# Model Ontological Commitments Using ORM$^+$ in T-Lex

Yan Tang and Damien Trog

Semantic Technology and Application Research Laboratory (STARLab),
Department of Computer Science, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussel, Belgium
{yan.tang, Damien.trog}@vub.ac.be

**Abstract.** When designing and developing ontology based applications, we semantically ground them by ontologically committing the application rules to their respective domain. These rules can be, for instance decision rules for a decision support system. For the DOGMA framework we have introduced ORM$^+$, a novel extension of ORM for modeling, visualizing and interchanging ontological commitments. In this paper, we illustrate our ongoing research on ORM$^+$ and T-Lex as its supporting tool. We demonstrate in the field of on-line customer management.

## 1 Introduction

An ontology is a semiotic representation of agreed conceptualization in a subject domain [1, 8]. As an ontology modeling framework and ontology engineering methodology, DOGMA (Developing Ontology-Grounded Methods and Applications, [14, 18]) was designed and inspired by the tried-and-tested principles from conceptual database modeling. In DOGMA, an ontology is represented in two layers using the *double-articulation* principle: the *lexon* layer and the *commitment* layer.

A *lexon* is modeled as a quintuple $<\gamma, t_1, r_1, r_2, t_2>$, where $t_1$ and $t_2$ are terms that represent two concepts in some language. $r_1$, $r_2$ are roles ($r_1$ corresponds to “role” and $r_2$ - “co-role”) referring to the relationships that the concepts share with respect to one another. $\gamma$ is a context identifier. It is assumed to point to a resource, usually a document in the general sense, where the terms $t_1$, $t_2$ are originally defined and disambiguated, and in which the roles $r_1$, $r_2$ become “meaningful”. For example, a lexon $<\gamma$, *order manager, accept, is accepted by, customer request*$>$ explains a fact that “order manager accepts customer request”.

An ontology is often designed for several tasks or applications. As Guarino mentioned, the value of an ontology is the reusability in the context of knowledge management [8]. In order to ensure the reusability of an ontology, the *commitment* layer in DOGMA is designed to separate the application layer and the lexon layer. It contains a set of ontological commitments, with which an application commits its local vocabulary and application semiotics to the meaning of the ontology vocabulary.

The commitments need to be expressed in a *commitment language* that can be easily interpreted by the domain experts and a machine. In [1, 18], the authors studied many advantages of using ORM (Object Role Modeling, [9]) as a commitment modeling manner. Furthermore, they argue that it is rather feasible to model and visualize the commitments using ORM, for ORM has an expressive capability in its graphical notations and verbalization possibilities. The ORM commitment models can be stored in XML files (ORM Markup Language, ORM-ML for short, [1]), with which agents can share the ontologies.

Suppose that a domain expert wants to constrain the lexon <*Customer Request, is Accepted By, accept, Order Manager*> with a uniqueness constraint like "*one* customer request is accepted by *at most one* order manager". Its ORM diagram is shown in **Fig. 1**.

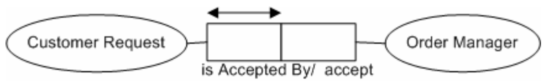

**Fig. 1.** Model ORM uniqueness constraint

The corresponding XML file is shown as below:

```
…
<Predicate id= "lexon-1">
<Object_Role ID= "lexon1_forward"
Object="CustomerRequest" Role="isAcceptedBy"/>
<Object_Role ID= "lexon1_backward"
Object="OrderManager" Role= "accept"/>
…
<Constraint xsi:type= "Uniqueness">
<Object_Role>lexon1_forward</Object_Role>
</Constraint>
…
```

ORM, ORM 2[1] and ORM-ML are the technology that is mainly used in DOGMA. However, it still lacks several operators and connectors while grounding the semantics for the application rules, e.g. the *sequences* and *dependences*. Moreover, ORM has certain limitations on using some specific operators, such as the *implication* operator. As the response, we have proposed ORM$^+$ (an extension to ORM) and ORM$^+$ ML (a hybrid language of ORM-ML and FOL Rule ML) in our early paper [21]. Seven new graphic notations, including *negation*, *conjunction* and *sequence* operators, were introduced. These notations are mainly used for modeling commitments used by ontology based decision support systems, such as Semantic Decision Tables [22].

