ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol. 36, No. 1 A

A GRAPH ORIENTED MODEL FOR ONTOLOGY TRANSFORMATION INTO CONCEPTUAL DATA MODEL

Justas Trinkunas, Olegas Vasilecas

Information Systems Research Laboratory, Vilnius Gediminas Technical University
Saulėtekio al. 11, LT-2040 Vilnius, Lithuania

Abstract. The paper analyses graph oriented method for ontology transformation into conceptual data model. A number of methods were proposed to develop conceptual data models, but only few deals with knowledge reuse. In this paper we present an approach for knowledge represented by ontology automatic transformation into conceptual data model. The graph transformation language is presented and adapted for formal transformation of ontology into conceptual model. Details and examples of proposed ontology transformation into conceptual data model are presented also.

1. Introduction

Conceptual data models of information systems are used to capture the meaning of an application domain as perceived by someone. The important requirement for developing conceptual data models is to reduce efforts, costs and time. This requirement can be implemented by the explicit use of enterprise knowledge for automatic or semiautomatic generation of conceptual data model. A number of methods were proposed to develop conceptual data models, but only few deals with knowledge reuse. In this paper we present an approach for knowledge represented by ontology automatic transformation into conceptual data model, which can be transformed into executable specification. The metamodels of ontology and conceptual data model are analysed and the method of automatic transformation is proposed. The developed prototype OntEr, which realises proposed method, is described in the case study section.

Why do we need ontology for conceptual data modelling? We can use ontology for conceptual data modelling at least for three different purposes. Firstly, ontology is a source of the knowledge and unexperienced designer can use ontology to get initial domain knowledge. Secondly, some parts of ontology can be used for conceptual data model development. For example we can adapt several concepts from the ontology to our needs and transform them into conceptual data model. And finally, all ontology after adaptation can be used for the conceptual data model.

The advantage of using ontology for conceptual data modelling is the reusability of domain knowledge. As a result of it the conceptual data model will be made faster, easier and with fewer errors than creating conceptual data model in usual way.

In earlier works we already demonstrated the benefits knowledge reuse for conceptual modelling [1]. Also we made ontology representation language analysis and conceptual modelling language analysis [2]. According this analysis we decided to use OWL DL as an ontology representation language and ER for data modelling. Consequently Protégé 3.3 tool was chosen for ontology creation and Sybase Power Designer 12.0 tool was chosen was data modelling. In this paper we continue our work.

The work is organised as follows. Firstly we give a theoretical background and discus related works, later we present transformation of ontology into conceptual data model based on graph, and finally we describe case study.

2. Theoretical background

2.1. Ontology

Many authors in their works propose different ontology definitions. We accept in [3] proposed ontology definition. Ontology defines the common terms and concepts (meaning) used to describe and represent an area of knowledge. An ontology can range in expressivity from a taxonomy (knowledge with minimal hierarchy or a parent/child structure), to a thesaurus (words and synonyms), to a conceptual model (with more complex knowledge), to a logical theory (with very rich, complex, consistent and meaningful knowledge).

The structure of ontology can be defined mathematically. However different authors provide different

definitions which can vary from 3-tuple definition where ontology is defined as O=(Concepts, Relations, Axioms) to 10-tuple definition, where ontology is defined in more details [4].

Mathematically we define ontology using graph formalism. In work [5] authors define an ontology O as a directed labelled graph G = (N, E) where N is a finite set of labelled nodes and E is a finite set of labelled edges. An edge e is written as a triplet (n_1, α, n_2) where n_1 and n_2 are members of N and α is the label of the edge. The structure of graph consisting from [6]:

1. a set of concepts (vertices in a graph);
2. a set of relationships connecting concepts (directed edges in a graph);
3. a set of instances assigned to a particular concepts (data records assigned to concepts or relation).

2.2. Ontology languages

In this chapter we briefly review ontology languages.

An RDF graph is a set of RDF triples [3]. The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph. A subgraph of an RDF graph is a subset of the triples in the graph. A triple is identified with the singleton set containing it, so that each triple in a graph is considered to be a subgraph. A proper subgraph is a proper subset of the triples in the graph. A ground RDF graph is one with no blank nodes.

