Implementing Commodity Flow in an Agent-Based Model E-commerce System

Maria Ganzha[1,2] Maciej Gawinecki[2], Pawel Kobzdej[2], Marcin Paprzycki[2,3] and Tomasz Serzysko[4]

[1] Elblag University of Humanities and Economy, Elblag, Poland
[2] Systems Research Institute Polish Academy of Science, Warsaw, Poland
[3] Warsaw Management Academy, Poland
[4] Warsaw University of Technology

Abstract. In our work we are developing a complete agent-based e-commerce system. Thus far we have been focusing on interactions between clients and shops (C2B relationships). In this work we discuss how the proposed system can be augmented with a logistic subsystem and discuss specifics of its implementation.

1 Introduction

In our work, we are developing a complete model agent-based e-commerce system ([3,7]). Thus far we have considered interactions between clients and shops (C2B relationships) and assumed that in each shop there exists a warehouse where products, to be sold through price negotiations, are stored. We did not address questions like: where are these products coming from, how are they restocked, etc. Only recently we have proposed an initial conceptualization of the logistics subsystem through which e-shops purchase products they sell ([8]). We have observed that while processes involved in client purchasing a product are very similar to these when stores re-stock their warehouses, there are also important differences concerning issues like: product demand prediction, offer selection criteria, interactions with wholesalers (including B2B portals as infomediaries), methods of price negotiations, trust management etc. These differences that underline the need for the special logistics subsystem, as well as assumptions about the business functions that shape it have been presented in [8] and are omitted here.

The aim of this paper is to discuss how the logistics subsystem has been implemented in the JADE agent environment. Specifically, we use two scenarios (failed and successful purchase) to illustrate flow of messages between agents supporting logistic functions. We discuss type and content of messages exchanged in agent interactions, while utilizing JADE's Sniffer to illustrate operation of the system.

Before proceeding let us make a few observations. First, note that the proposed logistics subsystem is not "stand alone" (e.g. similar to these considered in [1, 6, 10]). Instead, it has been created within the context of the system that we are developing, which has directly influenced its design. Second, focus of this work is on details of implementation of agents and their interactions. Therefore, we omit important topics like: how are sale forecasts derived and transformed into purchasing orders, how agents evaluate wholesaler offers, etc. Currently we have implemented rudimentary mechanisms that support these functions and encapsulated them into replaceable modules. Therefore, readers should assume that, for instance, when we write that "received offers are evaluated," then their favorite evaluation method has been utilized (i.e. an appropriate module was replaced with one containing that evaluation method).

2 System Description

Let us start our work from a brief description of the system. Its main functionalities have been depicted in the Use Case diagram in Figure 1. This diagram contains functions involved in system logistics and is conceptualized on a slightly higher level of abstraction than diagrams included in [3, 7]. Therefore these diagrams as well as the description of non-logistics related functions presented there should be consulted for all additional details.

Our system models a distributed marketplace in which e-shops sell products to incoming buyers. Specifically, *User-Clients* are represented by (1) a *Client Agent* that orchestrates actions involved in purchasing a product, (2) a *Client Decision Agent* that is responsible for data analysis and decision making in support of *User-Client* request, and (3) a pool of *Buyer Agent*s (*BA*) that represent *User-Client* in price negotiations.

In Figure 1 we can also see two *Client Information Center*s—central repositories where matchmaking information (which e-store sells which product, and which wholesaler can supply which product) is stored (see [9] for analysis of approaches to matchmaking). Let us note that the *Logistics CIC* is represented "outside of the system." This is to indicate that its services could be provided, for instance, by a B2B portal. Finally, the *Shop Agent* (*SA*) is the counterpart of

