KNOWLEDGE MANAGEMENT IN MULTI-AGENT SYSTEMS

Dejan Lavbič, Ana Šaša, Marjan Krisper
Information Systems Laboratory
Faculty of Computer and Information Science
University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
Tel: +386 1 47 68 367; fax: +386 1 47 68 704
e-mail: {Dejan.Lavbic, Ana.Sasa, Marjan.Krisper}@fri.uni-lj.si.si

ABSTRACT

As knowledge management has become an emerging paradigm its application in various domains is being researched. Ontologies as a means of knowledge representation are gaining their importance, even though its practical use is still somewhat limited. We believe that agent-oriented systems, due to their cognitive characteristics, provide a promising approach for knowledge management. On the other hand ontologies and ontology reasoners seem to be a suitable mechanism which would enable agents to efficiently use their knowledge. This could mean a big step forward in the agent-oriented world i.e. possibly a standardized approach to agents' mentalistic notions, which would be particularly useful in agent communication and knowledge sharing. In this paper we present a possible solution to use of ontologies in Multi-Agent System and explain how it was applied in our case study.

1 INTRODUCTION

There is a growing recognition in the business community about the importance of knowledge as a critical resource for organisations. Knowledge management can be defined as a method to simplify and improve the process of sharing, distributing, creating, capturing and understanding knowledge in an organisation. The purpose of knowledge management is to help organisations create, share and use knowledge more effectively, because that causes fewer errors, better decisions, less reinventing of wheels etc.

Ontologies were developed in artificial intelligence to facilitate knowledge sharing and re-use. The reason ontologies are becoming popular is largely due to what they promise: "a shared and common understanding of domain that can be communicated between people and application systems". As such, the use of ontologies and supporting tools offers and opportunity to significantly improve knowledge management capabilities in large organisations.

The purpose of this article is to present knowledge management within Multi-Agent System (MAS) that we developed for facilitating decision support in modern organisations. Case study is from the domain of mobile communications and is ontology based. The primary goal is to provide the knowledge worker an intelligent analysis platform that enhances management process. Thus mechanism for efficient knowledge exchange and management is required to facilitate cooperation between agents in MAS.

The remainder of this paper is organised in the following sections. In section 2 we provide agents and Multi-Agent Systems background. In section 3 we present ontologies and their roles in knowledge management. Then in section 4 we discuss how knowledge management can be addressed within Multi-Agent Systems and present our approach in using ontologies for knowledge sharing purposes. Finally, in section 5, we provide our summary, conclusions and guidelines for future work.

2 AGENTS AND MULTI-AGENT SYSTEMS

Multi-Agent Systems (MAS) are gradually becoming a new paradigm for developing distributed computing systems. This paradigm provides an appropriate architecture for design and implementation of integrative business information systems. Agent-based technology supports complex information systems development by providing natural decomposition, abstraction, and flexibility of management for organizational structure changes [Kishore, 2003; Luck, 2005]. In general, benefits of an agent-based information system include simplification of complex distributed computing, time savings, more and better information, better decisions, improved business processes etc.

The research on intelligent agents and multi-agent systems has been on the rise over the last two decades. The stream of research on business information systems and enterprise integration [Kang, 2003; Tewari, 2003; Yuan, 2003] makes the MAS paradigm a very appropriate platform for integrative decision support within business information systems and knowledge management. Similarities between the agent in the MAS paradigm and the human actor in business organisations in terms of their characteristics and coordination lead us to a conceptualisation where

intelligent agents in MAS are used to represent actors in human organizations.

While there is no universally agreed definition of an agent, the following is most widely accepted: "an agent is a computer system that is situated in some environment, and that is capable of autonomous actions in this environment in order to meet its design objectives" [Wooldridge, 2000]. Furthermore, it has been proposed that an intelligent agent is autonomous, reactive, proactive, and social [Bernon, 2005]. This characterization has also been adopted by AgentLink society that consists of European research groups and partners from industry in the field of agents and multi-agent systems. An agent is different from a traditional object. First of all, agents are commonly modelled using "mentalistic" notions, such as knowledge, belief, intention, obligation, while objects are modelled as simply encapsulating their internal structure as methods and attributes. The degree to which agents and objects are autonomous is quite different. An object does not have control over its behaviours, because it is invoked by others. On the contrary, an agent is able to decide whether or not to execute an action after receiving request.

