

## AGENTOWL: SEMANTIC KNOWLEDGE MODEL AND AGENT ARCHITECTURE

Michal LAČLAVÍK, Zoltán BALOGH, Marián BABÍK,  
Ladislav HLUCHÝ

*Institute of Informatics  
Slovak Academy of Sciences  
Dúbravská cesta 9  
845 07 Bratislava, Slovakia  
e-mail: [michal.laclavik@savba.sk](mailto:michal.laclavik@savba.sk)*

Manuscript received 6 May 2006; revised 5 September 2006  
Communicated by Jacek Kitowski

**Abstract.** MAS is a powerful paradigm in nowadays distributed systems, however its disadvantage is that it lacks the interconnection with semantic web standards such as OWL. The aim of this article is to present a semantic knowledge model of an agent suitable for discrete environments as well as implementation and a use of such model using the Jena semantic web library and the JADE agent system. The developed library allows interconnection of Agent and Semantic Web technologies can be used in an agent based application where such interconnection is needed. The defined model and methodology show the use of the library in knowledge management applications where the proposed model has been used and evaluated in the scope of the Pellucid and K-Wf Grid IST projects.

**Keywords:** Semantics, agent, architecture, knowledge model

### 1 INTRODUCTION

Multi-Agent Systems (MAS) is a powerful paradigm [1, 2, 3] for distributed heterogeneous information systems when representation and reasoning using knowledge is needed. At present, MAS lacks interconnection with current commercial technological standards and the results of semantic web research [1]. The MAS needs to incorporate results of many areas of computer sciences such as artificial intelligence,

or the knowledge management. In the MAS, knowledge is usually represented by states, rules or predicate logic [8, 7]. Although this is extremely powerful, it is hard to capture knowledge from a person or from current information systems in the form of rules or predicate logic clauses. Moreover, another difficulty is to present information and knowledge expressed in e.g. predicate logic to the end user [17]. Ontology as understood in the Semantic Web is closer to current information systems. It is based on XML/RDF [20, 19, 18], which can be more easily captured or returned from/to a person and existing information systems because information systems usually have XML based interfaces and XML can be easier presented to user after transformations [21, 22, 32]. The MAS architectures, Belief Desire Intention (BDI) [23], Behavioral [5, 23] or FIPA [24], have a representation of the knowledge model. In BDI [23], architecture's "belief" represents the knowledge model. In the Behavioral architecture, knowledge is hidden in variables and algorithm states or can be represented by additional mechanisms (rules, ontology etc.) FIPA [24] describes the knowledge model based on ontology, but leaves internal agent memory model, its manipulation and understanding of design of agent on developer's decisions. In addition, FIPA defines a knowledge manipulation based on content languages such as FIPA-SL [8], FIPA-KIF, which are powerful but lack any interconnection with commercial tools and standards. For example, the JADE agent system supports FIPA-SL language for message passing, but no FIPA-SL query engine or repository of such knowledge model is available. This is why we see ontology description from the semantic web area (DAML+OIL [19], OWL [18]) to be more appropriate for real application. Furthermore, FIPA defines FIPA-RDF, but again only concerning a message structure. For these reasons all implementations of FIPA architecture (current MAS standard) are weak and not suitable for developers who want to build their knowledge management systems using software agents. Therefore we have decided to integrate semantic web technologies into MAS and create architecture, methodology and software for such integration. In addition, we have developed a generic ontology suitable for representing an agent knowledge model for applications in fully observable environment, which can be described by discrete events. This model can be further extended and thus used in many areas especially for the knowledge and experience management [14, 17, 31].

Structure of the article is divided into 6 chapters:

**Introduction:** Explains the state of the art and needs for such work.

**Agent Knowledge Model:** Describes the proposed agent knowledge model using description logic compatible with OWL-DL.

**Modeling Methodology for Agent Design:** Explains a modeling methodology based on CommonKADS and Protégé ontology editor for use of such model in specific applications.

**Design and Specification of Agent Software Library:** describes developed library based on Jena, which supports such model in the JADE agent system

**Demonstration:** Shows a use of the agent model, methodology and library on a simple agent example.

**Conclusion:** concludes results of the presented work.

## 1.1 State of the Art

The agent architectures are fundamental engines underlying the autonomous components that support effective behavior in the real-world, dynamic and open environments. Many architecture typologies exist [23], the following basic types of agent architectures are considered in this work:

- Reactive Architecture
- Belief Desire Intention Architecture – BDI
- Behavioral Architecture.

