Combining Rule-Based and Plug-in Components in Agents for Flexible Dynamic Negotiations

Costin Bădică[1], Maria Ganzha[2], Marcin Paprzycki[3], and Amalia Pîrvănescu[1]

[1] University of Craiova, Software Engineering Department
Bvd.Decebal 107, Craiova, 200440, Romania
`badica_costin@software.ucv.ro`
[2] Gizycko Private Higher Educational Institute, Department of Informatics
Gizycko, Poland
`ganzha@pwsz.net`
[3] Oklahoma State University, Computer Science Department
Tulsa, OK, 74106, USA and
Computer Science, SWPS, 03-815 Warsaw, Poland
`marcin@cs.okstate.edu`

Abstract. For software agents to become part of e-commerce they have to be *flexible*—to engage in negotiations of forms which are not known in advance, and *mobile*—to migrate to remote locations. This note aims at combining flexibility with mobility by joining rule-based mechanism representation with modular mobile agents. Furthermore, we focus on a more complete e-commerce scenario and address questions like: what happens before negotiations start and after they are finished, where from the purchase is actually made etc. Description of agent interactions in such a complete e-commerce scenario is presented.

1   Introduction

Recent advances in auction theory have produced a general methodology of describing price negotiations [9]. To engage in negotiations, forms of which are unknown in advance, agents have to be appropriately flexible [3]. Furthermore, it is argued that they have to be mobile to be used in realistic applications [2]. However, mobile agents have to be lightweight to be able to swiftly move across the network. Unfortunately, *flexible agents cannot be lightweight* as they have to "carry" their intelligence with them [8].

In this note we describe architecture of a multi-agent e-commerce system that aims at combining flexibility and mobility. Our proposal builds on: (i) conceptual architecture of a multi-agent e-commerce system summarized in [3]; (ii) flexible framework that allows agents to participate in arbitrary negotiations described in [1], and (iii) lightweight modular agents that migrate to remote markets and engage in negotiations [3] (see also references quoted there). Furthermore, we extend the proposed approach beyond the "act" of negotiation. In [7] negotiations were extended to include matchmaking. In our work we consider: matchmaking, negotiating and purchasing. Interestingly, processes between completion of price negotiations and actual purchase, while involving a number of possibilities, are practically forgotten in literature.

2   Rule-Based and Plug-in Components for Automated Negotiation

We start by summarizing the framework for automated negotiation introduced in [1] and the architecture of mobile agents capable of dynamic negotiations elaborated in [3]. Authors of [1] analyzed the existing approaches to agent negotiations (primarily the FIPA protocols) and argued that they do not provide enough structure for the development of portable agent-based e-commerce systems. They also sketched a framework for implementing agent negotiations involving a number of infrastructure providing sub-agents: *Gatekeeper*, *Proposal Validator*, *Protocol Enforcer*, *Information Updater*, *Negotiation Terminator* and *Agreement Maker*. Central point of this framework consisted of a generic negotiation protocol and a taxonomy of JESS rules ([5]) used for enforcing specific negotiation mechanisms.

In our earlier work we have implemented (using JADE [4]) agents capable of negotiation adaptation via dynamically loadable modules ([3]). These agents consisted of three main components: (i) *communication module*—responsible for agent-agent communication, ii) *protocol module*—responsible for enforcing protocols governing negotiations, and (iii) *strategy module*—responsible for producing protocol-compliant actions necessary to achieve agent goals. Advantages of this architecture were threefold: (i) separation between functionality of each module, (ii) separation of a "private" strategy and a protocol that is "public" to the market, (iii) support for lightweight mobility.

Let us now see how it is possible to combine these two approaches. (1) Work presented in [1] assumes implicitly that *Buyer* agents are intelligent and furthermore carry with them a "generic negotiation protocol" thus making them very heavy, while our approach can help avoid this problem. (2) The *Gatekeeper* sub-agent does not play any role in actual price negotiations and thus can be placed "in the system" as a full-fledged agent. (3) Analysis presented in [1] involves only *Buyer* agents entering a given host and becoming involved in price negotiations; actions of the system preceding and following negotiations are not considered; we have thus included them in our system.