Whereas the popularity and applications of the agent technology in the business domain has grown over the recent years, the field currently deals with innovative approaches and architectures for solving business and information systems integration problem. There is a lack of unifying framework that would be used for business information systems (ERP, workflow, etc.), the MAS paradigm integration and also provide a foundation for conceptual analysis and modelling of integrative business information systems based on the MAS paradigm. There has been some research progress as mentioned in [Kishore, 2003], but mainly using Object-Oriented techniques and not MAS approach. In this article we propose ontology based solution towards integration of business systems with emphasis on knowledge management.

3 ROLE OF ONTOLOGIES IN KNOWLEDGE MANAGEMENT

Ontologies are increasingly gaining their importance in interoperable systems to capture meanings and relationships of concepts used in various domains. They are used to capture knowledge about some domain of interest, describe the concepts of the domain and also the relationships that hold between those concepts. [Gruber, 1995] defines the ontology as an explicit specification of a conceptualisation of the real-world entities of an application domain. Therefore ontologies are very useful whenever two or more actors have to work together. They play a very important role in our MAS due to collaborative nature of the system, agent-to-agent communication, knowledge management and interoperability reasons that exist between different database systems.

Knowledge representation is very important for information systems in knowledge aware organisations but it is not sufficient by itself. Knowledge has to be understood and furthermore users must know how to use that stored knowledge. To benefit from existing knowledge and to derive new knowledge, inference support is required, where rules play an important role. They are used for defining relationships among concepts of ontology and therefore semantically extend captured data. As depicted in figure 1 ontology construction is a step that can be placed just after the conceptualization of problem domain and before formalization.

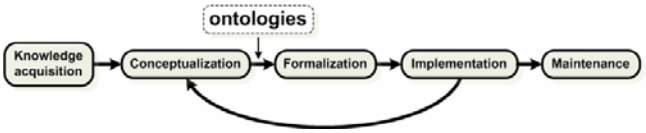

Figure 1: *Application lifecycle with high level of knowledge usage*

While there is no formal methodology for ontology construction, [Noy, 2000] suggests the following steps in ontology development: determine the domain and scope of the ontology, consider reusing existing ontologies, enumerate important terms in the ontology, define the classes and the class hierarchy, define the properties of classes, define restrictions, create instances, check for inconsistencies.

There are several languages available for ontology representation and different ontology languages provide different facilities. The most recent development in standard ontology languages is **OWL**, developed by the World Wide Web Consortium (W3C) [W3C, 2004]. OWL makes it possible to describe concepts but it also provides new facilities. It has a richer set of operators (e.g. and, or and negation) and it is based on a different logical model which makes it possible for concepts to be defined as well as described. Complex concepts can therefore be built up in definitions out of simpler concepts. Furthermore, the logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognise which concepts fit under which definitions. The reasoner can therefore help to maintain the hierarchy correctly. This is particularly useful when dealing with cases where classes can have more than one parent.

OWL ontology consists of **Individuals**, **Properties** and **Classes**. Individuals represent objects in the domain that we are interested in. Individuals are also known as instances and can be referred to as being instances of classes. Properties are binary relations on individuals – i.e. properties link two individuals together. They are also known as roles in description logics and relations in UML and other object oriented notions. In some other formalisms they are called attributes. OWL classes are interpreted as sets that contain individuals. They are described using formal descriptions that state precisely the requirements for membership of the class. Classes may be

organised into a superclass-subclass hierarchy, which is also known as taxonomy. The word concept is sometimes used in place of class, where classes are actually a concrete representation of concepts. We chose OWL for our case study implementation due to high level of semantic expressiveness and wide acceptance of the language in the Semantic Web community.

4 KNOWLEDGE MANAGEMENT IN MULTI-AGENT SYSTEMS

Every agent has to deal with its specific problems and therefore uses the ontology that corresponds to its problem domain. However some intersections of their ontology domains exist which allow agents to cooperate and thus satisfy their common goal. Several issues arise at this point. The first thing to consider is finding the best way to organize the ontology among agents of our MAS to enable an efficient way for them to communicate using the ontologies and of course to maximize performance. The second step is defining the way agents manage the ontologies and reason based on them.

4.1 Organizing ontologies among agents

There are at least four possible solutions to this issue and they will be discussed further in this section.