A name is a URI reference or a literal. These are the expressions that need to be assigned a meaning by an interpretation. A set of names is referred to as a vocabulary. The vocabulary of a graph is the set of names, which occur as the subject, predicate, or object of any triple in the graph. The assertion of an RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. The assertion of an RDF graph amounts to asserting all the triples in it, so the meaning of an RDF graph is the conjunction (logical AND) of the statements corresponding to all the triples it contains.

OWL graph is an RDF graph. Not all RDF graphs are valid OWL graphs, however. The OWLGraph class specifies the subset of RDF graphs that are valid OWL graphs.

An OWL ontology contains a sequence of annotations, axioms, and facts. Annotations on OWL ontologies can be used to record authorship and other information associated with ontology, including imports references to other ontologies. The main content of OWLOntology is carried in its axioms and facts, which provide information about classes, properties, and individuals in the ontology. Names of ontologies are used in the abstract syntax to carry the meaning associated with publishing ontology on the Web. The intent is that the name of ontology in the abstract syntax is the URI where it can be found, although this is not part of the formal meaning of OWL. Imports annotations, in effect, are directives to retrieve a Web document and treat it as OWL ontology.

[12] OWL ontologies may be categorised into three species or sub-languages: OWL-Lite, OWL-DL and OWL-Full. A defining feature of each sub-language is its expressiveness. OWL-Lite is the least expressive sub-language. OWL-Full is the most expressive sub-language. The expressiveness of OWL-DL falls between that of OWL-Lite and OWL-Full. OWL-DL may be considered as an extension of OWL-Lite and OWL-Full an extension of OWL-DL.

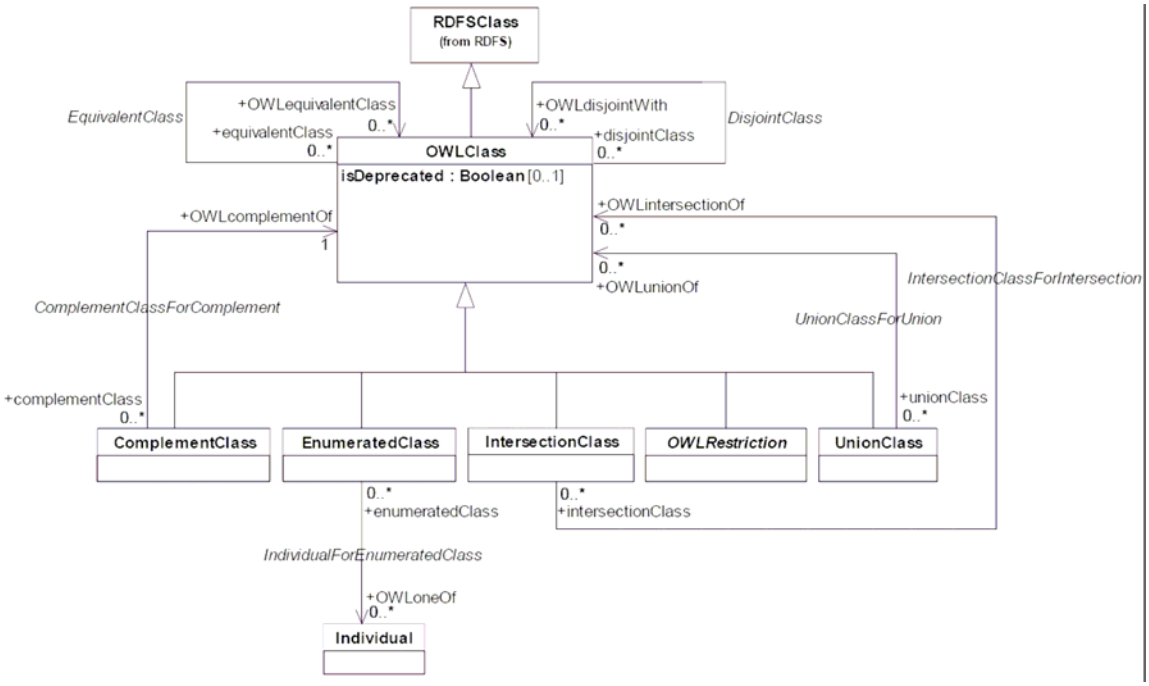

Figure 1. The OWL Class Descriptions Diagram [3]

In the table below we list most important elements of the ontology and define which element of graph represents it. The ontology elements are depicted from the OWL metamodel which is described in [3]. In Figure 1 we provide the part of the OWL metamodel.