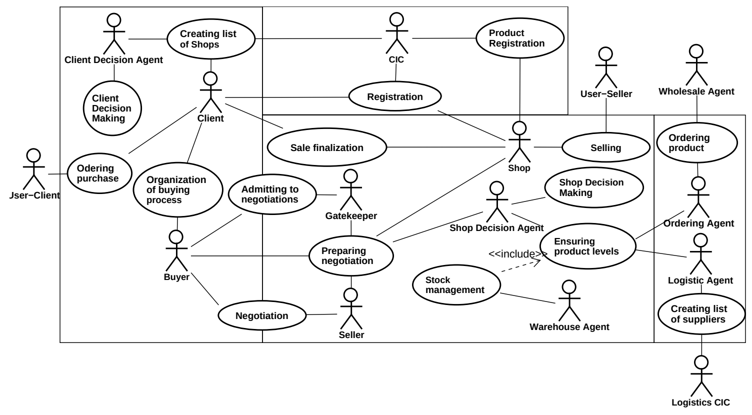


Fig. 1. Use Case diagram of the proposed system

the *Client Agent* and orchestrates all functions taking place in the e-store. Decisions made in the store are the result of data processing facilitated by the *Shop Decision Agent* (*SDA*). The *SA* is supported by the *Gatekeeper Agent* (*GA*) that is responsible for admitting incoming *BA*s into price negotiations, management of a pool of *Seller Agent*s, as well as negotiation preparation. The role of the *GA* ends when a group of *BA*s is released to negotiate prices with a *Seller Agent*.

Let us now focus our attention on logistics agents. First, the *Warehouse Agent* (*WA*), which is responsible: (1) for handling product reservations ([3]), and (2) for managing the warehouse, and this function we are interest in. To ensure appropriate supply of products the *WA* follows the forecast delivered by the *SDA* and utilizes (1) the *Logistics Agent* (*LA*) that is the "brain" behind ordering products, and (2) a pool of *Ordering Agent*s which are responsible for handling individual purchasing orders.

3 Product restocking process

We start from the UML sequence diagram of the restocking process (see Figure **??**). However, since we have implemented the logistics subsystem in JADE [2], we use output of the *Sniffer Agent* to describe and illustrate two restocking scenarios. The first of them, depicted in Figure 2, represents process that ultimately ended in a failure. In the description we use small letter agent names to

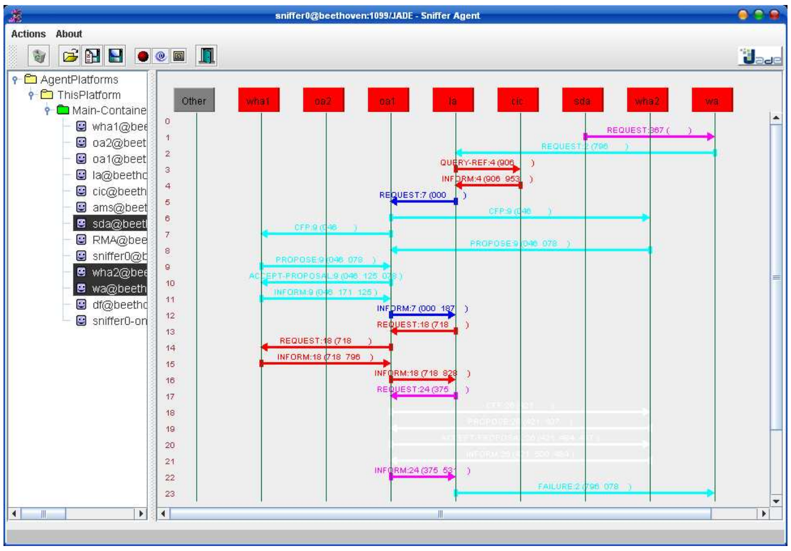


Fig. 2. Messages in the first restocking scenario as observed by the *Sniffer Agent*

match the naming convention from Figure 2; we denote messages as mn, where *n* is the message number. Obviously, Figures **??** and 2 should be considered together.

Let us assume that the *sda* has prepared forecast about required stock level of Canon EOS 10D camera, for a period between February 1st and February 28th. It states that the sales will be 4 items per week, with a deviation of 2 items. Furthermore, the *sda* requested that the purchase price stays below US$ 1,500. This prediction is communicated to the *wa* as a *FIPA Inform* ACL message $\{m1\}$ with the following value of the content slot:

```
(SDAPrediction
  :predictionDescription
   (PredictionDescription
    :globalProductID CanonEOS10D-566782
    :priceMax 1500
    :predictionDeviation 2
    :predictionPeriod (Period
      :from 20070201T0000000+02:00
      :to 20070228T0000000+02:00)
    :predictionAmount 4))))
```

Upon receiving this message, the *wa* checks the current stock level and concludes that it should purchase between 1 and (preferably) 5 cameras. Furthermore, these cameras have to be delivered strictly before 4 p.m. on January 31th; however, an offer with promised delivery time of January 29th at 10:00 am would be ideal. Based on these constraints, the *wa* prepares an ACL message $\{m2\}$ specifying the requested purchase and sends it to the *LA*. The content of this message is:

```
(action (agent-identifier
  :name la@beethoven:1099/JADE)
  (OrderRequest
    :orderDescription
      (OrderDescription
        :deliveryTimeRequired
          20070131T1600000+02:00
        :priceMax 1500
        :amountRequired 1
        :deliveryTimeAllowed
          20070129T1000000+02:00
        :amountPreferred 5
        :globalProductID
          CanonEOS10D-566782)))
```

