Development of an Agent based E-barter System

Sebla Demirkol
International Computer Institute
Ege University
Bornova, Izmir 35100
TURKEY
sebla.demirkol@ege.edu.tr

Sinem Getir
International Computer Institute
Ege University
Bornova, Izmir 35100
TURKEY
sinem.getir@ege.edu.tr

Moharram Challenger
International Computer Institute
Ege University
Bornova, Izmir 35100
TURKEY
moharram.challenger@mail.ege.edu.tr

Geylani Kardas
International Computer Institute
Ege University
Bornova, Izmir 35100
TURKEY
geylani.kardas@ege.edu.tr

Abstract— **A barter system is an alternative commerce approach where customers meet at a marketplace in order to exchange their goods or services without currency. In order to cope with challenges of electronic barter systems (e.g. efficient management of trades and determination of best-matching goods and their suppliers), several software systems based on intelligent agents have been proposed. However only a very few of these proposals consider exact development and implementation of multi-agent e-barter systems. Besides, most of the approaches consider matching between exchanged goods in which only price and quantity information are used. Hence, in this paper, we discuss design and implementation of a multi-agent e-barter system which utilizes ontology-based comparison for bid matching. Formal representation and decision-making criteria for agent mediated barter process are given and behavioral model of collaborating agents within the system is described. In addition to the traditional e-barter members, a new type of software agent is introduced in order to infer about semantic closeness between offered and purchased items. Related approach may enhance capabilities of e-barter systems in the way of finding the most appropriate matches between supplies and demands, considering not only price and quantities for goods.**

Keywords-software agent; multi-agent system; e-barter

I. INTRODUCTION

A barter system is an alternative commerce approach where customers meet at a marketplace in order to exchange their goods or services without currency. In barter marketplaces, amount of purchased goods or services are paid by manufactured goods or offered services. Electronic barter (shortly called e-barter) systems provide exchange of supplies in more reliable and useful way by using the capabilities of computers and Internet. Although e-barter systems seem similar to well-known e-commerce systems, in contrast with the usual meaning of e-commerce, transactions do not necessarily contain the exchange of money [1]. Money can be one of these goods.

In order to cope with challenges of e-barter systems (such as efficient management of trades and determination of best-matching goods and their suppliers), various studies exist (e.g. [2], [3], [4]) which propose agent based solutions. Considering autonomy, reactivity, pro-activeness and social ability of intelligent software agents, use of Multi-agent Systems (MAS) both in modeling e-barter markets and exact implementation of software systems in real-world e-barter applications is reasonable. In a MAS, agents interact with each other and these interactions can be either cooperative or selfish [5]. In other words, agents can share a common goal or they can pursue their own interests. That nature of MASs exactly fits to the requirements of e-barter applications.

Although existing studies include noteworthy proposals on agent based market simulation and/or researchers discuss theoretical design issues of agent based e-barter systems (such as learning, prediction, planning and behavioral model), unfortunately only a very few of them consider exact development and implementation of multi-agent e-barter systems within the software engineering perspective and guides to the engineers for developing such software systems starting from the scratch. In order to fill this gap, in this paper, we discuss development of a multi-agent based e-barter system by taking into consideration of software design and implementation with a widely-used MAS software development framework.

On the other hand, matching of most appropriate goods (or services) with required ones directly affects result of trades inside an e-barter system. In most of the e-barter applications, only amounts and exact (or direct) correspondence between exchanged goods are taken into account. That means a trade can not be realized until an exact counterpart is found for goods to be purchased. We think that some sort of *semantic* relations can be established between exchanged goods (or services) and these relations can be used during matching of the offered and purchased goods in order to find most appropriate counterparts in case of no exact matching exists. For instance, let us suppose that goods with type A are determined as directly exchangeable with goods with type B in a system. Also a type called C is defined as semantically related with type B. So, if a customer needs to exchange his goods with type A in the system and at the time of his request there does not exist any customer who sells goods with type B but there exists some customers selling goods with type C, system can automatically suggest our customer to negotiate with the customers selling goods with type C due to the reasoning on the predefined relation between types B and C.

To provide more than direct matching between exchanged goods, we propose a new member for agent based e-barter systems in this paper. That new member of the system use

Semantic Web [6] technologies in order to infer about semantic closeness between offered and purchased items based on the defined ontologies. We describe how such a system member can be modeled as a software agent and discuss interactions of this agent with other agents during barter trades.

