SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2009. Vol. 751
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 52–64 P.

Ontology Transformation: from Requirements to Conceptual Model*

Justas Trinkunas[1] and Olegas Vasilecas[2, 3]
[1] Information Systems Research Laboratory, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius-40, Lithuania, *justas@isl.vgtu.lt*
[2] Information Systems Research Laboratory, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University, Sauletekio al. 11, LT-10223 Vilnius-40, Lithuania, *olegas@isl.vgtu.lt*
[3] Department of Computer Science, Faculty of Natural Sciences, Klaipeda University, Herkaus Manto 84, LT-92294 Klaipeda, *olegas.vasilecas@ik.ku.lt*

Information systems are increasingly complex, especially in the enormous growth of the volume of data, different structures, different technologies, and the evolving requirements of the users. Consequently, current applications require an enormous effort of design and development. The fast-changing requirements are the main problem in creating and/or modifying conceptual data models. To improve this process, we proposed to reuse already existing knowledge for conceptual modelling. In the paper, we have analysed reusable knowledge models. We present our method for creating conceptual models from various knowledge models.

Keywords: transformation, ontology, data modelling.

1 Introduction

Most of information systems analysts have at least once considered how many times they have to analyse the same problems, create and design the same things. Most of them ask, is it possible to reuse already existing knowledge? Our answer is yes. We believe that knowledge can and should be reused for information systems development. The knowledge reuse could be really of help to information systems analysts and engineers who develop information systems in one particular area for many years or have one software product line.

However, even these days, most domain knowledge is elicited from documents, experts and usually waste previous efforts, time, and resources. In this paper, we present available knowledge sources which could be reused in software engineering process,

* 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.

and we present a new approach of building conceptual models from these sources. Real world domain knowledge or, as we call it, domain ontology can bring outstanding benefits in software engineering. We already proved in [11] that ontologies can be reused for conceptual data modelling.

In this paper we are proposing a method of knowledge reuse for those who seek an efficient and quality driven approach for data structure development, data integration strategies, enterprise data models, logical data models, database design, data warehouse design, or data mart design.

The paper is organised as follows: the next chapter describes the theoretical background; in Chapter 3 we present and analyse available knowledge sources which could be reused for conceptual data modelling; and in Chapter 4 we present the method for knowledge reuse for data modelling.

2 Related Work

In this chapter we present ontology and metamodel based transformations used in our proposed method. We also discuss quality metrics which could be used for evaluation of the method and data sources.

2.1 Ontology

Computer science defines ontology as “a formal, explicit specification of a shared conceptualization” [24]. This definition is based on the idea of conceptualization: a simplified view of the world that we want to represent. Conceptualization is the process by which human mind forms an idea about part of the reality. This idea is a mental representation free of accidental properties and based on essential characteristics of the elements. Therefore, the (computer science) ontology concept is joined to a domain or mini-world, and the specification represented in ontology is concerned with that domain. In computer science, the main idea to create ontologies is to take a concrete model of the world and, through a descriptive process (function), to create an abstract model that captures its essence.

Many authors propose different ontology definitions. We accept the ontology definition proposed in [18]. Ontology defines the common terms and concepts (meanings) used to describe and represent an area of knowledge. Ontology can range in expressivity from taxonomy (knowledge with minimal hierarchy or a parent/child structure) to thesaurus (words and synonyms) to a conceptual model (with more complex knowledge), to a logical theory (with very rich, complex, consistent, and meaningful knowledge).

2.2 Metamodel-Based Transformations

The notion of model transformation is central to Model Driven Engineering. A model transformation takes as input a model that conforms to a given metamodel and produces as output another model that conforms to a given metamodel. A model transformation may also have several source models and several target models. One of the characteristics of a model transformation is that a transformation is also a model, i.e. it conforms to a given metamodel. More information on metamodels can be found in [16].