Table 1. OWL elements

Element name	Type	Graph element
OWLOntology	Class	Node
OWLClass	Class	Node
ComplementClass	Class	Node
EnumeratedClass	Class	Node
DisjointClass	Class	Node
IntersectionClass	Class	Node
EquivalentClass	Class	Node
RestrictionClass	Class	Node
UnionClass	Class	Node
Property	Property	Node
OWLAnnotationProperty	Property	Node
OWLOntologyProperty	Property	Node
FunctionalProperty	Property	Node
OWLDatatypeProperty	Property	Node
OWLObjectProperty	Property	Vertex
InverseFunctionalProperty	Property	Node
SymmetricProperty	Property	Node
TransitiveProperty	Property	Node
OWLRestriction	Restriction	Node
HasValueRestriction	Restriction	Node
AllValuesFromRestriction	Restriction	Node
SomeValuesFromRestriction	Restriction	Node
CardinalityRestriction	Restriction	Node
MaxCardinalityRestriction	Restriction	Node
MinCardinalityRestriction	Restriction	Node
OWLDataRange	DataType	Node

2.3. Conceptual data model

Conceptual data model to model the overall logical structure of a database, independent from any software or data storage structure considerations [16].

A conceptual data model represents the overall structure of an information system. It describes the conceptual relationships of different types of information rather than their physical structures. A conceptual data model is independent of a particular database management system (DBMS).

For detail conceptual data model analysis we choose entity-relationship (ER) language.

An ER diagram is a graphical modelling notation that illustrates the interrelationships between entities in a domain. ER diagrams often use symbols to represent three different types of information.

Basic components of the ER language are [7]:

Entities. An entity is a phenomenon that can be distinctly identified. Entities can be classified into entity classes.

Relationships. A relationship is an association among entities. Relationships can be classified into relationship classes.

Attributes and data values. A value is used to give value to a property of an entity or relationship. Values are grouped into value classes by their types. An attribute is a function, which maps from an entity class or relationship class to a value class; thus the property of an entity or a relationship can be expressed by an attribute-value pair.

Additionally, we include domains and data types elements. And finally, ER model can be defined as quintuple:

$$ER = (E, A, D, R, DT), \quad (1)$$

where E is a set of entities, A – set of attributes, D – set of domains, R – set of relationships, DT – set of data types.

ER model can be represented as a graph. We define ER model using graph formalism. ER model is a directed labelled graph G_{ER} = (N, E) where N is a finite set of labelled nodes and E is a finite set of labelled edges. An edge e is written as a triplet (n_1, α, n_2) where n_1 and n_2 are members of N and α is the label of the edge.

In the table below we list most important elements of the ER model according to ER metamodel [8].

Table 2. ER elements

Element name	Type	Graph element
Model	Model	Node
Entity	Entities	Node
Attribute	Attributes	Node
AlternateKey	Attributes	Node
ForeignKey	Attributes	Node
InversionEntry	Attributes	Node
Key	Attributes	Node
PrimaryKey	Attributes	Node
Domain	Domain	Node
AtomicDomain	Domain	Node
DomainConstraint	Domain	Node
ListDomain	Domain	Node
UnionDomain	Domain	Node
EntityConstraint	Constraint	Node
Relationship	Relations	Vetex
Inheritance (Generalization)	Relations	Vetex
Association	Relations	Vetex
Association Link	Relations	Vetex
Link/Extended Dependency	Relations	Vetex

2.4. Metamodel based transformations

In the works [10, 18] authors describe metamodel based transformations. Authors argue that metamodel based transformations permit descriptions of

mappings between models created using different concepts from possibly overlapping domains and the transformation process facilitates the reuse of models specified in one domain-specific modelling language in another context: another domain-specific modelling language. Without the ability to perform model transformations, every existing model must be developed and understood separately, and/or has to be converted manually between the various modelling formalisms. This often requires as much effort as recreating the models from scratch, in another modelling language. However, when automatic model transformations are used, the mapping between the different concepts has to be developed only once for a pair of meta-models, not for each model instance.

Metamodel of ontology and conceptual data model should be based on meta-meta-model, defining all necessary constructs.

Model Driven Architecture (MDA) [11] defines three viewpoints (levels of abstraction) from which some system can be seen. From a chosen viewpoint, a representation of a given system (viewpoint model) can be defined. These models are (each corresponding to the viewpoint with the same name): Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). MDA is based on the four-layer metamodeling architecture, and several OMG's complementary standards. Layers are: meta-metamodel (M3) layer, metamodel (M2) layer, model (M1) layer and instance (M0) layer.

