Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp324-329)

Domain ontology based object-oriented and relational databases

J. SEBESTYÉNOVÁ
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava
SLOVAKIA
http://www.ui.sav.sk

Abstract: - The paper describes incorporation of ontology in database of decision support system. Implementation of persistent knowledge-base can be done using relational as well as object-oriented database. Creation of database schema according to classes with their specific properties and slots of domain ontology is described for both types of databases. Besides the database maintenance problems, various possibilities of database reasoning are shortly described.

Key-Words: - Ontology, Intelligent agents, Distributed computing and distributed databases, Decision support system

1 Introduction

Two entities can communicate only if they agree upon the meaning of the terms they use. **Ontology**, understood as an agreed vocabulary of common terms and meanings shared by a group of people, is a solution to that problem.

In order for an agent to make statements and ask queries about a subject domain, it must use a conceptualization of that domain. A domain conceptualization names and describes the entities and the relationships among those entities that are to be considered in that domain. It therefore provides a vocabulary for representing and communicating knowledge about the domain.

Explicit specifications of domain conceptualizations, called ontologies, are essential for the development and use of intelligent systems as well as for the interoperation of heterogeneous systems. They provide a vocabulary for a domain and can be used as building block components of knowledge bases, object schema for object-oriented systems, conceptual schema for data bases, structured glossaries for human collaborations, vocabularies for communication between agents, class definitions for conventional software systems, etc.

Ontology construction is difficult and time consuming. This high development cost is a major barrier to the building of large-scale intelligent systems and to widespread knowledge-level interactions of computer-based agents. Since many conceptualizations are intended to be useful for a wide variety of tasks, an important means of removing this barrier is to encode ontologies in a reusable form so that large portions of ontology for a given application can be assembled from existing ontologies available from ontology libraries. Intelligent systems are characterized by their ability to effectively reason with their knowledge.

Computer-understandable ontologies are represented in logical languages, such as the W3C OWL (Ontology Web Language) and the draft ISO standard, SCL (Simple Common Logic). However, logical languages are only a means to express content. Ontology is one way to use language and logic more effectively.

There is mounting psychological evidence that human cognition centrally involves similarity computations over structured representations, in tasks ranging from high-level visual perception to problem solving, learning, and conceptual change. Understanding how to integrate **analogical processing** into AI systems seems crucial to creating more human-like reasoning systems [5]. Yet similarity plays at best a minor role in many AI systems. Most AI systems operate on a first-principles basis, using rules or axioms plus logical inference to do their work. Those few reasoning systems that include analogy tend to treat it as a method of last resort, something to use only when other forms of inference have failed.

The exceptions are case-based reasoning systems, which started out to provide computational mechanisms similar to those that people seem to use to solve everyday problems. Unfortunately, CBR systems generally have the opposite problem, tending to use only minimal first-principles reasoning. Moreover, most of today's CBR systems also tend to rely on feature-based descriptions that cannot match the expressive power of predicate calculus. Those relatively few CBR systems that rely on more expressive representations tend to use domain-specific and task-specific similarity metrics. This can be fine for a specific application, but being able to exploit similarity computations that are more like what people do could make such systems even more useful, since they will be more understandable to their human partners. While many useful application systems can be built with purely first-principles reasoning and