Model Driven Architecture (MDA) [17] defines three viewpoints (levels of abstraction) from which a system can be viewed. From a chosen viewpoint, a representation of a given system (viewpoint model) can be defined. These models each correspond to the viewpoint with the same name, and they are Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM). MDA is based on four-layer metamodeling architecture, and several OMG's complementary standards. There are the meta-metamodel (M3) layer, metamodel (M2) layer, model (M1) layer, and instance (M0) layer.

We analyse ontology transformation to a data model. The mapping from OWL (ontology web language) to ER was described in [18]. However, this mapping is incomplete and it is not clear which elements from the OWL ontology are not transformed into data model. As a result, some information from OWL ontology cannot be used in data model. Metamodel-based transformations are shown in Figure 1.

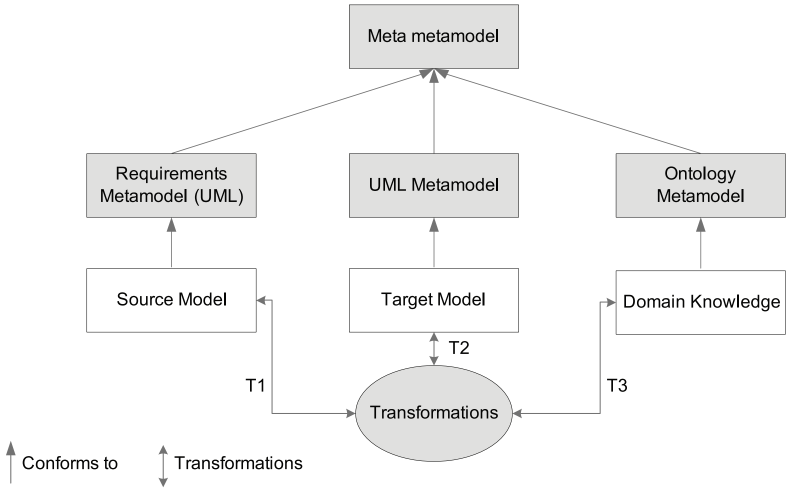


Fig. 1. Metamodel-based transformation

2.3 Quality Properties

We have to discuss the quality properties that could be used for transformation evaluation. The paper [19] provides a list of quality properties. The main quality properties are: annotation, appropriateness, completeness, conceptual clarity, consistency, correctness, expressiveness, testability, unambiguity, understandability, verifiability. However, this list is not very useful because we need a systematic approach to quality improvements.

Another author [20] defines the following properties.

- Legibility. To measure legibility which expresses the ease with which a conceptual schema can be read, we propose two subcriteria, namely clarity and minimality. Clarity is a purely aesthetic criterion. It is based on the graphical arrangement of the elements composing the schema. The second sub-criterion is minimality. A schema is said to be minimal when every aspect of the requirements appears only once.

- Expressiveness. A schema is said to be expressive when it represents users requirements in a natural way.
- Simplicity. A schema is said to be simple if it contains the minimum possible constructs.
- Correctness. The property is used in a wide range of contexts leading to very different interpretations. A schema is syntactically correct when concepts are properly defined in the schema.
- Completeness. A schema is complete when it represents all relevant features of the application domain [10]. More specifically, the completeness can be measured by the degree of coverage of user's requirements by the conceptual schema.
- Understandability. Understandability is defined as the ease with which the user can interpret the schema. This criterion is very important for the validation phase and, consequently, influences directly the measure of completeness. The understandability of a conceptual schema relies on how much modelling features are explicit.

Using these properties, we will evaluate transformation.

3 Available Domain Knowledge Sources

In this section we review and analyse available knowledge sources which could be reused for conceptual data modelling. It is very important to have good quality domain knowledge source because transformation quality straightforwardly depends on knowledge quality. Of course, the transformed model can and should be improved manually.

Information systems are increasingly complex, especially because of the enormous growth of the volume of data, different structures, different technologies, and the evolving requirements of the users. Consequently, current applications require an enormous effort of design and development. In fact, such applications require a detailed study of their fields in order to define their concepts and to determine the concepts' relationships [10].