Recently, we have been working on a modeling tool called T-Lex [23] for modeling ontological commitments using ORM$^+$. T-Lex is a tool for graphical ontology engineering that has a notation based on ORM.

By using T-Lex, the most significant constraints for ontologies (mainly based on ORM) are supported [24], which include most database consistency constraints.

[1] ORM 2 [11] is the second generation of ORM. Its graphical notations are improved based on industrial experiences.

Because of the tree representation, constraints are indexed instead of connected with a line to avoid cluttering.

In this paper, we focus on the ongoing researches on ORM$^+$ and discuss how to model, visualize and verbalize the ORM$^+$ models in T-Lex. The rest of the paper is organized as follows: section 2 is the related work. We compare our work to other ontology modeling tools, such as Protégé. Seven graphical notations are demonstrated in section 3. We conclude, open a discussion and discuss the future work in section 4.

## 2 Related Work

The current state of the art on ORM modeling is the NORMA tool from Neumont University [1]. It supports the new ORM2 notation and is able to map the schemas to the popular RDBMs.

Protégé [16] is an ontology development tool developed at Stanford University. For Protégé different visualization modules have been developed, with Jambalaya [19] as the most popular one. It uses a nested graph-based representation of hierarchical structures, together with nested interchangeable views.

Variations of the spring embedded algorithm [3] are also widely used in ontology engineering tools. Examples are the work of Mutton and Golbeck [15], the OIModeller plug-in for the KAON server [5], and the visualization user interface in OntoEdit [20]. The ontology is considered as a graph whose vertices represent concepts and whose edges represent relationships. Vertices are positioned randomly as their initial position. Each vertex is considered to cause a repulsive force on the others, while edges represent an attracting force. When minimum energy is reached, the visualization is complete. The advantage is that high-level structure can be detected. The disadvantage is that the number of iterations and the initial positions can create a new representation on each visualization.

Another approach is using cluster maps for visualizing populated, light-weight ontologies [4]. This visualizes the instances of a number of selected classes from a hierarchy, organized by the taxonomy.

Pretorius also visualized the lexon base by employing a fish eye view on an ordered visual representation [17]. The technique is well suited for an overview of very large lexon bases, but not for searching for particular terms.

An overview of other ontology tools can be found in several surveys [13, 6].

Unlike all the tools illustrated above, T-Lex [23] uses *NORM* tree to tackle the lexon base visualization problems in a *scalable* and *structured* manner. NORM is a recursive acronym of “NORM Ontology Representation Method”. It is a method to represent domain ontologies in an undirected rooted tree format. T-Lex is particularly suited for small to medium sized ontologies. This is partly supported by grouping lexons by context and by the assumption that the user already knows the starting point for searching. Just like the term NORM is a recursive definition, the NORM tree in general is recursive. It can be spanned infinitely. By using T-Lex, the users can have *more flexible* ontology views than other ontology modeling tools.

## 3 ORM$^+$ in T-Lex

In [21], we introduced the ORM$^+$ graphical notations of *negation*, *conjunction*, *implication*, *sequence*, *necessity* and *possibility*. Negation, conjunction and implication are borrowed from the basic operators[2] of propositional logics. Sequence is used in the commitments related to the time issue. The necessity and possibility operators are two basic ones in Modal logic. We illustrate their graphical notations in the following subsections.

### 3.1 Negation

It is possible to model a negation constraint using ORM. In ORM, one uses "closed-world" and specific "open-world" assumptions [9, pp. 61]. The "closed-world" assumption uses the *absence* of positive information (e.g. Customer request is accepted by Order manager) to imply the negative (e.g. Customer request is not accepted by Order manager). With an "open-world" approach, negative information is explicitly stored using negative predicates or status object types. I.e. "Customer request" has Acceptance status {'Accepted', 'Not Accepted'} and *each* "Customer request" has *at most one* Acceptance status (**Fig. 2**). Or, "Customer request" has two subtypes "Accepted customer request" and "Unaccepted customer request", which are mutually exclusive (**Fig. 3**).