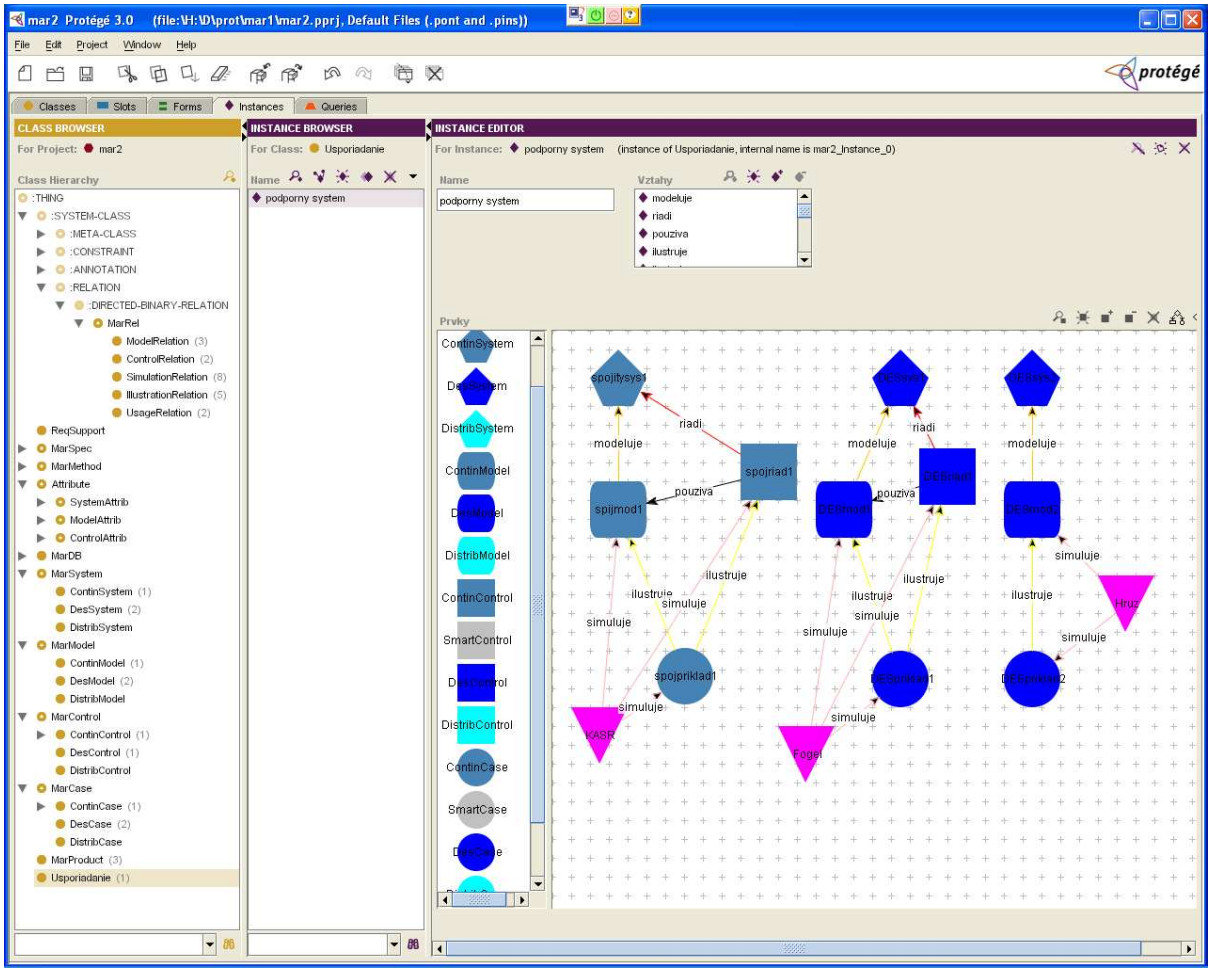

Figure 1 Ontology creation in Protégé

with today's CBR technologies, integrating analogical processing with first-principles reasoning will bring us closer to the flexibility and power of human reasoning.

One of the bottlenecks in the creation of AI systems is the difficulty of creating large knowledge bases. There have been a number of systems that capture some aspects of reasoning by analogy. No previous analogy systems have been successfully used with multiple, large general-purpose knowledge bases created by other research groups. While the majority of today's CBR systems have moved to feature-vector representations, there are a number of systems that still use relational information.

2 Domain Ontology

We can consider an ontology as a collection of agreements upon a vocabulary of common terms and meanings in some domains. **Concept** is an entity representing some "thing", the actual entity in the real world.

