FUNCTIONALIZING TRUST IN A MODEL AGENT-BASED E-COMMERCE SYSTEM

Maria Ganzha, Maciej Gawinecki, Paweł Kobzdej, Marcin Paprzycki
Systems Research Institute, Polish Academy of Science
Newelska 6, 01-447 Warsaw, Poland

Costin Badica
Computer Science Department, University of Craiova
Bvd. Decebal 107, Craiova, 200440, Romania

ABSTRACT

This note discusses the way that operations involved in trust management in a model agent-based e-commerce system are functionalized. Specifically, we use UML sequence diagrams to identify when, during an attempted purchase, trust-related information is exchanged between agents in the system. We also illustrate precise form and content of messages exchanged between agents.

1 INTRODUCTION

Currently we are in the process of developing a complete model agent-based e-commerce system. Its description can be found in [1] and in collected there references to our earlier work. In this system there exist a number of places where its behavior is influenced by what can be defined as a "trust relationship" between its components. In our earlier work [2] we have conceptualized, on a very general level: (1) precisely where in our system we have to deal with "trust management," (2) which "events" that take place in the system influence trust relationships, and (3) how do they influence them. The aim of this note is to look into trust management related processes from the functional point of view. Specifically, we are interested in establishing (1) which components of the system are involved in trust management, (2) when, during purchasing process, do these components communicate, and (3) what information is being transferred. Let us start from briefly summarizing the way our system has been designed, with special attention paid to trust management related issues.

2 SYSTEM SUMMARY

Our proposed model agent-based e-commerce system depicts an e-marketplace where *shop agents* and their co-workers, represent *User-Seller*s and attempt at selling products to *buyer agents*, which together with *client agents* represent *User-Client*s. In Figure 1 we present a partial use case diagram of the system, in which we focus our attention on these of its components that are directly involved in trust management (detailed description of the system, as well as it complete use case diagram can be found in [1]).

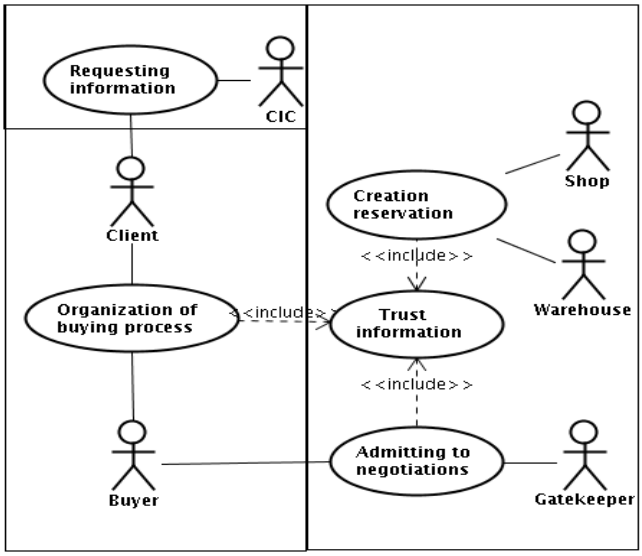

Figure 1: *Partial use case diagram of the system* (*only agents participating in trust relations and their interactions are depicted*)

In Figure 1 we can see three major agents (and three main "areas") of our system: (1) the *CIC* (*Client Information Center*) *agent* which manages white-page and yellow-page data about products sold in the system (this is the central information repository), (2) the *Client agent* (*CA*) which represents *User-Client* (this is the *User-Client* support infrastructure), and (3) *Shop, Warehouse* and *Gatekeeper agent*s, which represent *User-Seller* (this is the *User-Seller* support infrastructure). Let us now briefly describe each one these agents.

CIC agent (*CIC*) is responsible for providing information which e-store in the system sells which products. Information about products and stores is semantically represented – using OWL Lite demarcation – and persisted in a Jena [5] environment (for more details see [4]).

Client agent (*CA*) represents *User-Client* in autonomously making all necessary decisions related to the purchasing process. *Buyer agent*(*s*) (*BA*) help the *CA by* actually taking part in price negotiations.

