`{mjarrar, jdemey, meersman}@vub.ac.be`

Abstract. This paper tackles two main disparities between conceptual data schemes and ontologies, which should be taken into account when (re)using conceptual data modeling techniques for building ontologies. Firstly, conceptual schemes are intended to be used during design phases and not at the run-time of applications, while ontologies are typically used and accessed at run-time. To handle this first difference, we define a conceptual markup language (ORM-ML) that allows to represent ORM conceptual diagrams in an open, textual syntax, so that ORM schemes can be shared, exchanged, and processed at the run-time of autonomous applications. Secondly, unlike ontologies that are supposed to hold application-independent domain knowledge, conceptual schemes were developed only for the use of an enterprise application(s), i.e. "in-house" usage. Hence, we present an ontology engineering-framework that enables reusing conceptual modeling approaches in modeling and representing ontologies. In this approach we prevent application-specific knowledge to enter or to be mixed with domain knowledge. To end, we present DogmaModeler: an ontology-engineering tool that implements the ideas presented in the paper.

Keywords: Ontology, Conceptual data modeling, Context, Ontology tools, Reusability, DOGMA, DogmaModeler, ORM, ORM-ML.

1 Introduction and motivation

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]; they provide conceptual 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] [M99]. Nevertheless, this meaning is being represented in diagram formats, and typically used in an *off-time mode*, *i.e.* used during the design phases and not at the run-time of applications.

Nowadays, the Internet and the open connectivity environments create a strong demand for sharing and exchanging not only data but also data semantics. Therefore, emerging ontologies are intended to represent agreed and shared domain semantics. So that, based on such ontologies, computer systems can meaningfully communicate to exchange data and make transactions interoperate independently of their internal technologies. For example, by sharing an ontology, heterogeneous information resources can be integrated and searched through mediators [TSC01] [SOVZJSSM02], e-commerce applications can meaningfully communicate, etc.

Conceptual data schemes and ontologies are quite similar, as both consist of conceptual relations[1] and rules[2]. Thus, several researchers have proposed to (re) use those conceptual methodologies and tools for ontology modeling, e.g. [F02] [CHP01] [BKKHSHLA01] [M01]. Reusing conceptual modeling techniques for ontology engineering is ultimately beneficial: the large set of existing conceptual modeling methods, graphical notations, and tools can make ontologies better understandable, and easier to adopt, construct, visualize, etc. Furthermore, legacy conceptual schemes can be mined and/or "ontologized".

For such purposes, some extensions have been made to the foundation of the conceptual modeling constructs. For example, to effectively capture knowledge about application domains, [WSW99] redefined several conceptual modeling constructs of the ER approach, by defining some rules –based on ontological analyses- to resolve ambiguities that exist in current practice of modeling relationships. [GHW02] suggested *ontological semantics* for UML class diagrams, by rooting the UML constructs to the GOL upper level ontology. [BCD01][BB03][MC02] have shown how to reason about conceptual schemes.

However, complementary to these efforts, there are two main disparities between ontologies and conceptual data schemes that we aim to tackle in this paper:

1) Conceptual data schemes are being preserved in off-time mode diagrams; while, ontologies typically are *shareable and exchangeable at run-time*, i.e. machine-processable semantics.

2) Unlike conceptual data schemes that capture semantics for a *given application domain*, ontologies are supposed to capture semantics about real-world domains, independent from specific application needs, i.e. *"relatively" generic knowledge*. Therefore, *the genericity (/application-independency) of knowledge is a fundamental*

[1] Conceptual relations can be unary relations (usually called "concepts" or object types as called in ORM), or n-ary relations, which also are called fact types in ORM.

[2] Rules, formally, are well-formed formulae; defined in an ontology (or a conceptual schema) in order to specify and constrain the intended models that can hold. In conceptual data modeling they are commonly called "constraints". Notice that rules can be used for e.g. enforce integrity, derivation and inference, taxonomy, etc.

asset in ontology modeling, and that mostly distinguishes ontologies from conceptual data schemes.

The first goal of this paper is to provide a way for conceptual diagrams to be exchanged and shared at run-time. Therefore, we *define a conceptual markup language in which one can textually represent ORM conceptual diagrams in an open (nonproprietary) way*. The second goal is to present an ontology engineering framework (and tool) that enables ORM conceptual schemes to be used for modeling and representing *ontological commitments*[3]. Please notice that our approach is not restricted to ORM. The same techniques can be applied to other conceptual modeling methods.

Remark: Let us now clarify some terminology that we frequently use in the paper. The intended use of the term 'task', is related and limited to the inferential knowledge that is required to describe a task to be performed, e.g. it does not describe temporal aspects. An application may convey one or more *kinds* of tasks. However the term task is often interchanged with the 'application' that conveys one kind of task. Notice that the '*reusability*' of knowledge implies the maximization of using this knowledge among *different kinds* of applications; while increasing the *usability* implies maximizing the number of applications among the *same kind* of task [JM02b]. Moreover, we use the term '*generic task*' to refer to a reusable kind of task.

Structure of this paper: In section 2, we give a bird's eye introduction to ORM. The ORM markup language (ORM-ML) will be presented in Section 3. In section 4, we discuss the knowledge independency and genericity issues, then we present our framework that uses ORM in ontology modeling. In section 5, we briefly present our DogmaModeler Tool for ontology engineering. Finally, in section 6, we conclude and present some future work.

2 ORM background

Conceptual modeling methodologies are well developed and have proven to be quite successful for building information systems in a graphical way at the conceptual level. By representing the data on a higher level of abstraction conceptual models quality checks can be performed easily. ORM (Object-Role Modeling) [H01] is such a conceptual modeling approach that was developed in the early 70's. Originally it is a successor of NIAM (Natural-language Information Analysis Method) [VB82].

In ORM, the world is represented under the form of objects playing roles. There are two kinds of object types: lexical object types (LOTs) and non-lexical object types (NOLOTs). The distinction between LOTs and NOLOTs is a linguistic distinction. Lexical object types correspond to "utterable" entities (for instance: 'ISBN', 'Title', 'FistName'), while Non-lexical object types (for instance: Book, Person) refer to "non-utterable" entities [VB82]. The design process of information systems is simpli-

[3] See section 4 for the definition.

fied by using ORM, because ORM has an easy to understand graphical notation which includes most frequently used constraints.

Based on ORM, several conceptual modeling tools exist, such as Microsoft's VisioModeler™ and the older InfoModeler, which have the functionality of modeling a certain Universe of Discourse (UoD) in ORM, and support the automatic generation of a consistent and normalized relational database schema for a modeled UoD.

Also, ORM schema's 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.

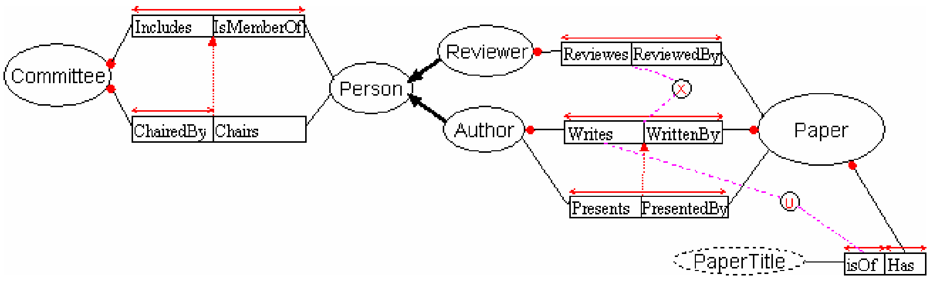


Fig. 1: An example of ORM schema

In Fig. 1, object types are shown as named ellipses. Logical predicates (fact types) appear as named sequences of roles, where each role appears as a box. Roles are connected by line segments to the object types that 'play' them.

In the Figure, the object types are Committee, Person, Reviewer, Author, Paper and PaperTitle. The dotted ellipse indicates that PaperTitle is a lexical object type while the others are non-lexical. The predicates in the figure can be verbalized as follows: ‘A Paper Has a PaperTitle and a PaperTitle IsOf a Paper’, ‘An Author Reviews a Paper and a Paper is Reviewed by a Reviewer’, ‘An Author Writes a Paper and A Paper is Written By an Author’, etc. The arrows connecting the object types Author and Reviewer to Person denote an is-a (predefined subtype) relationship.