Knowledge models are reusable models, in other words, 'templates' to help jump start and/or quality assure data modelling efforts. They can be used to save time and costs on efforts to develop enterprise data models, logical data models, database, and data warehouse.

There are many books and articles written on the subject of data modelling, and most system professionals know how to model data. Actually, what they need are reusable knowledge models which could be reused for real projects and could save many hours of work [2].

If we want to get a high quality data model after transformation, the knowledge model has to be simple, correct, and complete.

If the knowledge model is not simple after the transformation, we will get a complex conceptual schema, which will have to be improved manually.

If the knowledge model is not correct, after the transformation we will get the same mistakes in the conceptual model.

If the knowledge model is not complete (at least in our domain), after the transformation we will get an incomplete conceptual model.

We propose to improve knowledge model iteratively. All mistakes noticed in the conceptual model should also be rechecked in the knowledge model. Also, if the conceptual model is incomplete, we have to add the needed information into the knowledge model. Step by step we can create a sophisticated source of knowledge. We will not discuss the creation of ontologies in this paper because this topic needs more attention and is out of the scope of this paper.

In the next chapters, we analyse different knowledge sources which could be reused for conceptual data model building, and we try to evaluate which of them is the most suitable.

The knowledge sources can be classified into three main categories – commercial (for example, IBM data model, industry data models [2]), freely available (for example, SUMO and OpenCyc), and manually created for a specific purpose.

3.1 SUMO

In this section we present SUMO domain ontology which could be used as a knowledge model. The main difference between universal models and domain ontologies is that usually ontologies are more abstract. To reuse ontology is more complicated than to reuse a universal model. However, domain ontologies are a really good knowledge source that could be reused.

SUMO is the largest free, formal ontology [1]. It contains more than 20,000 terms and 70,000 axioms if all domain ontologies are combined. SUMO consists of SUMO itself, the MId-Level Ontology (MILO), and domain ontologies (communications, countries and regions, distributed computing, economy, finance, engineering components, geography, government, military, North American industrial classification system, people, physical elements, transportation, etc). SUMO is written in the SUO-KIF language.

SUMO ontology also provides check rules. Most significantly, SUMO ontology is freely available for everyone. Everyone can participate in the ontology development and improvement process. Other advantage is that ontologies provide a set of rules, i.e. we can restrict the model. However, most of these ontologies do not cover all domain areas.

3.2 Cyc and OpenCyc

OpenCyc [5] is the open source version of the Cyc technology, the world's largest and most complete general knowledge base and commonsense reasoning engine. The entire Cyc ontology contains hundreds of thousands of terms, along with millions of assertions that relate the terms to each other, forming an upper ontology, whose domain is all the human consensus about reality.

The Cyc project has been described as "one of the most controversial endeavours of the artificial intelligence history" [13]; hence, it has inevitably garnered its share of criticism.

OpenCyc is similar to full Cyc, but its knowledge base is just a few percent of the full knowledge base and its functionality is greatly reduced. Since Cyc's success lies in the completeness of its knowledge base, the only people who really know the extent of Cyc's progress are Cycorp employees [4].

Furthermore, with trying to cover the entire world, Cyc becomes too enormous and abstract. The reuse of this ontology is very complicated. Some issues about Cyc reuse are discussed in [14].

3.3 Wikipedia-Based Ontologies

In recent years some communities tried to extract structured information from Wikipedia. As a result, YAGO [7], DBpedia [8], and FreeBase [9] ontologies were created. In this section we shortly introduce these ontologies.

YAGO is a huge semantic knowledge base. According to [7], YAGO contains more than 2 million entities and 20 million facts about these entities. The authors of YAGO state that YAGO has a manually confirmed accuracy of 95%.