Rest of the paper is organized as follows: Related work is given in Section 2. Section 3 discusses requirement analysis and derivation of software agents for constructing the MAS. E-Barter agents with their behavioral model and agent interactions are discussed in Section 4. Section 5 covers implementation of the proposed system and Section 6 concludes the paper.

II. RELATED WORK

The system, introduced in [2], considers transaction and shipping costs of e-bartering. Customer Agents in the system have a utility function that represents the preferences of the customers and basket of resources indicating the items that they own. Another work [7] intends to provide the integrity between formal specification and a suitable design for an e-barter system.

There are also some other works about generation of automated test scenarios for e-barter systems. For instance, a specification and the application of a method and tool for test scenario generation for an e-barter system are introduced in [8]. In [4], authors propose a barter double auction which uses a MAS to represent the related barter system.

A general research about multi-agent learning [9] is related to our work since it focuses on a case study of bartering with MASs. Researchers apply case-based reasoning to their MAS and try to overcome problems that derived from distributed data over a collection of agent. Another work [3] is about formal specification of multi-agent e-barter systems. Authors intend to specify and analyze e-barter systems with a formal language that is based on classical process algebra. This is analogous to our work since we also study properties of e-barter systems represented in a specific notation. In our work, agents are indicated with some functions to simulate the preferences and tolerations of the customers that they represent.

A knowledge-based approach to e-barter trading is given in [10]. Researchers focus on an approach to find most appropriate matches between supplies and demands, considering not only price and quantities. It is aimed to consider semantic similarity between bid descriptions. It is also proposed the use of a logical language to express agent preferences. That common feature of both [10]'s and our proposals differs from the abovementioned studies and enhances efficiency of the barter transactions. We also believe that our study promotes the approach in [10] by practical implementation of semantic matching required during the barter trade.

III. AGENTIFICATION OF THE E-BARTER SYSTEM

In order to model organizational view of a MAS, it is vital to determine agents and roles, goals and tasks of agents in the system based on the system requirements. An agent-based e-barter system consists of agents that exchange goods or services of owners corresponding to their preferences. These agents perform their tasks considering the leverage of owners to procure needed products by the exchange of offered services or manufactured goods of others. Every agent in the MAS is named with the inspiration of his active role that he/she plays in the application [11]. Likewise, in our proposed system, the *Customer Agents* are responsible for adding and evaluating barter proposals. The *Barter Manager Agent* manages all trades in the system. This agent is responsible for collecting barter proposals, matching proper barter proposals and tracking the bargaining (bartering) process between Customer Agents. To provide semantic matching between exchanged goods, Barter Manager asks for an agent called *Semantic Web Service Agent* (shortly we call *SWS Agent*) to infer about semantic relation between exchanged goods based on his knowledgebase. After finalization of bargaining, Customer Agents send engagement message to the Barter agent. Then, the Barter agent notifies the *Cargo Agent* for transporting barter products between Customer Agents. Any further agents and their roles can also be defined (e.g. a *Bank* agent manages the payment transactions). To sketch out the organizational structure of the system, we group system's agents in two categories: First group is Service Management Agents Group. Cargo Agent, Barter Manager and SWS Agent are the members of this group. The second group is User Agents Group. Customer 1, Customer 2 … Customer N agents constitute this group. The groups are shown in Fig. 1. Service management group provides services for customers in user agent group. Meanwhile customer agents are negotiating with each other.