There are two kinds of conceptual knowledge: concept and set. A concept is defined by the essence of the objects it subsumes and not by their state. Such a definition allows us to focus on the essence of the concepts and not on their state. An essence is invariant, which is not the case of state. On the other hand, a set makes it possible to put together objects whose state shares some common properties. For instance, if "Human Being" refers to a concept, "Teenagers" refers to a set composed of human beings whose age is in given constraints.

Differences are the elementary units from which the meaning of terms is built. This means they have no meaning in themselves. A difference belongs to the essence of objects. Unlike an attribute it cannot be removed from the definition of an object without changing its nature; nor can it be valued. For example, for human beings "mortal" is a difference whereas "age" is an attribute.

A difference is a unit that builds meanings and divides concepts. Adding a difference to an existing concept makes it possible to create two new ones, the first to which it belongs and the second which will never be able to own it. That difference is called the specific difference of the former new concept. This is the reason

why differences are defined by couple of opposite differences, like “mortal” and “immortal”. Thus, owning a difference for a concept implies it will never contain the opposite difference, nor the concepts it could subsume.

Ontology terms are organized in structures called directed acyclic graphs where nodes can have multiple parents [7]. Ontologies can be considered as a set of concepts, which are connected by binary relations. Concepts are well-defined entities: they have a unique meaning, properties like a name (label), a description and an identifier. Fig. 1 shows, for example, the ontology created in Protégé tool [11].

Main directions in usage of ontologies are:
- Semantic web (computer – understandable semantics)
- Multi-agent systems communication (meaning of sent and received messages)
- Database systems (in the process of distributed databases reasoning, and/or in the process of ontology-based database creation).

There exist many large database systems (some of them with partially conjunctive domains), but searching any information in such distributed databases seems to be intractable, because of different database schemas. Definition of the domain ontology with properly given similarity relations (or equality relations, such as “same as”) between concepts may solve the problem. Queries to databases need to be modified according to the defined domain ontology, which can be done automatically. This approach is usable for simple queries [3], [6].

3 Ontology - based Databases

Another way to incorporate the ontology in database system is to create a database schema with respect to given domain ontology.

An entity-relationship diagram represents data model of a relational database. ER model can also represent concepts and relationships in the ontology (as a set of nodes, which are connected by edges). However, a tree or a net representation of the ontology is needed when a user browses an ontology, or when all "child nodes" of a given hierarchy have to be selected. The basic concept of ER modeling is not powerful enough for such complex applications and additional semantic modeling concepts are required:
- Specialization
- Generalization
- Categorization
- Aggregation.

Some new entity constructs are still needed:
- Superclass - an entity type that includes distinct subclasses that require to be represented in a data model
- Subclass - an entity type that has a distinct role and is also a member of a superclass.

If we focus on the representational aspects, the domain ontology can be viewed as a knowledge-based system using the following entities:
- Concepts,
- Cases,
- Schemes,
- Associations.

A straight comparison between a DB and an ontology needs to take the nature of the data into account. The advent of object orientated databases, improved logics and faster inference is making the distinction between DBs and ontologies more fuzzy.

Real-world semantics (term used in literature on semantic integration of databases) corresponds to the concepts in the real world that the objects in the model refer to. This type of semantics involves human interpretation (or meaning) and use of data or information. A huge role of ontology is not so much for processing, but for sharing meaning and for improving tacit knowledge transfer.

The DB schema is the structure of data, whereas the data are the facts. We can map the data structures in the database to ontology: table ↔ class, table column ↔ property, value of table column ↔ literal or resource, foreign key ↔ property pointing to other resource, table row ↔ instance of class.

3.1 Relational DB

Relational databases are the most common DB today. If we want to create the relational database schema according to a given domain ontology, first we have to create a table (or tables) of properties with all of their possible values. Secondly we have to create a table for representing the given class of the ontology. An instance of the class (individual) will be represented as a row in the table, and it will be characterized by the properties with given values (valuation is restricted to previously specified possible values). Such approach can provide variable creation of database schema from an application, though it is not usual or obvious for relational databases.