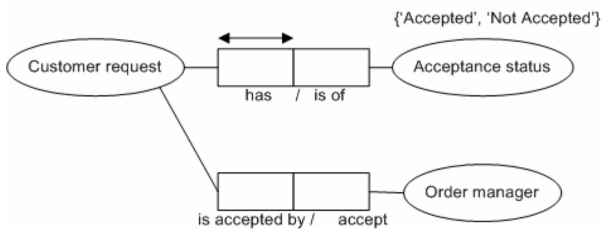

**Fig. 2.** Model negation in ORM – method 1

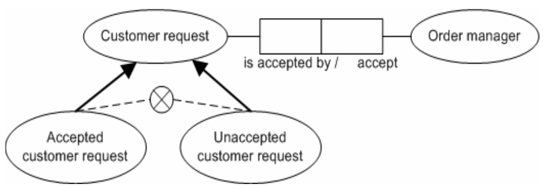

**Fig. 3.** Model negation in ORM – method 2

Although transferring the negation connective to the value constraint or the exclusive constraint doesn't lose any information of a negative proposition, it is still

[2] In propositional logic, there are five basic operators and connectors: *negation*, *conjunction*, *disjunction*, *implication* and *equivalence*. The disjunction can be modeled using the exclusive-or constraint in ORM. The *equivalence* can be modeled using the *equality* constraint in ORM. Therefore, the equivalence and disjunction are excluded in the paper.

not easy for a domain expert to know when the negative status is taken. For both positive and negative statuses of a type are modeled in the same schema. The domain expert has to make the analysis or reckon on extra information in order to know whether he uses the negative status or the positive one. To simplify the situation, we introduce the constraint of negation in ORM$^+$.

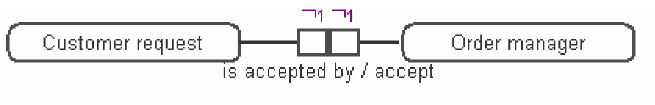

**Fig. 4.** Model ORM$^+$ negation in T-Lex

**Fig. 4** shows an ORM$^+$ diagram containing a notion of negation. A "¬"is marked above each applied role. The number (e.g. '1' in **Fig. 4**) after "¬" indicates the constraint group. Note that a negation constraint needs to be applied to both role and co-role. Because the lexon role and co-role often has the same meaning, e.g. 'is accepted by' and 'accept' in **Fig. 4**. It is strange for a role to have a negative situation while the co-role stays positive, and vice verse.

**Fig. 4** is verbalized as "a customer request is not accepted by an order manager; an order manager doesn't accept a customer request". When an application receives this commitment, it will give two resulting sets: one contains the customer requests that are not accepted by any order managers; the other contains the order managers that do not accept any customer requests.

The negation is often used with other connectors, such as the implication, which will be discussed later.

## 3.2 Conjunction

The *conjunction* binary operator is to construct a logical *AND* with the conjunction operator $\wedge$ . The conjunction operator is equivalently used as *set intersection* in the set theory. It is very useful to restrict the population of an object that plays a specific role. One writes such ontological commitments at the query level, e.g. list all the customers who are listed in a customer catalog *AND* whose state is normal".

ORM doesn't provide modeling techniques for the conjunction operator. In ORM$^+$, a " $\wedge$ " denotes the conjunction operator.

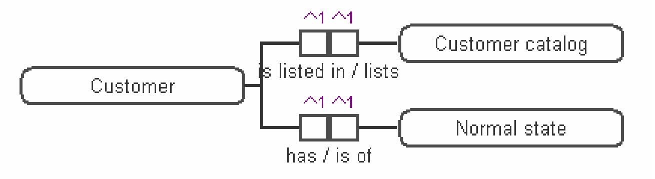

**Fig. 5.** Model ORM$^+$ conjunction in T-Lex

**Fig. 5** is an example containing a conjunction connector. We verbalize it as "a customer is listed in a customer catalog AND he has a normal state". Note that the conjunction connective is often applied to more than two lexons. It is not possible to be applied on only one lexon.

When a decision supporting system receives an ontological commitment shown in **Fig. 5**, it will give a record of ‘customer’ that is listed in a ‘customer catalog’ and he has a ‘normal state’. As the negation operator, the conjunction operator is also often used with the implication connective.