Every agent has knowledge about its problem domain and directly communicates with another agent whenever it needs information about certain subject other agents might have – the intersecting parts of the ontology they use are passed only by the direct communication between agents. The advantage of this approach is that every agent can decide for itself whether it should trust another agent and request information from the agent it finds the most credible. The major drawback is that in MAS, where a lot of communication is needed among agents, additional message passing can cause reduced performance of the overall system. There are two possible ways of passing the new information to other agent. The first is that an agent simply sends the new piece of information to all agents that might be interested. This is the way that the subscribe protocol works [FIPA, 2001], but it is not always the best option, especially in the case of messages that contain large ontologies. If we presume that the number of all agents interested in (subscribed to) certain subject is N, that the size of a message which contains an ontology is K and that the size of an inform message is I, where $I << K$, the size of all the messages passed in this case would be $N \cdot K$.

The other option is that **an agent only sends an inform message to all the agents which might be interested** (or subscribed) **that some new information on certain subject exists**. Other agents can request the new information when they need to update their knowledge. In this way we avoid sending unnecessary messages to agents which won't actually need it. If N' is the number of agents which update their information, the maximum number of messages passed would be:

- When a certain agent finds an information that might be of interest to other agents it notifies these agents about the change: $N \cdot I$.
- Each agent which at certain moment needs this information would then ask about the change: $N' \cdot I$.
- The first agent would then send the answer to any agent that requested it: $N' \cdot K$.

The size of all messages would thus be: $N \cdot I + N' \cdot I + N' \cdot K$, where $0 \le N' \le N$. Compared to the previous case: $N \cdot I < N \cdot K < 2 \cdot N \cdot I + N \cdot K$, which means that this choice is more appropriate when most of the possible agents will not need the information each time it changes.

Every agent has knowledge about its problem domain, but whenever something new arises about the common knowledge which might be of interest for other agents, it updates the common ontology, which is accessible to all other agents in the system; the common ontology thus comprises an intersection of all of the agents' ontologies.

This is a possible solution to avoid unnecessary message passing. Similarly to the previous case, when an agent finds some information that might be of interest to other agents it simply notifies other agents about the change and writes the change to the common ontology. All the agents that are concerned about this piece of information can thereafter acquire it in the common ontology. The maximum number of messages passed, when some new piece of information is found, is $N \cdot I$, which is better, compared to previous options. Thus agents practically do not need to communicate directly to share knowledge, but only to coordinate and other behavioural issues. There is however one limitation when compared to the previous possibilities. For an agent there is no way of telling whether he can really believe the new information, i.e. in the case its beliefs or knowledge are contradictory to the information in the common ontology. This is not appropriate in the MAS where there are more agents reasoning about the same issue. In that case some other mechanism of coordination is needed.

Every agent has knowledge about its problem domain, but whenever something new arises it writes it in the common ontology; the common ontology thus comprises a union of all of the agents' ontologies. The number and size of messages passed is the same as in the previous case, but the main improvement is that in the case that an agent experiences a failure it can immediately restore its knowledge when restarted. In our case study we are dealing with MAS where every agent reasons about its specific problem domain. Therefore we have chosen the last approach, which is described more in detail in the next section.

4.2 Ontology organization in our case study

Multi-Agent System presented in our case study is from the domain of mobile communications. To depict knowledge management in the environment of various agents our MAS uses heterogeneous systems such as Data Mining Decision Support System [Rupnik, 2005], Data Warehouse and different resource on the Internet.

The global goal that agents in our MAS strive to is supporting decision making process while using several existing systems for business analysis that already exist in organisation and employing information from the environment where organisations resides. A very important element of the environment is the World Wide Web, where agents play information retrieval role for the purpose of decision making. When all the roles that agents play were defined, a problem of coordination among agents arose. The situation is depicted in figure 2 with all agents responsible for distinct part of the system.

