Ontology-based Knowledge Management

Zina Houhamdi [#1], Belkacem Athamena [*2]
[#] Software Engineering Department, Al Ain University of Science and Technology
Al Ain, Abu Dhabi, UAE
[1] z_houhamdi@yahoo.fr
[*] MIS Department, Al Ain University of Science and Technology
Al Ain, Abu Dhabi, UAE
[2] athamena@gmail.com

Abstract—It is recognized that knowledge has a considerable usefulness in people's daily life and all businesses. The current paper discusses a knowledge description using ontology and its application in Multi Agent Systems (MAS). The presented work proposes to reach a powerful relationship between MAS and Knowledge Management (KM), principally by introducing results achieved in the semantic web domain to MAS. The proposed method is to apply KM in MAS using ontology, wherever undetermined knowledge is absent. This is an appropriate model for various cases particularly when we require experience management.

Keyword—Ontology, Multi Agent System, Knowledge Management, UML

I. INTRODUCTION

Knowledge is actually an essential attractive feature in the universe. The economy realm is changing in the direction of the so labeled knowledge economy, where the more esteemed resource is the knowledge [8]. This paper perceives knowledge as is symbolized by semantic web research. Nowadays KM approaches use principally centralized approaches, which don't enough, fulfill the requirements to include diverse and distributed context. KM systems are generally conceived for a real/small number of applications. In this situation, agents are appropriate solutions; however MAS approaches miss intelligence and knowledge. Agents perceive conventional knowledge in the same manner as human.

The agent use distributed approach; hence it is analogous to the real world. Nevertheless we require particular pivotal accessible locations. This study takes charge of collaborating intelligent agents, which can be mobile if necessary however don't have to be. The research concentrates on an agent oriented model for KM systems [20].

The *protégé* ontology editor and *commonKADS* methodology will be adopted for knowledge description. The JADE agent system will be used for the architecture implementation.

MAS or Agent Oriented Programming (AOP) is an extremely strong approach in heterogeneous non-centralized information systems which require knowledge in reasoning and description. For the time being, MAS lack link with actual standards of the commercial technology and the research results in the semantic web. MAS require to include finding of various flied of computer sciences including knowledge management, artificial intelligence, negotiation etc. [9]. Knowledge is normally defined by predicate logic, rules or states. These definitions are very robust however it is difficult to obtain information in that manner from operational information systems or from user. And also it is difficult to display information declared in predicate logic, for example, to people.

Ontology as known in modern information systems is similar to the semantic web. It is described using XML/RDF, which can be more simply obtained from or sent back to a current information system or user. For this reason, the knowledge is brought in to MAS, using semantic web outcomes, and an appropriate agent knowledge model is formulated. We have developed an agent library to uphold this model and consequently improve the JADE agent system (evaluated as the top implemented toolkit in existing MAS).

First, this paper presents the description of the agent architecture and knowledge model. Secondly, it specifies the methodology to produce an agent knowledge model and finally, it defines the proposed approach for managing MAS ontology using Java libraries.

II. BACKGROUND

This section presents the actual literature in subjects of concern: software agents, knowledge management and ontologies. Further, it explains the study background and motivation.

A. Agent

All software technologies are totally distinct from Agents since these technologies are principally function oriented. In contrast, the agent technology is the following step from Object Oriented Programming (OOP) to AOP.

When we deal with agents we signify agents in MAS, where we have several agents and these agents are able to communicate. Based on the research direction, the world agent is defined divergently. The literature contains various definitions of an agent. In this paper, we adopt the agent definition proposed by the Agent-Link II: *An agent is a computer system capable of flexible autonomous action in a dynamic, unpredictable and open environment* [11].

There are 2 types of agent (e.g. Fig. 1): Robotic agent (implanted on hardware) and Software agent (supported by software). Mobile agents describe the subset of software agents that can migrate between locations. The mobility can be weak and strong. The weak mobility signifies that an agent moves and takes just the program and attributes. The mobility is strong when an agent travels from one workstation to another it carries its execution state and its attributes and program.

Simulation Agent is a software agent type. It assists to imitate a discrete system. This kind of system is too complex or can't be defined by mathematical equations. Simulation of ants' behavior is the top popular example [19].

The agency is the software context where the intelligent agents operate. They communicate via messages passing. These agents apply the artificial intelligence methods (e.g. reasoning, negotiation, learning, and decision support). They can have besides a number of sensors to perceive outer environment and other techniques to affect special systems in the context. Intelligent agents learn and operate by passing message.