In order to create a database schema according to the given class of the domain ontology, algorithm of the database creation and manipulation part of the application follows these steps:

1. Create the table of properties with N+2 columns named: property_name, number_of_allowed_values, allowed_value_1, … , allowed_value_N, where N is maximum of the properties cardinality (specified previously).

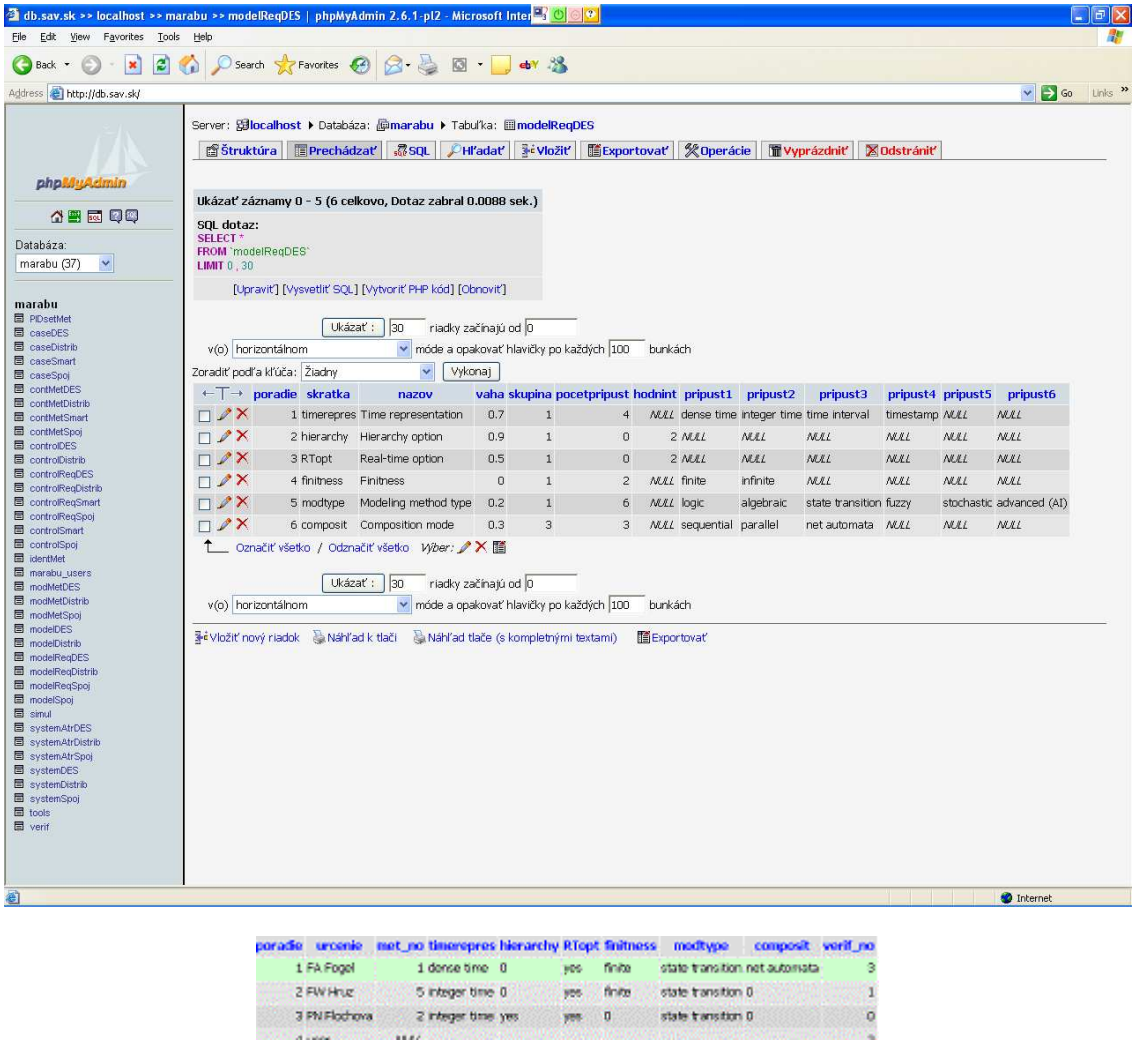

Figure 2 Creation of ontology-based relational database in MySQL - tables of properties and class individuals

