agent systems, modeling, simulation, e-commerce, implementation, distributed intelligence

Krzysztof CHMIEL[*], Dariusz CZECH[*], Marcin PAPRZYCKI[#]

AGENT TECHNOLOGY IN MODELLING E-COMMERCE PROCESSES; SAMPLE IMPLEMENTATION

It is often claimed, that intelligent agents are to play an important role in bringing intelligence to the networked world of information. While these claims resurface continually since at least 1994, they do not seem to be substantiated through the development and implementation of large scale agent systems (in terms of both types and sheer number of agents co-existing in the system). While there are a number of possible reasons why this may be the case, in this paper we present an attempt at remedying this situation. We have developed an e-commerce framework and implemented it using the current state-of-the-art agent platform (JADE). This agent framework can be utilized to experiment with development and implementation of distributed intelligence in context of e-commerce scenarios. Furthermore, the proposed tool can be extended to actually model factors pertinent to the world of e-business. In this paper we summarize main features of the developed framework.

1. INTRODUCTION

While origins of agents can be traced much further back in time, it can be said that it was 1994 when they became an important factor shaping programming landscape. It was then that P. Maes has published her seminal paper '*Agents that Reduce Work and Information Overload*' [1], in which she has presented a vision of intelligent software agents helping humans in dealing with important problems in the world of networked computers brimming with information. Interestingly, the idea of *intelligent* agents is not the only one in existence. There exists also a branch of agent system research that follows the biological metaphors and can be traced to, among others, work of Feber [2] and Liu [3]. Here, agents are understood mostly in terms of *extremely primitive* "beings" and intelligence is an emergent property of the system. While this approach is very interesting and leads to a large number of important theoretical and practical

[*] Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poland.
[#] Department of Computer Science, Oklahoma State University, USA, Department of Computer Science, SWPS, Warsaw, Poland, marcin.paprzycki@swps.edu.pl

results, in this paper we are interested in applications of *intelligent* agents. It has to be stressed, that while P. Maes has provided us with the metaphor and the scope of intelligent agent systems, there exists a problem with their all-agreed definition. A study of literature revealed a widely-varied array of features which are used to describe software agents [4]. This being the case, we will try to avoid the question of a precise definition and proceed with a general notion of an intelligent agent as an autonomous entity that represents its owner and acts in an environment consisting of other agents as well as other programming artifacts (databases, web-sites etc.). We will also assume that regardless of the criticism presented, among others, by H. Nwana and D. Ndumu [5], there is a great potential in agent systems. We will rather focus on the positive program introduced in [5], which can be summarized that for the agent systems to reach their potential developers must go beyond theoretical frameworks and generalized diagrams and start implementing actual agent systems (see also [10, 11, 12] for additional methodological considerations). Pursuing this goal, we have experimented with the JADE agent platform [13] and established that there are no limitations in the agent platform itself that would prevent large agent systems to be implemented [14]. Running experiments on outdated hardware with limited memory we were able to scale JADE to more than 1000 of agents and more than 40,000 messages and concluded that the primary limitation is the computer hardware itself.

Knowing that it is possible to implement large agent systems, we have proceeded to develop and implement a skeleton system that can be used in two capacities. First, it can be easily extended to support studies in distributed AI, in particular intelligent agent systems for e-commerce (by encapsulating intelligence into agents). Second, it can be utilized in modeling of various features of actual e-commerce systems (by modeling various "behaviors" of buyers and sellers, i.e. pricing policies, customer preferences, discounting policies etc. and running the system for a large number of cycles). Our system has been implemented in JADE 3.1 [13], currently one of the most comprehensive FIPA [9] compliant agent systems. In the next section we describe the most important features of the existing system. We follow with a number of comments that are to clarify some of the decisions that shaped the system design and present directions of its future development.