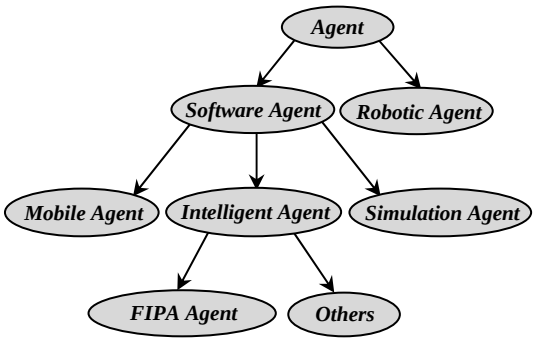

Fig. 1. Agents Classification

FIPA (Foundation for Intelligent Physical Agents) agent use FIPA standards [4]. Agents' communication is achieved by the FIPA ACL (Agent Communication Language) [7]. Interaction between agents uses ontology. The agent program requires sustaining a directory manager in which all agents can enroll.

Several hybrid approaches can be obtained by combining different types of agents together for example mobile-intelligent agents or software-robotic agents.

We give this recapitulation to clarify the agent definition since it is overcast in computer society the real meaning of the word agent. Whenever the term agent is cited in the paper, it refers to FIPA Agent.

B. Ontology

Ontology has usually different interpretations. It is perceived as thesaurus, taxonomy or vocabulary. We adopt the semantic web team definition for ontology: Ontology is a specification of problem context which represents the domain objects with their attributes and their relationships [2]. Ontology is recognized as a very crucial characteristic in diverse systems to give a semantic skeleton for KM. More precisely; ontology can be defined as a collection of descriptions of knowledge elements which are attributes, functions, classes and relationships. Ontology describes information concerning objects as hierarchical framework by classifying the objects based on their important characteristics.

The processing is not the biggest ontology utility. The exceptional advantages of ontology are meaning sharing, gradual occurrence and detection of holes and improvement of the implicit knowledge migration. Ontology includes structured knowledge described in a formal language as well as unorganized or undefined knowledge represented in a procedural style or natural language.

Ontology in computer language represents knowledge in organized and formal way. The ontology is conceived to act as a feedstock for problem solving and MAS. During the agents' inter-operation, the ontology describes formally the significance of the used vocabularies. This issue is essential since the external context and purposes adequacy can be perceived differently by the distinct agents. However they can still inter-operate to execute a shared task.

Ontology can be described by UML [13], RDF (Resource Description Framework) [15], DAML+OIL [3], OWL [14], any object oriented language. The semantic web society considers OWL–Web ontology language as the best ontology description. But, other descriptions are preferred by MAS community like Object Oriented Approaches [1] and Description Logic (DL) [10].

C. Knowledge Management

Information can be defined as a truth or data concerning particular topic or event. However *Knowledge signifies possessing information and perceiving it during practice* [9]. Knowledge in human world depends on information (which can be false or true) and also on undetermined information (which is partly false or true). Undetermined knowledge can be described by diverse methods for example computing with words, fuzzy logic or probability measures [18]. Several techniques are recognized to describe undetermined knowledge and can be applied in MAS for example FIPA-SL extended language [5]; but, undetermined knowledge is considerably complex and not instantly comprehensible particularly for the agent itself. Whenever undetermined knowledge is used, computer systems can't find out new knowledge by applying fundamental logic theory in current knowledge base. This situation is called knowledge inconsistency [20].

For this reason we discuss just determined knowledge describing reliable information. This kind of information are organized and represented by ontologies. This option allows to agents to perceive knowledge and to deduce further knowledge from available ones in a simple and unambiguous way. It seems that the use of such knowledge is adequate for very restricted applications domain; however by analysis of distinct applications [10], more precisely where experience KM is required, the concentration on determined knowledge is more appropriate than the undetermined knowledge.

Knowledge in the human world is shared between users and they use some techniques to understand and classify this knowledge by detecting the synonymous words, applying same vocabulary or producing central repository for example database, library or knowledge base. Most contemporary knowledge systems are centralized and use ontology [12], [17].

We propose a combination of centralized and distributed methods. In this hybrid approach, Agents have a direct access to local repositories where they can obtain needed knowledge otherwise they have to solicit other agents to supply them with desired knowledge which is managed just by a specific agent. This methodology incorporates centralized and distributed approaches thus it is remarkably advantageous for KM systems.

In this work, we will discuss the following points:

- Conception of agent framework by incorporating ontology to knowledge model.
- Description of agent's generation approach by using the proposed agent framework.

Definition and elaboration of Agent library which is used for agents' construction with the suggested methodology (with the expectation to introduce agents to conventional systems).

III. AGENT FRAMEWORK

An example of a task diagram describing the locate victim task is shown in Fig. 3. Actions within each state are executed sequentially and are written as functions. Locate victim is a reactive task, which means that it is initiated whenever a search (area) message is received from the find area to Search task. After the task receives a search area message, it plans a route to obtain to the area and then goes about executing the route. If route execution fails, the task re-plans the route and updates the map.

There are three fundamental architectures types: behavioral, BDI (Belief Desire Intention) and reactive. The hybrid approach which combines the three architectures types, is the more applied in the existing systems. In this paper, we focus on behavioral style by developing an agent repository to implement agent operations.

As first step, we define formally agent repository using DL and after that we explain how RDF-OWL can be used for its design and implementation. The main element in the proposed framework is event. This framework is an improvement of ontology model used in JADE (concept and predicates). As we mentioned in [1], JADE ontology model has some shortcoming.