The mapping from OWL to ER was described in [17] document. However this document was just a proposal and in final version of the document [3] OMG group did not left the OWL mapping to ER. However, this mapping is incomplete and it is not clear which elements from the OWL ontology are not transformed into ER model. As a result of it some information from OWL ontology can not be used in ER model.

2.4. The Graph Transformation Language

We have adopted graph transformation language used in [5, 9] works. The language consists of the five basic operations: node addition, edge addition, node deletion, edge deletion, and abstraction. Currently we need only two operations (node addition and edge addition).

Node Addition. Given the graph G, a node N and its adjacent edges $\{(N, \alpha_i, m_j)\}$ to add, the node addition results in a graph G'=(M', E') where M'=M$\cup$ N and E' = E $\cup$ $\{(N, \alpha_i, m_j)\}$.

Edge Addition. Given a graph G and a set of edges SE = $\{(m_i, \alpha_j, m_k)\}$ to add the edge addition operation EA [G, SE] results in a graph G'=(M,E') where E'=E$\cup$SE.

The node addition operation can be used to introduce new objects into ER model from the ontology. The edge addition operation is needed to build relationships between the ER objects.

3. Proposed approach

In this chapter we describe how ontology can be transformed into conceptual data model using graph formalism based on metamodelling.

Many authors work in the field of transformation of ontology [13, 14, 15]. However proposed transformation methods are informal and used for different purposes.

We adapted the schema of models transformation from [10] for the ontology transformation into conceptual data model.

Transformation from ontology G_O into conceptual data model G_{ER} can be presented as:

$$G_O \rightarrow G_{ER}, \quad (2)$$

where G_O is an ontology represented as graph which is based on OWL metamodel, G_{ER} is ER model represented as graph based on ER metamodel, $\rightarrow$ is transformation of the elements of graph.

Ontology G_O transformation into ER model G_{ER} consists of set of elementary transformations which can be presented as:

$$G_O(\text{OWLElement}) \rightarrow G_{ER}(\text{ERElement}), \quad (3)$$

where OWLElement is an element from OWL metamodel, ERElement is an element from ER metamodel and $\rightarrow$ is simple graph transformation which consists of Node Addition and Edge Addition operations defined above.

Elementary transformations are defined as follows:

1. OWLOntology element is transformed into Model element. A Model in ER consists of the various modelling elements (entities, relationships and domains) that can be used to describe and represent things of interest to an enterprise. Entities represent things within a subject area or across areas, and relationships represent the associations between them. Domains represent logical data types. OWL ontology contains a sequence of annotations, axioms, and facts. Annotations on OWL ontologies can be used to record authorship and other information associated with ontology, including imports references to other ontologies. The main content of OWLOntology is carried in its axioms and facts, which provide information about classes, properties, and individuals in the ontology. The transformation is as follows:

   $$G_O(\text{OWLOntology}) \rightarrow G_{ER}(\text{Model}). \quad (4)$$

2. OWLClass element is transformed into Entity element. All classes in OWL are identified by uri. An entity in ER is identified by name. OWLClass and Entity are nodes in the graph:

   $$G_O(\text{OWLClass}) \rightarrow G_{ER}(\text{Entity}). \quad (5)$$

3. OWLDataTypeProperty element is transformed into Attribute element. An attribute in ER represents a common characteristic of some entity instances. OWL data type properties are used to link individuals to data values. A data type property is defined as an instance of the built-in OWL class owl:DatatypeProperty. OWLDataTypeProperty and Attribute are nodes in the graph:

$G_O(\text{OWLDatatypeProperty}) \rightarrow G_{ER}(\text{Attribute})$. (6)

4. OWLObjectProperty element is transformed into Relatinoship element. An object property in OWL relates an individual to other individuals. An object property is defined as an instance of the builtin OWL class owl:ObjectProperty. Relationships represent connections, links, or associations between two or more entities. OWLObjectProperty and Relatinoship are vertexes in the graph:

$G_O(\text{OWLObjectProperty}) \rightarrow G_{ER}(\text{Relationship})$. (7)

5. OWLDataRange element is transformed into AtomicDomain element. A data range in OWL is either a literal type or an enumeration of literals. Atomic domains in ER are those having values, which are regarded as being indivisible. Atomic domains in ER restrict, in a manner described by their constraints, the value space of the datatype identified via the baseType attribute. OWLDataRange and AtomicDomain are nodes in the graph:

$G_O(\text{OWLDataRange}) \rightarrow G_{ER}(\text{AtomicDomain})$. (8).