In literature, the main focus is on the externals of the agents, their communication with environment and other agents. The internal knowledge model is left for an agent creator. Several tools allow creation of the BDI based agents [23]; however, these are sufficient for some simulations and tests only, definitely not for any real system. The FIPA does not cover this area of agent systems either. The FIPA specifications just describe how agents should communicate and how they can share, translate or communicate ontologies. In the FIPA compliant implementations of an agent system, different approaches for building the agent knowledge model can be found. The most advanced, but not a sufficient approach is in the JADE [5] agent system. The JADE support for ontologies or agent knowledge modeling [7] is based on the Java classes. The JADE agent model is not sufficient in several ways. Such model cannot support features of semantic ontology representations such as OWL, does not have a query engine for the FIPA-SL language [8] and a model based on predicate logic is hard to be communicated with a user and existing commercial systems. Thus, we have created an RDF/OWL based agent memory model. The agent communication based on the RDF/OWL and FIPA based agent system was presented in other works such as [9], but this work lacks a generic internal model, offering only a theory how RDF/OWL can be used for the agent communication. In this work we considered parts of those two models and extended them with the event based memory model. The event based model was previously used in many areas, but in the agent field only the work of Anderson [10] deals with the event model as typical reactive architecture. Another approach for the event based model is in Russell [4], where action – situation pairs produce events similar to those in our model. Russell's model is well described and more general than the proposed model but in practice its implementation is quite complicated. Our model is suitable for applications where an agent needs to search for information or knowledge in environment evolving in time. We think such model is useful for searching agents or agents in the knowledge management application. In the proposed model, events are used to take action or to have a reference of past actions, which provide us with a history

of an environment at any moment. History events can be preprocessed at any time enabling achievement of different results with the same knowledge model if we find out that processing algorithms created by an agent developer are not sufficient and need to be refined.

Figure 1 represents existing and missing features of FIPA compliant MAS. Dashed lines represent missing connection with GUI, external knowledge bases, external systems existing in an organization, ACL communication based on OWL and SPARQL content languages as well as the generic model based on OWL, which can be used for pool of applications. Such dashed connections as well as the generic extensible knowledge model suitable for discrete environments are covered in the architecture presented in the paper.

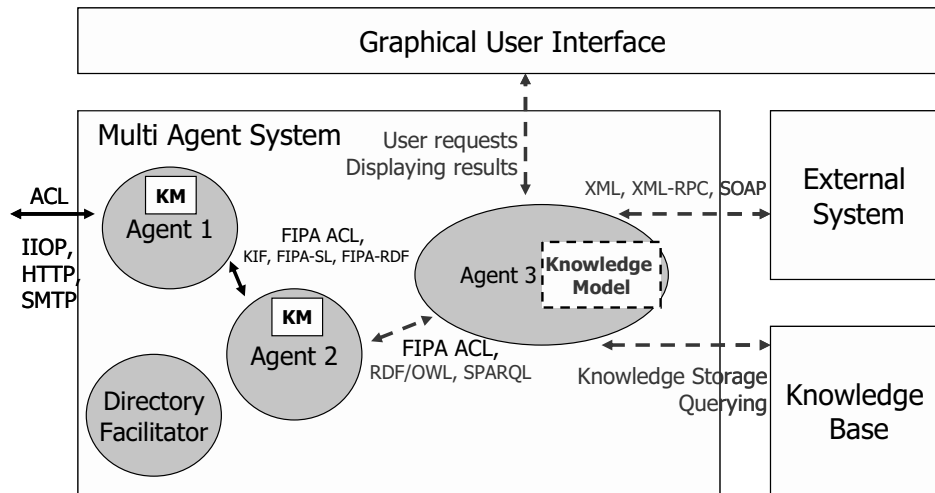


Fig. 1. Existing and missing features of agent architecture

According to the Agent Technology Roadmap [1], the result of AgentLink II [11] community, there are a number of broad technological challenges for research and development over the next decade in the agent technology. Within the presented work we are trying to partially cover the following challenges:

- Providing better semantic infrastructure (ontologies, knowledge models)
- Apply basic principles of software and knowledge engineering
- Make stronger connection between the MAS and existing commercial technologies.

## 2 AGENT KNOWLEDGE MODEL

Most of the agent architectures are combinations of basic architecture types – the so called hybrid architectures. We have focused on behavioral architecture, where

an agent memory model used in behaviors implementations was created within this work. The proposed Knowledge Model is generic and suitable for applications with discrete, fully observable environments where environment changes can be captured by discrete events. Such model is suitable only in applications where environment can be modeled by discrete events in time and context, contrary to Russell's model [4] which is more general but too complex to implement. Such applications are for example the information search agents or the agents in knowledge management applications where we tested the proposed model. The motivation for such model came from satisfying different information and knowledge management needs in organizations. Usually in an organization discrete events can be captured as defined in the model such as e-mails received, documents created, documents opened, activities finished or e-mails sent. An analysis of such events can provide useful information and knowledge about organization processes, documents or communication flows. The use of agents to reason on such model is essential if we deal with large distributed or virtual organizations in heterogeneous environment.

In this chapter we will describe the agent memory formally using description logic [16] and subsequently we will move to modeling and implementation using RDF/OWL. The model is based on events, while the idea is taken partially from the JADE ontology model as well as Russell's model [4]. As we described, the JADE ontology model [7] has some limitations.