DBpedia [8] is a knowledge base which allows to make sophisticated queries in Wikipedia, and to link other data sets on the Web to Wikipedia data. The DBpedia data set currently provides information on more than 2 million entities. Altogether, the DBpedia data set consists of 218 million pieces of information (RDF triples). The accuracy of DBpedia is not confirmed at the moment.

Freebase [9] is an open, shared database that contains structured information on millions of topics in hundreds of categories. This information is compiled from open datasets like Wikipedia, MusicBrainz, the Securities and Exchange Commission, and the CIA World Fact Book, as well as contributions from user community. The accuracy of Freebase is not confirmed at the moment.

YAGO, DPpedia, and Freebase knowledge sources are really valuable. At the moment DBpedia is the biggest Wikipedia-based ontology. The accuracy of YAGO has been confirmed.

3.4 Industry Data Models

The book [2] provides a series of industry universal data models for each phase of an enterprise's business cycle: people and organizations, products (services or physical goods), commitments which are established between people and/or organizations, transport shipment, work efforts, invoices, budgeting and accounting, human resources management and tracking.

An industry data model or universal data model is a model that is widely applied in some industry. Sufficiently effective industry data models have been developed in banking, insurance, pharmaceuticals and other industries to reflect the strict standards applied in customer information gathering, customer privacy, consumer safety, or "just in time" manufacturing.

The authors of the book [2] claim that 60% of a data model (corporate or logical) or data warehouse design consists of common constructs that are applicable to most enterprises. This means that most data modelling or data warehouse design efforts are at some point recreating constructs that have already been built many times before in other organizations.

The authors provide nine subject areas: accounting and budgeting, human resources, invoicing and billing, orders and agreements, people and organizations, product, shipments and deliveries, web and e-commerce, work effort and project management.

In addition, several industry specific universal data models are available, including banking, investments and financial services, healthcare, insurance, manufacturing, professional services, telecommunications, and travel.

These universal data models are very useful for data modelling. However, we can reuse only the structure; these models do not contain any business rules, which also are a very important knowledge resource.

3.5 Commercial Data Models

There are many commercial data models which could be reused. We analysed the well-known IBM data model M1. M1 database contains the Banking Data Warehouse Model. The model is composed of an entity-relationship model for application development. It contains 910 entities and 5237 attributes.

Let us examine the 'product' definition of the IBM data model.

'Product' identifies goods and services that can be offered, sold, or purchased by the financial institution, its competitors, and other involved parties, or in which the financial institution has an interest during the normal course of its business activity; for example, 'Product#220 (Xyz bank's private banking residential mortgage loan)', 'Product #988 (personal checking account)', 'Product #440 (securities trading service)'. 'Product' has 22 attributes and 28 relationships in the model. A small part from the model is provided in Figure 2.

The IBM data model has three levels – the abstract level, the middle level, and the model level. Such organisation of the model is very convenient for reuse.

Commercial data models usually are very expensive and there are many restrictions.

3.6 Other Ontologies

Protégé [6] provides more than 50 domain ontologies; however, none of them can be used for conceptual data modelling because most of them contain only a few dozens of concepts and are totally immature.

WordNet is a large lexical database of English [15]. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms (synsets) that each express a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. WordNet cannot be straightforwardly reused, but it is very useful for finding synonyms and more abstract terms.

4 The Proposed Approach

In the previous section we analysed available sources which could be reused for conceptual modelling. In this section we describe our method for knowledge reuse for conceptual modelling. We performed a few experiments, including reuse of knowledge from already existing resources and from the resources which were created manually.

We briefly describe the proposed method for building a conceptual model from reusable models. The method consists of four main steps.

1. Building of requirements using ontology.
2. Finding or creation of an appropriate knowledge source which can be used for transformation.
3. Transformation of the knowledge model into a specific data model with our plug in OntER. The created data model can be opened with Sybase Power Designer 12.0 tool and adapted for specific needs. Transformation rules can be found in [24].
4. The last step is the generation of the physical data model with Power Designer 12.0 for a particular database management systems (DBMS). This feature is already implemented in the original version of Power Designer 12.0.