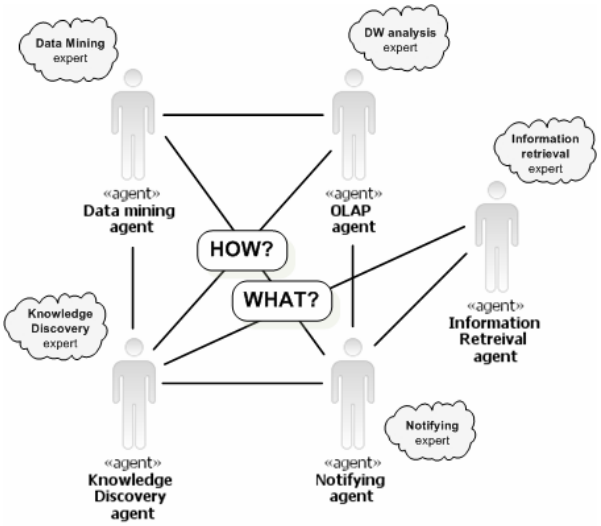

Figure 2: *Knowledge management and sharing between agents in our MAS*

In the view of an individual agent, coordination is not an issue that designer should be concerned about, but that becomes very important in the view of an agent as a part of the whole system [Wooldridge, 2000; Jian, 2005; Jiao, 2006; Soo, 2006]. Important questions like how to ask another agent for help, delegate work to and what is the content of the message requesting some sort of action from an agent that resides in MAS.

Even though the Semantic Web paradigm was created to represent document content, it makes sense to apply this concept to other areas where a common language and carefully designed definitions are necessary in form of common ontologies. In our approach we composed 3 domain ontologies that are based on organisational structuring and are as follows: notifying ontology, information retrieval ontology and data mining and warehousing ontology (see figure 3). In data warehouse area, definitions for figures, dimensions, cubes, reports and their attributes (e.g. data field of the legacy system) are captured in ontology. OLAP agent then, in the name of business user, analyses cubes and in case of major changes drills down to trace reason for deviation.

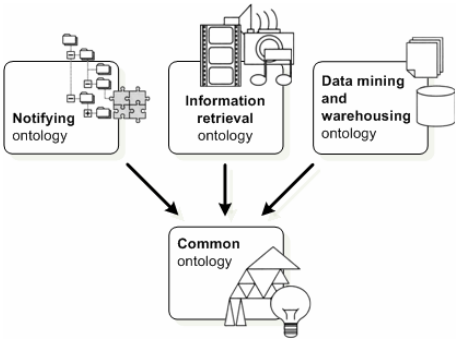

Figure 3: *Developing common ontology from multiple domain-specific ontologies*

Information retrieval ontology mainly deals with World Wide Web concepts and retrieval of information located on web sites. It is concerned with structured and semi structured sources of information and rules for extracting, cleaning and storing information into knowledge base.

The purpose of notifying ontology is to develop architecture for relevant information alocation, transformation and also generating alerts for business users. Elements of context are also presented for defining users needs and requests so that the right information can reach the right user at the right time.

The elements of common ontology, derived from domain ontologies were efficiently used in agent-to-agent communication. This modular approach results in very little effort in adding new agents to MAS and straightforward way of extending capabilities. Using ontologies for knowledge modelling gives an organisation opportunity to separate domain knowledge from single software applications. In doing this, independence is achieved, so that the data can be used as the knowledge basis in other applications within or outside MAS.

This approach is novel in a sense that there is a common ontology for the purpose of interconnection among various agents, where every agent still has its own distinct knowledge model. Thus inference is a concern of every agent that is responsible for its problem domain and when all the common facts (predefined or derived) are assembled, the inference on the highest level is conducted with Knowledge Discovery Agent.

4.3 Managing ontologies

Currently the most suitable choice for development of Multi-Agent Systems is JADE due to its popularity and support. JADE is a software framework fully implemented in Java language. It simplifies the implementation of Multi-Agent Systems through a middle-ware compliant with the

FIPA specifications and through a set of tools that supports the debugging and deployment phases [TILAB, 2000].

Therefore the next step to consider is how agents, written in Java, will use the OWL ontologies. During the past years there has already been research work devoted to management of OWL ontologies using the Java language. Two main approaches can be distinguished, both with a number of support tools:

- **Generating Java code from an ontology**. Examples of this concept are Protégé Bean Generator plug-in [Aart, 2002] and OWLBeans toolkit [Tomaiuolo, 2004]. They both provide fairly similar functionality, which is quite reduced compared to OWL. Bean Generator is implemented as a Protégé plug-in, whereas OWLBeans is a separate toolkit, that supports some additional functionalities such as reading OWL ontology and converting it to Java language for example.
- **Direct access to OWL ontology**. Examples are Protégé-OWL API [Stanford, 2004] and the OWL API of the JENA framework [HP, 2000]. They provide classes and methods not only to load or save OWL files, to query and manipulate OWL data models, but also to perform reasoning.