The *la* asks the *cic* agent (instance of *Logistics CIC*; see Figure 1) about suppliers of Canon EOS 10D camera, using a *FIPA QUERY-REF* message $\{m3\}$ (see [5] for details concerning product ontology and querying the *CIC*). The *cic* replies with a *FIPA INFORM* message $\{m4\}$ containing a list of *Wholesale Agent*s. This response could have the following content:

```
(all ?supplier (Supplies
  ?supplier
  CanonEOS10D-566782))
(set
  (agent-identifier
    :name wha2@beethoven:1099/JADE
    :addresses (sequence
      http://beethoven:7778/acc))
  (agent-identifier
    :name wha1@beethoven:1099/JADE
    :addresses (sequence
      http://beethoven:7778/acc)))
```

When the list is non-empty, the *la* “prunes” the list of potential suppliers—all suppliers that have their trust value below a threshold (trust information is stored in the *LA* database) will be removed from the list (see also [4]). The list of remaining wholesalers is supplemented with trust information (to be used to rank offers) and sent $\{m5\}$ to one of free *OA*'s (here *oa1*). Note that the message contains the *IssueOrder* action with the *OrderDescription* and the suppliers list and thus contains all necessary information to execute an order.

```
(action (agent-identifier
   :name  oa1@beethoven:1099/JADE)
   (IssueOrder
     :orderDescription (OrderDescription
        :deliveryTimeRequired
          20070131T1600000+02:00
        :deliveryTimeAllowed
          20070129T1000000+02:00
        :amountRequired 1
        :amountPreferred 5
        :priceMax 1500
        :globalProductID
          CanonEOS10D-566782
        :suppliers (sequence
          (SupplierDescription
            :trust 0.98
            :name (agent-identifier
              :name wha2@beethoven:1099/JADE
             :addresses (sequence
               http://beethoven:7778/acc)))
          (SupplierDescription
            :trust 0.7
            :supplier (agent-identifier
              :name wha1@beethoven:1099/JADE
              :addresses (sequence
                http://beethoven:7778/acc
                  )))))))
```

After obtaining the request from the *la*, the *oa* engages in *FIPA ContractNet Protocol* interactions with *Wholesale Agents* (*WhA*) from the list (here *wha1* and *wha2*. It sends the following *FIPA CallForProposal* message $\{m6, m7\}$, containing *Sell* action with initial contract requirements.

```
(action (agent-identifier
     :name wha2@beethoven:1099/JADE )
   (Sell
     :globalProductID
       CanonEOS10D-566782
     :deliveryTimeRequired
       20070131T1600000+02:00
     :deliveryTimeAllowed
       20070129T1000000+02:00
     :amountRequired 1
     :amountPreferred 5))
```

The *WhA*s evaluate the CFP and if terms contained there are acceptable respond by sending *FIPA Propose* messages; e.g *wha2* agent sends message proposing sale of 4 cameras, at US$ 1450 each to be delivered by January 30th at 14:00 $\{m8\}$:

```
(Proposition
   :amountSpecific 4
   :priceSpecific 1450
   :supplier (agent-identifier
     :name wha2@beethoven:1099/JADE
     :addresses (sequence
         http://beethoven:7778/acc))
       :deliveryTimeSpecific
    20070130T1100000+14:00)
```

The *oa* evaluates received offers and if there is at least one that satisfies its criteria, accepts the best one (here *wha1*'s) by sending an ACL message $\{m10\}$:

```
(PropositionAccepted
   :proposition
         (Proposition
      :amountSpecific 4
      :priceSpecific 1450
      :supplier (agent-identifier
        :name wha1@beethoven:1099/JADE
        :addresses (sequence
           http://beethoven:7778/acc))
          :deliveryTimeSpecific
      20070130T1100000+02:00)))
```

The *wha1* confirms an order with *FIPA Inform* message $\{m11\}$ containing the *PropositionConfirmed* predicate and a unique *orderID*. This re-confirmation is required by the *Contract Net Protocol* and has the following form:

```
(PropositionConfirmed
   :proposition
         (Proposition
       :amountSpecific 4
       :priceSpecific 1450
       :supplier (agent-identifier
       :name wha1@beethoven:1099/JADE
       :addresses (sequence
          http://beethoven:7778/acc))
       :deliveryTimeSpecific
         20070130T1100000+02:00)
    :orderID 4904904)
```

