Transforming Ontology Axioms to Information Processing Rules – An MDA Based Approach*

Diana Kalibatiene[1], Olegas Vasilecas[1], Giancarlo Guizzardi[2]

[1] Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius
diana@isl.vgtu.lt, olegas@isl.vgtu.lt
[2] Ontology and Conceptual Modeling Research Group (NEMO),
Federal University of Espirito Santo, Vitoria – ES – Brazil,
gguizzardi@inf.ufes.br

Abstract. In knowledge-based information systems development ontologies are used to represent domain knowledge and transform them to the components of an information system. However, the ontology-based development of application domain rules, constraining or directing different aspects of a domain, is not defined in a formal manner and is required to be developed. In this paper, the authors propose an MDA-based method for transforming ontology axioms to application domain rules. Such rules can be consecutively transformed to information processing rules and implemented by executable rules in a software component of an information system. The paper illustrates the proposed method by applying it to transform Protégé ontology axioms to OCL constraints, which are the part of a UML class diagram.

Keywords: ontology, axiom, information processing rule, MDA, transformation.

1 Introduction

In knowledge-based information systems (IS) development, a number of authors [1-3] suggest to represent knowledge by *domain ontology*, since the semantic content expressed by ontology can be transformed into IS artefacts, thereby reducing the costs of a conceptual modelling of the IS. In this context, researches are challenged to transform domain ontology to a conceptual data model. However, the ontology-based development of application domain rules is not defined in a formal manner.

In the IS development, a rule-based approach has achieved a lot of attention and obtained a strong motivation for its application [4-6]. A number of methods were proposed to develop and to implement rules: UML with OCL [7], Demuth et al method [8], the Ross method [9], SWRL [10], etc. The chosen IS development methodologies are compared according to the possibility of rules modelling in [11].

* The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)" Reg. No. B-07042

But none of the proposed languages or methods has been accepted as a standard technology yet, since they are not suitable for modelling all types of rules. Only few of them deal with reuse of knowledge acquired in the analysis of some particular application domain and automatic implementation of rules.

The authors of this paper consider that domain ontology should be used for the modelling and implementing of application domain rules. Since, first, both domain ontologies and conceptual data models are intended to capture knowledge about a certain subject domain. Secondly, from a syntactic perspective, in both artefacts, domain knowledge is expressed in a similar way: terms of concepts, their properties and relationships, and rules (in ontology – axioms). Third, as stated in [1], ontology axioms (and ontology as a whole) are typically expressed in a formal way. Therefore, they can in principle be transformed to application domain rules automatically.

The main idea of this paper is to investigate the automatic transforming of ontology axioms to application domain rules, which are implemented as information processing rules of an IS. Therefore, section 2 overviews the related works according to information processing rules and their implementation, section 3 presents a method of the transforming ontology axioms to information processing rules, section 4 presents an example of the proposed transformation, section 5 concludes the paper.

2 Information Processing Rules and Their Implementation

Definition of a rule depends on the level of abstraction, where this rule is used, or the development step of a system. System development using MDA framework [12], proposed by the OMG, implies creation of models of the following types: the computation independent model (CIM), the platform independent model (PIM), and the platform specific model (PSM). A *CIM* or an application domain model is a view of a system from the computation independent viewpoint. A vocabulary or application domain ontology is used to specify it. A *PIM* defines a set of components and services of a system. A *PSM* combines the specifications in the PIM with the specifications how that system uses a particular type of platform. It contains all information necessary to generate the final code. A *mapping* between CIM and PIM, PIM and PSM should exist to ensure the integral, complete and appropriate transformation of models. However, OMG specifies only transforming PIM to PSM. A number of IS development tools, like *PowerDesigner* [13] or *MagicDraw* [14], support only the transforming IS models to software system models (PIM to PSM).

The concept of an application domain rule can be analysed according to each of these levels of abstraction. At the *CIM level*, rules are statements that define or constrain some aspects of a particular application in a declarative manner. At the *PIM level*, rules are statements that define information processing rules using a rule-based language, like OCL [7]. At the *PSM* level, rules are statements in a language of a specific execution environment, like MS SQL Server 2008 [15] or ILOG JRules [16].

According to the implementation of application domain rules at IS, they can be classified to: (a) *Structural rules* (terms, definitions, facts, and integrity constraints), which can be implemented by a conceptual data model of an application domain, e.g., entity-relationship or UML class model; and (b) *Dynamic rules*, which can be

expressed by ECA rules and implemented, like SQL triggers or SQL views. A dynamic rule is: (a) a dynamic constraint, restricting transition from one state of the application domain to another; or (b) a derivation rule, calculating or logically concluding new information from existing information; or (c) a reaction rule, testing a condition and upon finding it true performing a defined action.