In what follows, we briefly name and explain the constraints appearing in the diagram, in fact by giving an (approximate) verbalization of the example in Fig. 1. For other types of ORM constraints we refer to [VB82] or [W90]; the notation and definitions used here are taken from [H01].

Black dots indicate a mandatory role constraint. Example verbalization of Fig. 1: ‘Each Paper must be WrittenBy at least one Author’’. The arrow-tipped bars above the roles are uniqueness constraints. For example: ‘Each Committee is ChairedBy at most one Person. Uniqueness constraints can span more than one role, indicating that ‘any combination that instantiates these roles should be unique’. E.g. for the predicate (Author writes paper, Paper is WrittenBy author), there holds that every combination of author and paper is unique, so an author can only once be the author of the same paper. In ORM, one can also define constraints between different predicates or fact types. For instance, the circled ‘X’ (which stands for eXclusion constraint) between

‘Reviewer Reviews/ReviewedBy Paper’ and ‘Author Writes/WrittenBy Paper’ means that an Author (Person) who has written a certain Paper is not allowed to be a reviewer of the same paper. An arrow between two predicates indicates a subset constraint between the roles involved: ‘Each Author who presents a Paper must have written that Paper’.

3 ORM Markup Language

As we noted in the introduction, conceptual modeling approaches were developed to assist system developers during the design phases. They were not meant to be accessed and processed at the run-time of applications. Besides, conceptual diagrams are typically saved in graphical formats, which are proprietary and therefore are limited to use inside specific CASE tools.

In this section we show how conceptual diagrams can be marked up and thus accessed and processed at run-time of applications. We illustrated this by defining a new open syntax markup language to represent ORM conceptual diagrams textually. ORM Markup Language is based on the XML syntax, and is defined in an XML-Schema [ORMML-XMLS] that acts as its complete and formal grammar, thus any ORM-ML file 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 as a “save as” or “export to” functionality in ORM tools. In section 6 we will present an ontology engineering tool that makes use of ORM-ML.

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 A. 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 loss of information or change in semantics, except for the geometry and topology (graphical layout) of the schema (e.g. location, shapes of the symbols), which we provide as a separate *graphical style sheet* to the ORM Schema.

We represent the ORM document as a one node element called ORMSchema, which consists itself of two nodes: ORMMeta and ORMBody. As a header to an ORM document, and to illustrate the "ORM Schema Document" (instance) nature of the described schema, ORMMeta node includes *meta data* about the ORM *document* using the 16 Dublin Core[4] Meta Tags [RFC2413]; an example of their use appears in Table 1 below.

```
…<ORMMeta>
     <dc:Title>Bookstore</dc:title>
     <dc:Creator>Mustafa Jarrar</dc:creator>
     <dc:contributor>Jan Demey</dc:contributor>
    <dc:contributor>Robert Meersman</dc:contributor>
```

[4] Dublin Core is a standard of 16 metadata tags that can be used for “annotating” any information object.

```
        <dc:Description>An example of ORM-ML </dc:description>
        <dc:Language>English</dc:language>.
        …..
</ORMMeta>….
```

Table 1. Example of an ORMMeta Node in an ORM-ML File

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

We adopt in the sequel the ORM modeling technique as defined in [H01]. Object elements are abstract XML elements and 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. 2 in Example 2). Objects might have some Value or ValueRange elements, which are used for value constraints on the Object Type (not present in Fig. 2). A ValueRange element has 2 attributes: begin and end, with obvious meanings. Object Types 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]). 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 2 attributes: ref_name, the name of the reference and numeric, a boolean to indicate whether it is a numeric reference.

Example 2

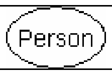
	… <Object xsi:type="NOLOT" Name="Person"/> …

Fig. 2 **Table 2**. ORM-ML representation of Fig. 2

Subtype elements are used to represent subtype relationships between non-lexical object types. A subtype element is required to have two attributes: parent and child, which are references to object elements (see Example 3).

Example 3

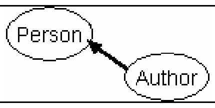
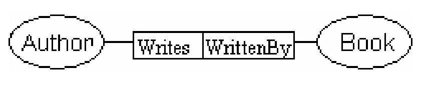
	… <Object xsi:type="NOLOT" Name="Person"/> <Object xsi:type="NOLOT" Name="Author"/> <Subtype Parent="Person" Child="Author"> …

Fig. 3. **Table 3.** ORM-ML representation of Fig. 3

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. Note that we did not put the 'ID' attribute to refer to predicates rather than to object_role elements. The reason therefore is that we need to be able to refer to specific object-roles 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.

Example 4. This example shows a simple binary predicate as in fig 4, and how it is represented in ORM-ML in Table 4.



```
…
<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>
```

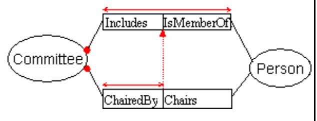
Fig. 4 **Table 4.** ORM-ML representation of Fig. 4

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, frequency, and ring constraints: irreflexive, anti-symmetric, asymmetric, symmetric, intransitive, and acyclic constraints. As mentioned above, we use the ID-s of the Object_Role elements to define constraints (except for value constraints on an object type, since these are defined in the corresponding object element).

Uniqueness and mandatory constraint elements possess only Object_Role elements (at least one). 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 Example 6 below).

The representation for subset, equality and exclusion constraints is analogous, so we will only discuss them in general terms. Each of these latter constraints has exactly two elements that contain references to (combinations of) object_role elements. For example, to represent an equality constraint between two predicates, we create a subset element, containing 2 elements ‘First’ and ‘Second’. In the first element we put references to the object_roles from the first predicate, and in the second we put references to the object_roles from the second predicate (see Example 6).

Example 6. This example shows the representation of the constraints from Fig. 6.



```
…<Predicate>
   <Object_Role ID="2" Object="Committee" Role="Includes"/>
   <Object_Role ID="3" Object="Person"  Role="IsMemberOf"/>
</Predicate>
<Predicate>
   <Object_Role ID="4" Object="Committee" Role="ChairedBy"/>
   <Object_Role ID="5" Object="Person" Role="Chairs"/>
</Predicate>
<Constraint xsi:type="Mandatory">
    <Object_Role>2</Object_Role></Constraint>
<Constraint xsi:type="Mandatory">
    <Object_Role>4</Object_Role></Constraint>
<Constraint xsi:type="Uniqueness">
   <Object_Role>4</Object_Role></Constraint>
<Constraint xsi:type="Uniqueness">
```

	<Object_Role>2</Object_Role> <Object_Role>3</Object_Role> </Constraint> <Constraint xsi:type="Subset"> <Parent> <Object_Role>2</Object_Role> <Object_Role>3</Object_Role> </Parent> <Child> <Object_Role>4</Object_Role> <Object_Role>5</Object_Role> </Child> </Constraint> …

Fig. 6. **Table 6.** ORM-ML representation of Fig. 6

Finally, *ring constraint* elements simply contain references to the object_roles they are put on, and frequency constraints have two attributes: a reference to the object_role the constraint is placed on and an attribute called 'Frequency' which contains the declared frequency number.

A markup language such as ORM-ML has more advantages than just enabling the use of conceptual schemas at the runtime of applications. We name some of them:

- Building Style-sheets. As ORM-ML is very easy to parse and interpret, we can easily build style-sheets to translate ORM-ML files into other languages. For instance, we have already built a style-sheet to translate ORM-ML files into pseudo natural language sentences, which is used in DogmaModeler (see fig. 13 and fig. 16 in section 5). Moreover, it would be very useful to write style-sheets that translate ORM-ML into rule engine's languages. Experiments with Haley's Authorete™ [Haley] are being planned in the near future. Of course, such a style-sheet would open doors for ORM as a very powerful business rule modeling language. Other style-sheets can be built to translate ORM-ML into other types of conceptual schemes, ontology languages e.g. DAML, or into first order/description logic formalisms, etc.
- Easier schema integration and transformation. For integrating information systems, it is in general easier to integrate or *align* the conceptual models of these systems than integrate their logical or the physical internal representation as demonstrated by the literature on view- and schema integration (e.g. [SP94]). Therefore, ORM-ML as a standardized syntax for ORM models may assist interoperation tools to exchange, parse or understand the ORM schemes.
- Conceptual queries over the web. In open and distributed environments, building 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 more expressive and 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, NIAM/ORM-based query languages (e.g. RIDL [VB82], [M81], ConQuer [BH96]) would gain from ORM-ML by representing queries in such an exchangeable representation.