Next, the *oa* forwards the confirmation to the *la* (a success message $\{m12\}$). The *la* checks the delivery time and waits. After the promised delivery time has passed, and no delivery notification was received from the *wa*; if there is still time before the hard deadline the *la* contacts $\{m13\}$ an available *OA*(here *oa1* again; *FIPA Request* message reminding about the order *oa1* $\{m14\}$). Agent *oa1* contacts the supplier by sending it the reminder containing the ID of the order:

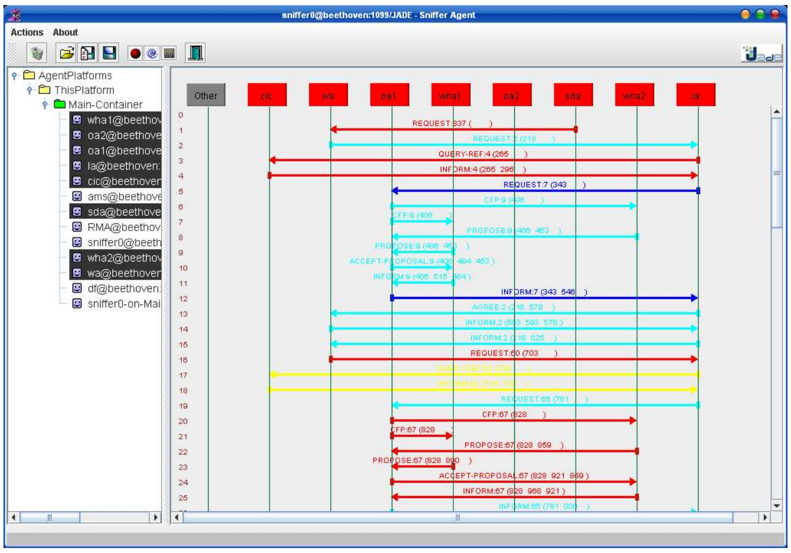
```
(action (agent-identifier
   :name oa1@beethoven:1099/JADE)
   (Deliver
     :orderID 4904904))
```

Let us now assume that the *wha1* re-confirms $\{m15\}$ an order with the new expected time for delivery of January 31th, 1 p.m. This information is again forwarded to the *la* ($\{m16\}$), which checks the delivery time and waits again. After the promised time passed, and if no delivery notification was received from the *wa* and there is still time to the deadline, the *la* contacts an available *OA* (the *oa1* again). This time however, a reminder has been already sent to this supplier. So *wha1* is removed from the list of potential suppliers, and a new order request is sent ($\{m17\}$). The *oa1* initiates the *FIPA ContractNet Protocol* with the last remaining supplier (the *wha2*), which automatically wins and its offer is accepted and confirmed ($\{m18-m21\}$). Confirmation is forwarded to the *LA* ($\{m22\}$). The *LA* checks the delivery time and waits yet again. After the time promised passes, and no delivery notification was received from the *wa*, since this time deadline is crossed, the ordering process fails, and a proper notification is sent to the *wa*:

```
(Result
  (action (agent-identifier
       :name la@beethoven:1099/JADE)
     (OrderRequest
        :orderDescription
           (OrderDescription
              :deliveryTimeRequired
```

```
          20070131T1600000+02:00
        :priceMax 1500
        :amountRequired 1
        :deliveryTimeAllowed
          20070129T1000000+02:00
        :amountPreferred 5
        :globalProductID
          CanonEOS10D−566782)))
  (Failure
   :reason SupplierDidNotDelivered))
```

While the process depicted in Figure 2 ended in a failure, in Figure 3 we depict a restocking process that ended in success. Now we focus our attention only on



```
0
1  REQUEST:337 (     )
2  REQUEST:2 (218     )
3  QUERY-REF:4 (265     )
4  INFORM:4 (265  296   )
5  REQUEST:7 (343     )
6  CFP:9 (406     )
7  CFP:9 (406     )
8  PROPOSE:9 (406  453   )
9  PROPOSE:9 (406  453   )
10 ACCEPT-PROPOSAL:9 (406  484  453 )
11 INFORM:9 (406  515  484 )
12 INFORM:7 (343  546   )
13 AGREE:2 (218  578   )
14 INFORM:2 (593  593  578 )
15 INFORM:2 (218  625   )
16 REQUEST:60 (703     )
17 [illegible]
18 [illegible]
19 REQUEST:65 (781     )
20 CFP:67 (828     )
21 CFP:67 (828     )
22 PROPOSE:67 (828  859   )
23 PROPOSE:67 (828  890   )
24 ACCEPT-PROPOSAL:67 (828  921  859 )
25 INFORM:67 (828  968  921 )
   INFORM:65 (781  000   )
```

