A Cooperative Information Exchange

Zina Houhamdi and Belkacem Athamena

Software Engineering Department, Faculty of Science and IT, Al-Zaytoonah University, Jordan

Summary
Multi-agent systems are often very complex applications with many modules and issues that need to be taken into account. The Cooperative Information exchange system is about multi- agent system implementation that would implement each agent to act on behalf of the user and exchange information with other agents. The system would make use of ontologies in order to make knowledge exchange more easy and convenient process.
In this paper, we present *Expertorer*, a system of distributed collaborative agents to help ts owner find information most relevant to current needs by asking other agents, present in the system, whether they have certain knowledge about the subject or not. *Expertorer* provides needed services for users to share and learn information.

Key words:
Artificial Intelligence, Multi agents system, Ontology, Jade.

1. Introduction

Today we live in 'Information Society' where information is the crucial part of everyday life. There are sufficient amount of new information being exchanged and produced every day. The Internet revolution has made an abundance of information resources available for direct and easy access on the users desktop. However, finding appropriate information has become a significant problem for many users. Organized information spaces are easier to search, but finding or authoring these is difficult.

Current WWW search engines allow users to locate information of interest, but often return vast amount of irrelevant information. We are not able to investigate all data that are potentially available for us. Therefore often people are not able to get access to the all desired news due to the number of new web sites and information sources that appear every day.

However the semantic web idea and Artificial Intelligence are new technologies introduced to help us with such variety of information. In the future web applications will be able to receive much higher accuracy by using semantic information available on the web. Cooperative Information Exchange will investigate and implement a system, which will utilize these technologies, which could eventually help with the information and knowledge exchange that could speed and increase the accuracy of this process [5].

The objective of this paper is to present *Expertorer*, a system that uses Multi Agent Technology to allow its user to exchange information on a special topic with the other agents present in the system. A Multi-Agent System MAS is one that consists of number of agents, which interact with one another. Agents will be acting on the behalf of users/other agents with different goals and motivation. The Agents in MAS work in a team to achieve common goal by interacting with one another. To successfully interact, they require ability to cooperate, coordinate and negotiate with each other in order to obtain valuable information. The agents will share the same ontology, which will allow exchanging meaningful and understandable information for all the parties involved [6]. Agents will also use protocols available with JADE to search for agents who have the information that the agent is seeking on behalf of its user.

2. System overview

2.1 Agent – database interaction

Knowledge exchange will be the most important part of the system and therefore each agent representing user has a database with all user's knowledge. Agents will be able to query each other. However only the agent itself has the ability to look for the knowledge in its database. After the search it will return the answer or block the operations. Therefore the agent querying others will only receive the answers from those agents, which has it. User will be able to receive and see the needed information from the system. This information will be displayed in the GUI of the agent.

2.2 Agent – agent interaction

Agent must be able to communicate with other agent present in the system in order to achieve what user requested the agent to do. There is a message exchange capability that each agent has. Each user will have his/her own agent, which will serve as the users' representative and will all together form a MAS consisting of many users.

2.3 Interaction in Multi-agent system

One of the most important issues arising from MAS is the communication between agents and coordination of this communication. The main systems' requirements will be accomplished through agent communication. Agents will speak directly to each other and will attempt to find the

Manuscript received January 5, 2011
Manuscript revised January 20, 2011

information that the user request. JADE provides each agent with a globally unique identifier AID. It also gives the platform addresses that agent operates on along with all other agent addresses needed for communication [2].

The agent management will be done through JADE available features such as graphical user interface to manage agents with Agent Management System (AMS), sniffer agent to look at the message exchange, transport and agent migration services. Those JADE build in modules help to manage Multi-agent system.

2.4 Agent deployment

Systems developed with JADE have the ability to become distributed systems. The agent platform can be split on several hosts. There could also be several platforms in the system. Each host acts as a container of agents and provides a complete run time environment for agent execution and allows several agents to execute concurrently. Each agent is then a single Java Thread. The first container started is the main container maintains a central registry of all the other containers agents can discover and interact with each other across the whole platform [3].

In the *Expertorer*, there is one platform, which would be placed on the server. Each user would have his or her own agent placed on the host machine. Therefore the system operates as a distributed environment consisting of many agents sharing its resources.