Of course, *similar to ORM-ML, a markup language could be defined for any other conceptual modeling method.* We have chosen ORM to *illustrate* adopting conceptual modeling methods for ontology engineering purposes because ORM has several strengths over other methods [H01]: 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, anti/a/symmetric… derivation rules, etc.). Furthermore, ORM has an expressive and stable graphical notation since it captures many rules graphically and it minimizes the impact of change to models[5]. 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 be a stepwise methodology arriving at "semantics" of a business application's data based on natural language communication.

In section 4 and 5, we show how ORM-ML is used as ontological commitment language.

4 DOGMA approach for ontology Engineering

In this section, we present the second topic of this paper: we first discuss some application-independency issues of ontologies and conceptual data schemes, i.e. domain vs. application conceptual modeling; then we present our DOGMA[6] ontology engineering framework that enables the use of conceptual modeling methods, such as ORM and its ORM-ML, for modeling and representing ontologies.

Similar to conceptual data schemes, ontologies consist of interrelated concepts and rules (e.g. identity, mandatory, value, cardinality, taxonomy, etc.) that constrain and specify the intended meaning of the concepts. However, since conceptual schemes were developed only for the use of an enterprise application(s), thus, building such schemes depends on the specific needs and tasks that are planned to be performed within a certain enterprise. In comparison, building ontologies is a challenging job, since ontologies are supposed to hold application-independent domain knowledge. The consensus level about ontological content is the main requirement in ontology modeling, and mainly distinguishes it from conceptual data modeling.

An expected question may arise, namely: how can one decide whether knowledge is application-independent? Certainly, ontology modelers will not manage to come to an agreement for all applications that can possibly exist in a domain. This is why we believe, like [CJ93], that there is no strict line between the levels of dependent and independent (or generic and specific) knowledge. *When building an ontology, there will always be intended or expected needs and tasks “at hand”*, which will influence the independency level of the ontology. In short, *there is a clash between building an application-independent ontology and encountering the intended goals for building this ontology*. In the problem solving research community, this issue is called the *interaction problem*, which influences the independency of the problem-solver’s domain

[5] 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.

[6] Developing Ontology-Guided Mediation for Agents

knowledge [HSW97]. For example, Bylander and Chandrasekaran argued in [BC88] that:

> "Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem."

Solving such a clash requires a *principled methodology* to guide ontology modelers towards more genericity, as well as meeting requirements at hand. Notice that the methodologies that emphasize on the requirements "at hand", or that evaluate ontologies based only on how they fulfill specific requirements, will lead to *application ontologies*, similar to conceptual data schemas containing less reusable knowledge. Likewise, the methodologies that emphasize only on the genericity of the knowledge will lead to less usable ontologies, since they have no intended use by ignoring requirements at hand [BBH96][RVMS99].

In our approach, we introduce an essential principle that underpins the foundation of ontologies. Unlike ontology engineering proposals, which consider an ontology as one "unit", holding both conceptual relations and rules together (e.g. [G95], [G98], [FHVDEK00]): we decompose an ontology into an ontology base and a layer of ontological commitments. The ontology base holds conceptual relations, as domain knowledge. The commitment layer consists of a set of ontological commitments, where each commitment holds ontology rules, which *formally and explicitly* provide an interpretation of an application or task in terms of the domain knowledge, (see fig. 7). We will show that ontology rules are mostly application/task-dependent knowledge, i.e. strongly influenced by the intended use of the knowledge and requirements at hand. Therefore, *as a result of the decomposition, the genericity of the knowledge in the ontology base level is increased, while rules influenced by requirements at hand are kept separated in the commitment layer. Hence, a conceptual schema can be seen as an ontological commitment defined in terms of the domain knowledge*. In what follows, we respectively describe the ontology base and the commitment layer, illustrating both by means of a detailed example. For further details about the DOGMA approach, see [JM02a] and [JM02b].

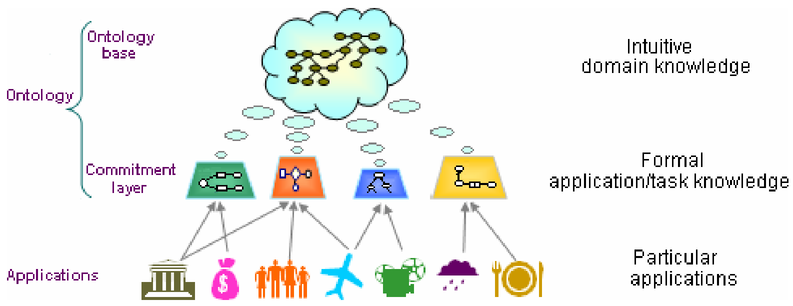
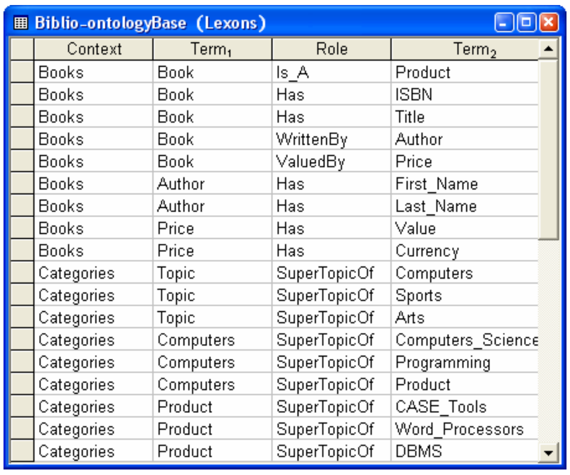


Fig. 7. Knowledge organization in the DOGMA approach

<u>The ontology base</u> consists of "plausible" intuitive domain fact types, represented and organized as a set of context-specific binary conceptual relations, called *lexons*. A lexon is formally described as a 4-tuple of the form <γ: Term$_1$, Role, Term$_2$>, where γ

is a context identifier, used to group lexons that are intuitively and informally "related" in an intended conceptualization of a domain. For each context γ and Term T, the pair (γ, T) is assumed to refer to a *concept*.

Fig. 8. shows an example of an ontology base represented in a table format (see fig. 13 for its corresponding "tree" representation using DogmaModeler). The ontology base in this example consists of two contexts: "Books" and "Categories". Notice that the term "Product" that appears within both contexts refers to two different concepts: the intended meaning of "Product" within the context "Categories" –that deal with the subject classification of books- refers to a topic or a subject of a book, while within the context "Books", it refers to a real world's entity. More about our notion of context will be discussed at the end of this section.



Context	$Term_1$	Role	$Term_2$
Books	Book	Is_A	Product
Books	Book	Has	ISBN
Books	Book	Has	Title
Books	Book	WrittenBy	Author
Books	Book	ValuedBy	Price
Books	Author	Has	First_Name
Books	Author	Has	Last_Name
Books	Price	Has	Value
Books	Price	Has	Currency
Categories	Topic	SuperTopicOf	Computers
Categories	Topic	SuperTopicOf	Sports
Categories	Topic	SuperTopicOf	Arts
Categories	Computers	SuperTopicOf	Computers_Science
Categories	Computers	SuperTopicOf	Programming
Categories	Computers	SuperTopicOf	Product
Categories	Product	SuperTopicOf	CASE_Tools
Categories	Product	SuperTopicOf	Word_Processors
Categories	Product	SuperTopicOf	DBMS

Fig. 8. Example of an ontology base

The layer of ontological commitments mediates between the ontology base and its applications. Each commitment consists of: (1) an *ontological view* that specifies which lexons from the ontology base are *relevant*[7] to this commitment, i.e. selection of lexons, and (2) rules that formally constrain and specify the intended meaning of the selected lexons. For simplicity, one can see a commitment as a conceptual schema, where its conceptual relations correspond to lexons in the ontology base. Applications that use (or more precisely, "commit to") a certain commitment must satisfy all rules declared in this commitment. In other words, any possible world, for an application, must conform to the rules declared in its commitment(s) (cf. model-theoretic semantics).