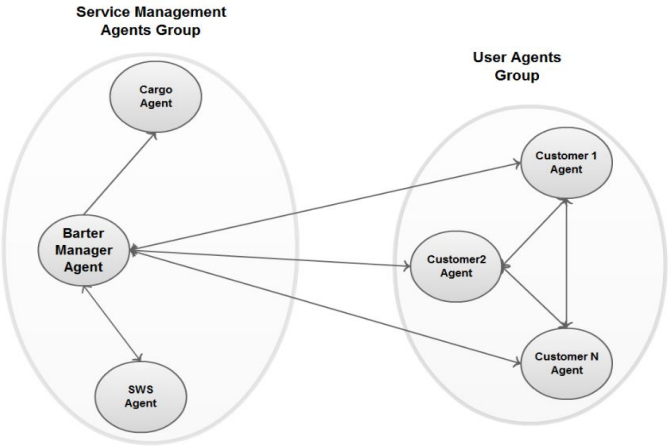

Figure 1. Organization view of multi-agent e-barter system

Let us analyze the e-barter system with formal representations. Suppose that the system has k agents and n different goods. Assume that a subset of agents will be willing to make exchange. We can formally consider that $A=\{i_1,...,i_m\}$ is the set of agents and quadruple *(S, B, ts, tb)* is a barter transaction for any $i \in A$. The first component of the quadruple denotes the *selling value of the good*. The second one denotes the *buying value of the good* and the last two components denote the *percentage of tolerance* for selling and buying respectively. Exchanges will occur if the value of the candidate good is more or less than the tolerated value. For instance, the agent can sell the related item at least an amount of (S – S * ts / 100) and buy at most an amount of (B + B * tb / 100).

IV. System Agents and Their Interaction Model

After determination of system agents and their relations inside the big picture, next step is to define and elaborate behavior model of the agents. Internal planning mechanism of each agent and coordination and communication between other agents should be given. In the following subsections, every system agent is discussed within this context. Last subsection discusses whole interactions between system agents and their coordination.

A. Barter Manager Agent

Four different behaviors are defined for Barter Manager Agents: *ProposalManager* behavior, *SWSandMatchingManager* behavior, *ResponseManager* behavior and *WaitForOtherManager* behavior. Some of these behaviors can be triggered by others. For example, ProposalManager and ResponseManager behaviors exist in the system from the beginning. However SWSandMatchingManager and WaitForOtherManager behaviors are created by existing behaviors. Behavioral model of Barter Manager Agent is given in Fig. 2. Figure shows main behaviors of the agent and tasks composed by these behaviors.

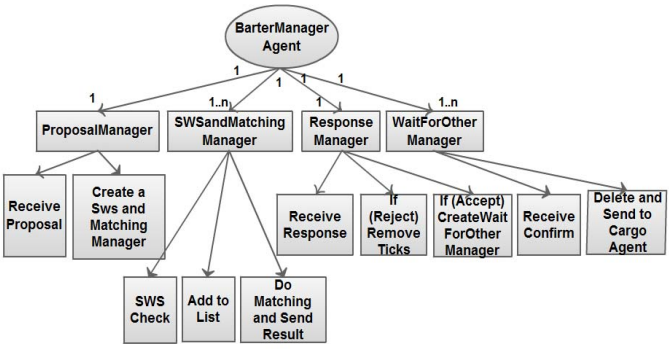

Figure 2. Behavioral model of Barter Manager Agent

ProposalManager behavior provides interaction between Barter Manager and Customer. It collects proposals, matches them and keeps track of bargaining between customers. Every proposal contains tolerance degree of the goods. Selling value and buying value can be indicated with the proposal or checked by EvaluateManager behavior of the Customer Agent. ProposalManager behavior awakes when the customer offers or requests something. Each time it starts, it receives a proposal and then creates an instance of SWSandMatching behavior. There is only one ProposalManager behavior instance for every Barter Manager, because one behavior will get all the requests or offers.

ProposalManager behavior and SWSandMatchingManager behavior are separate behaviors. If there was only one behavior for doing these two tasks, customers would have to wait until the matching finishes. As a result, a customer can offer or request consecutively. Thus the SWSandMatchingManager behavior is also necessary for the system. This behavior realizes the Semantic Web Service checking, addition of the goods, and then matching. It is created by the ProposalManager behavior. There can be n instances of that behavior during runtime of the Barter Manager Agent.

As depicted in Figure 2, ResponseManager has three tasks. First task is *ReceiveResponse* task. It receives the result of negotiation from the first customer. The other two tasks are separated by the result of negotiation. The system provides verifying the results by asking both of the customers. If both of the customers accept negotiation, an instance of *WaitForOtherManager* behavior is created.

After Barter Manager Agent finishes matching, evaluation starts between two customers. Consequently, matched customers will be ticked. It means that they are making evaluation now. For instance, Table 1 shows a matching table for a bartering process. A customer, numbered as 5, offers goods with type A, and requests goods with type B. On the other hand, a customer, numbered as 9, offers type B and requests type A. At the beginning, the evaluation values are NULL. It shows that customers are not busy. In other words, they are not evaluating anything.