## 2.1 Formal Description of Model Using Description Logic

The model is based on five main elements: Resources, Actions, Actors, Context and Events. Figure 2 shows a formal graph representation using the same terms as the model described in this chapter, where we describe the knowledge model using description logic [16] compatible with OWL-DL [18]. In the proposed model we expect that all agents share same ontology.

The *Resource* class stands for all the resources in the agent environment. Many subclasses representing resource types can be defined as part of customization of the model. Important subclass of *Resource* is *Actor*.

$$\begin{aligned} Actor &\subseteq Resource \\ \{actor\} &\in Actor \end{aligned} \tag{1}$$

The *Actor* class denotes actors in the environment. *Actor* individuals can take actions  $\{action\}$ , which are individuals of the *Action* class.

$$\{action\} \in Action \tag{2}$$

The *Task* class symbolizes done or to be done tasks in the environment. Depending on application it can represent problems too. The *Domain* class stands for application domain extension of ontology, while all extensions should be subclasses of this class. *Context* class represents context of actors, environment or else.

$$\begin{aligned}
&Resource \subseteq Context \\
&Action \subseteq Context \\
&Domain \subseteq Context \\
&\{domain\} \in Domain \\
&Task \subseteq Context \\
&Context \supset Resource \cup Action \cup Domain \cup Task \\
&\{context\} \in Context
\end{aligned} \tag{3}$$

An important property of the *Task* is a *domain*. This symbolizes domain related application concepts. It was identified that such connection is useful for setting appropriate knowledge management algorithms for the actor context and resource updating.

$$Task \subseteq domain.Task(domain) \cap Context \tag{4}$$

The *Event* class represents events in the system. The *Event* individual  $\{event\}$  is an  $\{action\}$  taken by an  $\{actor\}$  on particular  $\{resource\}$  in the situation described by the  $\{context\}$ . The properties of the *Event* class are *context.Event*, *resource.Event*, *action.Event*, *actor.Event*.

$$\begin{aligned}
&Event \subseteq \\
&\quad action.Event(\{action\}) \cap \\
&\quad resource.Event(\{resource\}) \cap \\
&\quad actor.Event(\{actor\}) \cap \\
&\quad context.Event(\{context\}) \\
&\{event\} \in Event
\end{aligned} \tag{5}$$

A special type of the *Actor* is the *Agent*. The *Agent* is used for a software agent representation in the system.

$$\begin{aligned}
&Agent \subseteq Actor \\
&\{agent\} \in Agent
\end{aligned} \tag{6}$$

Typical actions which can be performed by software agents are defined. They represent types of inter-agent communication such as the ACL QUERY-REF and ACL INFORM message [25]. When communication between agents is performed, events of such kind are generated.

$$\{aQuery, aInform\} \in Action \tag{7}$$

When actions such as creating, updating or deleting of resources are performed, events containing those kinds of action are stored and evaluated in the system.

$$\{aUpdate, aDelete, aCreate\} \in Action \tag{8}$$

The *Actor* class consists of important properties: *context.Actor*, *resource.Actor*.

$$\begin{aligned}
 Actor &\subseteq \\
 &resource.Actor(\{resource\}) \cap \\
 &context.Actor(\{actor\}) \cap \\
 &Resource
 \end{aligned} \tag{9}$$

The *context.Actor* represents actor's current context. The system or application environment is based on stored events. The events model the environment state. Current state of the environment or actor related environment/context is thus affected by relevant new events.

$$\begin{aligned}
 contextActor(\{actor\}) = \\
 f_C(\forall event; actor.Event(\{actor\}) \in \{event\})
 \end{aligned} \tag{10}$$

The *resource.Actor* property stands for all current resources of the actor. These resources are results of actors' intentions or objectives. Such resources are dependable on current actors' environment state/context (*context.Actor*).

$$\begin{aligned}
 resource.Actor(resource) = \\
 f_R(contextActor(\{actor\}))
 \end{aligned} \tag{11}$$

Functions/algorithms for context and resource updating are specified by (10) and (11). An advantage of such model is that it enables to achieve better results when such algorithms are changed in the future, using the same model and data. Due to storing all events we can model the environment at any moment from the past and process it later from any starting point with improved algorithms for context and resource updating. In addition, this model can be successfully used outside the MAS [14]. It can be used in knowledge intensive applications, where we need to model actors and their knowledge model. Often this is the case when we need to model users of the system who are mostly the main actors in any application.

### 3 MODELING AND DEVELOPMENT METHODOLOGY FOR AGENT DESIGN

Since this model is suitable mainly for knowledge management applications, we tried to define modeling methodology according to methodologies used in the knowledge management. Thus the methodology follows the CommonKADS [26] which was developed and used for the knowledge management analysis and modeling. However, CommonKADS is not tied to any modeling tool, knowledge representation or ontologies. CommonKADS is divided into two main parts:

- a knowledge model based on other three models:
  - an organizational or environmental model
  - an agent model
  - a task model
- and design of the system. In our case the design is based on the MAS.