Each ontological commitment corresponds to an explicit *instance* of an (intensional) first order *interpretation* of the domain knowledge in the ontology base. In other words, it is the role of commitments to provide the formal interpretation(s) of the

[7] Notice that the relevancy is an application-dependent choice.

lexons. Therefore, the lexons in an ontology base are free of a *particular* formal interpretation. This allows different formalizations and interpretations, even if sometimes they disagree about certain things, to co-exist as different commitments in the commitment layer and to share what they have in common.

In our approach, ontological commitments are not restricted to be expressed in a certain specification language. However, as the goal of this paper is to enable the use of conceptual modeling methods, we illustrate representing ontological commitments using the ORM-ML. In the next example, as well as in the next section, we illustrate how commitments can be modeled using the ORM graphical notation, while the corresponding ORM-ML is being generated automatically.

In the next example, we show how an ontology base -holding domain knowledge- may have different formal interpretations in different commitments, and that is because of the difference in the intended use of the same domain knowledge.

Example

The example given below is based on the Biblio-OntologyBase domain knowledge provided in Fig. 8. We present two different *kinds* of applications: library applications that need to interoperate with other libraries, and bookstore applications that additionally need to interoperate with other bookstores. Each kind of application has different rules (i.e. interpretation) that do not necessarily agree with the other's rules. E.g., unlike bookstores, pricing information is not relevant for library applications. Likewise, bookstores identify a book by its ISBN, while in library applications, ISBN is not a mandatory property for every instance of a book, so it cannot be used for identity[8] purposes; instead, they identify a book by the combination of its title and authors. So for bookstores, instances of a "Thesis", a 'Manual', etc. are not considered as books -since they do not have ISBN- while for libraries they are. However, suppose that both bookstore and library applications have the same agreement about categories (i.e. topics of books). Fig. 9, Fig. 10, and Fig. 11 illustrate the use of the ORM graphical representation of ontological commitments for 'Bookstore' 'Library', and 'Categories' respectively.

[8] Notice that in ORM, the identity of a concept is based on the Uniqueness and Mandatory properties together. For example, if an ISBN is a mandatory and a unique property for all instances of the concept book, then one can use the ISBN as an identify property for the concept book. Furthermore, in ORM, one can use the combination of two (or more) properties to identify a concept, e.g. the combination of (title, author) of a book. In that case both properties must be unique and mandatory for every instance of a book, at any time.

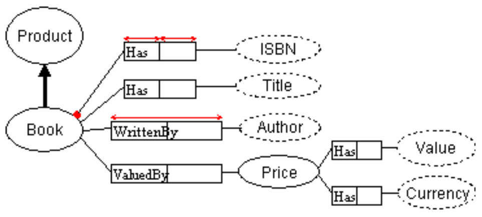


Fig. 9. ‘Bookstore’ Ontological Commitment

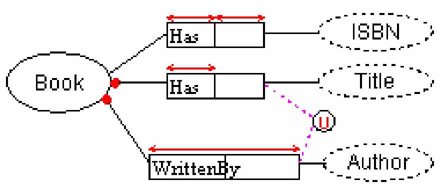


Fig. 10. ‘Library’ Ontological Commitment

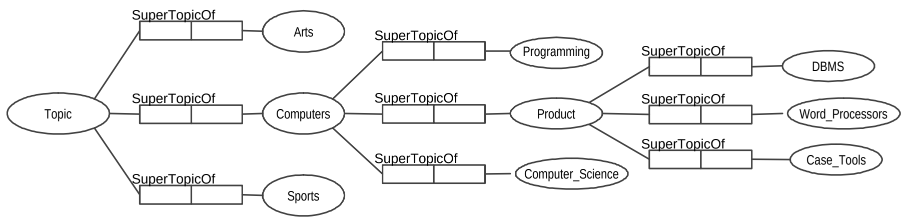


Fig. 11. ‘Categories’ Ontological Commitment[9]

All commitments share the same Biblio-OntologyBase shown in Fig. 8: all fact types in each commitment correspond to lexons in the ontology base. Notice that a commitment only uses the lexons that are relevant to it, i.e. the ontological view. For example, the lexons {<Books: Book, ValuedBy, Price>, <Books: Price, Has, Value>…} do not appear in the ‘Library’ commitment.

Fig. 12 shows commitments sharing the same Biblio-OntologyBase together with some bookstore applications committing to the ‘Bookstore’ and ‘Categories’ commitments, and some library applications committing to the ‘Library’ and ‘Categories’ commitments.

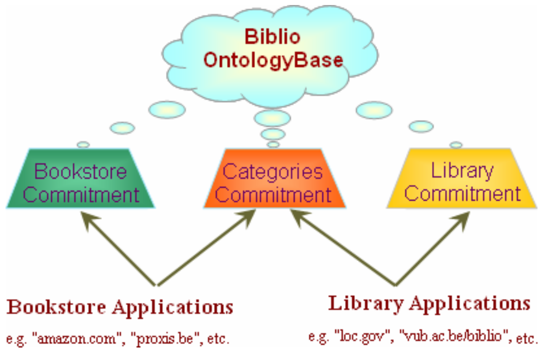


Fig. 12. Using ontological commitments

[9] Notice that we do not use the subtype relationship in ORM to represent the “SuperTopicOf” relationship in Fig. 11; since they do not have the same formal semantics. For more information about the formalization of topics and subjects, see [WJ99].

Discussion

One can see from the previous example that application-kinds, even within the same domain, may have different formal interpretations of the same knowledge. For example, the set of possible instances of the concept "book" for library applications is formally not the same as for bookstore applications, since both have different identity criteria, etc. Nevertheless, in reality, both kinds of applications intuitively share the same concept of what is really a book. For example, suppose that one assigns an ISBN for an instance of a "Master Thesis", then this instance can be considered as a book for bookstores, or that one removes an ISBN for an instance of a book, then this instance will no longer be a book, although, this instance remains the same real life object and is still being referred to and used as a book. Obviously, one of the main requirements at hand (i.e. intended use) for bookstores is "what can be sold", which influences the identity criteria and thus the formal interpretation of the domain knowledge. However, there is more to "say", than an ISBN, in order to identify a book in real world domains (e.g. structuring, formatting, authoring, publishing, printing, versioning, copying…). And, that the identity criteria should be based on essential properties that help to distinguish real world objects as being the same or not [G02]. In other words, for modelling ontologies at the domain level, the identity of a concept should serve to identify all of its instances in all applications.

Please note that the aim of this paper is not to discuss the identity rule, but we use this example to illustrate the difference (and requirements) between modeling knowledge at the domain level vs. modeling knowledge at the application level.

At the domain level, in our approach, we have introduced the notion of context; so that a term within a context refers *intuitively* to a concept. The intuition that a context provides here is: *a set of implicit or maybe tacit*[10] *assumptions that are necessary for all instances of a concept in all its applications.* In other words, a context is an abstract identifier that refers to implicit and tacit assumptions in a domain, and that maps a term to its intended meaning (i.e. concept) within these assumptions. Notice that a context in our approach is not explicit formal knowledge. In practice, we define a context by referring to a source (e.g. a set of documents, laws and regulations, informal description of "best practice", etc.), which, by *human understanding*, is assumed to "contain" the necessary assumptions.

We suppose that the ontology base –as intuitive domain knowledge- is free of any particular formal interpretation; or rather, lexons are assumed (by human understand-

[10] The difference between implicit and tacit assumptions, is that the implicit assumptions, in principle, can be articulated but still they have not, while tacit assumptions are the knowledge that cannot be articulated: it consists partially of technical skills –the kind of informal, hard-to-pin-down skills captured in terms like "know-how", "we know more than we can tell or put in words", etc. Even if tacit assumptions cannot be articulated, but they can be transferred through other means over than verbal or formal descriptions [Innovanet03] [N94]. In many cases, the knowledge about the real world (such as the identity of a person, book, etc.) is tacit knowledge [P96].

ing) to be “true within their context’s source”. The formal interpretation of the lexons is provided through ontological commitments, which are explicit and formal (and thus machine-understandable) knowledge.