2. SYSTEM DESCRIPTION

2.1. AGENTS IN THE SYSTEM

Our goal was to set up a skeleton of an e-commerce system that can be easily further expanded to allow studies in development of distributed intelligent systems as well as experimental modeling of e-commerce scenarios. In pursuing this goal we have simplified the e-commerce system structure presented in [6]. In our model we have the following types of agents:

- Simulation management agents:
    - General simulation manager
    - Simulation information gathering agent
- Actual simulation agents
    - Client information center agent
    - Client agents which represent clients, their buying requests and behaviors
    - Shop agents for representing e-stores and their policies
    - Seller agents (parts of an e-store) that negotiate with client agents
    - Database agents which secure access to actual databases

Let us now describe general scheme of the implemented system and use it to discuss in more details each of these agents separately. Let us start from the two managerial agents which do not take part in actual simulations:

General simulation manager agent – is receiving all of the simulation parameters, generates an appropriate number of JADE containers on the specified computers, and initiates the simulation.

Simulation information gatherer agent – during the simulation this agent collects information about various events happening during its course e.g. which stores have more clients, which stores sell-off their good faster, which stores have run out of goods to be sold etc.

Let us now devote our attention to agents taking part in the simulation. The most important parts of the system – agents and their interactions (for simplicity, depicted without the two managerial agents) – are presented in Figure 1.

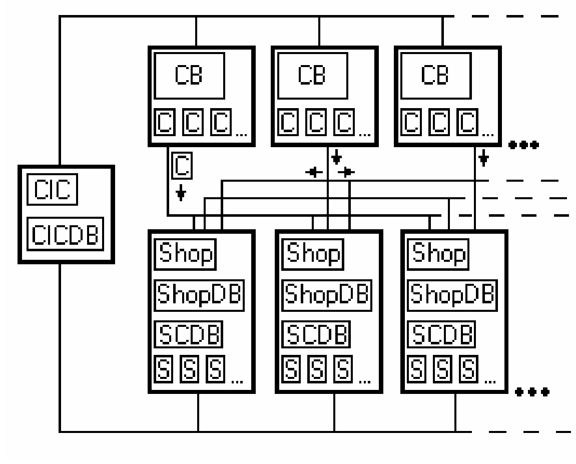

Fig. 1. General schema of the system

Client information center agent (one such agent in the system) – denoted as *CIC* – stores, manages and provides information about participants. For instance, it registers new shops and clients. Thus, if a shop or a client agent wants to participate in the system it must communicate with the *CIC* agent first. The *CIC* agent works on the basis of the information stored in the *CICDB* database. There are two types of information stored there: (1) unique id for all clients (client registry) and (2) information about of all e-shops known to the system (yellow pages). Thus any *client base* agent (new or returning) has to communicate with the *CIC* agent to find out which stores are available in the system at a given time. In this way we are following the general philosophy of agent system development, where each function is embodied in an agent (observe that it would be possible for the *Shop* and *Client* agents to access the *CICDB* directly). Furthermore, this provides us with a simple mechanism of handling access to a single repository without dealing with problems of mutual exclusion etc. All these problems are handled by the JADE through its servicing of the inter-agent communication.

Client Base agents (one agent for each "user" of the system) – denoted *CB* – represent particular requirements, preferences, buying policies etc. of individual users. After the *CB* agent is created it registers at the *CIC* agent and obtains a user id (which is stored in the *CICDB*). Upon receiving a request (from the user – in our case a simulated user) it communicates with the *CIC* agent to obtain list of available shops, creates *Client* agents and sends them to these shops to seek products to purchase. Upon receiving messages containing details of negotiated offers (from *Client* agents) it decides which one to accept (where to buy products requested by the user) and sends a purchase confirmation message to the proper *Client* agent. In the case of a successful transaction a given request is completed; in the case of a failed transaction the *CB* makes the second attempt at completing the request by selecting the next best offer and sending a message to the *Client* agent etc.

Client agents (number depends on the number of e-shops present in the system) – denoted as *C* – are created by the *CB* agent and sent to shop(s) with a list of products to purchase. Using its negotiation strategies *C* agent negotiates the best price with its *Seller* agent and sends results of negotiations to the *CB.* In the case of its offer being selected it attempts at purchasing products.

Shop agents (one agent in each e-chop) – denoted as *Shop* – represents the e-store (embodies all of the store policies, such as pricing strategies, individual discounts, volume discounts etc.). When created, *Shop* agent communicates with the *CIC* agent and registers in the *CICDB.* For each incoming *Client* agent it creates an appropriate *Seller* agent. This function is completed by utilizing the information stored in the shop certification database – denoted *SCDB*. *SCDB* stores information about clients that have already visited the store in the past, their transaction record etc. *Shop* agent also utilizes the *Shop database* – denoted as *ShopDB,* which stores information about available products, their prices etc.