Shop agent (*SA*) is the central manager of the e-store and autonomously makes all decisions pertinent to selling products offered by the store. The *SA* is helped by (1) the *Gatekeeper agent* (*GA*) that is responsible for admitting (or not) *BA*s to the host, managing the process of preparing negotiation which includes, among others, registration of participants and supplying them with negotiation template and protocol, and releasing *BA*s to price negotiations; (2) the *Warehouse agent* (*WA*) that is responsible for product reservations and inventory management; and, (3) multiple *Seller agents* (*SeA*) that are directly involved in price negotiations with *BA*s.

A typical system operations scenario is as follows (for a detailed description see [4]). Let us assume that system is already initialized and all information about all products sold by all e-stores has been registered with the *CIC*. *User-Client* formulates a request – what product she would like to purchase. The *CA* queries the *CIC* to find out which stores sell the requested product and attempts to "deliver" a *BA* to these stores it deems worthy of its *trust*. Depending on the *trust* that each store has in the *CA*, its *BA*s are allowed (or not) to enter. *BA*s participate in price negotiations and report results to the *CA*. Based on obtained results, the *CA* decides to (1) attempt purchase at one of the stores, (2) try to negotiate a better price, or (3) abandon purchase altogether. Let us note that *trust* that the *CA* has in shops (*SA*s) in which its representatives were winners in price negotiations plays a role in making these decisions.

In our system we utilize an *airline ticket reservation mechanism* to manage the purchasing process. Successful price negotiations result in a reservation being issued to the winner. Within a certain time, specified in the reservation, that winner can make a purchase of the product at the negotiated price. Time of the reservation depends on the *trust* that the *SA* has in a given *CA.* When the reservation expires the reserved product is returned to the pool of available products and the only way for the *BA* to make a purchase is through repeated participation in price negotiations. Note that expired reservation has a direct negative effect on *trust* that the *SA* has in a given *CA.*

In the above description, we have identified these situations which involve trust relationships. Let us now discuss in more detail how these processes actually take place in the system.

3 INTERACTIONS BEFORE NEGOTIATIONS

Let us start from the interactions between agents in the system that take place before price negotiations. Let us assume that for each store that a given *CA* interacted with in the past, it has computed its trust-value. Let us also note that in the actual implementation we have decided to split the *CA* into two agents. The *CA* itself became an orchestrator of the flow of information and a store manager – taking part in interactions with other components of the system. The *CDA* (*Client Decision Agent*), which was originally viewed as an integral part of the *CA*, is where actual decision-making takes place. It has been upgraded to a status of a full-blown agent. Metaphorically, the *CDA* is the brain of the *CA* that, for technical reasons, has been removed from its skull.

After *User-Client*'s request is formulated, the *CA* communicates with the *CIC* and obtains a list of stores that sell a given product. The *CA* analyses that list of stores and updates their trust-values [2]. Then it checks if there are any stores with trust-value below a "threshold of trust." Such stores are considered untrustworthy and removed from the list. Obviously, when only very few stores remain on the active list, the *CA* may decide to adjust the threshold value vis-à-vis a given (unpopular) product and as a result to increase number of stores that it will try to interact with. One of the reasons for such a decision may be to obtain a broader perspective on current valuation of the requested product within the marketplace. After adjusting the list of shops, the *CA* interacts with *GA*s representing them (each shop (*SA*) is actually represented by its *Gatekeeper agent*, which is the agent that any other agent has to interact with first when attempting to make a purchase).

The *GA* checks the trustworthiness of each of the *CA*s that approach it about entering the shop. Note that, similarly to the case of the *CA* and the *CDA*, we have decided to separate the manager / orchestrator functions of the *SA* from the decision-making functions that were delegated to the *Shop Decision Agent* (*SDA*). Thus, to assess the trust value of incoming *BA*s, the *GA* communicates with the *SDA.* These processes have been depicted in the sequence diagram in Figure 2. Note that all messages presented in this and subsequent figures reference FIPA ACL messages.