As a result and as we have illustrated before, we enable the use of conceptual modeling methods for modeling ontological commitments. The application-independency level of an ontology is increased, by separating the commitments (mostly application/task-dependent knowledge) from the ontology base (intuitive domain knowledge). In other words, the interaction problem has “neglectable” influence on the genericity of the ontology base level, because ontology modelers are *prevented* from entering their application-specific rules at this level.

Remark: In accordance to the given independency discussion, we emphasize that modeling ontological commitments should not be specific to certain needs, *they should be made more generic (e.g. describing application kinds, generic tasks, etc.), and seen as reusable components of knowledge*. In our approach, the ontological commitments that are specific to a limited number of applications do not affect the independency of other commitments in the same commitment layer. Rather, *Commitments -specially large ones- can even be modularized into smaller and interrelated commitments*, so that the general purpose –i.e. reusable- parts can be separated from the more specific parts [JM02b]. And therefore, not only the ontology base (i.e. lexons and the intuitive definitions of their terms) can be shared and reused among commitments, but also the ontological commitments themselves can be modularized and seen as a set of reusable knowledge components.

5 DogmaModeler ontology engineering tool

This section briefly outlines our DogmaModeler Tool prototype for ontology engineering. Its implementation is based on the approach described in this paper.

DogmaModeler supports functionalities for modeling, browsing, and managing both the ontology base and the commitments. It supports modeling ontological commitments using the ORM graphical notation, and it generates the corresponding ORM-ML automatically. In addition, DogmaModeler supports verbalization of ontological commitments into pseudo natural language. Fig. 13 shows a screenshot of DogmaModeler with demonstrates its three main windows: the ontology base window, the commitment modeling window, and the commitment library window. We will describe these windows in what follows.

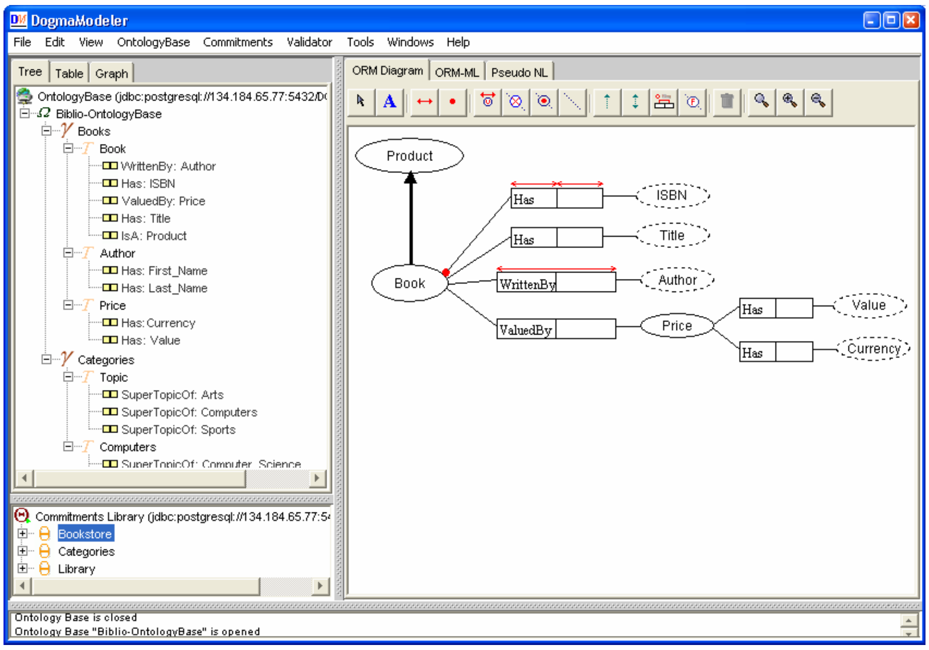


Fig. 13. A screenshot of DogmaModeler.

Ontology base window (The top left side of Fig. 13)

Before building ontological commitments, ontology builders should define their lexons in the ontology base window, in case it is empty. This window presents the set of lexons – {< γ : Term$_1$, Role, Term$_2$>} - in a tree-like structure[11]. The first level, (*W*) represents ontology bases (e.g. Biblio-Ontologybase). In the second level, each node (g) represents a context (e.g. Books). Within a context, each node (*T*), in the third level, represents a term; while nodes (▭) in the fourth level, represent the set of (Role, Term$_2$) for that term.

Notice that level 0 () in the tree represents an ontology base server, where the content of ontology bases is hosted and managed. All transactions on the ontology base (e.g. creating contexts, editing lexons, etc.) will be transmitted, verified and executed on the server side. As one can see in Fig. 13, DogmaModeler is connected with our DogmaServer[12], which stores and serves the ontology base and the commitment layer.

Commitment modeling window (The right side of Fig. 13)

This window consists of three panels: ORM, ORM-ML, and Pseudo NL. To build an ontological commitment, ontology builders can drag and drop lexons from the ontology base window into the ORM panel (i.e. defining the ontological view). When doing

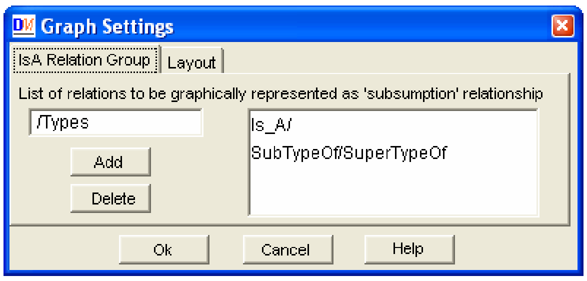
[11] The ontology base tree has advanced features, so it can also be browsed and seen as a graph.

[12] For more details, or to download DogmaServer, you can access: http://www.starlab.vub.ac.be/research/dogma/OntologyServer.htm.

so, lexons will be mapped automatically into ORM fact types. Then, in order to define rules on these lexons, ontology builders can use the ORM family of constraints; see icons in the top of the ORM panel.

Remark: mapping lexons as intuitive domain knowledge into ORM fact types that have predefined formal semantics [V82] is done as the following: a Term within a context is mapped directly into an Object Type in ORM, Roles within a context are also mapped directly into ORM Roles. While in case of ORM Subtype relations that have specific “build-in” semantics, commitment builders need to customize the “Graph settings” window, in order to specify which roles should be mapped, see Fig. 14. Further, DogmaModeler does not support ORM unary roles and nested fact types.

As we mentioned before, our approach is not restricted to ORM; the tool is designed with flexibility of adding new plug-ins in order to support modeling commitments in other languages, e.g. EER, UML, DAML, OWL etc.



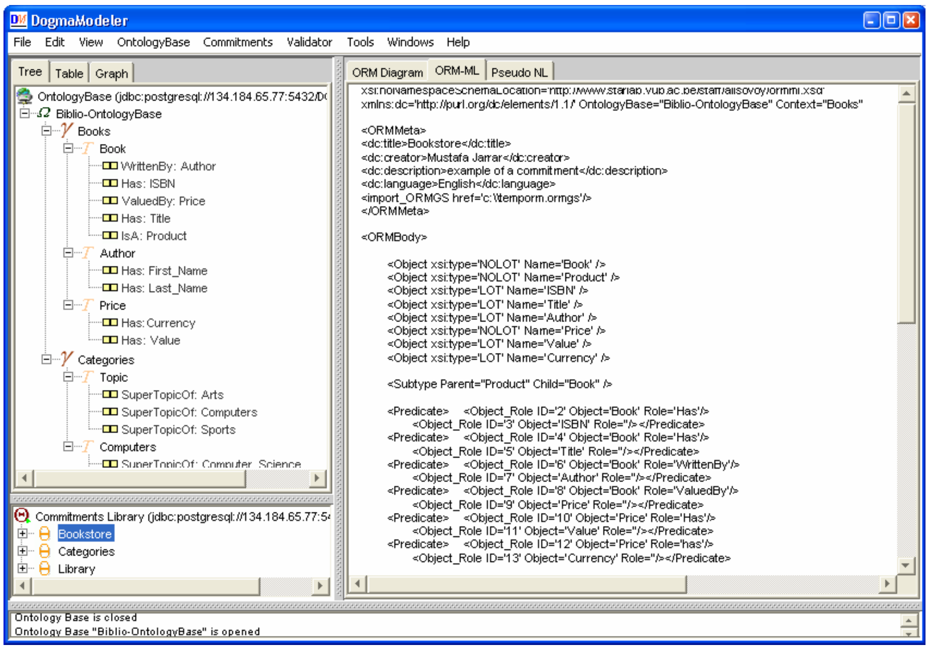
List of relations to be graphically represented as 'subsumption' relationship

/Types

Is_A/
SubTypeOf/SuperTypeOf

Fig. 14: Mapping to ORM Subtyping relationship

Fig. 15 shows the corresponding ORM markup language of the ORM model in Fig. 13, which is automatically generated by the tool. DogmaModeler supports saving ORM-ML into text files, or uploading it into an ontology server.