2. Provide interface to specification of some properties that will be written into the table as rows. This step can work with GUI, or by automatic translation of slots from the ontology.
3. Create the table of the class, that will be characterized by above specified properties, i.e. column names have to be read from the other table (all values of property_name).
4. Provide GUI for specification of class individuals that will be written into the table as rows. For all of the properties, user can select valuation from the previously specified allowed values.

Fig. 2 shows a table of properties and a table of the class in MySQL. For illustration, the class of discrete event systems model requirements has been used, which is the class of control theory domain ontology.

3.2 Object - oriented DB

Object-oriented DBs have many useful features: inheritance, classes, methods, associations, overloading, close links with OO programming languages, types, persistence, OQL.

New SQL standards utilize object - relational extensions: excellent performance results, well understood technology, good portability. Most problems which were considered OO only are relatively easy to implement on OR technology.

A task of creation of the database schema according to the classes of the domain ontology is easier in case of OO DB than in relational DB. For illustration, Fig. 3 shows this in JADE development environment [4] for developing applications using OO DB. JADE enables us to model and construct information systems in terms of a set of self-contained components called objects. JADE object model combines data and operations into objects, uses messages to communicate between objects, groups similar objects into classes, and maintains a class hierarchy to provide inheritance of data and procedures.

JADE provides the RootSchema, which is always at the top of the schema hierarchy. The RootSchema provides essential system classes, e.g. the Object class -