TABLE I. Goods Table at the Beginning

Customer	Supplies	Demands	Evaluation
5	A	B	NULL
9	B	A	NULL

If customer 5 and 9 can be matched then the NULL field will be ticked by signing the customers with their peers (as shown in Table 2). At the end of the evaluation, if the result is accept, both of the customers will be removed from the table. If the result is rejected, Barter Manager will try to match customer 5 with the other customers, and customer 9 with the others.

TABLE II. Goods Table after Matching

Customer	Supplies	Demands	Evaluation
5	A	B	9
9	B	A	5

WaitForOtherManager behaviour, created by ResponseManager behaviour, has two tasks. The first one is *ReceiveConfirm*. The other one is *DeleteAndSenttoCargo*. Again, there can be n WaitForOtherManager behaviours.

B. Customer Agent

There can be any number of Customer Agents in our system. A customer agent has two behaviours; *RequestOfferManager* and *EvaluateManager*. RequestOfferManager can be triggered by a customer over a GUI (Graphical User Interface) application. EvaluateManager behaviour is created by RequestOfferManager after it completes his tasks. Behavioral model of Customer Agent is given in Fig. 3.

RequestOfferManager behavior provides interaction between Barter Manager and Customer Agents and is composed of three tasks. First task is *DoRequestorOffer*. If customer sends (A, B) it means the customer wants to give A and get B. Both of the offer and request are defined in this transaction. If customers send (A, X) this means that, customer

wants to give A and wait for anything. It is represented with (A, X) since, customer has decided what to give but, has not decided what to want. The second task is *CommunicationwithBarterManager*. In this task, customer's offer or request is recorded by BarterManager. The last task is *ReceiveaProposeandCreateEvaluate*, it receives a respond and creates EvaluateManager behavior. A Customer Agent has n separate RequestOfferManager behaviors. Because customers make n requests or offers and each task works for different request-offer pairs.

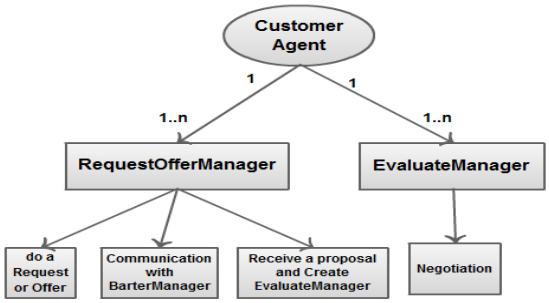

Figure 3. Behavioral model of Customer Agent

Within EvaluateManager behavior, a customer agent interacts with other customer agents. This behavior has the *Negotiation* task. Negotiation includes checking the quality of service and searching for the cost. Before the negotiation, BarterManager gives customers sequential numbers. Thereby it indicates customers which one will start to communicate first. Considering the formal representation of the negotiation, suppose that the information associated with the agent $i \in A$ is given by the quadruple (S_i, B_i, ts_i, tb_i) and there exists a customer j that SWSandMatchingManager behavior of the BarterManager Agent has matched. Suppose, $j \in A$ and have the information (S_j, B_j, ts_j, tb_j). During negotiation, EvaluateManager behavior of each customer agent checks the following inequalities:

$$Bo_j \leq (S_i - S_i * ts_i / 100) \quad (1)$$

$$Bo_i \geq (B_i + B_i * tb_i / 100) \quad (2)$$

$$Bo_i \leq (S_j - S_j * ts_j / 100) \quad (3)$$

$$Bo_j \geq (B_j + B_j * tb_j / 100) \quad (4)$$

Bo denotes amount of each agent's *buying offer*. Agent i should check whether the buying offer of Agent j (Bo_j) is within the range of Agent i's acceptable selling amount. To do this, Agent i uses (1) during the negotiation process. In each iteration of the negotiation, Agent j may change his buying offer. However, he also controls his offer such that it should not be more than his maximum affordable amount. Hence, Agent j uses (4) to determine his next buying offer (Bo_j). Considering the counterpart of the negotiation, Agent i uses (2) to determine his next offer to buy goods sold by Agent j and Agent j uses (3) to decide whether buying offer of Agent i (Bo_i) is acceptable.

C. *Semantic Web Service Agent*

The main task of Semantic Web Service Agent (shortly SWS Agent) is to infer about semantic closeness between offered and purchased items based on the defined ontologies. In other words, it supports semantic bid matching.

SWSManager behavior of SWS Agent provides a decision mechanism to categorize the objects. For instance, if a customer agent looks for a kind of meat and if perch is already registered to the system; SWS Agent decides that perch is a fish, and fish is a meat. Thus, it is able to catch the matching beyond the categorization.