DogmaModeler
File Edit View OntologyBase Commitments Validator Tools Windows Help
Tree Table Graph
ORM Diagram ORM-ML Pseudo NL

```
OntologyBase (jdbc:postgresql://134.184.65.77:5432/D
  Biblio-OntologyBase
    Books
      Book
        WrittenBy: Author
        Has: ISBN
        ValuedBy: Price
        Has: Title
        IsA: Product
      Author
        Has: First_Name
        Has: Last_Name
      Price
        Has: Currency
        Has: Value
    Categories
      Topic
        SuperTopicOf: Arts
        SuperTopicOf: Computers
        SuperTopicOf: Sports
      Computers
        SuperTopicOf: Computer_Science
```

```
Commitments Library (jdbc:postgresql://134.184.65.77:5
  Bookstore
  Categories
  Library
```

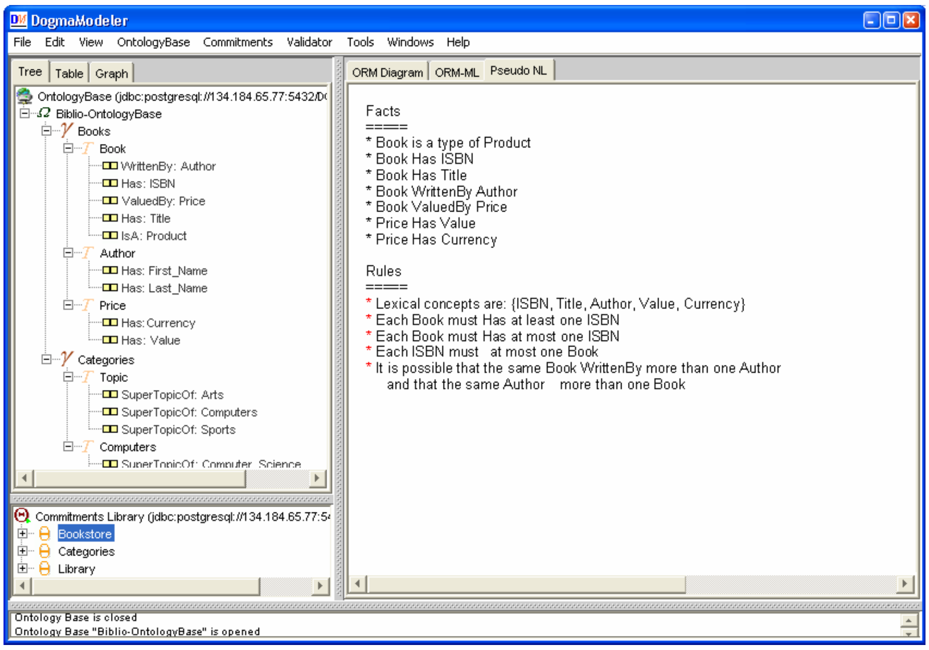
```xml
xsi:noNamespaceSchemaLocation='http://www.starlab.vub.ac.be/staff/ansovoy/ormml.xsd'
xmlns:dc="http://purl.org/dc/elements/1.1/" OntologyBase="Biblio-OntologyBase" Context="Books"

<ORMMeta>
<dc:title>Bookstore</dc:title>
<dc:creator>Mustafa Jarrar</dc:creator>
<dc:description>example of a commitment</dc:description>
<dc:language>English</dc:language>
<import_ORMGS href='c:\\temporm.ormgs'/>
</ORMMeta>

<ORMBody>

    <Object xsi:type='NOLOT' Name='Book' />
    <Object xsi:type='NOLOT' Name='Product' />
    <Object xsi:type='LOT' Name='ISBN' />
    <Object xsi:type='LOT' Name='Title' />
    <Object xsi:type='LOT' Name='Author' />
    <Object xsi:type='NOLOT' Name='Price' />
    <Object xsi:type='LOT' Name='Value' />
    <Object xsi:type='LOT' Name='Currency' />

    <Subtype Parent="Product" Child="Book" />

    <Predicate>    <Object_Role ID='2' Object='Book' Role='Has'/>
        <Object_Role ID='3' Object='ISBN' Role=''/></Predicate>
    <Predicate>    <Object_Role ID='4' Object='Book' Role='Has'/>
        <Object_Role ID='5' Object='Title' Role=''/></Predicate>
    <Predicate>    <Object_Role ID='6' Object='Book' Role='WrittenBy'/>
        <Object_Role ID='7' Object='Author' Role=''/></Predicate>
    <Predicate>    <Object_Role ID='8' Object='Book' Role='ValuedBy'/>
        <Object_Role ID='9' Object='Price' Role=''/></Predicate>
    <Predicate>    <Object_Role ID='10' Object='Price' Role='Has'/>
        <Object_Role ID='11' Object='Value' Role=''/></Predicate>
    <Predicate>    <Object_Role ID='12' Object='Price' Role='has'/>
        <Object_Role ID='13' Object='Currency' Role=''/></Predicate>
```

```
Ontology Base is closed
Ontology Base "Biblio-OntologyBase" is opened
```

Fig. 15 The ORM-ML panel window

Fig. 16 shows the corresponding Pseudo Natural language (fixed-syntax English sentences) of the ORM model from Fig. 13. It is automatically generated by the tool by applying predefined templates to the commitments' content. We believe that this allows non-experts to (help to) check, validate or build the commitment rules and will simplify the modeling process.

<u>Commitment library window</u> (Under the ontology base window)
The purpose of this window is to enhance the reusability, management, and organization of ontological commitments. The current implementation allows ontology builders to access and browse ontological commitments stored in a library (Q). Each node (⊖) in the first level of the tree represents a commitment. By expanding a commitment node, the set of lexons and the set of rules -subject to this commitment- will appear in the second level. Advanced features e.g. indexing, modularization/composing, versioning, etc. of ontological commitments are ongoing research issues.


DogmaModeler
File Edit View OntologyBase Commitments Validator Tools Windows Help
Tree Table Graph
ORM Diagram ORM-ML Pseudo NL

```
OntologyBase (jdbc:postgresql://134.184.65.77:5432/D
  Biblio-OntologyBase
    Books
      Book
        WrittenBy: Author
        Has: ISBN
        ValuedBy: Price
        Has: Title
        IsA: Product
      Author
        Has: First_Name
        Has: Last_Name
      Price
        Has: Currency
        Has: Value
    Categories
      Topic
        SuperTopicOf: Arts
        SuperTopicOf: Computers
        SuperTopicOf: Sports
      Computers
        SuperTopicOf: Computer_Science

Commitments Library (jdbc:postgresql://134.184.65.77:5
  Bookstore
  Categories
  Library
```

```
Facts
=====
* Book is a type of Product
* Book Has ISBN
* Book Has Title
* Book WrittenBy Author
* Book ValuedBy Price
* Price Has Value
* Price Has Currency

Rules
=====
* Lexical concepts are: {ISBN, Title, Author, Value, Currency}
* Each Book must Has at least one ISBN
* Each Book must Has at most one ISBN
* Each ISBN must   at most one Book
* It is possible that the same Book WrittenBy more than one Author
    and that the same Author   more than one Book
```

```
Ontology Base is closed
Ontology Base "Biblio-OntologyBase" is opened
```

Fig. 16. The Pseudo NL panel window

6 Conclusions and Future Work

This paper has presented and discussed two main disparities between conceptual data schemes and ontologies, and we have shown how these disparities can be tackled when adopting conceptual data modeling techniques for ontology engineering purposes. First, we have presented how conceptual diagrams can be marked up and thus accessed and processed at run-time of applications. We have illustrated this by defining a conceptual markup language in order to textually represent ORM conceptual diagrams. Second, we have discussed and analyzed the differences between modeling knowledge at the application level vs. modeling knowledge at the domain level and thus conceptual data schemes vs. ontologies. We have presented an ontology engineering approach and tool that increase the application-independency of an ontology by decomposing it into an ontology base (that holds intuitive domain knowledge) and a set of ontological commitments (that hold application specific knowledge). Hence, we have enabled conceptual data modeling methods to be used for modeling and representing ontological commitments.