The other elements of the ontology can not be transformed straight forward into the ER model but can be used for different purposes which are not discussed in this paper.

4. Case study

We have chosen Protégé 3.3 for the development of the ontology. A free version of the software provides all features and capabilities required for the present research. Protégé 3.3 can be downloaded from the site http://protege.stanford.edu. With Protégé 3.3 tool we built Salary ontology showed in Figure 3. In this ontology we describe main concepts and relationships which describe domain area of salaries.

Here we briefly describe proposed method of building conceptual model from the OWL DL ontology. The method consists of four main steps:

1. The first step is knowledge acquisition from the word, documents, people, conceptual data models, ontologies and other sources. All extracted knowledge is written in the domain ontology in OWL DL format. We use Protégé 3.3 tool, however other tool could be chosen for ontology development. Domain ontology is created manually. But we are expanding our work and in near future we will propose semiautomatic method for ontology development from existing conceptual models, ontologies and other sources.
2. The second step is the transformation of domain ontology into conceptual data model with our plug-in OntER. Created conceptual data model can be opened with Sybase Power Designer 12.0 tool and adapted for your needs.
3. The third step is verification of conceptual data model. If we made changes with Power Designer 12.0 we need to verify if conceptual data model is valid. The conceptual data model is compared with the domain ontology. This step is not implemented yet, however. Good thing, this step is not obligatory, too.
4. The last step is the generation of physical data model with Power Designer 12.0 for a particular DBMS. This feature is already implemented in the original version of Power Designer 12.0.

[16] Through a simple generation procedure, you can transfer the solid design framework of the conceptual data model to the physical data model. The physical data model adapts your design to the specifics of a DBMS and puts you well on the way to complete physical implementation.

All steps are showed in Figure 3.

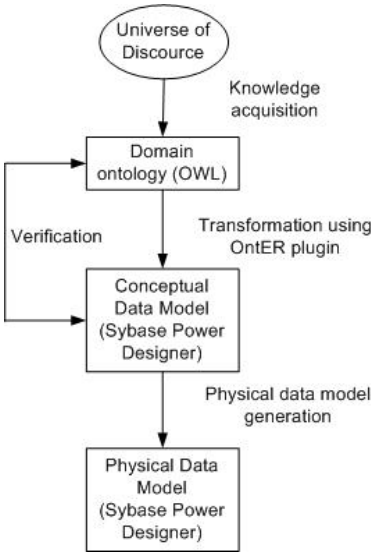

Figure 3. Main processes

4.1. Payroll system ontology

The ontology was created with Protégé 3.3 tool. The knowledge was extracted from Lithuanian Work Codex and other documents.

The main concepts described in Salary ontology are: wage, taxes, employment contract. The employment contract type can be terminable, not terminable or other. Terminable contracts are those contracts, which have validity date. Terminable contracts can be seasonal contract, temporary contract, and terminal contract. Other concepts are showed in the Figure 4.

Figure 4. Payroll system ontology

4.2. Developed plug-in OntER

OntER is the plug-in written in Java. Conceptual data model is build from the set of patterns which are filed with the needed data. Below we give the example of the attribute pattern. To build conceptual data model using patterns is very convenient. In case the conceptual data model of Power Designer will be changed in the future, we can change the patterns and update the plug-in.

<c:Attributes>
<o:EntityAttribute Id=Value>
<a:ObjectID>Value</a:ObjectID>
<a:CreationDate>Value</a:CreationDate>
<a:Creator>Value</a:Creator>
<a:ModificationDate> Value
</a:ModificationDate>
<a:Modifier>Value</a:Modifier>
<c:DataItem>
<o:DataItem Ref=Value/>
</c:DataItem>
</o:EntityAttribute>
</c:Attributes>

5. Conclusions and future work

We presented graph-oriented model for ontology transformation into conceptual data model based on metamodels. The advantage of proposed method is formally defined transformation of ontology transformation into conceptual model. However, it is not possible to transform all elements from OWL DL ontology into conceptual data model straight forward because OWL DL is semantically richer when data conceptual model.