A. Formal Agent Knowledge Model Specification

In this section, we describe the agent model by applying DL. Fig. 2 represents formal graph definition. Our model contains eight principal features: *resources*, *actors*, *operations*, *tasks*, *domain*, *contexts*, *events* and *agents*.

The set of the existing resources in the agent context is described by *Resource* class. The sub-classes corresponding to different sort of resources are represented by the model specialization. The *Actor* is the main sub-class of *Resource.*

$Actor \subseteq Resource$

$\{actor\} \in Actor$

Actor instances can perform operations {*operation*} which are instances of *Operation* class.

$\{operation\} \in Operation$

Depending on the application domain, *Task* class stands for problems or performed works or works require to be performed in the agent context.

The ontology extension in the system realm is represented by *Domain* class which is the super-class for the entire extensions set. *Context* class merges the environments of task, actors, resources and operation classes.

$Task \subseteq Context$

$Operation \subseteq Context$

$Domain \subseteq Context$

$Resource \subseteq Context$

$Context \supset (Task \cup Operation \cup Domain \cup Resource)$

$\{context\} \in Context$

$\{domain\} \in Domain$

Domain is a crucial task attribute. It denotes the beliefs associated to the application field. As well known, these relationships are helpful for selecting suitable KM algorithms used to update the resource and actor context.

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

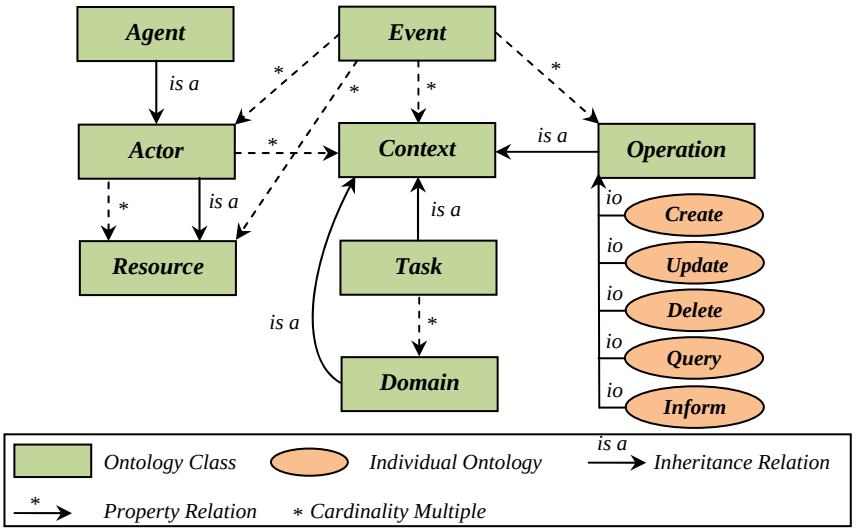

Fig. 2. Ontology for Knowledge Modeling

System events are represented by event class. {*event*} is an instance of event class which describes {*operation*} performed on special {*resource*}, defined by {*context*}, by {*actor*}. The *actor.Event, operation.Event, context.Event* and *resource.Event* are event class attributes.

$\{event\} \in Event$

$$Event \subseteq actor.Event(\{actor\}) \cap resource.Event(\{resource\}) \cap$$
$$context.Event(\{context\}) \cap operation.Event(\{operation\})$$

Agent class symbolizes a specific *Actor type.* This class describes agents in the system.

$Agent \subseteq Actor$

$\{agent\} \in Agent$

Software agents execute standard operations which are well specified. These operations define different kind of interaction between agents including ACL messages (INFORM and QUERY-REF). Events of this type are produced whenever an inter-agents interaction is executed.

$\{Inform, Query\} \in Operation$

But if operations like resources *deleting*, *creating* or *updating* are executed, the system will save and assess events representing those operation types.

$\{Create, Update, Delete\} \in Operation$

Actor class contains two essential attributes which are *Resource.Actor and Context.Actor*. The *Resource.Actor* attribute represents set of resources used by the actor. These resources are consequences of actor's plans or goals. Thus they depend on actual actor's context. The *Context.Actor* defines actual context of the actor. Concerning system context, it can be deduced from saved events which illustrate external context state. Therefore, new pertinent events influence actor context or the external system context.

Algorithms for updating resource and context are defined as follows.

$$Actor \subseteq Resource.Actor(\{resource\}) \cap Context.Actor(\{actor\}) \cap Resource$$

$$Context.Actor(\{actor\}) = f_c(\forall\{event\}; Actor.Event(\{actor\}) \in \{event\})$$

$$Resource.Actor(\{resource\}) = f_R(Context.Actor(\{actor\}))$$

These algorithms represent the system brain. If these algorithms are modified in the future, this model allows obtaining more suitable results by utilizing exactly the same data and framework: Since all events are stored, we can describe the context in any instant from past and treat it afterward from the begging with upgraded algorithms to update the resource and context. This is the strong point of our model.