3   Agents in an E-Commerce Environment

Let us now present details as to how the two approaches can be actually combined to balance flexibility and mobility. Fundamentally, our environment acts as a distributed marketplace that hosts e-stores and allows e-clients to visit them to purchase products. Buyers negotiate with sellers and choose where to make a purchase [3].

Figure 1 presents the complete UML activity diagram of the proposed system (illustrated from both *Client* and *Shop* "perspectives"). Note that box named *Negotiation Process* includes inside **all** processes conceptualized and illustrated by UML diagrams in [1]. Let us sketch functioning of the system depicted in figure 1 (further details can be found in [3] and [1]).

Client agent receives orders from customer, and attempts to make a purchase. In the system there exists a central repository (*yellow pages*), where all e-stores advertise information about products [7]. Therefore, *Client* queries the *yellow pages agent*, and then dispatches *Buyer* agents to each *Shop* selling the requested product. Hereafter, the *Client* agent enters a composite state, attempting to make purchase(s), as results of
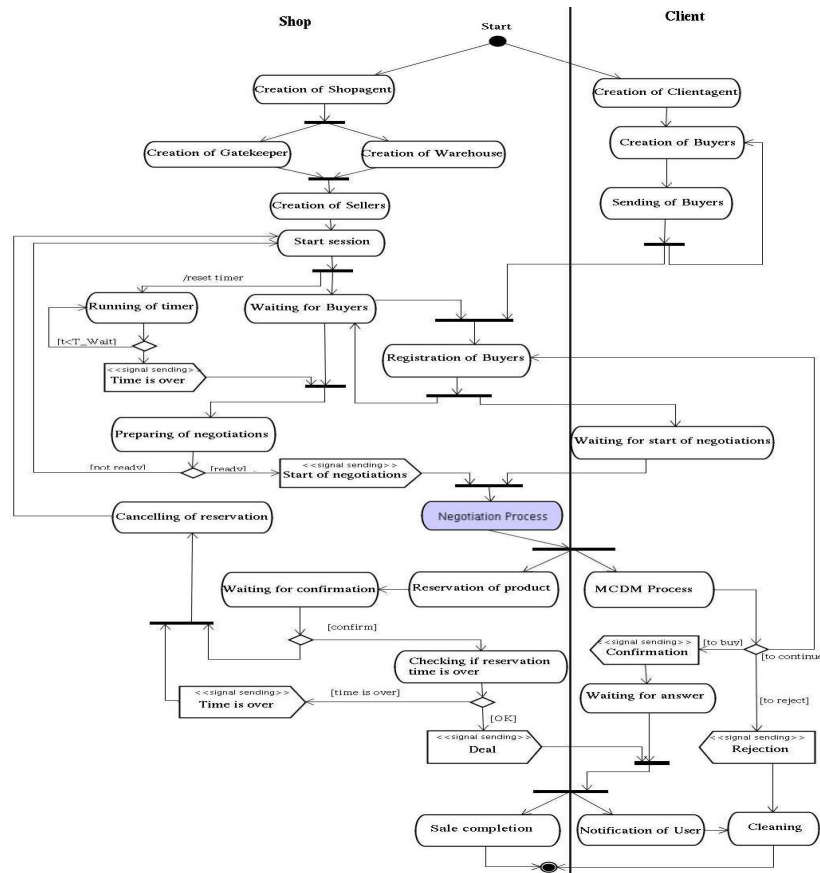
Fig. 1. UML activity diagram of the system operation

Buyer notifications. Whenever a *Buyer* agent reports a successful negotiation, the *Client* agent goes through a multi-criteria decision procedure (*MCDM Process*) that has three possible outcomes: (i) to complete the purchase, (ii) to cancel the purchase (awaiting better opportunity), or (iii) to declare the purchase impossible and notify the customer. The *Client* agent will terminate when all orders have been either honored or abandoned.