The next our future step is to evaluate the effectiveness of our proposed method for creating conceptual data models from the ontology.

References

[1] **O. Vasilecas, D. Bugaite, J. Trinkunas.** On Approach for Enterprise Ontology Transformation into Conceptual Model. *B. Rachev, A. Smirkarov (eds.), Proc. of the International Conference on Computer Systems and Technologies "CompSysTech'06", Varna, Bulgaria*, 2006, IIIA.23-1- IIIA.23-6.

[2] **J. Trinkūnas, O. Vasilecas.** Ontologijos vaizdavimui ir koncepciniam modeliavimui skirtų kalbų analizė (Analysis of Ontology and Conceptual Modeling Languages). *Informacinės technologijos* 2007, *Kaunas: Technologija*, 2007, 217-221.

[3] OMG. Ontology Definition Metamodel Specification. *Adopted Specification* 2006-10-11. *http://www.omg.org/docs/ptc/06-10-11.pdf* (2007-03-20).

[4] **B. Motik, A. Maedche, R. Volz**. A Conceptual Modeling Approach for Building Semantics-Driven Enterprise Applications. *Proc. of* 1*st Int'l Conf. Ontologies, Databases, and Application of Semantics* (ODBASE-2002), *Springer-Verlag*, 2002, 1082–1099.

[5] **P. Mitra, G. Wiederhold, M. Kersten.** A graph oriented model for articulation of ontology interdependencies. *Proc. Extending DataBase Technologies, Springer, Berlin Heidelberg*, 2000, LNCS 1777, 86–100.

[6] **J. Davies, R. Studer, P**. **Warren.** Semantic Web Technologies Trends and Research in Ontology-based Systems. *John Wiley & Sons Ltd*, 2006.

[7] **P.P. Chen.** The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1), 1976, 471-522.

[8] OMG. *Ontology Definition Metamodel. Preliminary Revised Submission to OMG RFP ad*/2003-03-04. *http://www.omg.org/docs/ptc/03-03-04.pdf* (2007-03-20).

[9] **M. Gyssens, J. Paredaens, J.V. d. Bussche, D.V. Gucht.** A Graph-Oriented Object Database Model. *IEEE Transactions on knowledge and Data Engineering*, 1994, *Vol*.6, *No*.4, 572-576.

[10] **T. Levendovszky et al.** Model Reuse with Metamodel-Based Transformations. *C. Gacek (ed.): Proc. of ICSR-7. Springer*, 2002, *LNCS* 2319, 166-178.

[11] **J. Miller, J. Mukerji (eds.).** MDA Guide Version 1.0. OMG Document: omg/2003-05-01**.** *http://www.omg.org/docs/omg/03-05-01.pdf* (2007-03-20).

[12] **S. Brockmans et al.** A Model Driven Approach for Building OWL DL and OWL Full Ontologies. 5*th International Semantic Web Conference, Athens, GA, USA*, 2006, *LNCS* 4273, 187-200.

[13] **E. Vysniauskas, L. Nemuraite.** Transforming Ontology Representation from OWL to Relational Database. *Information Technology And Control, Kaunas, Technologija*, *Vol*.35A, *No*.3, 2006, 333 - 343.

[14] **V. Sugumaran, V. C. Storey.** The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems (TODS), ACM Press, Vol*.31, *No*.3, 2006, 1064 – 1094.

[15] **J. Conesa, X. de Palol, A. Olive.** Building Conceptual Schemas by Refining General Ontologies. 14*th International Conference on Database and Expert Systems Applications - DEXA* '03. 2003, *LNCS* 2736, 693-702.

[16] Sybase PowerDesigner. *Conceptual Data Model Version* 9.5, 2002. *http://download.sybase.com/pdfdocs/pdd0950e/cdgs.pdf* (2007-03-20).

[17] OMG. *Ontology Definition Metamodel Preliminary Revised Submission,* 2005. *http://www.omg.org/docs/ptc/05-08-01.pdf* (2007-03-20).

[18] **M. Gogolla, A. Lindow, M. Richters, P. Ziemann.** Metamodel Transformation of Data Models. *In Bezivin, J., France, R. (eds.): Proc. UML*'2002., 2002, *http://citeseer.ist.psu.edu/gogolla02metamodel.html* (2007-03-20)

Received March 2007.