Mapping OWL ontology to Java has several advantages [Kalyanpur, 2004]:

- The Java API generated from ontology (schema) can be used to readily build applications (or agents) whose functionality is consistent with the design-stage specifications defined in the schema.
- The use of any Java IDE to debug (or customize) the application or ontology easily.
- The use of JavaDoc to generate an online documentation of the ontology automatically.

As three main concepts in OWL are classes, individuals and properties, at first glance OWL may not seem to be all that different from the object-oriented data model. In OWL, individuals are instances of classes as are in object-oriented data model objects instances of classes. Properties describe classes in more detail as do attributes which provide data belonging to a class. However the mapping of OWL ontology to an object-oriented representation is not straightforward. There are numerous problems which need to be taken into account due to the differences between the object-oriented representation and OWL ontologies. Here we will state only a few of them, a more exhaustive list with detailed descriptions can be found in [Tomaiuolo, 2004].

An object in Java is an instance of only one class. In OWL an individual can be an instance of multiple classes. Moreover in OWL, classes are assumed to overlap if not explicitly stated that they are disjoint. Furthermore they can be mutually disjoint or the contrary, they can be complete (their union completely covers another, more general, class). The first requirement can be achieved fairly simply, for example with an object of a class with several ancestors, whereas disjointness and completeness are very difficult to express.

In object-oriented data model there is no explicit notion for an OWL property. An OWL property can be an Object property, which indicates a relationship between individuals, or a Datatype property, which links an individual to an XML Schema Datatype value or an RDF literal. This can be achieved to a certain extent with attributes and variables, but in search of an appropriate mapping we should also consider support for other property characteristics, such as hierarchies, symmetry, transitivity, equivalence and inversion.

Thus if we decide for the option of generating Java code from an OWL ontology a compromise between losing some information in translation and increased complexity of implementation has to be considered. This is not appropriate if all the system relies only on this compromise, whereas it is sufficient and also the most reasonable option in certain parts of the system:

- **Agents' reasoning**: An agent should have a support for reasoning which is not too complex but is still complex enough for an agent to achieve its goals.
- **Communication between agents**: Even if it does not support all the functionality of the OWL ontology, it is sufficient for the purpose of communication between agents. As explained earlier agents exchange only the basic information. For example, instead of sending complex messages an agent only sends a notification with a description of the new information, whereas the complete information can be found in the common ontology if needed.

On the other hand each agent works only with its own ontology, which means that even if the generated Java code would enable full OWL functionality, reasoning on the overall ontology might not be achieved – certain inferences might be left out, because the union of the inferences made by all agents does not necessarily cover the inferences that could have been made, based on all the knowledge base.

Therefore some other mechanism for reasoning on the overall ontology is needed to accomplish this missing part and also to provide reasoning based on full OWL functionality. Considering the possibilities and existing tools, an API with a programmatic interface for directly accessing the ontology and communication with a reasoner is the most appropriate choice. Thus a complete support of OWL ontologies is ensured for the common ontology, which meets both requirements.

For construction of our ontology we used Protégé [Stanford, 2004], since it is currently one of the most powerful and widespread tools for this purpose. For OWL to Java mapping we use Bean Generator, which is a reasonable choice to use with the Protégé. It provides the basic transformation of OWL to Java with the expected reduced functionality. To access the common ontology we use the OWL API of the JENA framework. JENA is chosen mainly due to integrated rule and inference engine that can be directly used in manipulation of ontology.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a practical solution to an efficient use of OWL ontologies in multi-agent systems. We have pointed out several possibilities for distributing ontology of the overall system among its agents and their main advantages and drawbacks, which indicate the kind of a system where each one of them should be used. Furthermore we have proposed an implementation of such a system based on some existing tools.

Our current work is focused on testing the presented approach and trying to find possible improvements, especially in augmenting the functionality of translated OWL ontologies to the Java language and to adapt the agents' reasoner efficiently to it. Our aim is that this solution would work as a JADE add-on, which would facilitate and possibly expand its use.