Seller agents (number depends on the number of *Client* agents present in the store) – denoted as *S* –are generated for each *Client* agent visiting it. Using initial parameters obtained from the *ShopDB* and the *SCDB,* via the *Shop* agent, these agents perform price negotiations with their *Client* agents. They should be able to utilize different price negotiation (e.g. auction) protocols (e.g. English auction, Dutch auction etc.). Such multiple price negotiation protocols will allow seller agents to interact with clients that use them (different client agents may use different negotiation protocols – see [7, 8] for more details).

Database agents – for instance the *SCDB* agent – have been created following, again, the general agent system design approach, where each function is embodied by an agent. Here, we have implemented a database agent for each database in the system. In this way we separate the database and its implementation for the system itself (database can change, while the agent-agent interactions remain the same; only the database agent needs to be able to interact with the new database). Since their role is only auxiliary and is rather obvious, we omit them in what follows.

Finally let us note that all agent-agent interactions have been implemented using the FIPA compliant agent communication language ACL.

2.2. SAMPLE PURCHASE DESCRIPTION

To illustrate the work of the system, as well as the usage of ACL messaging, let us follow what happens in the e-store during the purchase made on user request. Obviously, a similar description can be made also for the (much simpler) operations happening within the "client side" of the system.

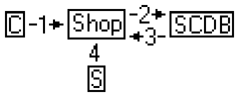

Fig. 2. Creation of an appropriate *Seller*

A. When the *Client* agent enters the store it introduces itself by its id (gained earlier by the *CB* agent from the *CICDB* via the *CIC* agent) and request from the *Shop* agent to create a *Seller* agent to work with. It is shown in Figure 2 as message 1 (messages are depicted as arrows).

B. *Shop* agent obtains necessary information (based on the *Client* id) from the *SCDB* agent (messages 2 and 3) and utilizes it to create a *Seller* agent with an appropriate negotiation strategy (step 4 in Figure 2).

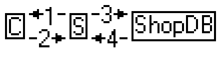

Fig. 3. Getting information about requested products

C. *Seller* agent introduces itself to the *Client* agent and receives its request (Figure 3, messages 1 and 2).
D. *Seller* agent then proceeds to obtain information about availability of requested products (messages 3 and 4) and makes its initial offer to the *Client* agent.

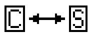

Fig. 4. *Client – Seller* negotiations

E. *Client* and *Seller* negotiate conditions of sale (Figure 4).
F. *Client* agent informs its *CB* agent (via an ACL message) about the negotiated pricing conditions and awaits request to proceed with purchase, or an order to end its existence.
G. *CB* agent selects the best offer and orders an appropriate *Client* agent to realize the order.

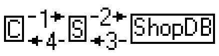

Fig. 6. Attempt at completing order

H. Selected *Client* agent attempts at completing the purchase (as depicted in Figure 6).
I. In case of failure (e.g. in the meantime goods might have been purchased) *Client* agent informs its *CB* agent about failure to allow it to select another *Client* to complete the transaction. If operation ends successfully *CB* is informed and communicates with its *Client* agents, which in turn communicate with their *Seller* agents. At this stage both the *CB* agent and the *Shop* agents update appropriate databases with details of the event (i.e. in Figure 7 a given *Shop* agent stores information about the negotiations with a given client and their result). Finally, all unnecessary agents are told to self-destruct through an appropriate ACL message send by the *CB* and the *Shop* agents.

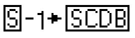

Fig. 7. Updating information about client after completing negotiations

To have this model completed and implemented one has to define content of messages to be exchanged in all situations described above. It is recommended to use protocols compatible with FIPA [9] standards which cover a number of needed communication scenarios (we have not done this completely at this stage of implementation of the system, but plan to do this in the near future). Here, let us present only an example of updating the information managed by the *SCDB* agent. As it is shown in Figure 7, upon completion of the transaction, the *Seller* agent sends an ACL message to the *SCDB* agent. This message is of type *inform* and its content could represent an amount of money spent by client with a given id, and in this case would have the following format: "ID(client_number) SALE(amount_of_money)". Obviously, as the system is developed further, to support realistic situations, the content of the message would need to be extended. It could contain, among others, details of the negotiation protocol and applied negotiation strategy (see also [7], [8]).