According to the observation in [1], [17-19], ontology defines the basic concepts, their definitions and their relationships, comprising the vocabulary of an application domain, and the axioms for constraining interpretation of concepts and expressing complex relationships between concepts. Consolidation and epistemological axioms are used to restrict concepts and their relationships in some manner [20]. Derivation axioms derive some new information from existing [20]. In ontology development tools, like Protégé [21], axioms are implemented using a particular language, like Protégé Axiom Language (PAL) [22] or Ontology Web Language (OWL) [23]. These axioms correspond to the consolidation and derivation axioms from [20].

Therefore: (i) consolidation axioms can be modelled by dynamic constraints and / or reaction rules, (ii) derivation axioms can be modelled by derivation rules.

3 Transforming Ontology Axioms to Information Processing Rules

In this section, we apply the MDA framework to demonstrate the transforming ontology axioms to application domain rules. The schema is presented in Fig. 1.

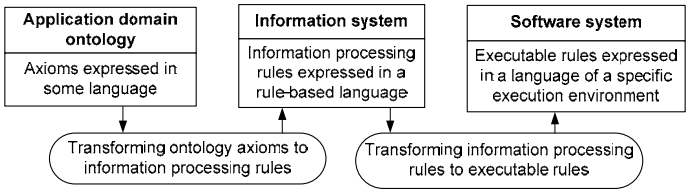

Fig. 1. The MDA-based transformation of ontology axioms to rules

In Fig. 1 an application domain is presented by an application domain ontology with ontology axioms. These axioms are transformed to information processing rules. The information processing rules are transformed to the corresponding executable rules.

Since the approach presented here relies on the definition of an explicit transformation between the meta-models of ontology axioms and information processing rules, the meta-models of these two paradigms are also defined. Fig. 2 presents general ontology meta-model, which can be adapted to a particular ontology meta-model, like OWL or Protégé. Derivation and consolidation axioms are aggregated of statements, which can be states or conditions. Condition can be composed of one or more conditions. A state is atomic in all cases.

Fig. 3 presents a general meta-model of a conceptual data model, which can be adapted to a meta-model of a particular conceptual data model, like entity-relationship [23]. The authors use [23] and [24] to define the meta-model of a conceptual data model. One ECA rule can constrain one or several entities. One entity can be

constrained by zero or several ECA rules. The ECA rule is an aggregation of one event, which activates rule, zero or one condition and one or two actions. If the ECA rule has no condition, it is assumed that the condition is satisfied in all cases. According to the rule properties defined in [5], the ECA rule should be atomic, e.g. it cannot be split into separate rules. However, the condition itself can be composed of one or more conditions. The condition and the action are aggregated of rule clauses. The rule clause is aggregated of facts (associated entities and / or attributes) of a conceptual data model.

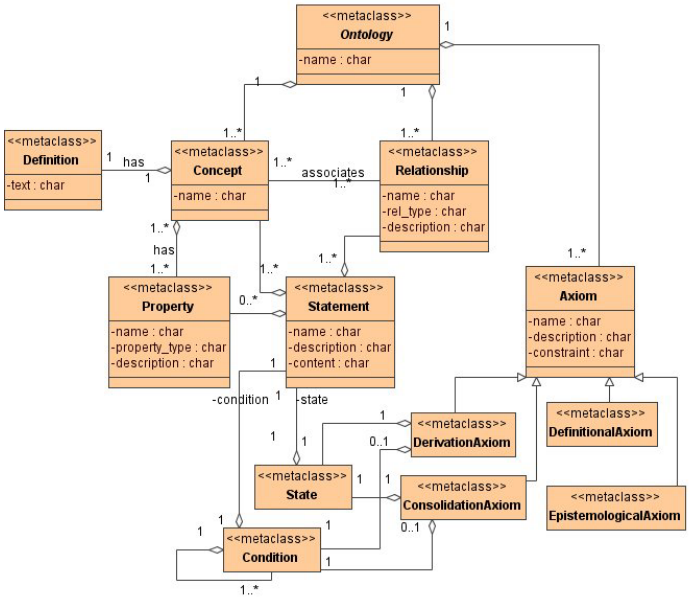

Fig. 2. The meta-model of an ontology

The action in the ECA rule is atomic in all cases. The ECA rule can have two actions: the first one is performed, when the condition is satisfied, and the second one is performed, when the condition is not satisfied.