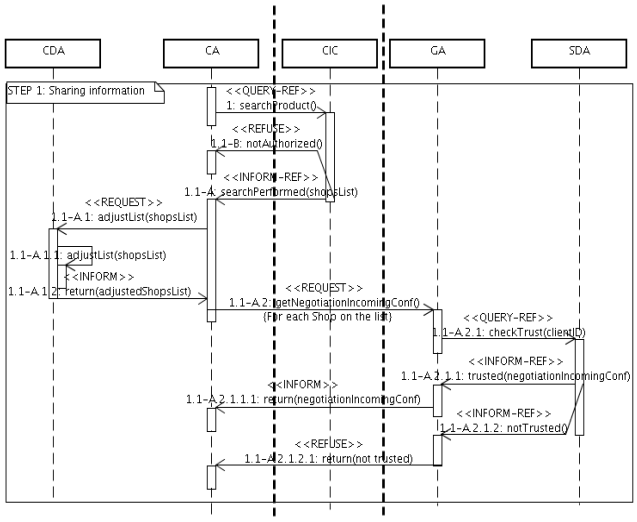

Figure 2: *The sequence diagram of ACL messages exchange between CA and CIC and CDA*

We can see here, first, exchange of messages between the *CA* and the *CIC* (*QUERY-REF* message from the *CA* is either responded to by a *REFUSE* message – when the *CA* is not authorized in the system or by the *INFORM-REF* message containing the requested list of stores). Next, the *CA* communicates with its "brain," the *CDA*, to adjust the list of shops to interact with (here we can see a simple *REQUEST – INFORM* pair of messages). For each store

on the list, the *CA* sends a *REQUEST* message to its *GA*. Such a message requests to be admitted to the store and to specify conditions of admission – the store may admit *BA*s, or create them internally (for more details see [3]).
In the way that our system was set up, where the *CA* and the *GA* are located on two separate servers named *beethoven* and *bach*, when running JADE agent environment, the message that the *CA* sends to the *GA* will have the form:

```
(request
  :sender (agent-identifier
    :name ca@beethoven:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc)
    :X-team-id "client-1")
  :receiver (agent-identifier
    :name ga@bach:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc))
  :content
    (action
      (agent-identifier
        :name ga@bach:7771/JADE
        :addresses (sequence
http://10.1.1.2:7770/acc))
      (get-negotiation-incoming-conf)
    )
)
```

The *GA* communicates with the *SDA* to evaluate the trust value of the incoming *CA* (in Figure 2, a pair of *QUERY-REF – INFORM-REF* messages). Depending on the content of the *INFORM-REF* message that it receives from the *SDA*, the *GA* sends to the *CA* either: (1) an *INFORM* message – in the case it is to be admitted – which will have the following form (note that here the *GA* informs the *CA* not only that it is ready to get involved in interactions, but also informs the *CA* that it can both accept its representative – `can-welcome-buyer true`, and create a buyer agent locally – `can-create-buyer true`):

```
(inform
  :sender (agent-identifier
    :name ga@bach:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc)
    :X-team-id "shop-1")
  :receiver (agent-identifier
    :name ca@beethoven:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc))
  :content
    (result
      (action
        (agent-identifier
          :name ga@bach:7771/JADE
          :addresses (sequence
http://10.1.1.2:7770/acc))
        (get-negotiation-incoming-conf)
      (negotiation-incoming-conf
        :can-create-buyer true
        :can-welcome-buyer true)
    )
)
```

or, otherwise, (2) when the *CA* is not going to be admitted to negotiations, a *REFUSE* message – which will have the form:

```
(refuse
  :sender (agent-identifier
    :name ga@bach:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc)
    :X-team-id "shop-1")
  :receiver (agent-identifier
    :name ca@beethoven:7771/JADE
    :addresses (sequence
http://10.1.1.2:7770/acc))
  :content
    (result
      (action
        (agent-identifier
          :name ga@bach:7771/JADE
          :addresses (sequence
http://10.1.1.2:7770/acc))
        (get-negotiation-incoming-conf)
      (reason not-trusted)
     )
)
```