2.3. SIMPLE CASE STUDY

In previous sections we have described an agent system which can be used as a base for a large number of e-commerce (and, in general, commerce) simulations. We have designed our implementation in such a way, that the skeleton system can be easily modified by adding functional modules representing “actions” to be carried by agents (adding intelligence and / or functionality to the system).

While, at this stage of the project (see below), our goal was not to attempt at developing an intelligent and / or realistic system, we have decided to pursue our implementation slightly further and implement a demonstrator system modeling the following real-life scenario. Let us assume that “users” purchase PC parts on the Internet, while e-shops sell them. To be able to test the complete functionality of our system, we proceeded with a very simple (even simplistic) model and implemented straightforward rules guiding the behavior of individual agents. At the beginning of the simulation, a number of *CB* agents register in the *CIC* agent and a number of *Shop* agents do the same. Initially, e-shops hold randomly generated stores of PC parts. We have created a “*user generator*” agent that contacts *CB* agents with a sequence of requests for purchases of PC parts (representing a stream of users). Upon receiving a request *CB* agents attempt at fulfilling it by contacting the *CIC* agent and obtaining the list of currently available stores and generating an appropriate number of *Client* agents and sending them to these e-shops. Our *Client* and *CB* agents weren’t intelligent at all (as intelligence was not our goal) and always bargained for the lowest price and selected the best offer based only on the price factor. At the same time the discount policy of the shop depended only on the initial parameters and the previous transactions record of a given clients. Our negotiation protocol was also very simple – *Client* agent receives the initial price from the *Seller* agent and makes its bid; in the case the bid of the *Client* agent is not accepted it gradually increases the acceptable price (by halving the difference between the current buy-price and the initial bid obtained from the *Seller* agent) until its bid is accepted by the *Seller* agent (or until it is established that a common price level cannot be found).

2.4. SIMMULATION RESULTS

The above described simplified system works and in this way it verifies our general system design. We haven't observed any particular JADE-related problems during the implementation phase. Furthermore, reconfirming results reported in [14], our system is practically-scalable with the only limitation for number of shops and clients being the available RAM and the CPU speed. While our model is extremely simplified, especially when the behaviors of the *Client* and *Shop / Seller* agents are concerned, we have observed the following behaviors:

- when the number of clients and orders is small, buyers prefer shops with lower initial prices,
- when the number of clients is "medium" and from time to time delivery orders could not have been completed, some clients after buying in shops with higher initial prices became "attached" to them because of the discount policy that gave them significantly discounted price; the level of discount depended on the total amount of past purchases,
- when the total number of clients was large, many clients stopped buying at the "lower initial price shops" and preferred shops with best long-term cooperation discount policy (for the same reasons as the above),
- when clients floods shops almost immediately from the start, all goods have "evaporated" regardless of the discount policy and clients bought everything so that the largest profits have been generated by shops with highest prices.

Obviously, these results were predictable because all *Client* agents used identical purchasing strategies and neither of them adapted their behaviors over time. For instance, *Client* agents were incapable to modify their purchasing strategies, while *Shop* agents (thus e-shops) could not change their initial prices, or discount strategies (no supply and demand dependencies were implemented at all). At the same time, it is interesting, how many "rational" behaviors such a simplified system has captured. This indicates that the approach proposed here is correct. It is possible to model behavior of complex systems by applying agent-based decomposition and modeling on the level of individual agents.

3. CONCLUDING REMARKS

In this paper we have presented a skeleton agent system that can be used to model e-commerce systems. This system was implemented in JADE and experimented with using a simplistic PC-parts buying scenario. It is important to stress that, at this stage, our goal was not realistic modeling of e-commerce systems. Rather, we wanted to design, implement and test a framework system. In this system most of functionalities necessary for modeling e-commerce are already in place. We have fully functioning client side of the system, seller side of the system, and the communication infrastructure. At this stage we will be able to proceed in a number of directions: (1) more "intelligent" behavior of agents (primarily *Client*, *Shop* and *Seller* agents) – including more realistic and diversified auctioning strategies as well as diversified negotiation protocols, (2) proceed with large scale simulations which will include heterogeneous buyer and seller agents (different clients and stores using different policies and strategies), (3) introduce basic learning into the system to allow agents representing stores and clients to adapt their long-term behavior etc. Moving in this direction we will, first, modify agent-agent communication to make it fully compatible with FIPA standards. Second, we will proceed to introduce into the system support for ontologies to allow for additional experiments involving semantically oriented negotiations. We will report on these developments in the near future.