Future research on our approach concerns the development and engineering of commitment libraries, which open new several research issues such as e.g. indexing, versioning, distributed development, modularization, etc. of ontological commitments. Currently, our main priority is to develop and formalize a methodology for modularizing (and thus composing) ontological commitments. As a work in progress, we have defined an inclusion interrelationship between ontological commitments, so that all concepts and constraints introduced in the included commitment will be inherited in the including commitment.

Acknowledgments. The authors are grateful to Jan De Bo, Peter Spyns, Sven Van Acker, Ruben Verlinden, Pieter De Leenheer, and Andriy Lisovoy for their comments and suggestions on an earlier version of this paper.

References

[BB03] Borgida, A., Brachman, R.: Conceptual Modeling with Description Logics. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., (eds): The Description Logic Handbook, Theory, Implementation and Applications. ISBN: 0521781760 (2003).

[BBH96] Beys, P., Benjamins, R., van Heijst, G.: Remedying the reusability-usability trade-off for problem solving methods. In B.R. Gaines and M. Mussen, (eds): Proceedings of the KAW-96, Banff, Ca, (1996)

[BC88] Bylander, T., Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: The right level of abstraction for knowledge acquisition. In: Gaines B., Boose, J. (eds): Knowledge Acquisition for Knowledge Based Systems. Vol. 1. Academic Press, London, (1988) 65-77

[BCD01] Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML Class Diagrams using Description Logic Based Systems. In: Workshop on Applications of Description Logics- ADL-2001, (2001)

[BH96] Bloesch, A., Halpin, T.: ConQuer: a Conceptual Query Language. In: Thalheim, B. (ed.): Conceptual Modeling - ER'96 Proceedings, Lecture Notes in Compute Science, Springer-Verlag, (1996) 121-33

[BHW91] van Bommel, P., ter Hofstede, A., van der Weide, Th. P.: Semantics and verification of object-role models. Information Systems, 16(5), October (1991) 471- 495

[BKKHSHLA01] Baclawski, K., Kokar, M., Kogut, P., Hart, J., Smith, J., Holmes, W., Letkowski J., Aronson, M.: Extending UML to Support Ontology Engineering for the Semantic Web, 4th International Conference on UML, (2001) 342-360

[CHP01] Cranefield, S., Haustein, S., Purvis, M.: UML-Based Ontology Modelling for Software Agents. In: Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, Montreal (2001) 21-28

[CJ93] Chandrasekaran, B., Johnson T.: Generic Tasks and Task Structures: History, Critique and New Directions. In: David, J., Krivine, J., Simmons, R. (eds.): Second Generation Expert Systems, Springer, (1993) 233 – 272

[CP99] Cranefield, S., Purvis, M.: UML as an ontology modelling language. In: Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence, IJCAI-99, (1999)

[F02] Franconi, E.: Tutorial on Description Logics for Conceptual Design, Information Access, and Ontology Integration: Research Trends. 1st International Semantic Web Con. (2002)

[FHVDEK00] Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., Klein, M.,: Oil in a nutshell. 12th International Conference on Knowledge Engineering and Knowledge Management EKAW 2000, Juan-les-Pins, France, (2000)

[G95] Gruber T.R.: Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human-Computer Studies, 43(5/6) (1995)

[G98] Guarino, N.: Formal Ontology in Information Systems. In: Proceedings of FOIS'98, IOS Press, Amsterdam, (1998) 3-15

[GHW02] Guizzardi, G., Herre, H., Wagner G.: Towards Ontological Foundations for UML Conceptual Models. 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE'02), Lecture Notes in Computer Science, Vol. 2519, Springer-Verlag, Berlin (2002) 1100-1117

[GW02] Guarino, N. and Welty, C.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM, 45(2): 61-65 (2002).

[H89] T.A. Halpin. A logical analysis of information systems: static aspects of the data-oriented perspective. PhD thesis, University ofQueensland, Brisbane, Australia, 1989.

[H01] Halpin, T.: Information Modeling and Relational Databases. 3rd edn, Morgan-Kaufmann

[Haley] The HaleyEnterprise, http://www.haley.com.

[HP95] Halpin, T., Proper, H.: Subtyping and polymorphism in object-role modeling. Data & Knowledge Engineering 15(3), (1995) 251-281

[HPW93] ter Hofstede, A., Proper, H., van der Weide, T.: Formal definition of a conceptual language for the description and manipulation of information models. In: Information Systems 18(7), October (1993) 471-495

[HSW97] van Heijst, G., Schreiber, A., Wielinga, B.: Using Explicit Ontologies in KBS Development, International Journal of Human-Computer Studies, 46, (1997) 183-292.

[Innovanet03] Persidis A., Niederée C., Muscogiuri C., Bouquet P., & Wynants M.: Innovation Engineering for the Support of Scientific Discovery. Innovanet Project (IST-2001-38422), deliverable #D1, 2003.

[JM02a] Jarrar, M., Meersman, R.: Formal Ontology Engineering in the DOGMA Approach. In: 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE'02), Lecture Notes in Computer Science, Vol. 2519, Springer-Verlag, Berlin (2002) 1238 - 1254

[JM02b] Jarrar, M., Meersman, R.: Scalability and Knowledge Reusability in Ontology Modeling, In: Proceedings of the International conference on Infrastructure for e-Business, e-Education, e-Science, and e-Medicine (SSGRR2002s) (2002)

[M81] Meersman, R.: Languages for the High-Level End User. In: InfoTech State of the Art Report, Pergamon Press (1981)

[M99] Meersman R.: The Use of Lexicons and Other Computer-Linguistic Tools. In Zhang Y., Rusinkiewicz M, & Kambayashi Y. (eds.): Semantics, Design and Cooperation of Database Systems, in The International Symposium on Cooperative Database Systems for Advanced Applications (CODAS 99), Springer Verlag, Heidelberg, (1999) 1 – 14

[M01] Meersman, R.: Ontologies and Databases: More than a Fleeting Resemblance. In d'Atri A., Missikoff, M. (eds): OES/SEO 2001 Rome Workshop, Luiss Publications (2001)

[MC02] Meisel, H., Compatangelo, E.: EER-ConcepTool: a “reasonable” environment for schema and ontology sharing. In: Proc. of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’2002), IEEE Computer Society Press (2002) 527–534

[N94] I. Nonaka, A dynamic theory of organizational knowledge creation, In: Organizational Science, Vol. 5, No. 1 (1994), pp. 14-37

[ORMML-XMLS] http://www.starlab.vub.ac.be/ORMML/ormml.xsd

[P96] Polany, M.: The Tacit Dimension. Doubleday, Garden City-N.Y. (1996)

[RFC2413] http://www.ietf.org/rfc/rfc2413.txt (Dublin Core definition)

[RVMS99] Russ, T., Valente, A., MacGregor, R., Swartout, W.: Practical Experiences in Trading Off Ontology Usability and Reusability. In Proceedings of the Twelfth Banff Knowledge Acquisition for Knowledge-based Systems Workshop, pp. 4-11-1 to 4-11-20, (1999)