## 3.3 Implication

In ORM, the implication operator is modeled using a *subset* operator. **Fig. 6** shows an ontological commitment that the set of the members of ‘Driver’ who has ‘Driver’s license’ is the subset of the members of ‘Driver’ who has ‘License’. ORM uses a dotted arrow-tipped bar running from the subset role to the superset role for the subset constraint. It is comparable to the logical connective $\rightarrow$, which can be verbalized as “IF…, THEN” alike sentences. For example, **Fig. 6** is verbalized as “IF a driver has driver’s license, THEN he has license”.

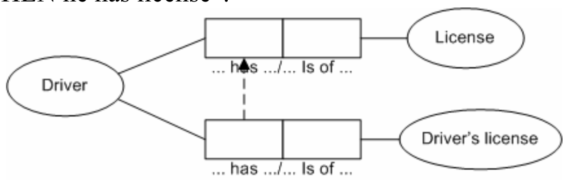

**Fig. 6.** Model implication using ORM subset constraint

However, there are two limitations while modeling the implications using ORM. One is that users can only model *monotonic* rules in ORM. In practice, we often encounter non-monotonic rules. E.g., the ontological commitments for the ontology based decision support systems often include *non-monotonic* rules, which are verbalized as “IF…THEN…ELSE” alike sentences and cannot be modeled using ORM.

The other limitation is, ORM implication constraint can only be applied to one object type (lexon term), such as “Driver” in **Fig. 6**. It is normal to model ORM implication in this way, because the ORM subset (implication) constraint is initially used for *set comparison* instead of *logical reasoning*. We need a more powerful notation to model event-driven decision commitments.

We use the symbol $\Rightarrow$ to designate the *implication* logical operator in the ORM$^+$ diagram (**Fig. 7**). The condition is indicated with $\Rightarrow$ followed by a group number and the action is illustrated by $\overset{\blacktriangle}{\Rightarrow}$ started with a group number. We verbalize **Fig. 7** as: *IF* a customer is *NOT* listed in a customer catalog (a customer catalog does *NOT* list a customer), *THEN* an order manager creates a new customer.

**Fig. 7** is a simple example that consists of a monotonic decision rule. We are able to model as well non-monotonic decision rules using ORM$^+$. **Fig. 8**, for example, is verbalized as “*IF* a customer is *NOT* listed in a customer catalog, *THEN* an order manager creates a new customer, *ELSE* the order manager approves the customer request.” Note that the group number is very important in T-Lex. It categorizes the modeling information. For example in **Fig. 7** and **Fig. 8**, the negation operator is grouped together with the action “an order manager creates a new customer”.

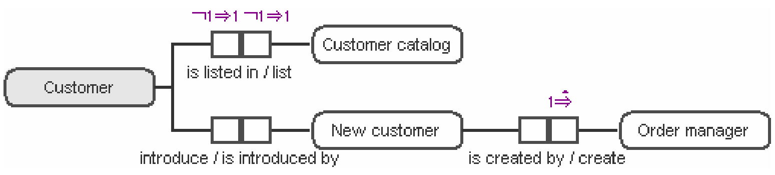

**Fig. 7.** Model ORM$^+$ implication in T-Lex

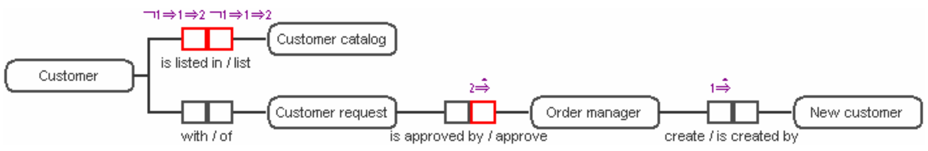

**Fig. 8.** Model non-monotonic decision rules in T-Lex

## 3.4 Sequence

The *sequence* operator in ORM$^+$ is application oriented. ORM doesn’t provide modeling methods for this kind of operators.

The definitions of the sequence constraint may differ between domains [21]. However, the core message is the same. That is, the issue of *order*, regardless of in the measure of time or space. We intend to use the sequence operator to reason on *orders*, e.g. the execution order of processes. Suppose that we have a rule “an order manager verifies a customer request *AFTER* the order manager receives the customer request”, which constraints the execution order of two processes.