Fig. 3. Messages in the second restocking scenario as observed by the *Sniffer Agent*

these messages that are different from the process described before. Therefore, we start form the moment when the *la* successfully finished ordering a product and communicates this event to the *wa* (as a *FIPA Agree* message $\{m13\}$):

```
((action (agent−identifier
  :name la@beethoven:1099/JADE)
  (OrderRequest
    :orderDescription
      (OrderDescription
        :deliveryTimeRequired
          20070131T1600000+02:00
        :priceMax 1500
        :amountRequired 1
        :deliveryTimeAllowed
          20070129T1000000+02:00
        :amountPreferred 5
        :globalProductID
          CanonEOS10D−566782)))
  (Order :orderID 4904904))
```

When ordered product is delivered to the warehouse, the *wa* confirms it to the *la* by sending a *FIPA Inform* message $\{m14\}$, and in turn the *la* informs the *wa* about success in delivery process $\{m15\}$. This last message may seem spurious, as the *wa* already "knows" that the Canon EOS cameras have arrived. However, we proceed here to complete the "communication loop" that started with the *wa* sending out the original purchasing order. By sending this last message we close the case of this order without distracting the communication protocol.

4 Concluding remarks

In this paper we have described in some detail the way in which we have implemented the logistics subsystem that was recently added to our model agent based e-commerce system. Our description was based on two actual runs of the JADE-implemented subsystem as well as the UML sequence diagram of the process. These two runs represented two basic scenarios of product restocking: one ending with a success, and one ending with a failure. We have focused our description on message types, forms and content. Currently the logistics subsystem is being integrated with the core of the system and we expect this process to be completed shortly. The next step in our research will be (1) to enhance the ways in which sale predictions are made, ordering requests issued (on the basis of these predictions) as well as offers are evaluated; (2) to extend the scope of negotiations to include items like: price of insurance, multiple options and prices of product delivery etc. We will report on our progress in subsequent reports.

References

1. Agentis Software, 2007. `http://www.agentissoftware.com/`.
2. Jade—java agent development framework. TILab, 2007. `http://jade.tilab.com/`.
3. C. Bădică, A. Bădiţă, M. Ganzha, and M. Paprzycki. Implementing rule-based automated price negotiation in an agent system. *Journal of Universal Computer Science*, 13(2):244–266, 2007.
4. C. Bădică, M. Ganzha, M. Gawinecki, P. Kobzdej, and M. Paprzycki. Towards trust management in an agent-based e-commerce system—initial considerations. In A. Zgrzywa, editor, *Proc. of the MISSI 2006 Conference*, pages 225–236, 2006.
5. C. Bădică, M. Ganzha, M. Gawinecki, P. Kobzdej, M. Paprzycki, M. Scafes, and G.-G. Popa. Managing information and time flow in an agent-based e-commerce system. In D. Petcu and et. al., editors, *Proceedings of the Fifth International Symposiom on Parallel and Distributed Computing*, pages 352–359, Los Alamitos, CA, 2006. IEEE CS Press.
6. C. A. Butler and J. T. Eanes. Software agent technology for large scale, real-time logistics decision support. *US Army Research Report ADA392670*, 2001. 23 pages.
7. M. Ganzha, M. Paprzycki, C. Bădică, and A. Bădiţă. *E-Service Intelligence—Methodologies, Technologies and Applications*, chapter Developing a Model Agent-based E-commerce System, pages 555–578. Springer, Berlin, 2007.

8. T. Serzysko, M. Gawinecki, P. Kobzdej, M. Ganzha, and M. Paprzycki. Introducing commodity flow to an agent-based model e-commerce system. Submitted for publication.
9. D. Trastour, C. Bartolini, and C. Preist. In *WWW '02: Proceedings. of the 11th international conference on World Wide Web*, pages 89–98, NY, USA, 2002. ACM Press.
10. W. Ying and S. Dayong. Multi-agent framework for third party logistics in e-commercestar. *Expert Systems with Applications*, 29(2):431–436, August 2005.