By a simple generation procedure, the conceptual data model can be transferred 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.

The transformation process is shown in Figure 2.

The first experiments were carried out with the domain ontology found in Protégé site [6]. Other experiments were performed with ontology created by us. Finally, we tried to experiment with other ontologies.

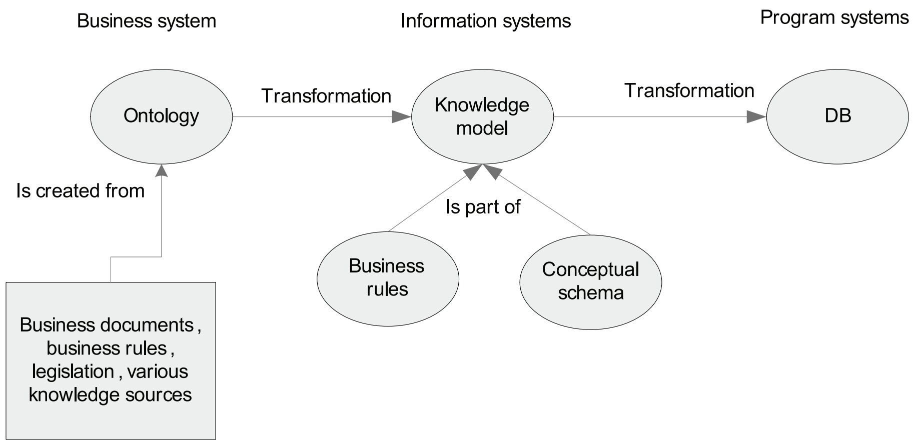
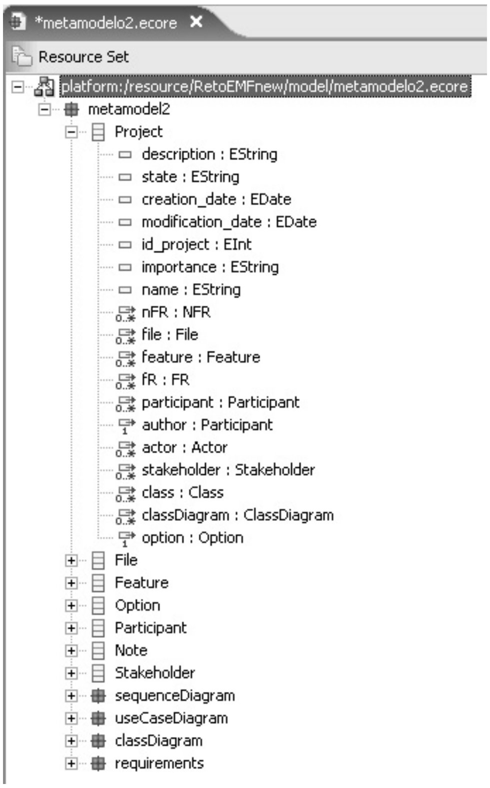