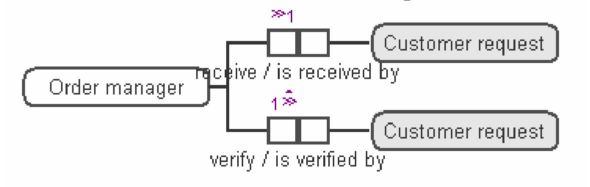

**Fig. 9.** Model ORM$^+$ sequence in T-Lex

The event that happens earlier is indicated with a ≫ followed by a group number. The one that happens later is marked with a $\hat{\gg}$ after the group number.

## 3.5 Other ORM$^+$ graphic notations in T-Lex

With regard to other ORM+ graphic notations, there are two important operators in the Modal Logic – the Necessity and Possibility. In the research field of object-role molding, Halpin categorizes rule modalities into *alethic* and *deontic* [10]. Alethic rules “impose necessities, which cannot be violated by any applications because of physical or logical law…”, e.g. there is only one sun in the sky. A deontic rule “imposes obligations, which may be violated, even though they ought not …”, e.g. no one is allowed to kill another person.

In ORM 2 [11], an alethic modality of *necessity* □ is used for positive verbalizations by default. For example, the fact ‘a customer is listed in a customer catalog’ may be explicitly verbalized as ‘a customer is NECESSARILY listed in a customer catalog’ by default. Halpin interprets it in terms of *possible world semantics*, which are introduced by Saul Kripke et al. in the 50’s [12]. A proposition is “*necessarily* true *if and only if* it is true in all possible worlds”. The *facts* and *static constraints* belong to a possible world, in which they *must* exist at some point in time. Therefore, the necessity operator may explicitly append on the fact ‘a customer is listed in a customer catalog’ by default.

The necessity and possibility operators are important in the decision support world, e.g. in e-court. Therefore, their graphic notations are explicitly introduced. The necessity constraint is indicated with □ above the applied roles (**Fig. 10**).

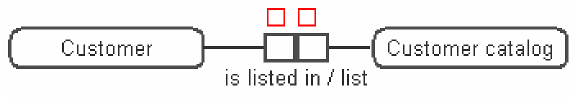

**Fig. 10.** Model ORM$^+$ necessity in T-Lex

## 4 Conclusion, Discussion and Future Work

In this paper, we have discussed our ongoing work on ORM$^+$, which is an extension to ORM, and T-Lex as its supporting tool. The work is based on our experience on modeling ontological commitments for decision support.

As Halpin discussed, there are, in principle, *infinitely* many kinds of constraints [9, pp. 16]. This principle is general. It is not only for ORM, but also for many other modeling languages. As new problems bring forward new needs, one can always extend a modeling tool.

However, the modeling means will be more and more complicated until we cannot handle it. Therefore, we need to be very careful when we introduce new notations. Before extending an existing modeling tool, the following questions need to be answered: 1) Can the new constraint be modeled using a combination of existing constraints? 2) Is this new constraint really useful? How many applications will use it? In our problem settings, the graphical notations introduced in this paper are mainly used in the context of ontology-based decision support systems. If our applications are relevant to decision support, the above two questions can be answered.

Currently, we get more and more requirements on reasoning on these notations. In the future, we’ll focus on the reasoning issue of ORM$^+$.

**Acknowledgement** We’re pleased to thank Gu Yan for programming. The research is partly supported by EC Prolix project.