The mapping of meta-classes from Fig. 2 and Fig. 3 is defined as follows: a concept maps to an entity, a definition maps to an entity description, a property maps to an attribute, a relationship maps to a relationship, a statement maps to a rule clause, an axiom maps to an application domain rule, a state maps to an action or a condition, and a condition maps to a condition. For the detailed mapping see the case study. For more details about the mapping of an ontology to a conceptual data model see [23] and [25]. For the detailed study of the transforming axioms to information processing rules the authors choose the PAL, which is used to define axioms in Protégé ontology, and Object Constraint Language (OCL), which is used to define information processing rules in UML class diagrams.

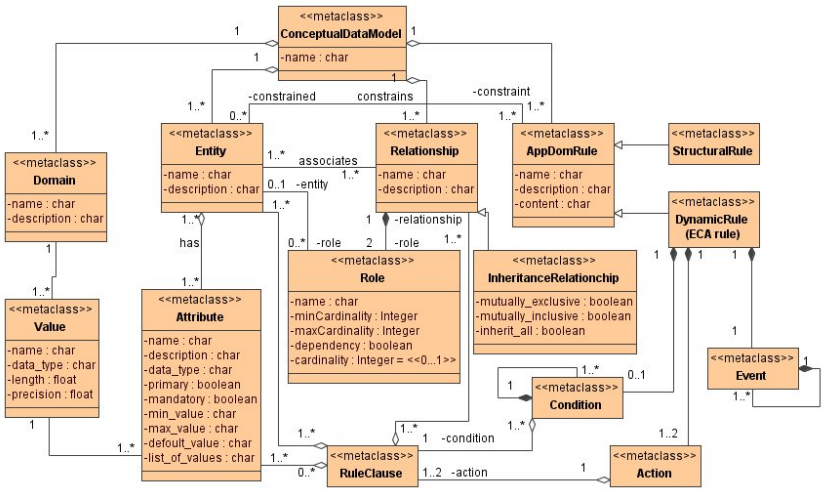

Fig. 3. The meta-model of a conceptual data model

4 Transforming PAL Constraints to OCL Constraints

In this section the authors present the transforming PAL constraints to OCL constraints. We have chosen Protégé because it allows for the open source software to be installed locally and provides all features and capabilities required for the present research. UML is the most popular for modelling of business and IS. OCL is proposed as a formal language to express rules, since UML diagrams are typically not refined enough to express those rules explicitly. However, OCL does not have any graphical notation and thus does not account for easily comprehensible language.

The authors define Protégé ontology (*ProtégéOntology*) as a set of $CLASS^{PO}$, $SLOT^{PO}$, REL^{PO}, VOC^{PO}, $VALUE^{PO}$, $DOMAIN^{PO}$, *PALCONST* elements:

$$ProtégéOntology := \{CLASS^{PO}, SLOT^{PO}, REL^{PO}, VOC^{PO}, VALUE^{PO}, DOMAIN^{PO}, PALCONST\}, \quad (1)$$

where $CLASS^{PO} = \{class^{PO}_i \mid i : N\}$ is a set of concepts in the *ProtégéOntology*, like a *customer* or an *order*. $SLOT^{PO} = \{slot^{PO}_i \mid i : N\}$ is a set of slots presenting properties of classes, like *customer_name*, and their relationships with other classes, like a slot *customer* in the class *order*. $REL^{PO} = \{is\text{-}a, inverse, has\}$ is a set of relationships, where *is-a* presents the hierarchical relationship between classes, *inverse* – the inverse relationship between slots, and *has* presents slots of a class. $VOC^{PO} = \{docum^{PO}_i \mid i : N\}$ is a set of class definitions. Each class has one particular definition. $VALUE^{PO} = \{value^{PO}_i \mid i : N\}$ is a set of values in an ontology. A value can be a *string*, a *symbol*,

a *class*, etc. $DOMAIN^{PO} = \{domain^{PO}_i \mid i : N\}$ is a set of domains in the *ProtégéOntology*. A *domain* ($domain^{PO}_i$) is a set of possible values ($value^{O}_i$) of slots ($slot^{PO}_i$). In Protégé ontology all axioms are simply implemented by PAL constraints. The authors define PAL constraints as follows:

$$PALCONST := \{palconst^{PO}_i \mid i : N\}, \quad (2)$$

where *PALCONST* is a set of axioms defined by PAL constraints. Each element of *PALCONST* is a PAL constraint ($palconst^{PO}_i$) defined be a PAL-name ($palname^{PO}_i$) or a label of a constraint, a PAL-documentation ($paldocum^{PO}_i$) or a description of a constraint, a PAL-range ($palrange^{PO}_i$), which is a set of local and global variables that appear in the statement of the PAL constraint, and a PAL-statement ($palstat^{PO}_i$). A PAL-range ($palrange^{PO}_i$) consists of a set of classes ($class^{PO}_i$). A PAL-statement ($palstat^{PO}_i$) consists of a set of statements ($statement^{PO}_i$), which are connected using logical connectives and / or quantifiers. A *statement* ($statement^{PO}_i$) is composed of a class ($class^{PO}_i$) or a slot ($slot^{PO}_i$) associated by a relationship (rel^{O}_i) with a class ($class^{PO}_i$) or a slot ($slot^{PO}_i$) or a value ($value^{PO}_i$).

Analysing and detailing a UML class diagram, which can be used to represent a particular conceptual data model, falls outside the principal scope of this paper. The authors define only an OCL constrain here. For more details about mapping of the Protégé ontology and a UML class diagram can be seen in [26]. According to [7], the following components of OCL constraints are defined:

$$OCLCONST := \{oclconst_i \mid i : N\}, \quad (3)$$

where *OCLCONST* is a set of OCL constraints in a UML class diagram. Each element of *OCLCONST* is an OCL constraint ($oclconst_i$) defined be a context ($context^{OCL}_i$), a type ($statetype^{OCL}_i$) and an OCL statement ($statement^{OCL}_i$).

The context can be a particular class, an attribute or a method of a UML class diagram. Statement types can be stereotypes (invariant, precondition or post-condition), an initial value and derived value. A *statement* ($statement^{OCL}_i$) is composed of a class ($class^{UML}_i$), an attribute ($attribute^{UML}_i$) or a method ($method^{UML}_i$), which is associated by a mathematical operator with a class ($class^{UML}_i$), an attribute ($attribute^{UML}_i$), a method ($attribute^{UML}_i$) or a value ($value^{UML}_i$). An example of a statement is *self.numberOfEmployees > 50*, where *numberOfEmployees* is an attribute and 50 is a possible value of this attribute. The reserved word *self* is used to refer to the contextual instance. For instance, if the context is the *Company* class, then *self* refers to an instance of the *Company* class.

The transformation of a PAL constraint to the corresponding OCL constraint follows: (a) a PAL-range ($palrange^{PO}_i$) is transformed to the context of an OCL constraint ($context^{OCL}_i$); (b) a PAL-statement ($palstat^{PO}_i$) is transformed to an OCL statement ($statement^{OCL}_i$); (c) a PAL-name ($palname^{PO}_i$) is transformed to the name of an OCL constraint; (d) a PAL-documentation ($paldocum^{PO}_i$) may be transformed to comments of an OCL constraint, which are denoted by two dashes (--). The analysis of PAL shows that all PAL-statements correspond to OCL invariants.

4.1 An Example of Transforming PAL to OCL

A prototype of the *PAL OCL transformation* plug-in was developed to carry out the experiment of automatic transformation of PAL constraints to OCL constraints. In this prototype it is necessary to specify a file, where OCL constraints will be stored and all PAL constrains will be automatically transformed to OCL constraints.

An example of transforming a PAL constraint from the *Newspaper* ontology [21], restricting that *the Employee end date should be after the start date*, to the corresponding OCL constraint follows:

- A PAL constraint:

```
(%3APAL-RANGE "(defrange ?Employee :FRAME Employee)\n")
(%3APAL-STATEMENT "(forall ?Employee \n (< (
'start_date' ?Employee) ('end_date' ?Employee)))\n"))
```

- An OCL constraint:

```
context Employee inv start_date_before_end_date:
self.end_date > self.start_date
```

5 Conclusions

The analysis of the related works in the field of knowledge-based information systems development using the domain ontology shows that rules used in application domain are presented in the ontology by axioms.

In this article, we contribute with an MDA-inspired method for transforming ontology axioms to information processing rules. In order to illustrate the presented method of transforming ontology axioms to information processing rules, we show the transforming of PAL constraints to OCL constraints.

The case study as well as the implemented prototype shows that the suggested method can be implemented and used for the automation of ontology axioms transformation to the information processing rules of an information system.