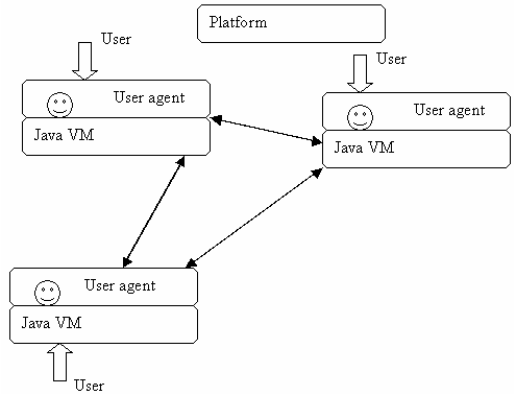

Fig. 1 *Expertorer*'s MAS structure.

3. Design phase

To allow one agent to query other agents to send it some kind information, communication between agents will be based on the template similar to the FIPA-Query Interaction Protocol (IP).

Explanation of the Protocol Flow: The Initiator asks the Participant to perform *inform* action using one of two query communicative acts, *query-if* or *query-ref*. The *query-if* performative is used when the Initiator wants to query whether a particular statement is true or false. The *query-ref* communication is used when the Initiator wants to query for some information. The Participant receives the *query-if* or *query-ref* and makes a decision of whether to act and reply or refuse the query request. When Participant decides to refuse, then "*refused*" becomes true and the Participant agent communicates a refuse. The other option is that "*agreed*" becomes true [7].

In the *Expertorer*, we modified the idea of FIPA Query Interaction Protocol so that the agent has to answer to the query of another agent. We used Behaviors class available in JADE to mimic and change the idea of FIPA Query Interaction Protocol.

Interaction protocol and interaction between agents in general means to exchange some kind of data. In case of Multi-agent systems it usually means that agent requests or query the receiver agent to execute some kind of an "*inform*" that is, to answer to the "*query-ref*". For example agent *i* sends query to agent *j* because he wants to know something and he thinks that agent *j* has this information. If agent *j* really has this information it sends the "inform" performative to agent *i* with the message containing this information. So, it is not possible to reject the query and fail or refuse to the query issued by another agent due to the fact that Multi-agent system presented in the system has to be a collaborative one.

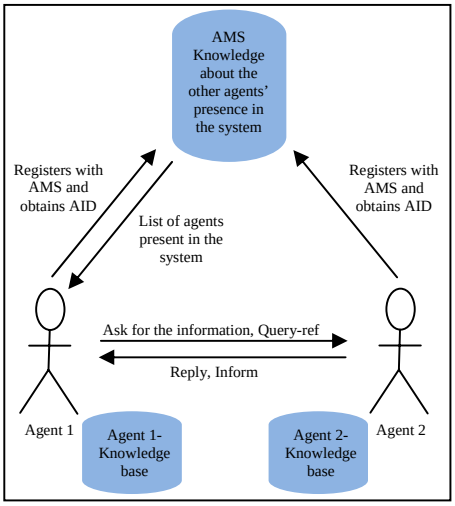

Fig. 2 Program flow chart.

The system will work as follows: Agent, which is represented by its user, will register with AMS of the platform. Then the agent will obtain its AID. Each agent has its own database where the user could enter his/her terms and definitions which would be the knowledge represented by him/her. Consequently, each agent will represent the user knowledge in the system.

When user will need to know the definition of a term he/she will ask the agent. Agent will then obtain all of other agents' addresses from the AMS. Then the agent will use modification of FIPA-query protocol to contact all of

them to ask them whether they have definition for the term. If one or several of them will have the definition for the term they will answer and send this definition only to agent that requested it. The definition will be displayed in the requestor agents' user GUI.

3.1 Agent Management System (AMS)

The AMS is compulsory feature in agent platform. It is responsible of creation and deletion of agents and its possible migrations. Each agent must register with an AMS in order to obtain an AID, which is then retained by the AMS as a directory of all agents present within agent platform and their current state (e.g. active, suspended or waiting) [2]. When agent deregisters with AMS it is deleted from the platform. The AID of the deleted agent can then be taken by the other agent, which would request to become a part of the platform. AMS contains Agent description and platform description. Then the request could be done in order to obtain this information. Single agent platform could be expanded to be present on several hosts and each platform could have only one AMS. Therefore this AMS would manage the whole platform.

In the *Expertorer*, each agent is able to discover all other agents in the system through the AMS which provides the mechanism of discovering agents automatically. Therefore it is used as the most reliable way to do that among all methods provided by JADE.

Message Transport Service (MTS) allows exchanging FIPA-ACL messages over many different agent platforms and across single platform. The message carries for example the information about the intended recipients of the message. This is done in the envelope. FIPA message structure consists of envelope with transport information, payload, which is, encoded message, message that is message parameters, and message content.