We present a knowledge model based on the OWL ontology [18] and we model it in the Protégé ontology editor [27]. Thus defined methodology for the knowledge model is similar to [12]. Ontology modeled with Protégé reflects the CommonKADS models and has several commonalities with the JADE ontology model. Design of the system is based on UML, AUML [28] and MAScommonKADS [29]. Using this methodology can bring good results only after several iterations of the modeling process, which should be performed after the first developing, use and evaluation of the first system version. The iteration method is common and used in most knowledge management methodologies.

### 3.1 Specification of Method for Creating Ontology and Agents Model

When modeling a knowledge model for an application we follow the first three CommonKADS models:

- The Organizational or in our case the Environment Model
- The Task Model
- The Agent or Actor Model.

When modeling the knowledge model we have to extend the generic agent model (Figure 2) by new elements and relations. This model is the same as that described using description logic in the previous chapter.

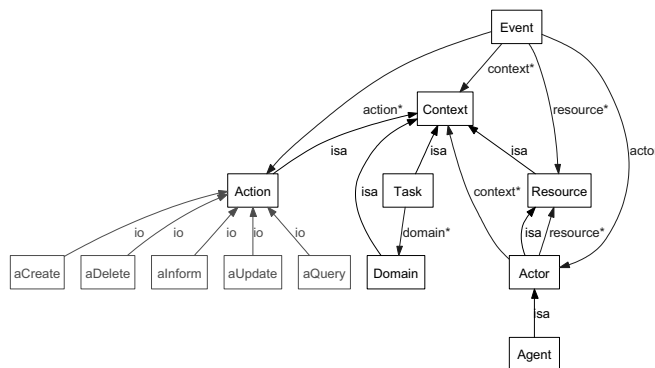


Fig. 2. Basic ontology for knowledge modeling

**The Environment Model** models the environment of application and that of actors. Thus, modeling a resource element needs to be extended by new types of resources such as documents, contacts, goods or services. A domain element needs to be extended by all application/domain specific concepts (possibly including resources), which model a problem domain, especially concepts used by actors to make decisions or accomplish a task (e.g. documents, contacts in administration applications). The resources considered as results of actor's goals have to be modeled as well. Usually results of actors' goals can be considered as resources e.g. in searching agents (found documents) or selling agents (sold goods).

**The Actor Model** models actors performing tasks and actions. The actor can be a human, a software agent or another entity (e.g. information system) performing actions and which can be monitored by the system. An important part of the actor model is actor's context which defines current actor's environment and actor's resources which are the results of accomplished goals. Thus, the Actor model includes definitions of 2 algorithms for the actor context (10) and resources updating (11) defined in the actor/agent model.

**The Task Model** models tasks, activities, processes and actions relevant to actors. An important property of a task is a Domain. Tasks are often related to some resources or other domain entities and we need this relation to define functions for updating the actor context and the resources mentioned above.

Defining ontology of mentioned models should be followed by iterative refining in order to include all needed elements in the model. Refining the model requires consideration whether combination of actions, resources and actors could be captured as events and whether all possible events could be created from defined ontology elements. The outcome of models is the knowledge model consisting of:

- Ontology developed in Protégé in the OWL format.
- Algorithms for each agent (actor) updating actors' context  $f_C$  (10) and resources  $f_R$  (11). Often such algorithms are similar or the same for all actors.

### 3.2 Design of Agent Based System with the Presented Model

The design of the system should be based on three UML diagram types, similarly to those in the object oriented programming and AUML [28], namely:

- Use Case Diagram
- Sequence Diagram
- Class Diagram.

**The Use Case Diagram:** creating the Use Case Diagrams for the MAS, we identified that it is important to design a diagram for each agent type, where the agent is understood as system boundaries of the use case. The number of agents

should be clear from the overall knowledge model of the system especially from the Actor model.

**The Sequence Diagram:** similar to the OOP sequence diagram. It represents communication among agents and interaction with external systems (GUI, sensors or other interfaces). Interactions among agents explain the type of the ACL message (e.g. Inform(), Query-ref()). An additional table describing each message type can be attached to a diagram.

**The Class Diagram:** represents design of each agent type. The main difference in comparison with the OOP UML class diagram is that all agent behaviors need to be described. Behaviors are considered as something owned by agents and thus should be displayed similarly as methods in the agent class. More details with examples of such UML diagrams can be found in [17].