Interestingly, the fact that at this stage the *GA* sends back a positive response does not mean automatically that the representative of the *CA* will be admitted to negotiations. Specifically, if the shop *creates BA*s locally, then the *BA* will be created for the *CA*. Situation changes when the shop receives incoming *BA*s. In this case it is possible that between moment of the first request about trust which is presented on the Figure 2 and the moment when *BA* arrives in the shop, trust relationship between *SA* and *CA* could change. To explain this we have to recall that the trust value depends, for instance on the fact that the *CA* has won price negotiations but did not make a purchase, or that it confirmed purchasing a product, but did not make a payment. Now, we have to consider the fact that a given *CA* can be involved in multiple interactions with a given shop (*SA*). Therefore, between the time that the *CA* receives a positive answer from the *GA* and prepares the *BA*; and the time that *BA* arrives at the shop, trust value could have changed because of actions of other *BA*s representing the *CA*.

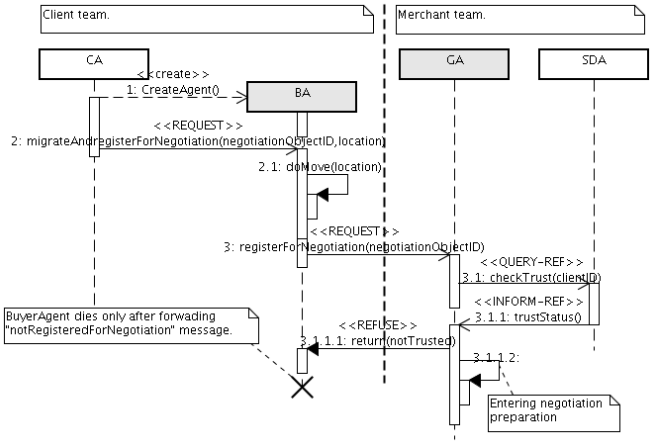

Figure 3: *The sequence diagram of ACL message exchange between arriving BA and GA.*

Therefore, when the *BA* arrives at the Shop and informs the GA: "I am here" the *GA* has to re-check the trust status of the *CA* that the *BA* represents. This process has been depicted in Figure 3. There we can see first the *CA* creating the *BA* and sending it a *REQUEST* message to migrate to a given *GA* and register for price negotiations. Upon receiving this *REQUEST* message, the *BA* moves to the specified location and sends a *REQUEST* to register to the local *GA*. The *GA* sends a *QUERY-REF* message to the *SDA* to (re)check if the *CA* is (still) trusted. In the case of a trusted *CA*, the *GA* initiates the process of preparing negotiations. If the *CA* is not to be trusted, then the *GA* sends a *REFUSE* message to the *BA*. In this case the *BA* sends a *REFUSE* message back to the *CA* (to inform it about the situation) and kills itself.

4 INTERACTIONS AFTER NEGOTIATIONS

The last moment in which trust is taken into account is when a given *BA* is a winner of price negotiations. As stated above, in our system we use an airline ticket reservation model. In this model, winning price negotiations means that a limited-time reservation will be issued for the winning *BA*. Furthermore, the time of reservation will depend on the level of trust (the more trusted the given *CA* is, the longer the reservation time is going to be (longer reservation time is a reward for being a good client). In this situation the *SA* asks the *SDA* about reservation time for the given *BA*. *SDA* establishes its duration based on the trust level [2]. After receiving an answer the *SA* asks the *WA* to reserve the product for a specific time. Finally, the *SA* informs the *BA* about the reservation time. This process (combined with the *CA* decision making that also involves trust considerations – we may not want to buy cheap products from stores that have a low trust value and rather buy more expensive products from trusted sources) is represented in Figure 4.

5 CONCLUDING REMARKS

In this note we have discussed functional aspects of trust management in a model agent-based e-commerce system. We have used UML sequence diagrams to formally represent all situations when agents exchange messages related to trust management. We have also presented form and content of trust-related messages exchanged between agents in the system.
Currently our system is being implemented and the above presented sequence diagrams are used as aid in this process.