REFERENCES

[1] P. Maes, Agents that Reduce Work and Information Overload, Communications of the ACM, 37(7), 1994, 31-40

[2] J. Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999

[3] J. Liu, Autonomous agents and multi-agent systems, World Scientific, 2001

[4] V. Galant, J. Tyburcy, Inteligentny Agent Programowy, Prace Naukowe AE Wrocław, Nr 891, Wrocław, 2001, 46 – 57, in Polish.

[5] H. Nwana, D. Ndumu, A perspective on software agents research, The Knowledge Engineering Review, 14 (2), 1999, 1-18

[6] V. Galant, J. Jakubczyc, M. Paprzycki, Infrastructure for E-Commerce, in: Nycz M, Owoc ML (eds.), *Proceedings of the 10th Conference on Knowledge Extraction from Databases*, Wrocław University of Economics Press, 2002, 32-47.

[7] C. Nistor, G. Parakh, M. Paprzycki, Dynamically Loaded Reasoning Models in Negotiating Agents, *Proceedings of the 3rd European E-COMM-LINE 2002 Conference*, Bucharest, Romania, 2002, 199-203

[8] G. Parakh, S. Rani, M. Paprzycki, A. Abraham, J. Thomas, Agents Capable of Dynamic Negotiations, in: M. Paprzycki (ed.), *Electronic Commerce; Research and Development*, ACTEN Press, Wejherowo, Poland, 2003, 113-120

[9] FIPA, http://www.fipa.org

[10] M. Paprzycki, Developing Intelligent Agent Systems, in: R. Akerkar (ed.), *Artificial Intelligence and its Applications*, TMRF Press, Kolhapur, India, 2003, 1-3

[11] M. Paprzycki, A. Abraham, Agent Systems Today; Methodological Considerations, in: *Proceedings of 2003 International Conference on Management of e-Commerce and e-Government*, Jangxi Science and Technology Press, Nanchang, China, 2003, 416-421

[12] M. Paprzycki, Software Agents in the Real World, in: M. Bohanec et. al. (ed.), *Intelligent and Computer Systems*, Josef Stefan Institute Press, Ljubljana, Slovenia, 2003, 5-8

[13] JADE, http://jade.cselt.it/

[14] K. Chmiel, D. Tomiak, M. Gawinecki, P. Karczmarek, M. Szymczak, M. Paprzycki, Testing the Efficiency of JADE Agent Platform, Proceedings of the ISPDC 2004, to appear

TECHNOLOGIA AGENTOWA W MODELOWANIU HANDLU ELEKTRONICZNEGO; PRZYKŁADOWA IMPLEMENTACJA

W literaturze przedmiotu często pojawiają się stwierdzenia, że inteligentni agenci programowi mają do odegrania istotną rolę w tworzeniu nowej generacji oprogramowania dla połączonych w sieć i zalanych informacją komputerów. Wprawdzie twierdzenia te pojawiają się systematycznie co najmniej od roku 1994, nie wiążą się one implementacjami dużych systemów agentowych (gdzie duży oznacza tak liczę typów jak i liczbę rzeczywistych agentów występujących w systemie). Wprawdzie istnieje wiele możliwych powodów, dla których powyższa sytuacja ma miejsce, w poniższym tekście miast zajmować się problemami, prezentujemy pewną propozycję mającą na celu poprawę sytuacji. Prezentujemy tutaj szkielet agentowego systemu handlu elektronicznego zaimplementowanego w jednej z najlepszych istniejących platform agentowych – JADE. Zaprezentowane narzędzie może być wykorzystane w badaniach naukowych nad systemami rozproszonej sztucznej inteligencji (zastosowanymi w handlu elektronicznym). Równocześnie oprogramowanie to może zostać wykorzystane w modelowaniu czynników kształtujących e-biznes. W artykule podsumowujemy główne cechy zaimplementowanego narzędzia.