#### 4 DESIGN AND SPECIFICATION OF AGENTOWL LIBRARY

In this chapter we provide description of developed AgentOWL software library to be used for creation of agents with the OWL knowledge model. The developed library is based on the JADE agent system [5] and the Jena semantic web library [30]. It covers functionalities that we identified as missing in current agent architectures such as JADE. The library can be used independently of the model and development methodology described in previous chapters. It can be used in any agent based application where interconnection with semantic web and commercial technologies is required. Missing components are as follows:

- Agent Knowledge Model based on RDF/OWL
- Action-resource pair (bases of events) as basics for communication included in OWL ontology model
- Sending ACL based RDF/OWL messages
- Receiving ACL based RDF/OWL messages
- Including received information (events) into a model
- XMLRPC receiving messages
- XMLRPC returning RDF/OWL and plain XML
- Inference Engine
- SPARQL messages handling.

The library consists of the following classes:

**The class Ontology** has basic constants related to OWL ontology and the used ontology.

**The class Memory** has the functionality to load, store and manipulate the agent memory based on an RDF or an OWL model. The Jena library is used for this manipulation. The memory can load an OWL file developed in Protégé which

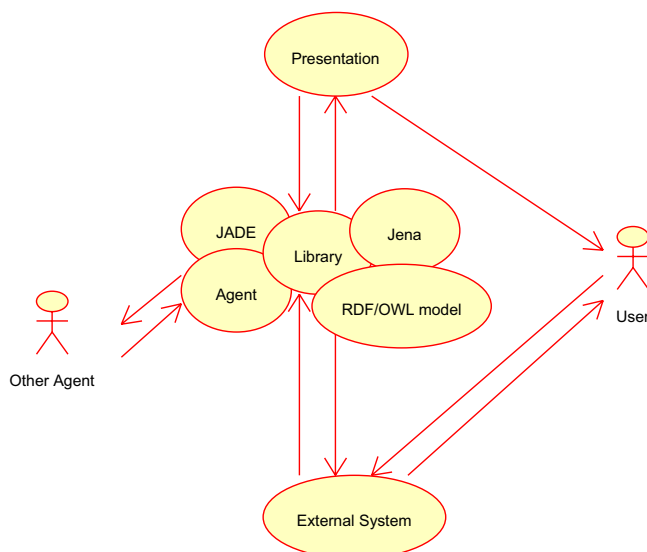


Fig. 3. Agent library functionality diagram

becomes an ontology model of the agent memory. The memory can be stored in a database backend such as MySQL, other RDBS supported by Jena or into the OWL file. When using a database model, an agent can migrate between nodes and does not need to carry its memory with itself. It just needs to disconnect from memory and connect after execution starts on a different node if a use of mobile agents is required.

**The class Message** contains static methods for creating ACL messages or transforming models to the RDF or XML strings. Some of those methods can be used for XML-RPC communication when wrapped with the XML-RPC server. Figure 3 shows an agent library functionality diagram of using this library. The JADE based agent can be developed using this library to support Jena OWL model as the agent memory, and furthermore it is possible to include the XML-RPC based functionalities for presenting knowledge or receiving events from external systems as RDF messages based on a used ontology. Furthermore it allows communication among agents based on RDF/OWL as FIPA-ACL content language.

The developed library is published on the JADE website as the third party software [13, 33] as a way to put together Jena RDF/OWL and JADE Multi-agent system features. The developed library is quite popular and in 2005–2006 it was downloaded 735 times.

More details with demonstration examples can be found in [17] or the on library website [33].

## 5 DEMONSTRATION

This demo shows a use of the developed agent library. We created a simple ontology in Protégé. In the demo we developed two agents:

- AskAgent
- AnswerAgent.

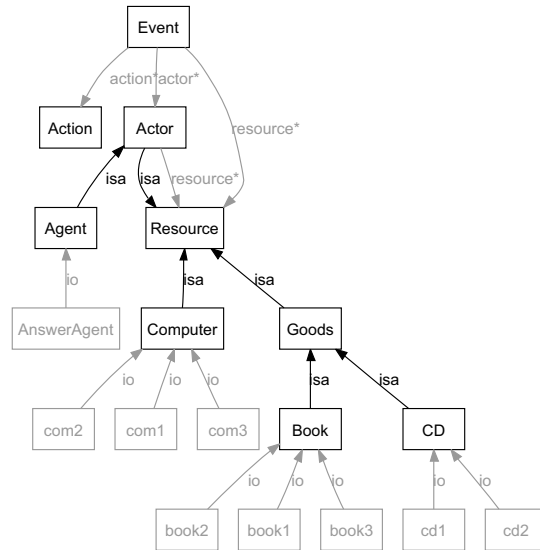


Fig. 4. The Agent model of the AnswerAgent. The AskAgent uses the same model with the only difference that it does not have any resource individuals.

Both agents use the same ontology, but with different individuals. In Figure 4 you can see the model of the AnswerAgent. The AskAgent uses the same ontology with only one individual “AskAgent”. The AskAgent knows different resource types such as books, CDs or computers, but its memory does not contain any individual of that kind. The AnswerAgent memory with different resource individuals can be seen in Figure 4. For example, the model contains 3 computer individuals.