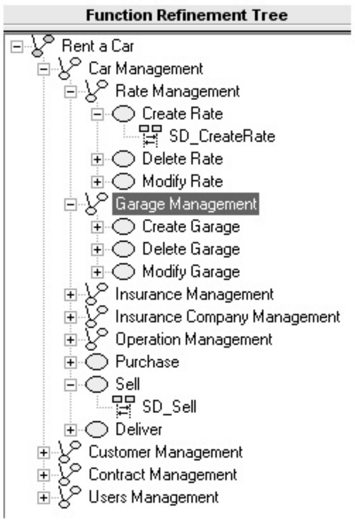
Fig. 2. The transformation process

We created a requirement metamodel with the Eclipse tool (Figure 3). We also used the requirement model taken from [22]. This requirement model [21] is composed of a Function Refinement Tree (Figure 4) to specify the hierarchical decomposition of the system, a Use Case Model to specify the system communication and functionality, and Sequence Diagrams specify the required object-interactions that are necessary to realize each Use Case. The ontology is presented in Figure 5.



```
*metamodelo2.ecore
Resource Set
platform:/resource/RetoEMFnew/model/metamodelo2.ecore
  metamodel2
    Project
      description : EString
      state : EString
      creation_date : EDate
      modification_date : EDate
      id_project : EInt
      importance : EString
      name : EString
      nFR : NFR
      file : File
      feature : Feature
      fR : FR
      participant : Participant
      author : Participant
      actor : Actor
      stakeholder : Stakeholder
      class : Class
      classDiagram : ClassDiagram
      option : Option
    File
    Feature
    Option
    Participant
    Note
    Stakeholder
    sequenceDiagram
    useCaseDiagram
    classDiagram
    requirements
```

Fig. 3. Requirement metamodel



```
Function Refinement Tree
Rent a Car
  Car Management
    Rate Management
      Create Rate
        SD_CreateRate
      Delete Rate
      Modify Rate
    Garage Management
      Create Garage
      Delete Garage
      Modify Garage
    Insurance Management
    Insurance Company Management
    Operation Management
    Purchase
    Sell
      SD_Sell
    Deliver
  Customer Management
  Contract Management
  Users Management
```

Fig. 4. Function Refinement Tree

Below a small piece of ontology “Enterprise” in Protégé (frames) format (defclass Supplier “someone whose business is to supply a particular service or commodity”) is provided.

```
   (is-a Enterprise)
   (role concrete)
   (single-slot city
;+         (comment “a place (city) where supplier is located”)
           (type INSTANCE)
;+         (allowed-classes City)
;+         (cardinality 0 1)
           (create-accessor read-write))
   (single-slot country
;+         (comment “country where supplier is located”)
           (type INSTANCE)
;+         (allowed-classes Country)
;+         (cardinality 0 1)
           (create-accessor read-write))
   (single-slot name_
;+         (comment “name of supplier”)
           (type STRING)
;+         (cardinality 1 1)
           (create-accessor read-write)))
```

Below the result of “Enterprise” ontology transformation into a conceptual model in Power Designer native format is provided.

```
( <o:Entity Id=”o58”>
<a:ObjectID>8B135BB3-7264-4809-910F-3DD4BEFC7DE0</a:ObjectID>
<a:Name>Supplier</a:Name>
<a:Code>Supplier</a:Code>
<a:CreationDate>1144263520</a:CreationDate>
<a:Creator>Justas</a:Creator>
<a:ModificationDate>1144263684</a:ModificationDate>
<a:Modifier>Justas</a:Modifier>
<c:Attributes>
<o:EntityAttribute Id=”o92”>
<a:ObjectID>07DEFDAC-0AD3-4A1C-AE20-4AD63A6A065A</a:ObjectID>
<a:CreationDate>1144263539</a:CreationDate>
<a:Creator>Justas</a:Creator>
<a:ModificationDate>1144263539</a:ModificationDate>
<a:Modifier>Justas</a:Modifier>
<c:DataItem>
<o:DataItem Ref=”o93”/>
</c:DataItem>
</o:EntityAttribute>
</c:Attributes>
</o:Entity>
```

Below a small piece of ontology “Salary” in Protégé (OWL) format and in graphical form (Figure 5) is provided.

```
<owl:Class rdf:ID=”WageRate”>
  <rdfs:comment xml:lang=”en”>Wage rate is amount of money
paid per unit of time or per unit of products</rdfs:comment>
  <rdfs:subClassOf>
  <owl:Class rdf:about=”#Wage”/>
  </rdfs:subClassOf>
  <owl:disjointWith>
  <owl:Class rdf:ID=”Quantity”/>
  </owl:disjointWith>
  <owl:disjointWith>
  <owl:Class rdf:ID=”Time”/>
  </owl:disjointWith>
</owl:Class>
```