B. Knowledge Sharing and Exchange in Defined Architecture

Centralized approaches are, today, frequently employed for KM system construction. Behind the world distributed we assume all MAS. We believe that each application needs a specific approach and none of these approaches is classified as the best. In the paper we represent how to employ both distributed and centralized and we leave the choice to the developer of KM system to select the appropriate approach.

To conceive the KM system, we adopt a hybrid approach. The KM system must contain central points (like libraries, internet portal, etc.) in which agents can obtain/share knowledge or use ontologies. This situation is called *blackboard approach* [20] there is one central location used by all agents to write their knowledge which are accessible by each agent. Similarly, we suggest designing a unique repository supported by the proposed agent knowledge framework. In this case, we authorize knowledge sharing between agents. This is very useful specifically if we consider collaborative agents. Thus, structured repositories are the best choice [8], where information sharing among agents can be achieved by using a query language for RDF. Common shared ontology (which is one of the blackboard approaches) is advantageous for collaborative agents as they must understand the used worlds in the same way. Consequently, they use exactly alike ontology during their communication. FIPA ontology specification defines the standards for ontology agents [6].

In FIPA definition, important are two principal messages between agents or precisely actors: *informs* and *requests.* In the former, an actor *informs* the other actor about something. In the later, one actor *requests* another actor to perform something or for something.

- *If informed*: The actor modifies its repository or its knowledge to be update.
- *Up request*: The asked actor asks/explores (optionally modify) its repository or its knowledge.

The interactions execution produce events and when an actor receives an event, it updates its specification (context as well as resources) by performing described algorithms C^A_{New} *and* R^A_{New} .

IV. Approach Description

The proposed approach combines various portions of several approaches. Principally, it is similar to *CommonKADS*. But this methodology does not work with any ontologies, knowledge description or modeling tool. *CommonKADS* methodology consists of two basic steps:

- Knowledge Specification containing three other sub-specifications: *Task Specification*, *Agent Specification*, and *Structural Specification.*
- System Design. We focus on Multi Agent System.

OWL ontology is used for knowledge specification which is modeled in the editor of *Protégé* ontology. Consequently, our approach for modeling knowledge is comparable to [16]. Modeling ontology with *Protégé* illustrates models of *CommonKADS*. System design is based on MAS *CommonKADS*, AUML and UML.

A. Agent Modeling and Ontology Creation Approach

The three models of *CommonKADS* are used as the starting point for knowledge modeling for an application: *Task Specification, Structural Specification,* and *Actor Specification.*

1. *Task Specification*: Describes the important processes, operations and tasks of the actors. Domain is the more significant attribute of the task. Usually, tasks are associated to a number of resources or other entities in the domain and these relations are needed to describe algorithms which update the context of the actor and resources.
2. *Structural Specification*: Describes the application context and the actors' context. In this manner the description of resource attributes requires to be enhanced with additional type of resources like service, contact, item or document. We extend the domain attributes with all particular features of the domain or application (resources are included), which describe problem area. Thus they are used by actors in task accomplishment or decisions making. The resources are treated as outcomes of actor goals and should be described also. In general outcomes of actor's goals can be seen as resources for example in auction agents (sold items) or hunting agents (locate documents). Entities characteristics and their relations are established as a main step of our model.
3. *Actor Specification*: Describes actors executing operations and tasks. The actor models a software agent, a human or any object executing operations which is under the system. Actor's context is the more significant attribute in the actor specification. It specifies the actual resources and context of the actor. Consequently, we define two algorithms for updating the actor context and resources.

$$C^{A}_{New} = f_c(e^{A}, C^{A}_{Old})$$

$$R^{A}_{New} = f_R(C^{A}_{New}, R^{A}_{Old})$$

Once the modeling step of entities, attributes and their relationships is completed, we define the needed ontology which will be improved iteratively to contain all required entities. The model improvement needs to check if the aggregation of actors, operations and resources are perceived as events also if the proposed ontology entities are able to deduce all events. The model output is knowledge model that includes:

- Two procedures for updating the resources and context of each actor (R^{A}_{New} , C^{A}_{New}). Usually procedures are identical or equivalent.
- Ontology elaborated using *Protégé.*

B. Agent Based System Design

Proposed system methodology use 3 UML diagrams:

- *Class Diagram*: Depicts the classes within the model. Each agent type is described by a class. Classes have attributes (member variables), methods (member operations) and relationships with other classes.
- *Sequence Diagram*: Shows the communications between objects and interaction with external systems (e.g. sensors, Graphical User Interface or other interfaces) in the sequential order that those interactions occur. Communications among agents illustrate ACL message type (i.e. *Query*, *Inform*). An auxiliary table is attached to a diagram to describe the type of each message.
- *Use Case Diagram:* Overviews the usage requirements for a system. It describes the behaviour of the target system from an external perspective. In MAS, each agent is represented by use case diagram. As consequence, the developers and designers can predict the intended system behaviour.