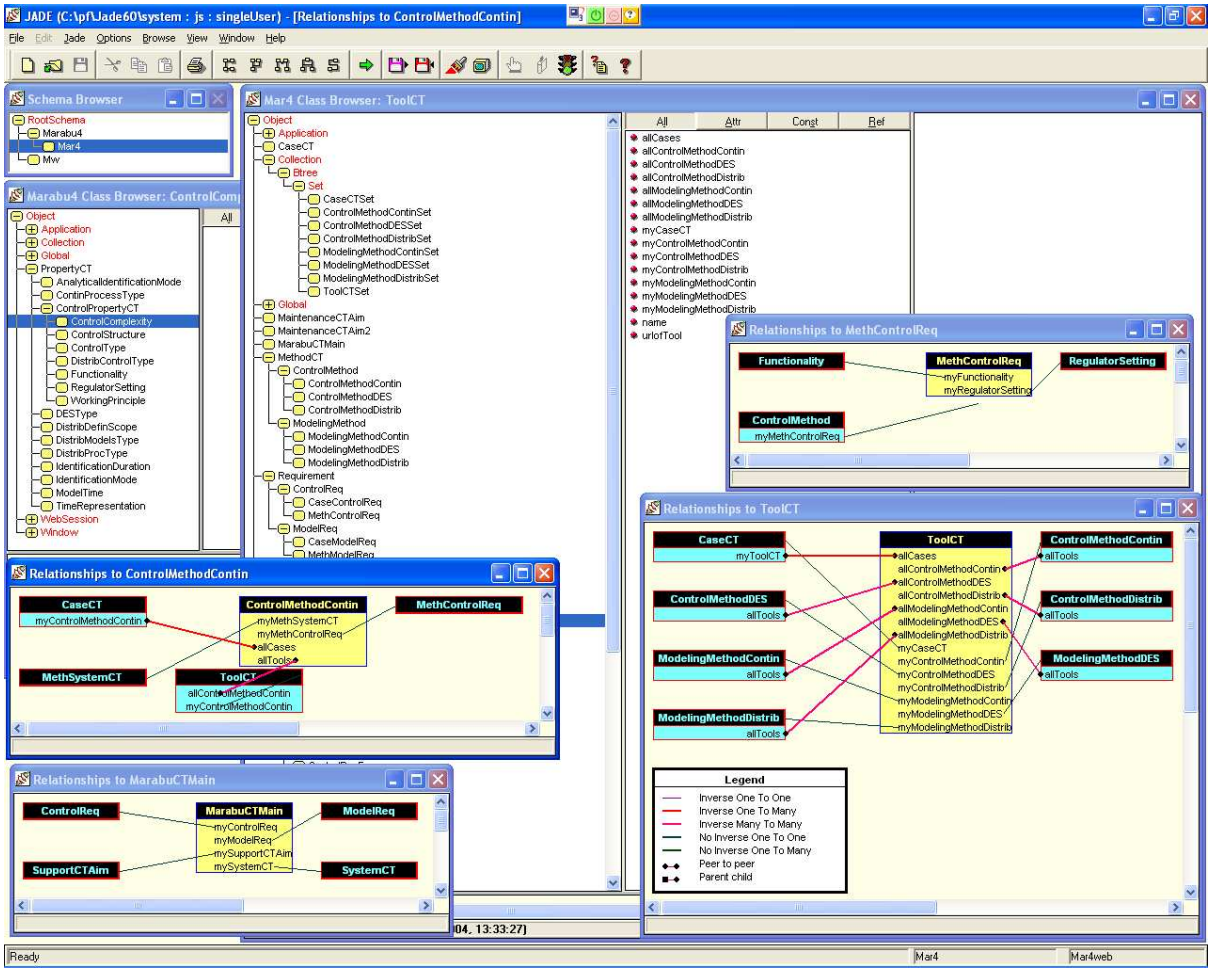

Figure 3 Creation of ontology-based object-oriented database in JADE

root of the class hierarchy, Collection classes, and the File, Exception, and Form classes. Because these classes are defined in the RootSchema, they can be accessed from all subschemas.

An object is any entity, either real or abstract, that exhibits some well-defined behavior and that has a unique identity. A class groups all objects that share the same set of properties and methods. Every object is an instance of one, and only one, class. A class describes common characteristics of a set of objects. An object is often referred to as an instance of the class that describes it.

Properties (attributes and references) represent the internal data storage of an object. The state of an object at any time is determined by values stored in each of its properties. Properties can be:

- Attributes: primitive variables such as numbers, strings, or Boolean values (e.g. age, name, and gender)
- Single-valued references: references to other objects
- Multiple-valued references: references to collections of other objects (for example, a collection of employees in a company object).

4 Decision support system

A knowledge base of a multi-agent system contains an ontology-based representation of the data relevant to the structure of the domain as well as supplementary functional data. For example, an agent-based decision support system using rule-based deductive reasoning is described in [12].

Decision theory is a means of analyzing which of a series of options should be taken when it is uncertain exactly what the result of taking the option will be. Decision theory provides a powerful tool with which to analyze scenarios in which an agent must make decisions in an environment.

4.1 DB Reasoning

Forward chaining is an example of the general concept of **data-driven reasoning** - that is, reasoning in which the focus of attention starts with the known data. It can be used within an agent to derive conclusions from incoming percepts, often without a specific query in mind. New facts can be added to the agenda to initiate new inferences.

The backward-chaining algorithm, as its name suggests, works backwards from the query. If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base that conclude q.

Backward chaining is a form of **goal-directed reasoning**. It is useful for answering specific questions such as "What shall I do now?" Often, the cost of backward chaining is much less than linear in the size of the knowledge base, because the process touches only relevant facts. In MAS, an agent should share the work between forward and backward reasoning, limiting forward reasoning to the generation of facts that are likely to be relevant to queries that will be solved by backward chaining.

In most case-based reasoning systems, cases are stored as named collections of facts in a memory. They are designed for a specific range of problems. Each case is a set of features, or attribute-value pairs, that encode the context in which the ambiguity was encountered. The case retrieval algorithm is mostly a simple k-nearest neighbors algorithm. The basic case-based learning algorithm performs poorly when cases contain many irrelevant attributes. Unfortunately, deciding which features are important for a particular learning task is difficult.