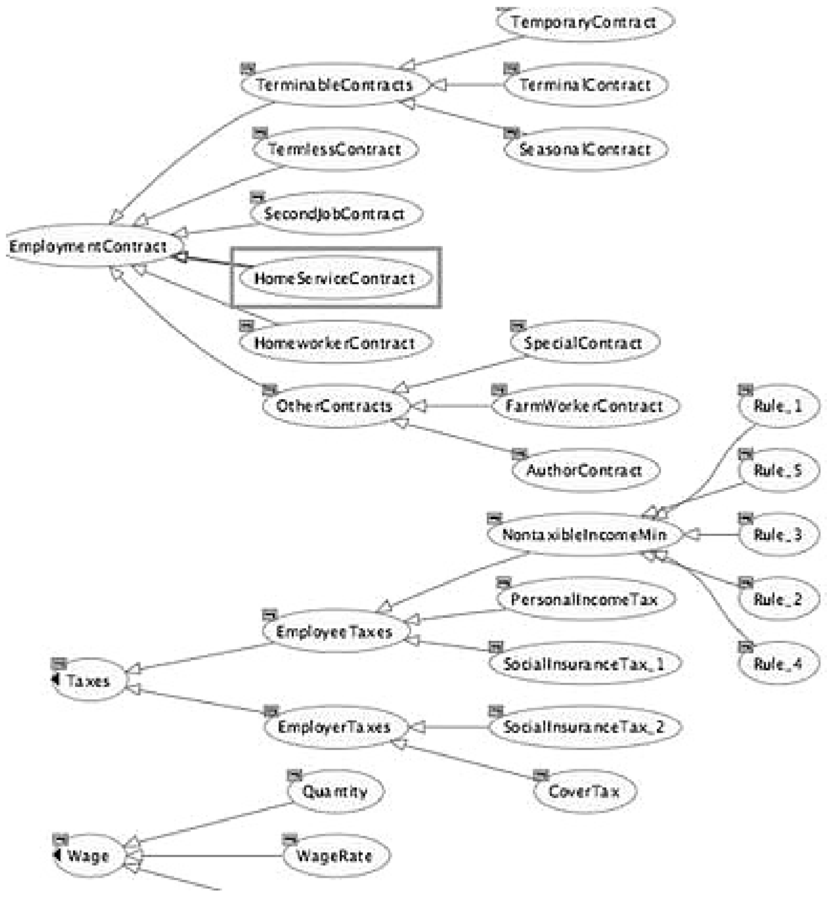


Fig. 5. Payroll ontology

Below the result of “Salary” ontology transformation into a conceptual model in Power Designer native format is provided.

```
( <o:Entity Id=”o40”>
 <a:ObjectID>2E0D229D-A725-4078-9EFD-D6F2F036E775</a:ObjectID>
 <a:Name>WageRate</a:Name>
 <a:Code>WageRate</a:Code>
 <a:CreationDate>1144314871</a:CreationDate>
<a:Creator>Justas</a:Creator>
<a:ModificationDate>1144314877</a:ModificationDate>
<a:Modifier>Justas</a:Modifier>
```

5 Conclusions and Future Works

We analysed available knowledge sources which could be reused for conceptual data modeling and proposed a method for knowledge reuse. The experiment showed that the proposed method is really effective. However, it is very important to have a good enough quality domain knowledge source because evaluation of the transformation process showed that quality strongly depends on the quality of the knowledge model.

After a thorough analysis of available knowledge sources, we decided that the most completed ontology is CYC. The most expressive is SUMO. However, the most suitable by all the quality properties listed in the Chapter 2 are universal and commercial data models.

Nevertheless, after series of transformations carried out, we found that many tools do not follow already accepted standards, and this situation makes research work even more difficult. We plan to extend the research work and create OWL to UML transformation in Eclipse environment with ATL language.