V. DESIGN OF AGENT SOFTWARE LIBRARY

In this section we define the proposed library which is supported by the Jena library and the JADE agent system. The library includes set of functions which are missed in modern agent architectures like JADE [9]. Library contains three classes which are *repository*, *message* and *ontology* and represented in Fig. 3 with UML.

The class *Message* holds required functions for models transformation to the XML/RDF and also ACL messages generation. A sub-set of those functions are employed for XML-RPC conversation.

The class *ontology* contains fundamental attributes concerning the proposed ontology and OWL ontology.

The class *repository* is responsible to manage, save and charge the agent repository using models such as OWL or RDF. For management, Jena library is employed. An OWL file elaborated in *Protégé* is loaded in the repository. This file will be changed into ontology model used by the repository. This repository is saved in an OWL file, in a database tail (e.g. MySQL) or in other Relational database supported by Jena. This database

model allows to an agent to move between computers without transferring its repository. It only requires disjoin from current repository and link after execution begins on a distinct node when a mobile agents' use is needed.

Fig. 4 displays an agent library behavior diagram showing the library usage. This library can be used to develop a JADE agent to prop OWL model based on Jena library. Moreover, now it is easy to incorporate the XML remote procedure call functions for knowledge description or events reception like RDF messages from systems in external environment by using the proposed ontology. In addition, it permits interaction among agents supported by RDF/OWL.

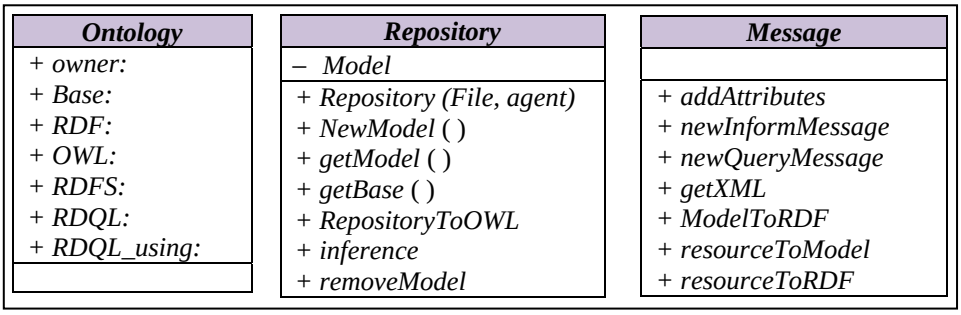

Fig. 3. UML Classes of the Library

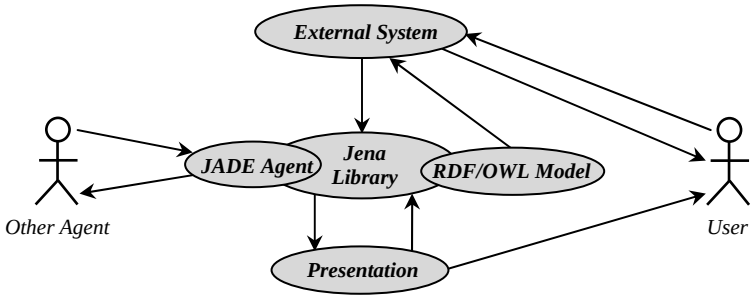

Fig. 4. Agent Library Behavior Model

VI. CASE STUDY

This example illustrates the application of the proposed agent library. In this case, a simple ontology is defined in *Protégé* and there are two agents:

– Query Agent (QA)

– Reply Agent (RA)

The two agents utilize the identical ontology however with distinct instances. In Fig. 5, we observe that QA maintain information about various resource kinds (like Computer, CD and Book) however no instances of that resources are saved in its repository. Fig. 6 shows the RA repository containing set of resource instances. In this case the model includes three different computer instances: *com1*, *com2* and *com3*. QA should look for the resource type selected by the user. The user chooses, for example, a computer in the QA interface. The resource type is sent to QA agent who creates a query message and sends it to the RA. The QA invites the RA to reply it all instances existing in its repository. The RA executes the query on its repository. The result is sent as different ACL inform messages containing the RDF specification of desired resource. RA generates events and saves it in its repository mentioning that resources were passed to QA. Thus RA conserves information concerning the external context. When an ACL-inform message is received by the QA, it keeps its context into its repository. Events concerning received resources are stored in QA repository. It also appends references to answered resources to the QA instance resource attributes. When a user asks for XML format in the QA interface, the QA instance from the QA repository is displayed in the XML style where you can observe resources which are in this time included in the QA repository.