References

1. Guarino, N.: Formal Ontology and Information Systems. In: Proc. of FOIS'98, pp. 3-15, IOS Press (1998)
2. Jarrar, M., Demey, J., Meersman, R.: On Using Conceptual Data Modeling for Ontology Engineering. In: Spaccapietra, S. et al (eds.) Journal on Data Semantics. LNCS, vol. 2800, pp. 185-207. Springer (2003)
3. Wand, Y., Storey, V.C., Weber, R.: An ontological analysis of the relationship construct in conceptual modeling. ACM TODS, 24(4), 494–528 (1999)
4. Morgan, T.: Business Rules and Information Systems: Aligning IT with Business Goals, Addison-Wesley (2002)
5. von Halle, B.: Business Rules Applied: Building Better Systems Using the Business Rules Approach. John Wiley & Sons (2002)

6. Ross, R. G.: Principles of the Business Rule Approach. Addison Wesley (2003)
7. OMG: UML 2.0 OCL Specification. OMG Final Adopted Specification (2003).
8. Demuth, B., Hussmann, H., Loecher, S.: OCL as a Specification Language for Business Rules in Database Applications. In: Gogolla, M., Kobryn, C. (eds.) Proc. of UML 2001. LNCS, vol. 2185, pp. 104-117. Springer-Verlag (2001)
9. Ross, R. G.: The Business Rule Book. Classifying, Defining and Modeling Rules. Business Rules Solutions. LLC Houston (1997)
10. The Rule Markup Initiative. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, version 0.6. (2004)
11. Herbst H., et al.: The Specification of Business Rules: A Comparison of Selected Methodologies. In: Verrijn-Stuart, A.A., Olle, T. W. (eds.) Methods and Associated Tools for the Information System Life Cycle, pp. 29-46. Elsevier (1994)
12. Miller, J., Mukerji, J. (eds.): MDA Guide Version 1.0.1. OMG (2003)
13. PowerDesigner. Sybase, Inc. (2009) (February, 2009): http://www.sybase.com/products/modelingdevelopment/powerdesigner
14. MagicDraw. No Magic, Inc. (2009) (February, 2009): http://www.magicdraw.com/
15. MSSQL. Microsoft SQL Server 2008. (2008) (September, 2008): http://www.microsoft.com/sqlserver/2008/en/us/overview.aspx
16. ILOG: ILOG JRules. BRMS Resource Center. (2008) (September, 2008): http://blogs.ilog.com/brmsdocs/2008/06/15/ilog-jrules-6-for-architects-and-developers-2/.
17. Bozsak, E., et al.: KAON - towards a large scale semantic web. In: Bauknecht, K., et al (eds.) Proc. of E-Commerce and Web Technologies 2002. LNCS, vol. 2455, pp. 304–313. Springer (2002)
18. Sugumaran, V., Storey, V. C.: The Role of Domain Ontologies in Database Design: An Ontology Management and Conceptual Modeling Environment. ACM TODS, 31(3), 1064-1094, ACM Press (2006)
19. Hu, Z., Kruse, E., Draws, L.: Intelligent Binding in the Engineering of Automation Systems Using Ontology and Web Services. IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 33(3), 403-412 (2003)
20. Guizzardi, G., Falbo, R. A., Pereira Filho, J. G.: Using Objects and Patterns to Implement Domain Ontologies. Special Issue on Software Engineering, 8(1) (2002) (September, 2008): http://www.scielo.br/scielo.php?pid=S0104-65002002000100005&script=sci_arttext&tlng=en
21. Protégé. Stanford Center for Biomedical Informatics Research (2009) (April, 2009): http://protege.stanford.edu
22. Grosso, W. Quick Guide to the Protégé Axiom Language and Toolset (PAL). Stanford University (2005) (March, 2009): http://protege.stanford.edu/plugins/paltabs/pal-quickguide/
23. OMG: OntologyDefinition Metamodel. (2005)
24. AIS: PowerDesigner Dictionary Meta-Model. Applied Information Science International (1996) (September, 2008): http://www.aisintl.com/case/products/PowerDesigner/sdp/meta_model.html
25. Vasilecas, O., Bugaite, D.: Ontology-based Information Systems Development: the Problem of Automation of Information Processing Rules. In: Neuhold, E., Yakhno, T. (eds.) Proc. of ADVIS'2006. LNCS, vol. 4243, pp. 187-196. Springer (2006)
26. Knublauch, H.: UMLBackend. Stanford University (2007) (March, 2009): http://protege.cim3.net/cgi-bin/wiki.pl?UMLBackend