3.2 Architecture description

Simple reactive architecture is implemented. This architecture type joins directly situation to an action and it is based on a stimulus-response mechanism triggered by sensor data. It is similar to finite state machines that possess some kind of sensors transmitting information. Higher levels of behaviors have more control over decision making. Decisions are made through goals to be achieved. This architecture is much less complicated than BDI and artificial intelligence models don't have to be used in that case.

3.3 Ontology

The ontology is the set of concepts and symbols which allow understanding a message that agent send or receive. The agents present in the system must share the same understanding in order to exchange and store shared information. Ontologies are usually domain specific [8]. Those are the elements that should be distinguished when JADE will read the content of the message:

- Predicates are elements, which are used to express the status of the world. They are used usually for the INFORM or QUERY-IF messages and express true or false. In the *Expertorer*, the predicates will not be used but the terms, which will be another element instead.
- Terms are another element identified by ontologies. They describe entities that exist in the world. Agents may reason about. In the *Expertorer*, Term and Definition will be two elements recognized by JADE as Terms in ontology.
- Actions are other elements of the ontology that suppose to indicate some agent defined action. They are usually used in such message performative as REQUEST or QUERY. In the *Expertorer*, we use the call for an action in the form of getsendDefinition() requests. It will request other agents to send the definition for the term.

The ontology presented in the *Expertorer* has two concepts. One is the Term that needs to be search and the other concept is Definition for the Term. There is also an action, which will be to sendDefinition. We have built a class that extends ontology and declares and defines all schemas, actions and concepts in that class. Each schema will be associated with class. Therefore there is a Glossary.java with main concepts and Send_glossary.java with action of the program.

The Figure 3 presents the basic architecture of the agent instance build to create multi-agent system. Many of the same agents will cooperate in the system.

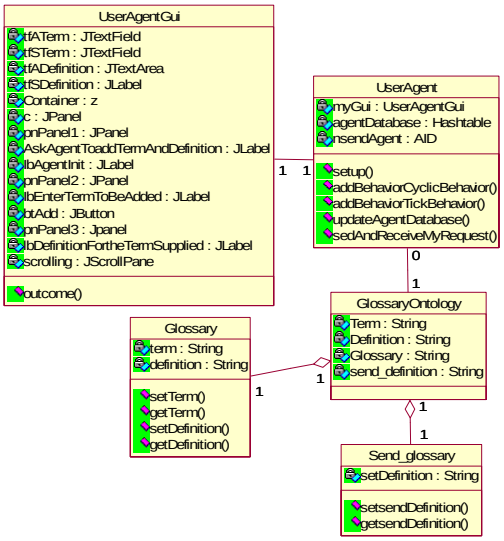

Fig. 3 UML diagram of the agent instance.

The general structure of the classes is as follows: All the classes are stored in the glossary folder within the directory where there is an agent. GlossaryOntology.java

contains all concepts and action that agent has to execute with relations between them and with their initializations. Glossary.java class contains concepts their setting and retrieving with the setTerm(), getTerm(), setDefinition() and getDefinition() methods. Send_glossary.java creates an action and implements setsendDefinition() and getsendDefinition() methods. This class tells agent what to do.

The ontology in the *Expertorer* is designed to let the agent know what to do when the message arrives and to understand what it should do if a certain type of the message arrives. Agent would know that he should retrieve the definition if the query arrives and what actually the definition means. Therefore the whole expression will look like that:

(action
(Send_glossary
(Glossary : term "Software design "
: definition "It is a process of problem solving and planning for a software solution")))

3.4 Graphical User Interface

Here the user can ask the agent to look for a particular definition of a supplied term. Text box will provide the user with possibility of entering term and after clicking on the "Search" button the Definition label should show the definition(s) found by the agent. The last text box will provide the user with information about the search operation (the result was found in local or remote database, and if it is remote then the agent was informed by whom?).

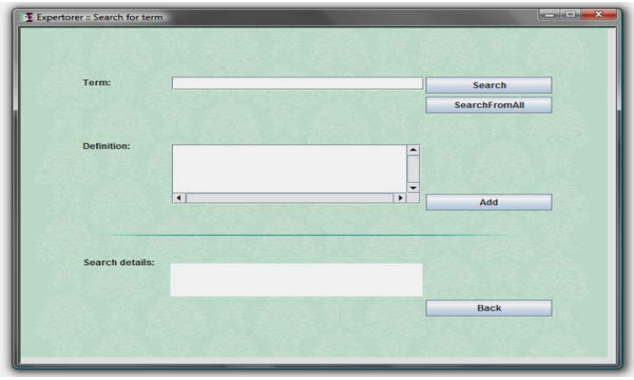

Fig. 4 Search entry part of the agent GUI.