A user selects a type of resource for which the AskAgent should search. The user selects a computer in the AskAgent GUI (Figure 5) and clicks the search button. The type of the resource is passed to an agent by the XML-RPC method call. One of the AskAgent behaviors is activated and the AskAgent produces the SPARQL query and passes the ACL QUERY message to the AnswerAgent.

```
(QUERY-REF
 :sender ( agent-identifier
```

```

    :name AskAgent@nb.laclavik.sk:1099/JADE
    :X-JADE-agent-classname
      agent.demo.askAgent.AskAgent )
:receiver (set ( agent-identifier
  :name AnswerAgent@nb.laclavik.sk:1099/JADE ) )
:content "
  PREFIX ont: <http://onto.ui.sav.sk/agents.owl#>
  PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  SELECT ?x WHERE {?x rdf:type ont:Computer}"
:language SPARQL
:ontology http://onto.ui.sav.sk/agents.owl
)

```

The AskAgent asks the AnswerAgent to return to it all computers it has in the memory. The AnswerAgent receives an ACL QUERY message and performs an SPARQL query on its memory. The result is passed as several ACL INFORM messages consisting of the RDF description of requested resource.

```

(INFORM
:sender ( agent-identifier
  :name AnswerAgent@nb.laclavik.sk:1099/JADE
  :X-JADE-agent-classname
    agent.demo.answerAgent.AnswerAgent )
:receiver (set ( agent-identifier
  :name AskAgent@nb.laclavik.sk:1099/JADE ) )
:content "
<rdf:RDF
  xmlns:j.0="http://onto.ui.sav.sk/agents.owl#"
  xmlns:rdf=
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://onto.ui.sav.sk/agents.owl">
  <owl:Class rdf:about="Computer">
    <rdfs:subClassOf rdf:resource="Resource"/>
  </owl:Class>
  <Computer rdf:about="com1">
    <title>Notebook Sony Vaio</title >
  </Computer>
</rdf:RDF>"
:language http://www.w3.org/1999/02/22-rdf-syntax-ns#
:ontology http://onto.ui.sav.sk/agents.owl
)

```

The AnswerAgent creates events in its memory that resources were sent to AskAgent. This way the AnswerAgent keeps information about the environment. When the AskAgent receives an ACL INFORM message, it stores its context into its memory model. Events about receiving resources are created in the AskAgent memory. In addition, it adds references to returned resources to the AskAgent individual resource property. When a user clicks on the “getXML” button (Figure 5) in the AskAgent GUI, the AskAgent individual from the AskAgent memory is returned in the XML format. In this XML you can see resources, which are now in the AskAgent memory.

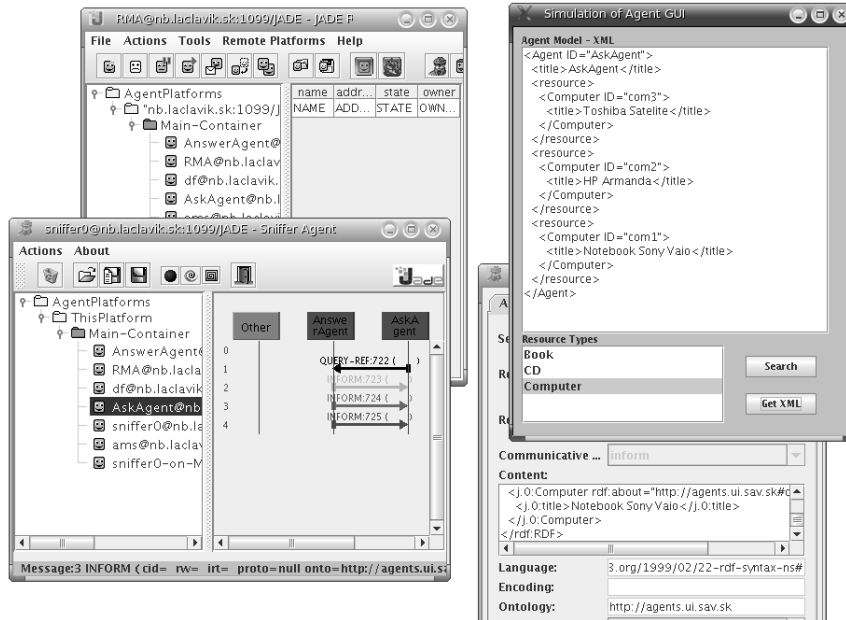


Fig. 5. Screenshot of running demo (top left: JADE window, bottom left: JADE Sniffer Agent, top right: AskAgent GUI, bottom right: ACL Inform Message)

```
<Agent ID="AskAgent">
  <resource>
    <Computer ID="com3">
      <title>Toshiba Satellite</title>
    </Computer>
  </resource>
  <resource>
    <Computer ID="com2">
      <title>HP Armanda</title>
    </Computer>
  </resource>
</Agent>
```

```
</resource>
<resource>
  <Computer ID="com1">
    <title>Notebook Sony Vaio</title>
  </Computer>
</resource>
<title>AskAgent</title>
</Agent>
```

XML that returned to GUI is plain XML differing from RDF/OWL based XML which can be gathered from a Jena Model. It is not possible to parse RDF/OWL based XML using XSL templates due to RDF resource referencing, while the AgentOWL library XML output of a Jena Model can be processed using XSL templates. Such plain XML can be then passed and showed in a different format in GUI. It would be better to develop GUI based on JSP and XSLT, where it is possible to transform returned XML to HTML. This option is easier for users to install, run and see the results, because to run the demo you just need Java installed. We created such GUI window for this demo purposes only. The solution adopting the XSL approach [14, 22] was created and used in the Pellucid and K-Wf Grid projects.