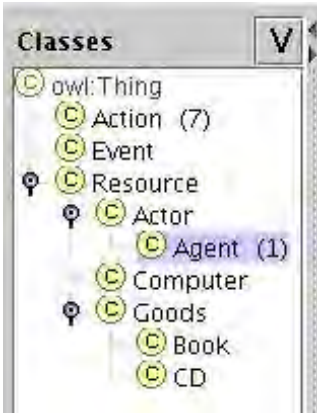

Fig. 5. QA Ontology-Protégé Screenshot

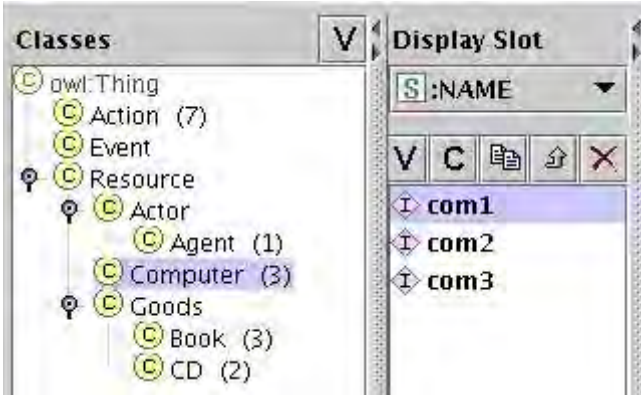

Fig. 6. RA Ontology with some Resource Instances

A. Usage of Defined Methodology

Here we present how proposed methodology is employed in the case study design and implementation. In this section agents repository models as well as knowledge model with brief summary are described to illustrate explicit example on proposed methodology. Fig. 7 represents the knowledge model and extends abstract knowledge model. Model is enhanced just by resources like Operations (e.g. sell, buy), CDs and books.

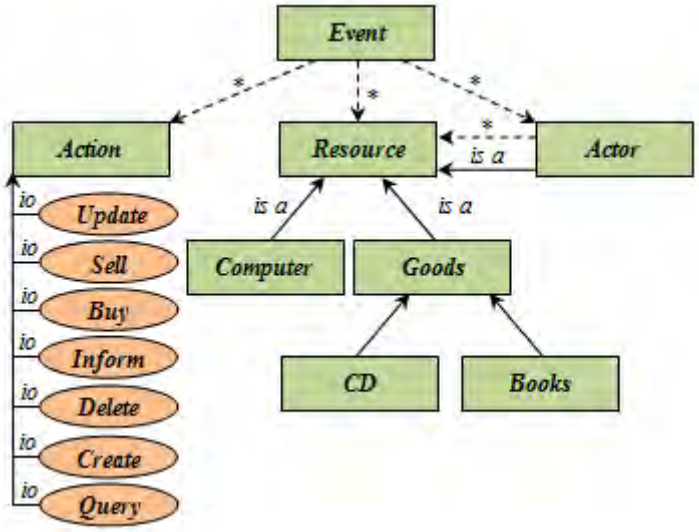

Fig. 7. Simple Knowledge Model

Fig. 8 represents the knowledge model of QA. It includes just one extra instance defining itself. When conversation begins, model is changed and extended with events summarizing the taken operations history also QA gent resources and context are modified as stated in the equations of R^A_{New} and C^A_{New}.

Fig. 9 shows the knowledge model of RA. It includes instances defining itself and other resources such as CDs or books. When conversation begins, the model is changed and extended with events summarizing the taken operations history.

The more important portion of the actor model is the description of the C^A_{New} and R^A_{New} algorithms. We present here only QA algorithms. RA just replies a wanted MySQL query however in a real situation where the QA buys items from RA we should define these algorithms for the both. The Algorithms are illustrated as flowchart for understandability reason. In fact they are written in Java.

To update the QA context, the C^A_{New} algorithm transfers all received resources to the recent context. Whenever a query event is created, the context is cleaned out (e.g. Fig. 10).

The R^A_{New} algorithm for QA resources habitually empties the resources list and fills it with a copy of the complete context (e.g. Fig. 11).