4.2 Active DB

Relational as well as OO DB systems posses a possibility of simple kind of event-driven processing [1], [2]. At a time instant specified in DB or after a given time interval elapse, an event occurs which subsequently starts an action (or activity). It can be used, for example, to automatically send e-mail containing relevant information reasoned from the knowledge-base according to the given context [8, 9, 10] and event.

5 Conclusion

Development of persistent knowledge-base for decision support systems was described. Creation of database schema according to concepts and their relations in domain ontology was described for relational and object-oriented databases. Besides the database maintenance problems, various possibilities of database reasoning were shortly mentioned.

Acknowledgements

The author is grateful to the Slovak Science and Technology Assistance Agency (grant No. APVT-51-024604 RAPORT) for partial support of this work.

References:

[1] Dayal U., E. Hanson, J. Widom, Active database systems, In: *Modern Database Systems*, W. Kim (Ed.), Addison-Wesley, Reading, MA, 1995, pp. 434–456.

[2] Dix J., S. Kraus, V.S. Subrahmanian, Temporal agent programs, *Artificial Intelligence* 127 (2001), pp.87–135.

[3] Greer D. S., B. Ludaescher, D. Fils, Ch. Baru, An ontology for integrating stratigraphic databases, *Proc. Denver Annual Meeting*, November 7–10, 2004.

[4] JADE: http://www.jadeworld.com/

[5] Kenneth D. Forbus, Thomas Mostek, Ron Ferguson, An analogy ontology for integrating analogical processing and first-principles reasoning, 2002, American Association for Artificial Intelligence, www.aaai.org.

[6] Kerschberg L., M. Chowdhury, A. Damiano, H. Jeong, S. Mitchell, J. Si, and S. Smith, Knowledge Sifter: Ontology-Driven Search over Heterogeneous Databases, http://eceb.gmu.edu/

[7] Kohler J., M. Lange, R. Hofestadt, S. Schulze-Kremer, Logical and Semantic Database Integration, *Proceedings of the IEEE Symposium Bioinformatics and Biomedical Engineering*, Eds. Danielle C. Young, Arlington, Virginia, USA, 8-10 November 2000.

[8] Laclavík M., Z. Balogh, L. Hluchý, R. Słota, K. Krawczyk, M. Dziewierz, Distributed Knowledge Management based on Software Agents and Ontology. *Proc. 5th Int. Conf. on Parallel Processing and Applied Mathematics PPAM'2003*, (ed. R.Wyrzykowski et.al.), 2004, *LNCS 3019*, Springer-Verlag, pp. 694-699, ISSN 0302-9743, ISBN 3-540-21946-3. September 2003, Czestochowa, Poland.

[9] Laclavík M., Z. Balogh, L. Hluchý, K. Krawczyk, M. Dziewierz, J. Kitowski, M. Majewska, Knowledge Management for Administration Processes, *Proc. of Znalosti 2004*, February 2004, pp.248-255. ISBN 80-248-0456-5.

[10] Lambert S., S. Stringa, G. Viano, J. Kitowski, R. Slota, K. Krawczyk, M. Dziewierz, S. Delaitre, M. B. Oroz, A. C. Gómez, L. Hluchý, Z. Balogh, M. Laclavík, M. S. F. Caparrós, M. Fassone, V. Contursi, Knowledge Management for Organisationally Mobile Public Employees, *Proc. of KMGov 2003*, Rhodes Island, Greece, *LNCS 2645*, Springer-Verlag, pp. 203-212, ISSN 0302-9743, ISBN 3-540-40145-8.

[11] Protégé: http://protege.stanford.edu/

[12] Sebestyénová J., Design and Verification of an Agent-based System, In: *Proc. 8th WSEAS Int. Conference on Computers CSCC 2004*, Athens, July 2004, 6 pages, CD, ISBN: 960-8052-99-8.