The demo together with the library is published on the JADE website as third party software [13, 33] as a way to put together Jena and JADE features.

## 6 CONCLUSION

The paper describes how semantic web technologies can be applied in multi-agent systems. An agent knowledge model was created enabling the possibility to model the agent environment, agent context and its results. The agent library to support such an agent knowledge model was developed and extends the JADE agent system, which is currently the most popular MAS toolkit. The model is extensible for the knowledge and experience management [14, 17, 31]. It was proved that such model and the agent architecture are implementable and can be used in different knowledge intensive applications in discrete environments. The approach was successfully used and evaluated in several R & D projects, mainly in the Pellucid IST project [6] (A Platform for Organisationally Mobile Public Employees EU 5FP IST-2001-34519). The developed library for using such semantic model was published on the JADE agent website and it is available to the worldwide JADE community.

The presented work will be further developed and extended. Such process has already been started and we are trying to apply the developed model, methodology and infrastructure to solve various problems in information search and the knowledge management application in organizational environment.

## Acknowledgment

This work was partially supported by projects K-Wf Grid EU RTD IST FP6-511385, NAZOU SPVV 1025/2004, RAPORT APVT-51-024604 and VEGA No. 2/6103/6.

## REFERENCES

- [1] LUCK, M.—MCBURNEY, P.—PREIST, C.: Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing, 2003.
- [2] KELEMEN, J.: The Agent Paradigm – Foreword. In Computing and Informatics, Vol. 22, 2003, No. 6, p. 513–519.
- [3] SINGH, P.: Examining the Society of Mind. In Computing and Informatics, Vol. 22, 2003, No. 6, p. 521–543.
- [4] RUSSELL, S.—NORVIG, P.: Artificial Intelligence – A Modern Approach. Second Edition, Prentice Hall Series in Artificial Intelligence, ISBN 0-13-790395-2, Chapter 10.3 Actions, Situations, and Events. pp. 328–340.
- [5] Telecom Italia: JADE (Java Agent DEvelopment Framework) Website. <http://jade.csel.it/>, 2004.
- [6] Pellucid Consortium: Pellucid IST Project Website. <http://www.sadiel.es/Europa/pellucid/>, 2004.
- [7] CAIRE, G.: JADE Tutorial Application-defined Content Languages and Ontology. <http://jade.csel.it/>, 2002.
- [8] FIPA: FIPA SL Content Language Specification, 2000.
- [9] OBITKO, M.—MARIK V.: OWL Ontology Agent based on FIPA proposal. Znanosti 2004, Brno, Czech Republic, 2004.
- [10] ANDERSON, J.: An Agent-Based Event Driven Foraging Model. Natural Resource Modeling, Vol. 15, 2002, No. 1.
- [11] AgentLink II EU Project, 2002, <http://www.agentlink.org/>.
- [12] SCHREIBER, G.—CRUBEZY, M.—MUSEN, M.: A Case Study in Using Protégé-2000 as a Tool for CommonKADS. 2001,
- [13] LACLAVÍK, M.: AgentOWL – OWL Agent memory model. 2005, <http://jade.tilab.com/community-3rdpartysw.htm#AgentOWL>.
- [14] LACLAVÍK, M.—GATIAL, E.—BALOGH, Z.—HABALA, O.—NGUYEN, G.—HLUCHÝ, L. : Experience Management Based on Text Notes (EMBET). In: Proc. of eChallenges 2005 Conference, 19–21 October 2005, Ljubljana, Slovenia, Innovation and the Knowledge Economy, Volume 2, Part 1: Issues, Applications, Case Studies; Edited by Paul Cunningham and Miriam Cunningham; IOS Press, pp. 261–268, ISSN 1574-1230, ISBN 1-58603-563-0.
- [15] K-Wf Grid Consortium: K-Wf Grid IST Project Website. 2006, <http://www.kwfgrid.net/>.
- [16] BAADER, F.—MCGUINNESS, D.—NARDI, D.: The Description Logic Handbook. ISBN 0521781760, January 9, 2003.