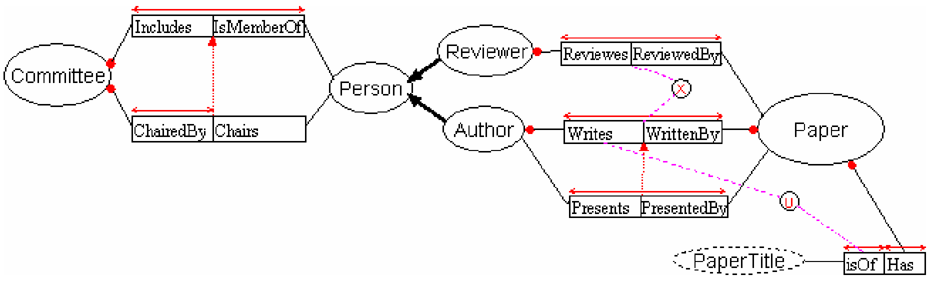
[SOVZJSSM02] Spyns, P., Oberle, D., Volz, R., Zheng, J., Jarrar, M., Sure, Y., Studer, R., Meersman, R.: OntoWeb - a Semantic Web Community Portal. In Karagiannis, D., Reimer, U., (eds.): Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM02), LNAI 2569, Springer Verlag, (2002) 189 – 200

[SP94] Spaccapietra, S., Parent, C.: View Integration: A Step Forward in Solving Structural Conflicts. In: IEEE Transactions on Data and Knowledge Engineering 6(2), (1994)
[T96] de Troyer, O.: A Formalization of the Binary Object-Role Model based on Logic. In: Data & Knowledge Engineering 19, North-Holland Elsevier (1996) 1-37
[TM95] de Troyer, O., Meersman, R.: A Logic Framework for a Semantics of Object-Oriented Data Modelling. In: Papazoglou, M.P. (eds): Proceedings of 14th International Conference Object-Orientation and Entity-Relationship Modelling (OO-ER'95), Lecture Notes in Computer Science 1021, Springer (1995) 238-249
[TSC01] Tzitzikas, Y., Spyratos, N., Constantopoulos, P.: Mediators over Ontology-based Information Sources. In: Second International Conference on Web Information Systems Engineering, WISE'01, (2001)
[V82] Van Griethuysen, J.J., (Ed.): Concepts and Terminology for the Conceptual Schema and Information Base. International Standard for Standardization, Publication No. ISO/TC97/SC5- N695, (1982)
[VB82] Verheijen, G., van Bekkum, P.: NIAM, aN Information Analysis Method. In: Olle, T.W., Sol, H. and Verrijn-Stuart A. (eds.), IFIP Conference on Comparative Review of Information Systems Methodologies, North-Holland, (1982) 537-590
[W90] Wintraecken, J.J.V.R.: The NIAM Information Analysis Method: Theory and Practice. Kluwer, Deventer. (1990)
[WJ99] Welty, C., Jessica, J.: An Ontology for Subject. In J. Data and Knowledge Engineering. 31(2)155-181. Elsevier. (1999)
[WSW99] Wand, Y., Storey, V., Weber, R.: An Ontological Analysis of the relationship Construct in Conceptual Modelling. In: ACM Transactions on Database Systems, Vol. 24, No. 4, (1999) 494-528

Appendix A

A complete example of an ORM Schema with the associated ORM-ML file, ORM pseudo NL generated by the DogmaModeler tool.

ORM Schema



ORM-ML

```
<?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/ormml.xsd
xmlns:dc="http://purl.org/dc/elements/1.1/">
    <ORMMeta>
            <dc:title>ORM ML example</dc:title>
            <dc:creator> Mustafa Jarrar </dc:creator>
            <dc:description>A complete example of an ORM ML file</dc:description>
```

```
        <dc:contributor>Jan Demey</dc:contributor>          </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='LOT' Name='PaperTitle' />
       <Subtype Parent='Person' Child='Author'/>
       <Subtype Parent='Person' Child='Reviewer'/>
       <Predicate>
                 <Object_Role ID='2' Object='Committee' Role='Includes'/>
                 <Object_Role ID='3' Object='Person' Role='IsMemberOf'/> </Predicate>
       <Predicate>
                 <Object_Role ID='4' Object='Committee' Role='ChairedBy'/>
                 <Object_Role ID='5' Object='Person' Role='Chairs'/>    </Predicate>
       <Predicate>
                 <Object_Role ID='6' Object='Reviewer' Role='Reviewes'/>
                 <Object_Role ID='7' Object='Paper' Role='ReviewedBy'/> </Predicate>
       <Predicate>
                 <Object_Role ID='8' Object='Author' Role='Writes'/>
                 <Object_Role ID='9' Object='Paper' Role='WrittenBy'/>  </Predicate>
       <Predicate>
                 <Object_Role ID='10' Object='Author' Role='Presents'/>
                 <Object_Role ID='11' Object='Paper' Role='PresentedBy'/> </Predicate>
       <Predicate>
                      <Object_Role ID='12' Object='PaperTitle' Role='isOf'/>
                      <Object_Role ID='13' Object='Paper' Role='Has'/> </Predicate>
       <Constraint xsi:type='Mandatory'> <Object_Role>2</Object_Role></Constraint>
       <Constraint xsi:type='Mandatory'><Object_Role>4</Object_Role></Constraint>
       <Constraint xsi:type='Mandatory'><Object_Role>6</Object_Role></Constraint>
       <Constraint xsi:type='Mandatory'><Object_Role>8</Object_Role></Constraint>
       <Constraint xsi:type='Mandatory'><Object_Role>9</Object_Role></Constraint>
       <Constraint xsi:type='Mandatory'><Object_Role>13</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'><Object_Role>4</Object_Role></Constraint>
       <Constraint xsi:type='Subset'>
              <Parent><Object_Role>2</Object_Role><Object_Role>3</Object_Role> </Parent>
              <Child> <Object_Role>4</Object_Role> <Object_Role>5</Object_Role> </Child>
       </Constraint>
       <Constraint xsi:type='Uniqueness'>
              <Object_Role>2</Object_Role><Object_Role>3</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'>
       <Constraint xsi:type='Uniqueness'>
              <Object_Role>10</Object_Role><Object_Role>11</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'>
              <Object_Role>8</Object_Role><Object_Role>9</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'>
              <Object_Role>6</Object_Role><Object_Role>7</Object_Role></Constraint>
       <Constraint xsi:type='Exclusion'>
              <First><Object_Role>8</Object_Role><Object_Role>9</Object_Role></First>
              <Second><Object_Role>6</Object_Role><Object_Role>7</Object_Role></Second>
       </Constraint>
       <Constraint xsi:type='Uniqueness'>
              <Object_Role>12</Object_Role><Object_Role>8</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'><Object_Role>13</Object_Role></Constraint>
       <Constraint xsi:type='Uniqueness'><Object_Role>12</Object_Role></Constraint>
       <Constraint xsi:type='Subset'>
              <Parent>  <Object_Role>8</Object_Role>  <Object_Role>9</Object_Role> </Parent>
              <Child> <Object_Role>10</Object_Role> <Object_Role>11</Object_Role> </Child>
```

```
    </Constraint>
  </ORMBody>
</ORMSchema>
```

ORM Verbalization
(Pseudo NL sentences, generated by DogmaModeler)

●	Each Committee must ChairedBy at least one Person.
●	Each Committee must Includes at least one Person.
●	Each Reviewer must Reviewes at least one Paper.
●	Each Author must Writes at least one Paper.
●	Each Paper must WrittenBy at least one Author.
↔	Each Paper must Has at most one PaperTitle.
↔	Each PaperTitle must isOf at most one Paper.
↔	Each Committee must ChairedBy at most one Person.
⟷	It is disallowed that the same Committee Includes the same Person more then once, and it is disallowed that the same Person IsMemberOf the same Committee more then once.
⟷	It is disallowed that the same Author Presents the same Paper more then once, and it is disallowed that the same Paper PresentedBy the same Author more then once.
⟷	It is disallowed that the same Author Writes the same Paper more then once, and it is disallowed that the same Paper WrittenBy the same Author more then once.
⟷	It is disallowed that the same Reviewer Reviewes the same Paper more then once, and it is disallowed that the same Paper ReviewedBy the same Reviewer more then once.
↑	Each Person who Chairs a Committee must also IsMemberOf that Committee.
↑	Each Paper who WrittenBy a Author must also PresentedBy that Author.
⊗	Each Paper which is WrittenBy a Person must not ReviewedBy with that Person.
Ⓤ	Each (PaperTitle, Author) as a combination refers to at most one Paper.