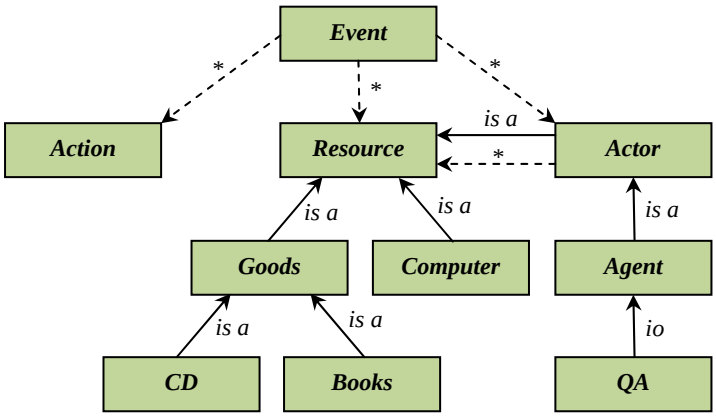

Fig. 8. Knowledge Model of QA

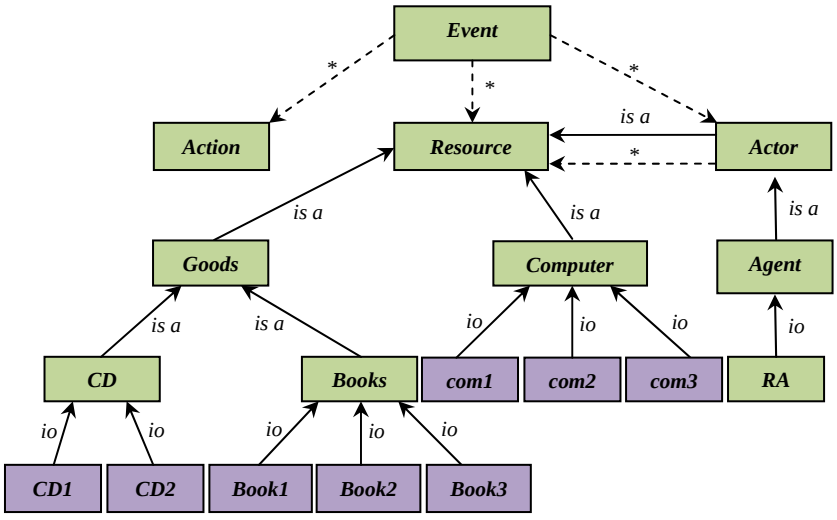

Fig. 9. Knowledge Model of RA

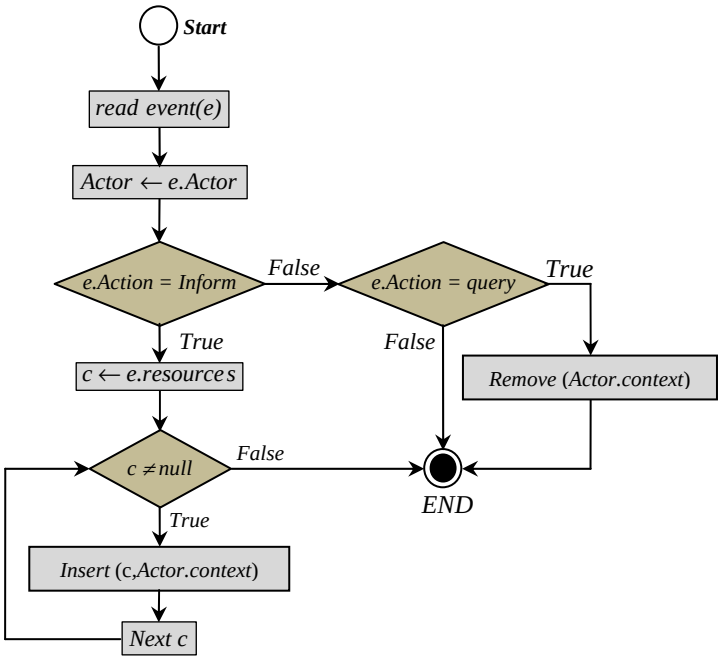

Fig. 10. Update Context Flowchart

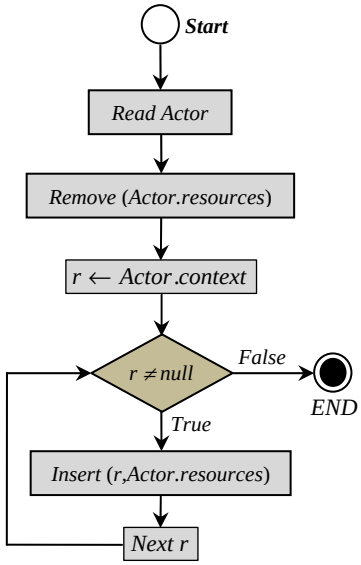

Fig. 11. Update Resources Flowchart

B. Definition and Design

In this section, we draw set of UML diagrams to illustrate the proposed methodology explicitly. Fig. 12 describes a use case diagram for QA. On Fig. 13 presents another use case from the RA perspective where the RA is the system border.

With set of use cases (one use case by an agent kind), a developer can describe a system behavior and manipulation in clear way.

Sequence diagram describes communication between external systems and agents (e.g. Fig. 14). In order to communicate the agents use ACL messages (Query or Inform).