4. Related Works

A lot of research and commercial organizations are involved in the realization of agent applications and a considerable number of agent construction tools have been realized. Some of the most interesting are:

- AgentBuilder is an integrated tool suite for constructing intelligent software agents [10].
- AgentTool is a Java-based graphical development environment to help users analyze, design and implement multi-agent systems [9].
- Bee-gent is a software framework based on multi-agent systems [1].
- FIPA-OS (FIPA Open Source) is an open agent platform originating from Nortel Networks [12].
- The Open Agent Architecture (OAA) is a distributed application framework that promotes a new programming paradigm called Delegated Computing [4].

5. Conclusion

Multi-agent systems will become more widely used in the future as they make available many useful things for the Internet and for the businesses. However the standards need to be developed for allowing those systems to cooperate with each other and with the outside environments. Multi-agent systems are often very complex applications with many modules and issues that need to be taken into account. However in this work, we have implemented an agent that would be functional for the user and that would be easy to create and use.

The agent consists of only one instance from which the multi-agent system can be created. The agent is sender and receiver at the same time and has the ability to create an instance of the database for each user. The system was build so that the agents cooperate with each other and share information. This is not fully completed and still needs enhancement. The ontology could be extended with data handling for future help. Security is another very important issue that has to be taken into account in the future. Each agent would have to have its information secured so that agent from outsider could not access it.

References

[1] Bee-agent Knowledge Media Laboratory Corporate Research & Development Center TOSHIBA Corporation. 2005. http://www.toshiba.co.jp/rdc/beegent/index.htm.
[2] Bellifemine. JADE: A Java Agent Development Framework. In "Multi-Agent Programming: languages, platforms and applications". Springer 2005.
[3] Bellifemine et al. Developing Multi-Agent Systems with JADE, Wiley. 2007.
[4] Cheyer A. & Martin D. The Open Agent Architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4 , no. 1, March 2001, pp. 143-148.
[5] Daconta M., Obrst L. & Smith K. The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management. Wiley. 2003.

[6] Denzinger J. *Ontology-Guided Learning to Improve Communication between Groups of Agents*, Proc. AAMAS 2007, Hakodate, 2007, pp. 923-930.
[7] FIPA, FIPA-Query Interaction Protocol, Standard Number SC00027H, FIPA TC Communication, 2002.
[8] Hadzic M., Chang E., Wongthongtham P. & Dillon T. Ontology-based Multi-agent Systems, Springer 2010.
[9] Juan C., DeLoach, & Robby. AgentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. Proceedings of the 24th Annual ACM Symposium on Applied Computing, USA. March 2009.
[10] Ling T. AgentBuilder. http://www.limsi.fr/~jps /enseignement/examsma/2005/1plateformes _3 /index-Ling.html
[11] Padgham L. & Winikoff M. Developing Intelligent Agent Systems: A practical guide, John Wiley and Sons. 2004.
[12] Mihaela U., Mircea C. & Douglas N. A FIPA-OS based multi-agent architecture for global supply chain applications. IPMM 2001 International Conference on Intelligent Processing and Manufacturing of Materials, July 29-August 3, 2001.

Zina Houhamdi received the M.Sc. and PhD. degrees in Software Engineering from Annaba University in 1996 and 2004, respectively. She is currently an Associate Professor at the department of Software Engineering, Al-Zaytoonah University of Jordan. Her research interest includes Agent Oriented Software Engineering, Software Reuse, Software Testing, Goal Oriented methodology, Software Modeling and Analysis, Formal Methods.

Belkacem Athamena was born in Algeria. He received the M.Sc. and PhD. degrees in Computer Engineering and System/Software Modeling and Analysis from Annaba University in collaboration with UCL University, Belgium, in 1994 and 2004, respectively. He is currently an Associate Professor at the department of Software Engineering, Al Zaytoonah University of Jordan. His research interests include System/Software Modeling and Analysis, Multi-agent, Fuzzy Logic, Neural Networks, Petri Nets, UML, VVT, Formal Methods, Fault Diagnosis. He has published over 40 papers, chapter in books, and conferences.