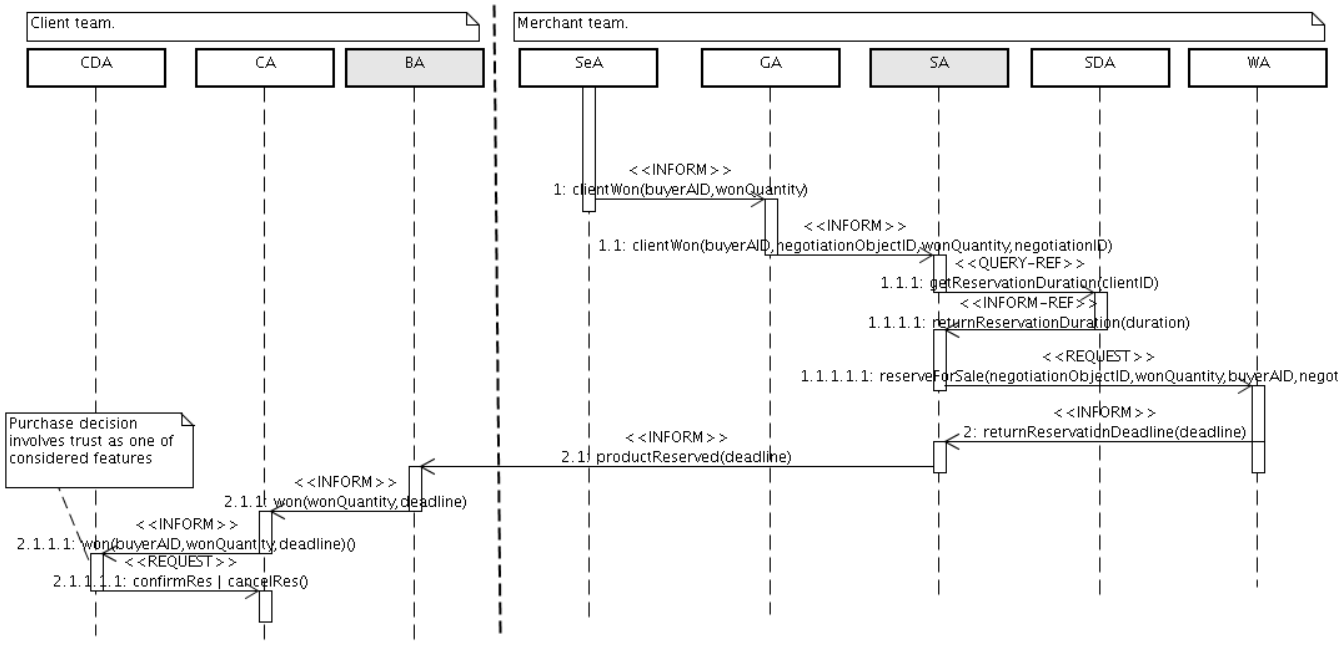

Figure 3: *The sequence diagram of ACL message exchange when negotiations are completed and there was a winner*

References

[1] C. Badica, M. Ganzha, M. Paprzycki, *Developing a Model Agent-based E-commerce System,* in: Jie Lu et. al. (eds.) E-Service Intelligence - Methodologies, Technologies and Applications, Springer, 2006, to appear

[2] C. Badica, M. Ganzha, M. Gawinecki, P. Kobzdej, M. Paprzycki, *Towards trust management in an agent-based e-commerce system – initial considerations*, Proceedings of the MISSI 2006 Conference, to appear

[3] C. Badica, M. Ganzha, M. Paprzycki, *Two Approaches to Code Mobility in an Agent-based E-commerce System*, in: C. Ardil (ed.), *Enformatika*, Volume 7, 2005, 101-107

[4] M. Gawinecki, M. Ganzha, P. Kobzdej, M. Paprzycki, C. Badica, M. Scafes, G. Popa, *Managing Information and Time Flow in an Agent-based E-commerce System,* in: Proceedings of the ISPDC'2006 Conference, to appear

[5] Jena, http://jena.sourceforge.net/