6 References

1. Suggested Upper Merged Ontology (SUMO), *http://www.ontologyportal.org/*, [2008-04-28].
2. Silverston L. (2001) The Data Model Resource Book: A Library of Universal Data Models for All Enterprises, Revised Edition, Volume 1. John Wiley & Sons.
3. Embarcadero Technologies, *http://www.embarcadero.com/*, [2008-04-20].
4. Friedman J. The Sole Contender for AI, *Harvard Science Review* 2003, *http://www.scribd.com/doc/1814/An-Article-about-the-Cyc-Project* [2008-04-18].
5. OpenCyc 1.0.2, *www.opencyc.org*, [2008-04-19].
6. Protégé ontologies, *http://protege.stanford.edu/download/ontologies.html*, [2008-04-19].
7. Yago, *http://www.mpi-inf.mpg.de/~suchanek/downloads/yago/* [2008-11-03].
8. DBpedia, *http://dbpedia.org/About* [2008-11-03].
9. Freebase, *http://www.freebase.com/* [2008-11-03].
10. Mhiri M., Chabaane S., Mtibaa A., Gargouri F. *An Algorithm for Building Information System’s Ontologies*. Eight International Conference on Enterprise Information Systems, Paphos, Cyprus, 2006, 467–470.
11. Trinkunas J., Bugaite D., Vasilecas O. Formal Transformation of Domain Ontology into Conceptual Model. *Izvestia of the Belarusian Engineering Academy*, 2006, Vol. 1 (21)/2, 2006, 112–117.
12. Brewster C. et al. Data driven ontology evaluation. *Proceedings of Int. Conf. on Language Resources and Evaluation*, Lisbon, 2004.
13. Bertino E., Zarri G. P., Catania B. Intelligent Database Systems. Addison-Wesley Professional, 2001. ISBN 0-201-87736-8

14. Conesa J., Palol X., Olive A. Building Conceptual Schemes by Refining General Ontologies. 14th International Conference on Database and Expert Systems Applications – DEXA '06, volume 2736 of LNCS, 2003, 693–702.
15. Wordnet – a lexical database for the English language, Princeton University Cognitive Science Laboratory, *http://wordnet.princeton.edu/* [2008-04-19].
16. Kanai S., Kishinam T., Tomura T. (2000) Object-oriented Graphical Specification and Seamless Design Procedure for Manufacturing Cell Control Software Development. *Proc. of the 2000 IEEE lntemational Conference on Robotics & Automation*, San Francisco, 401–407.
17. Miller J., Mukerji J. (eds.) (2003) MDA Guide Version 1.0. OMG Document: omg/2003-05-01. *http://www.omg.org/docs/omg/*03-05-01.pdf (2007-03-20).
18. OMG (2006) Ontology Definition Metamodel Specification. Adopted Specification 2006-10-11. *http://www.omg.org/docs/ptc/06-10-11.pdf* (2007-03-20).
19. Lindland O. I., Sindre G., Sølvberg A. (1994) Understanding Quality in Conceptual Modeling, IEEE Software, v. 11 n. 2, 42–49.
20. Cherfi S., Akoka J., Comyn-Wattiau I. (2002) Conceptual Modeling Quality – From EER to UML Schemes Evaluation. In: Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi, editors, *Proceedings of the 21st International Conference on Conceptual Modeling*, Volume 2503 of Lecture Notes in Computer Science, Tampere, Finland. Springer-Verlag, 414–428.
21. Abrahão, S., Genero, M., Insfran, E., Carsí, J. A., Ramos, I., Piattini, M. (2008) Quality-Driven Model Transformations: From Requirements to UML Class Diagrams. In: *Model-Driven Software Development: Integrating Quality Assurance*. IGI Publishing.
22. Reto Tool, *http://reto.dsic.upv.es/reto* [2008-06-19].
23. Gruber T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5 (2), 1993, 199–220.
24. Trinkunas J., Vasilecas O. A Graph-Oriented Model for Ontology Transformation into Conceptual Data Model. *Information Technology and Control, Kaunas, Technologija*, 2007, Vol. 36, No. 1A, pp. 126–131.