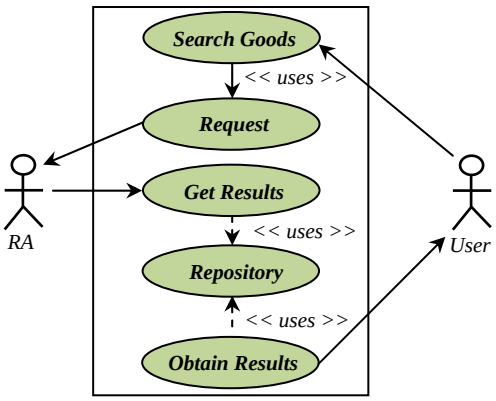

Fig. 12. QA Use Case Diagram

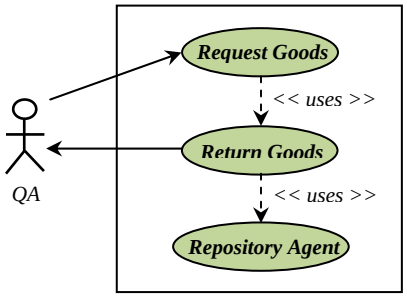

Fig. 13. RA Use Case Diagram

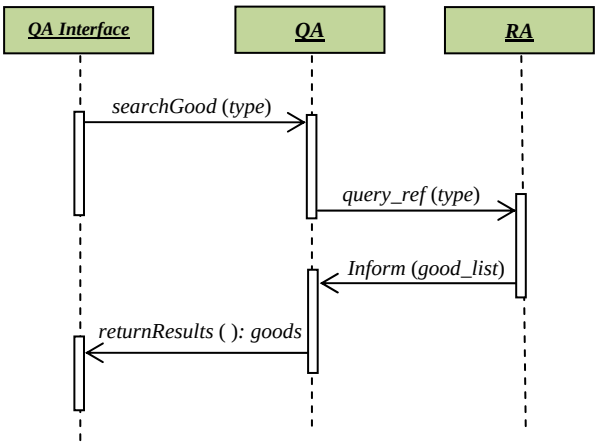

Fig. 14. Sequence Diagram

VII. CONCLUSION

The paper illustrates the application of the semantic web issues in MAS. We have proposed an agent knowledge model. This model can describe the external context of the agent, its context and resources. We have designed algorithms for updating the model while the external context evolves.

The proposed agent framework and model and can be implemented and applicable in several Knowledge Management domain.

The current work affects various domains in computer sciences.

– *Agent Library:* This paper presents the implementation of the proposed model and the agent framework. A significant advantage of the methodology is in using the semantic web outcomes to MAS. The application of this library allows the agent society to use models based on OWL/RDF in MAS and also to link MAS with commercial technologies and to express explicit and formal knowledge to the users.

– *Agent Repository Model:* The main components of the model are actors, resources and events. The usage of this model guides designers to deal with required modeling expansions. The model explicitly describes the application as well as the actor context.

– *Methodology Model*: The suggested methodology uses current system design and KM approaches. Its significant enhancement resides in usage of current tools and approaches for agent modeling which makes easy the application of this model.

The proposed model, architecture and library can speed up the translation from scientific outcomes to industrial application.

REFERENCES

[1] Caire, G.: JADE Tutorial: Application Defined Content Languages and Ontology, 2002. http://jade.cselt.it/
[2] Cui, Z., Jones, D., O'Brien, P.: Issues in Ontology based Information Integration, Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, Vol.47, pp.141-146, 2001.
[3] DARPA Agent Markup Language (DAML), 2002. http://www.daml.org/
[4] IEEE Foundation for Intelligent Physical Agents (FIPA), 2012. http://fipa.org/specs/ fipa00097/SC00097B.pdf
[5] FIPA SL Content Language Specification, 2001. http://www.fipa.org/ specs/fipa00008/XC00008G.pdf
[6] FIPA Ontology Service Specification, 2001. http://www.fipa.org/ specs/fipa00086/XC00086D.pdf
[7] FIPA ACL Message Structure Specification, 2000. http://www.fipa.org/ specs/fipa00061/XC00061E.pdf
[8] Kuntz, R., Matta, N.: Knowledge Management and Organizational Memories, Springer Science and Bussiness Media, New York, August 1, 2002.
[9] Laclavik, M., Balogh, Z., Gatial, E., Habala, O., Nguyen, G., Hluch, L.: e-Collaboration and Knowledge Sharing based on Text Notes, Informatika 2005, Bratislava, Slovakia,2005.
[10] Lambert, S.: Deliverable D4: Environment and Requirements Analysis, Pellucid Consortium, November 2003.
[11] Luck, M., McBurney, P., Preist, C.: Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent Based Computing, Agentlink community, 2003.
[12] Motta, E.: Ontoweb Consortium, Universidad Politécnica de Madrid, July 2004.
[13] OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Object Management Group, OMG Available Specification, November 2007.
[14] OWL Web Ontology Language Overview, OWL Working Group, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-features/
[15] Resource Description Framework (RDF), RDF Working Group, 2004, http://www.w3.org/RDF/
[16] Schreiber, G., Crubezy, M., Musen, M.: A Case Study in Using Protégé 2000 as a Tool for CommonKADS, EKAW '00 Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management,pp. 33-48 Springer Verlag London, UK, 2000.
[17] Stuckenschmidt, H.: Ontology-Based Information in Dynamic Environments, Proceedings of the Twelfth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), 2003.
[18] Wang, P.: Computing with Words, Wiley Series on Intelligent Systems (Book 3), Wiley-Interscience, 1 edition, 748 pages, 2001.
[19] Willmott, S., Thompson, S., Bonnefoy, D.: Agent Cities: Towards Dynamic Value Creation in Online Open Environments, IST Conference Milan, Italy, 2003
[20] Wooldridge, M.: An Introduction to Multi Agent Systems, Wiley, 2nd edition, 484 pages, 2009.J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.