SWS Agent discovers and executes Semantic Web Services (SWS) which provide semantic matching of ontological concepts. Semantic Web Services are web services with semantic interface to be discovered and executed. Capabilities of such web services are defined in service ontologies such as OWL-S [12]. Hence, when SWS Agent of our system receives a match request from Barter Manager, he first discovers appropriate SWS (e.g. an OWL-S service) and executes that service to support inference on the given ontological concepts.

D. *Cargo Agent*

As it name already depicts, Cargo Agent transmits the exchanged goods among the customer agents. Based on the description of his *TransmitManager* behavior, Cargo Agent achieves transmission requests after a successful barter operation and realizes real exchange of the bartered goods for the customer agents.

E. *Modeling the Interaction between System Agents*

After each individual e-barter agent and his behavior are given, it is worth discussing collaborations between these agents and coordinated execution of their behaviors. Fig. 4 depicts the collaboration between system agents during an e-barter transaction.

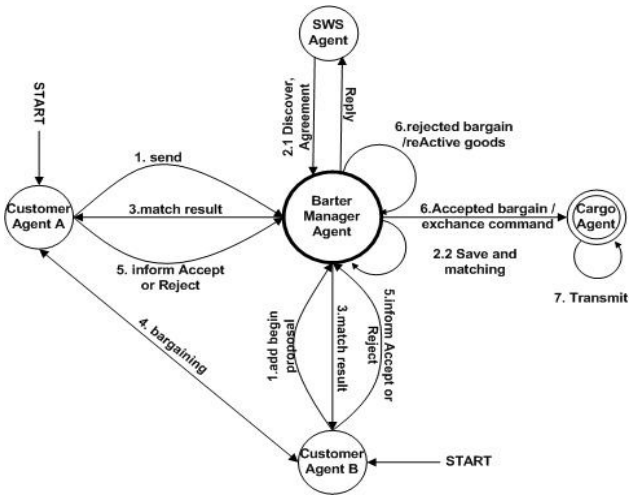

Figure 4. Interactions between e-barter agents

Customer Agent A (shortly Customer A) sends a proposal to the Barter Manager Agent (interaction 1 in Fig. 4). Barter Manager calls SWS Agent to discover service of goods, makes an agreement and executes the service (interaction 2.1 in Fig. 4). SWS Agent makes agreement and replies to the Barter Manager. Then Barter Manager determines appropriate customer(s) for the given proposal. In other words, he realizes matching (interaction 2.2 in Fig. 4). After matching is completed, Barter Manager sends match results to Customer A (interaction 3 in Fig. 4). Customer A finds the corresponding customer (Customer B) and begins to negotiation (or bargaining) (interaction 4 in Fig. 4). As soon as bargaining finishes, customers inform Barter Manager (interaction 5 in Fig. 4). Barter Manager receives the result and makes a decision. If the negotiation fails, Barter Manager reactivates goods (interaction 6 in Fig. 4). If the negotiation succeeds, Barter Manager gives exchange and transmission command to Cargo Agent (interaction 7 in Fig. 4). Finally, Cargo Agent transmits goods to Customer Agents.

Fig. 5 shows execution of agent behaviors during an e-barter transaction. At the beginning, RequestOfferManager sends a proposal via a GUI. Then, ProposalManager receives the proposal and creates a SWSandMatchingManager behavior instance. This instance checks for a SWS (message passing numbered 1 in Fig. 5) and receives check results from SWSManager (message passing numbered 2 in Fig. 5). After receiving results, SWSandMatchingManager sends match result to EvaluateManager (message passing numbered 3 in Fig. 5). EvaluateManager is activated during the negotiation with the other customer. After negotiation is finished, EvaluateManager creates ResponseManager. Response Manager receives an accept message and creates WaitForOtherManager to ensure the accuracy of the result. When WaitForOtherManager receives confirmation from both of the customers, it gives exchange command to TransmitManager. Finally, TransmitManager realizes exchange of the goods.