## References

1. Curland, M. & Halpin, T. 2007, ‘Model Driven Development with NORMA’, Proc. 40th Int. Conf. on System Sciences (HICSS-40), 10 pages, CD-ROM, IEEE Computer Society.
2. Demey, J., Jarrar, M., & Meersman, R. (2002). *Markup Language for ORM Business Rules*, In proc. Of International Workshop on Rule Markup Languages for Business Rules on the Semantic Web (RuleML-ISWC02 workshop)
3. Eades, P.: A heuristic for graph drawing. Congressus Numerantium 42 (1984) pp. 149–160
4. Fluit, C., Sabou, M., van Harmelen, F.: Supporting user tasks through visualisation of light-weight ontologies. In: Handbook on Ontologies: Staab, S., Studer, R. (eds.). Int. Handbooks on Information Systems. Springer (2004) pp. 415–434
5. Gabel, T., Sure, Y., V¨olker, J.: Kaon – ontology management infrastructure. SEKT informal deliverable 3.1.1.a, Institute AIFB, University of Karlsruhe (2004)
6. Gomez-Perez, A., Corcho, O., Fernandez-Lopez, M.: Ontological Engineering. Springer-Verlag New York, LLC (2003)
7. Gruber, T. R. (1993), Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In Workshop on Formal Ontology, Padva, Italy. In book “Formal Ontology in Conceptual Analysis and Knowledge Representation”. Guarino, N., and Poli, R. (Eds.). Kluwer Academic Publishers
8. Guarino, N. (1997), Understanding, Building, and Using Ontologies: A commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga." International Journal of Human and Computer Studies 46: 293-310, http://citeseer.ist.psu.edu/guarino97understanding.html
9. Halpin, T.A. (2001), Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design, ISBN-13: 978-1-55860-672-2, ISBN-10: 1-55860-672-6, San Francisco, California, Morgan Kaufman Publishers
10. Halpin, T.A., *Business Rule Modality*, http://www.orm.net/pdf/RuleModality.pdf. Proc. Of Eleventh Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD’06), 2006.
11. Halpin, T. A., & Curland, M. (2006), Automated Verbalization for ORM 2. In proc. of On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, R. Meersman, Z. Tari, P. Herrero (Eds.), pp. 1181-1190, LNCS 4278, Springer, ISBN 3-540-48273-3, Montpellier, France, October 29 - November 3, 2006
12. Kripke, S., *Semantical Considerations on Modal Logic*, APF 16, pp. 83 – 94, 1963.
13. Lambrix, P., Edberg, A.: Evaluation of ontology merging tools in bioinformatics. In: Pacific Symposium on Biocomputing. (2003) pp. 589–600
14. Meersman, R. (1999), The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems. In Y. Zhang, M. Rusinkiewicz, and Y. Kambayashi, editors, The Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99), pages 1–14. Springer
15. Mutton, P., Golbeck, J.: Visualization of semantic metadata and ontologies. In: IV ’03: Proceedings of the Seventh Int. Conference on Information Visualization, Washington, DC, USA, IEEE Computer Society (2003) pp. 300
16. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Knowledge Systems Laboratory, Stanford University, Stanford, CA, 94305, USA (2001)
17. Pretorious, J.A.: Lexon visualization: Visualizing binary fact types in ontology bases. In: IV. (2004) pp. 58–6310. Lambrix, P., Edberg, A.: Evaluation of ontology merging tools in bioinformatics. In: Pacific Symposium on Biocomputing. (2003) pp. 589–600

18. Spyns P., Meersman R., & Jarrar M. (2002), Data modeling versus Ontology engineering. SIGMOD Record: Special Issue on Semantic Web and Data Management, 31(4):12 - 17, December 2002
19. Storey, M., Musen, M., Silva, J., Best, C., Ernst, N., Fergerson, R., Noy, N.: Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in protégé (2001)
20. Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D.: OntoEdit: Collaborative ontology development for the Semantic Web, LNCS 2342 (2002) pp. 221–235
21. Tang, Y., Spyns, P. & Meersman, R. (2007), *Towards Semantically Grounded Decision Rules Using ORM*$^+$, Proc. of International RuleML Symposium on Rule Interchange and Applications (RuleML'07), in, Adrian Paschke and Yevgen Biletskiy (eds.), Springer Verlag, LNCS 4824
22. Tang, Y. and Meersman, R., On constructing semantic decision tables, in proc. of 18th International Conference on Database and Expert Systems Applications (DEXA'2007), LNCS 4653, Springer-Verlag, Berlin Heidelberg, 3~7th September, 2007, in, R. Wagner, N. Revell, and G. Pernul (Eds.), Regensburg, Germany, p.34-44, 2007
23. Trog, D., Vereecken, J., Christiaens, S., De Leenheer, P., and Meersman, R., T-Lex: a Role-based Ontology Engineering Tool, In proc. of ORM 2006, Springer-Verlag, Volume 4278, Montpellier, France, 2006
24. Trog, D., Tang, Y., and Meersman, R., Towards Ontological Commitments with O-RIDL Markup Language, Proc. of International RuleML Symposium on Rule Interchange and Applications (RuleML'07), in, Adrian Paschke and Yevgen Biletskiy (eds.), Springer Verlag, LNCS 4824