The *Shop* agent creates a *Gatekeeper*, a *Warehouse* and *Seller* agents (one *Seller*/one product)and then enters a complex state where it supervises negotiations. First, the *Shop* agent is waiting for finish of any negotiation. If it was successful, a given *Seller* informs the *Shop* which asks the *Warehouse* to reserve the product (for a specific amount of time). Then, if the winning *Buyer* confirms purchase, *Shop* asks the *Warehouse* to check reservation. If the reservation expired, then *Shop* hands rejection to the *Buyer*. Otherwise *Shop* informs *Buyer* about acceptance of transaction. This starts the final stage—named *Sale completion* which includes such actions as payment or delivery. If the *Client* rejects purchase (and informs the *Shop* about it through the *Buyer*), then *Shop* asks the *Warehouse* to cancel the reservation.

The *Gatekeeper* monitors incoming *Buyer*s and controls their admission to negotiations. When a minimum number of *Buyer*s have arrived or a timeout is triggered, the *Gatekeeper* passes identifiers of registered *Buyer*s to the *Seller* thus initiating negotiations. When negotiations are finished, the list of participating *Buyer*s is emptied and the admission/monitor process is restarted (assuming that the *Seller* is still alive). System allows loosing *Buyer*s to stay at the host and re-enter negotiations after updating protocol templates.

The *Warehouse* obtains from the *Shop* information about products and their quantities and saves them into a database. Then it waits for notifications or for timer events. The *Shop* notifies the *Warehouse* about: (1) registration of a new products for sale, (2) product reservations, (3) purchase confirmations and terminations. Time event triggers checking of existing reservations. All expired reservations are canceled, reserved products added to the pool of products for sale and the *Shop* is informed about a new amount of available goods. Note that the information about canceled reservation is provided to the *Shop* only when a purchase is requested by the *Buyer* and the *Shop* is checking if a transaction can be completed. Finally, if quantity of some product becomes 0 then *Warehouse* informs *Shop* accordingly and *Shop* terminates the corresponding *Seller* informing also the yellow pages agent that the product is not available anymore.

4   Concluding Remarks

In this paper we presented a multi-agent system that combines rule-based and mobile agent technologies for implementing flexible automated negotiations. Proposed system is being re-implemented using JADE and JESS (its earlier version, while fully functional, did not involve the general framework introduced in [1]). We are also working on agent strategies and decisions. We will report on our progress in subsequent papers.

References

1. Bartolini, C., Preist, C., Jennings, N.R.: A Software Framework for Automated Negotiation. In: *Proceedings of SELMAS'2004*, LNCS 3390, Springer Verlag (2005) 213–235.
2. Fuggetta, A., Picco, G.P., Vigna, G.: Understanding Code Mobility. In: *IEEE Transactions on Software Engineering*, vol.24, no.5, IEEE Computer Science Press (1998) 342–361.
3. Maria Ganzha, Marcin Paprzycki, Amalia Pîrvănescu, Costin Bădică, Ajith Abraham, JADE-based Multi-agent E-commerce Environment: Initial Implementation, In: *Analele Universității din Timişoara, Seria Matematică-Informatică* (2005) (to appear)
4. JADE: Java Agent Development Framework. See `http://jade.cselt.it`.
5. JESS: Java Expert System Shell. See `http://herzberg.ca.sandia.gov/jess/`.
6. Tamma, V., Wooldridge, M., Dickinson, I: An Ontology Based Approach to Automated Negotiation. In: *Proceedings AMEC'02: Agent Mediated Electronic Commerce*, LNAI 2531, Springer-Verlag (2002) 219–237.
7. Trastour, D., Bartolini, C., Preist, C.: Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In: *Proceedings of the WWW'02: International World Wide Web Conference*, Hawaii, USA, ACM Press, New York, USA (2002) 89–98.
8. Wooldridge, M.: *An Introduction to MultiAgent Systems*, John Wiley & Sons, (2002).
9. Wurman, P, Wellman, M., Walsh W.: A Parameterization of the Auction Design Space. In: *Games and Economic Behavior*, 35, Vol. 1/2 (2001), 271–303.