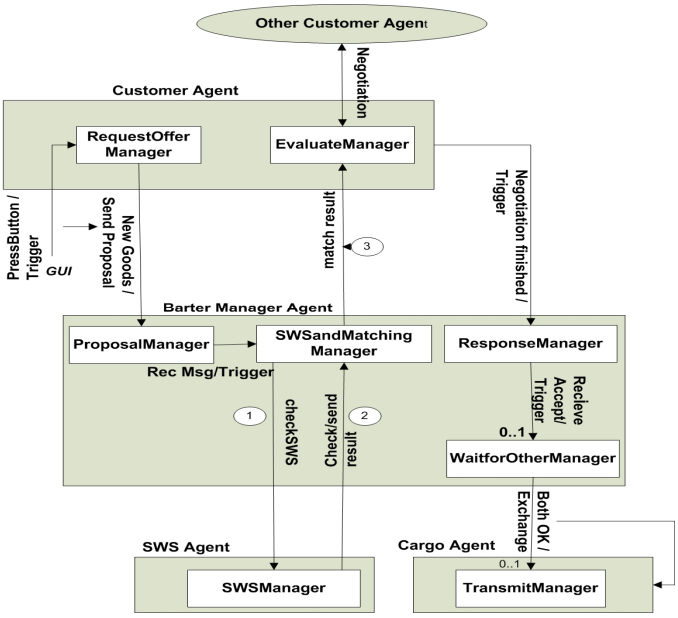

Figure 5. Behavioral perspective of an e-barter transaction

V. IMPLEMENTATION OF THE E-BARTER SYSTEM

The multi-agent e-barter system, proposed in this paper, has been implemented by using the JADE framework. JADE [13] is currently one of the most widely used Java-based MAS development software frameworks. It simplifies the implementation through a middle-ware that complies with the IEEE Foundation for Intelligent Physical Agents (FIPA) specifications [14] and through a set of graphical tools that supports the debugging and deployment phases.

Although it is impossible to cover full implementation of our system in here, implementation of two crucial agent behaviors are briefly discussed to give some flavor of how we utilize behavior and FIPA Agent Communication Language (ACL) messaging libraries of JADE.

SWSandMatchingManager is the most important behavior of Barter Manager Agent. It extends the JADE *Behavior* class and is activated when it accepts a proposal message. SWSandMatchingManager can send a SWS discovery request, wait for receive SWS reply, save new goods and do matching search. A switch-case selection structure may supply these tasks. So, the *action* method of this behavior has been written as it includes this structure. Following is a code fragment from the written class for SWSandMatchingManager behavior with including appropriate comments:

```
public class SWSandMatchingManager extends Behaviour {
...
public void action() {
   switch(currentStep){
    case 0:
     // sending a SWS discovery request
     msg.setPerformative(ACLMessage.REQUEST);
     msg.addReceiver(myOwnerAgent.SWSandMatching);
     msg.setContent(toTake+" "+toGive);
     myAgent.send(msg);
     mt=MessageTemplate.and(
           MessageTemplate.MatchPerformative(
           ACLMessage.INFORM),
           MessageTemplate.MatchSender(
           myOwnerAgent.SWSandMatching));
     currentStep = 1
     break;
    case 1:
     // wait for receive SWS reply
    case 2:   // save new goods
     // extract SWS info for goods from message
     Record r = new Record();
     // insert info to fields of new record
        myOwnerAgent.BargainList.addElement(r);
        currentStep = 3;
      break;
    case 3:
      // do matching search
      boolean matchResult=true;
      // will be set according to matching search
      CustomerB = null;
      // will be set according to matching search
      if (matchResult){
          ACLMessage newMsg = new
          ACLMessage(ACLMessage.PROPOSE);
          newMsg.addReceiver(CustomerA);
          newMsg.setContent(CustomerB+"A");
          // sending type of receiver agent and his partner
          myAgent.send(msg);
          newMsg = new
          ACLMessage(ACLMessage.PROPOSE);
```

```
            newMsg.addReceiver(CustomerB);
            newMsg.setContent(CustomerA+"B");
            // sending type of receiver agent and his partner
            myAgent.send(msg);
            endOfBehaviour = true;
          }
          break;
      }
    }
```