- [17] LACLAVÍK, M.: Ontology and Agent based Approach for Knowledge Management. Ph. D. Thesis; Institute of Informatics, SAS, field: Applied Informatics, June 2005.
- [18] W3C: Web Ontology Language (OWL). 2006, <http://www.w3.org/TR/owl-features/>.
- [19] DARPA: DAML Website. 2002, <http://www.daml.org/>.
- [20] W3C: Resource Description Framework RDF. 2006, <http://www.w3.org/RDF/>.
- [21] W3C: XSLT. 2006, <http://www.w3.org/Style/XSL/>.
- [22] LACLAVÍK, M.—BALOGH, Z.—NGUYEN, G.—GATIAL, E.—HLUCHÝ, L.: Methods for Presenting Ontological Knowledge to the User. In: L. Popelinsky, M. Kratky (Eds.): Znalosti 2005, Proceedings, VSB-Techicka universita Ostrava, Fakulta elektrotechniky a informatiky, 2005, pp. 61–64. ISBN 80-248-0755-6. February 2005, Vysoké Tatry, Slovakia.
- [23] WOOLDRIDGE, M.: Introduction to MultiAgent Systems. ISBN 047149691X, 2002.
- [24] FIPA: Foundation for Intelligent Physical Agents website. 2002, <http://www.fipa.org/>.
- [25] FIPA: FIPA Specification ACL Message Structure. 2000.
- [26] SCHREIBER, A. et al.: Knowledge Engineering and Management: The CommonKADS Methodology. ISBN 0-262-19300-0, 2000.
- [27] Stanford University: Protégé Ontology Editor. 2006, <http://protege.stanford.edu/>.
- [28] FIPA modeling Group: Agent-based Unified Modeling Language – AUML. 2004, <http://www.auml.org/>.
- [29] IGLESIAS, C.—GARIJO, M.—GONZALEZ, J.—VELASCO, J.: Analysis and Design of Multiagent Systems Using MAS-CommonKADS. 1998.
- [30] HP Labs and Open Source Community: Jena Semantic Web Library. 2006, <http://www.sf.net/>.
- [31] LAMBERT, S.: Pellucid Consortium: Knowledge Management for Organisationally Mobile Public Employees. KMGov 2003, Rhodes Island, Greece, 2003.
- [32] NÁVRAT, P.—BIELIKOVÁ, M.—ROZIJANOVÁ, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: Proc. International Conference on Computer Systems and Technologies – CompSysTech 2005, Varna 2005 <http://ecet.ecs.ru.acad.bg/cst05/Docs/cp/SIII/IIIB.7.pdf>.
- [33] LACLAVÍK M.: SouceForge.net: AgentOWL: Agents with OWL Ontology Models Using JADE Agent System and Jena. 2006, <http://agentowl.sourceforge.net/>.



**Michal LACLAVÍK** is a researcher at the Institute of Informatics of the Slovak Academy of Sciences. In 1999 he received his M. Sc. degree in computer science and physics. He received his Ph. D. degree in applied informatics with focus on knowledge oriented technologies in 2006. He is the author and co-author of several scientific papers and he participates in the Pellucid and K-Wf Grid IST projects as well as in several national projects. His research interests include artificial intelligence, agent technologies, knowledge management and semantic web.



**Marián BABÍK** is a researcher at the Institute of Informatics of the Slovak Academy of Sciences at Center for Intelligent Technologies (CIT) and a member of the computing group at the Institute of Experimental Physics, SAS. In 2003 he received his M. Sc. degree in artificial intelligence from the Technical University, Košice. He was a member of the CDF collaboration at the Fermi National Laboratory and worked as a technical student at the European Organization for Nuclear Research (CERN). He is the author and co-author of several scientific papers and participates in the EU IST projects K-Wf Grid, EGEE-2, Degree,

as well as in several national projects. He is experienced in the Semantic Web technologies and languages; development of large scale systems; development of knowledge based systems and semantic web services.



**Zoltán BALOGH** is a researcher at the Institute of Informatics of the Slovak Academy of Sciences. In 1999 he received his M. Sc. degree in management. Since then he is employed at the institute. He is the author and co-author of several scientific papers. He participates in Pellucid and K-Wf Grid EU IST projects as well as in several national projects (VEGA, APVT, SPVV). His research interests include web services, ontologies, case-based reasoning, instance-based learning and application of semantics in business systems.



**Ladislav HLUCHÝ** is the director of the Institute of Informatics of the Slovak Academy of Sciences and also the head of the Department of Parallel and Distributed Computing at the institute. He received M.Sc. and Ph.D. degrees, both in computer science. He is R&D Project Manager, Work-package Leader in a number of 4FP, 5FP and 6FP projects, as well as in Slovak R&D projects (VEGA, APVT, SPVV). He is a member of IEEE, ERCIM, SRCIM, and EuroMicro consortiums, the editor-in-chief of the journal *Computing and Informatics*. He is also (co-)author of scientific books and numerous scientific papers,

contributions and invited lectures at international scientific conferences and workshops. He also gives lectures at Slovak University of Technology and is a supervisor and consultant for Ph.D., master and bachelor studies.