References

1. **[Kishore, 2003]** Kishore, R., Zhang, H., Ramesh, R.: Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems, *Decision Support Systems*, Elsevier (2003)
2. **[Kang, 2003]** Kang. N., Han. S.: Agent based e-marketplace system for more fair and efficient transaction, *Decision Support Systems, Vol. 34*, Elsevier (2003), 157–165
3. **[Tewari, 2003]** Tewari, G., Youll, J., Maes, P.: Personalized location-based brokering using an agent-based intermediary architecture, *Decision Support Systems, Vol. 34*, Elsevier (2003), 127–137
4. **[Yuan, 2003]** Yuan, S.T.: A personalized and integrative comparison – shopping engine and its applications, *Decision Support Systems, Vol. 34*, Elsevier (2003), 139–156
5. **[Wooldridge, 2000]** Wooldridge, M.: An Introduction to Multi-Agent Systems, *Wiley Publishing*, Chichester, England (2000)
6. **[Gruber, 1995]** Gruber, T.: Towards principles for design of ontologies used for knowledge sharning, *International Journal of Human-Computer Studies, Vol. 43*, Elsevier (1995), 907–928
7. **[Noy, 2000]** Noy, N. F., McGuinness, D. L.: Ontology Development 101: A Guide to Creating Your First Ontology, Stanford, http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html, 2000
8. **[W3C, 2004]** W3C: OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features, 2004
9. **[Lavbič, 2004]** Lavbič, D.: Uporaba inteligentnih agentov, *Diploma thesis*, University of Ljubljana, Faculty of Computer and Information Science, 2004.
10. **[Rupnik, 2005]** Rupnik, R., Krisper, M.: Aplikativni sistemi odkrivanja zakonitosti v podatkih kot nov tip sistemov za podporo odločanju v informacijskih sistemih, *Uporabna informatika, Volume 13, Issue 2*, April/May/June 2005, pp. 61–73
11. **[TILAB, 2000]** Telecom Italia Lab: Java Agent DEvelopment Framework, http://jade.tilab.com
12. **[FIPA, 2001]** FIPA: Foundation for Intelligent Physical Agents: Subscribe Protocol Interaction Specification, http://www.fipa.org/specs/fipa00035/DC00035E.pdf
13. **[Aart, 2002]** Chris van Aart: BeanGenerator plug-in for Protégé, http://acklin.nl/page.php?id=34
14. **[Šaša, 2005]** Šaša, A.: Modeliranje večagenetnih sistemov, *Diploma thesis*, University of Ljubljana, Faculty of Computer and Information Science, 2005.
15. **[Kalyanpur, 2004]** A. Kalyanpur, D. Pastor, S. Battle, J. Padget: Automatic mapping of owl ontologies into java. *In proceedings of Software Engineering and Knowledge Engineering Conference, SEKE 2004*, Banff, Canada
16. **[HP, 2000]** Hewlett-Packard Development Company, Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net
17. **[Stanford, 2004]** Stanford University of School of Medicin, Stanford Medical Informatics, Protege-OWL, http://protege.stanford.edu/overview/protege-owl.html
18. **[Tomaiuolo, 2004]** Tomaiuolo, M., Turci, P., Bergenti, F., Poggi, A.: An Ontology Support for Semantic Aware Agents, *7th International Workshop on Agent-Oriented Information Systems (AOIS), Fourth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2004.
19. **[Bernon, 2005]** Bernon, C., Cossentino, M., Pavón, J.: An Overview of Current Trends in European AOSE Research, *Informatica, Vol. 29*, The Slovenian Society Informatika (2005)
20. **[Bordini, 2006]** Bordini, R. H., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-Sanz, J. J., Leite, J., O'Hare, G., Pokahr, A., Ricci, A.: A Survey of Programming Languages and Platforms for Multi-Agent Systems, *Informatica, Vol. 30*, The Slovenian Society Informatika (2006)
21. **[Luck, 2005]** Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology: Computing as Interaction, A Roadmap for Agent Based Computing, http://www.agentlink.org, AgentLink III (2005)
22. **[Jiao, 2006]** Jiao, J., You, X., Kumar, A.: An agent-based framework for collaborative negotiation in the global manufacturing supply chain network, *Robotics and Computer-Integrated Manufacturing, Vol. 22*, Elsevier (2006), 239–255
23. **[Soo, 2006]** Soo, V. W., Lin, S. Y., Yang, S. Y., Lin, S. N., Cheng, S. L.: A cooperative multi-agent platform for invention based on patent document analysis and ontology, *Expert Systems with Applications, Article in Press*, Elsevier (2006)
24. **[Jian, 2005]** Jiang, Y. C., Jiang, J. C.: A multi-agent coordination model for the variation of underlying

network topology, *Expert Systems with Applications, Vol. 29*, Elsevier (2005), 372–382