Another important behavior is SWSManager of SWS Agent. That behavior has been implemented as a JADE *CyclicBehaviour*. Referring back to Fig. 5, SWSManager behavior includes a communication with Barter Manager Agent during barter matching. SWSManager receives a message from BarterManager and sends an inform message to Barter Manager for ontology-based comparison of the bartered goods. Following is a code fragment from the written class for SWSManager which demonstrates ACL messaging of SWS Agent with Barter Manager for this purpose.

```
public class SWSManager extends CyclicBehaviour {
  …
  public void action() {
    MessageTemplate mt =
    MessageTemplate.and(MessageTemplate.MatchSender(
      BarterManager), MessageTemplate.MatchPerformative(
      ACLMessage.REQUEST));
    ACLMessage msg = myAgent.receive(mt);
    // extract msg info for goods
    // Do SWS Discovery using his ontology engine
    // returning result
    ACLMessage reply = new
      ACLMessage(ACLMessage.INFORM);
    reply.addReceiver(msg.getSender());
    reply.setSender(myAgent.getAID());
    reply.setContent("SWS result");
    myAgent.send(reply);
  }
}
```

VI. CONCLUSION

Design and implementation of a multi-agent e-barter system are discussed. Formal representation and decision-making criteria for agent mediated barter process are given and behavioral model of collaborating agents within the system is described. Perhaps the major advantage of the proposed system is the semantic web enabled bid matching during barter trades. In addition to the traditional e-barter MAS members, a new type of software agent is introduced and employed inside the proposed MAS in order to infer about semantic closeness between offered and purchased items based on the defined ontologies. Related approach may enhance capabilities of e-barter systems in the way of finding the most appropriate matches between supplies and demands, considering not only price and quantities for goods.

Currently, we are working on the reorganization of the internal structures of e-barter agents according to the well-known Belief-Desire-Intention (BDI) agent architecture. That new organization will provide us to examine collaboration of the agents and compare throughput of the system with the current system based on the reactive behavioral model. Another planned future work is to develop mobile versions of the MAS proposed here in order to use the e-barter system in various mobile devices.

ACKNOWLEDGEMENTS

This study is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 109E125.

REFERENCES

[1] N. Lopez, M. Nunez, I. Rodríguez, and F. Rubio, “A formal framework for E-barter based on Microeconomic Theory and Process Algebras”, Lecture Notes in Computer Science, vol. 2346, pp. 217-228, 2002.

[2] N. Lopez, M. Nunez, I. Rodríguez, and F. Rubio, “A multi-agent system for e-barter including transaction and shipping costs", In proc. of the 2003 ACM symposium on Applied computing (SAC'03), Melbourne, FL, USA, pp. 587-594, 2003.

[3] M. Nunez, I. Rodríguez, and F. Rubio, “Formal specification of multi-agent e-barter systems", Science of Computer Programming, vol. 57, issue 2, pp. 187-216, 2005.

[4] R. Tagiew and Y. Kovalchuk, "Barter double auction as model for bilateral social cooperations", In proc. of the 1st Computer Science and Electronic Engineering Conference (CEEC'09), Colchester, UK, 2009.

[5] K. Sycara, "Multiagent systems", AI Magazine, vol. 19, issue 4, pp. 79-92, 1998.

[6] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, vol. 284, issue 5, pp. 34-43, 2001.

[7] M. Bravettia, A. Casalboni, M. Nunez, and I. Rodríguez, "From theoretical e-barter models to an implementation based on web services", Electronic Notes in Theoretical Computer Science, vol. 159, pp. 241-264, 2006.

[8] A. Cavalli and S. Maag, "Automated test scenarios generation for an e-barter system", In proc. of the 2004 ACM symposium on Applied computing (SAC'04), pp. 795-799, 2004.

[9] S. Ontanon and E. Plaza, "A bartering approach to improve multiagent learning", In proc. of the 1st international joint conference on Autonomous agents and multiagent systems: part 1 (AAMAS'02), Bologna, Italy, pp. 386-393, 2002.

[10] A. Ragone, T. Di Noia, E. Di Sciascio, and F. M. Donini, "Increasing bid expressiveness for effective and balanced e-barter trading", Lecture Notes in Computer Science, vol. 5397, pp. 128-142, 2009.

[11] G. Kardas, E. E. Ekinci, B. Afsar, O. Dikenelli, and N. Y. Topaloglu, "Modeling tools for platform specific design of multi-agent systems", Lecture Notes in Artificial Intelligence, vol. 5774, pp. 202-207, 2009.

[12] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, (last access: February 2011).

[13] F. Bellifemine, G. Caire, and D. Greenwood, Developing Multi-agent Systems with JADE, San Francisco, CA, USA: John Wiley and Sons, 2007.

[14] IEEE FIPA Specifications, http://www.fipa.org/specifications/, (last access: February 2011).