AN AGENT-MEDIATED MARKETPLACE FOR TRANSPORTATION TRANSACTIONS

Nikos Karacapilidis, Alexis Lazanas
Industrial Management Lab, MEAD, University of Patras, 26504 Rio Patras, Greece
Email: {nikos, alexlas}@mech.upatras.gr

Pavlos Moraïtis
Department of Computer Science, University of Cyprus, 75 Kallipoleos str., Nicosia, Cyprus
Email: moraitis@ucy.ac.cy

Abstract. This paper reports on the development of an innovative agent-mediated electronic marketplace, which is able to efficiently handle transportation transactions of various types. Software agents of the proposed system represent and act for any user involved in a transportation scenario, while they cooperate and get the related information in real-time mode. Our overall approach aims at the development of a flexible framework that achieves efficient communication among all parties involved, constructs the possible alternative solutions and performs the required decision-making. The system is able to handle the complexity that is inherent in such environments, which is mainly due to the frequent need of finding a modular" transportation solution, that is one that fragments the itinerary requested to a set of sub-routes that may involve different transportation means (trains, trucks, ships, airplanes, etc.). The system's agents cooperate upon well-specified business models, thus being able to manage all the necessary freighting and fleet scheduling processes in wide-area transportation networks.

Keywords. Software Agents, E-Marketplace, Transportation Management.

1. INTRODUCTION

In the last few years, multi-agent systems have successfully addressed a variety of business problems, such as business process management (Jennings et al., 2000), supply chain management (Fox et al., 2000), and manufacturing scheduling (Shen and Norrie, 1998). The above success is partially due to the nature of software agents, which autonomously perform well-defined activities, based on rules and procedures coded into their behaviour. It is widely argued that such systems become more powerful and may solve complex problems when groups of different kinds of agents begin communicating in an efficient and effective way (Sycara and Zeng, 1996).

This paper reports on the exploitation of software agent technology in transportation management. More specifically, it discusses analysis and design issues raised during the development of an innovative agent-mediated electronic marketplace (Bakos, 1998; Karacapilidis and Moraitis, 2001b), which is able to efficiently handle transportation transactions of various types. Agents of the proposed system represent and act for any user involved in a transportation scenario, such as customers who look for efficient ways to ship their products and transport companies that may - fully or partially - carry out such requests, while they cooperate and get the related information in real-time mode. Our overall framework is based on flexible models that achieve efficient communication among all parties involved, coordinate the overall process, construct possible alternative solutions and perform the required decision-making.

In addition, the proposed system is able to handle the complexity that is inherent in such environments, which is mainly due to the frequent need of finding a "modular transportation solution". To further explain this concept, consider the case where a customer wants to convey some goods from place *A* to place *B*, while there is no transport company acting directly between these two places. Supposing that two available carriers *X* and *Y* have some scheduled itineraries from *A* to *C* and from *C* to *B*, respectively, it is obvious that a possible solution to the above customer's request is to involve both *X*

and *Y* and fragment the intended overall itinerary to the related sub-routes. It should be also noted here that these carriers might be associated with diverse transportation means, such as trains, trucks, ships and airplanes.

The system is also able to manage all the necessary freighting and fleet scheduling processes in wide-area transportation networks. Its agents will cooperate upon well-specified business models, which may efficiently carry out processes such as negotiation and establishment of contracts between customers and transport companies. Finally, it integrates state-of-the-art Internet technologies in order to best serve all parties involved.

Real applications offering freighting and fleet scheduling services in international transportation networks are rather limited. More specifically, there exist some web-based approaches that only offer to the interested customers the ability to register their transportation request, which will be then handled in the traditional way (see, for instance, Spitrans Brokering Inc.: `http://www.spitrans.com/`, CargoWorld Inc.: `http://www.cargoboss.com.hk/about_cargoworld.htm`, and The Global Book Agents Network: `http://www.globalbook.net/quote.html`). In fact, the companies that run these sites act like brokers (shipping and forwarding agents) in that, upon reception of the customers' requests, they attempt to satisfy them by intervening between customers and transport companies, while keeping all parties informed about the progress of the associated transactions. Some of the existing approaches also have information seeking capabilities; however, despite their efficient management of the underlying databases (where information about transport companies, costs, itineraries etc. is kept), the role of the existing approaches is rather *passive*. On the contrary, the role of our approach is *active*, in that it offers: i) continuous search and dynamic update of all users involved for alternative feasible solutions (push technology), ii) dynamic undertaking of initiatives (due to the semi-autonomy of agents) to investigate the satisfaction of a transport request and the establishment of an agreement, iii) a multi-criteria evaluation of alternative solutions integrated with automated assistance for the negotiation of these solutions between customers and carriers, and iv) dynamic maintenance of the profiles of all users involved and exploitation of the above, during the evaluation of a certain transaction, in issues such as specification of the right policy, definition of priorities among customers, etc.

The remainder of this paper is structured as follows. Section 2 gives a process-oriented description of the proposed e-market, and highlights the analysis and design approaches followed during its development. Section 3 reports on the specification of agents' roles and interactions, while Section 4 discusses design issues concerning models of agents, services and acquaintances. Finally, concluding remarks and future work directions are given in Section 5.

2. THE E-MARKET

Through user-friendly interfaces, the proposed system will initially offer users the ability to get connected to the transport services e-market and declare the type of services they are interested in selling or buying. Such declarations can be made at a different level of abstraction, upon the users' wish. For instance, a transport company, aiming at serving a related request from a customer (i.e. selling of services), may declare its total capacity and the full network of itineraries it is active on. However, adopting another level of abstraction, it may just declare the available capacity in some already scheduled freights and itineraries (perhaps by simultaneously lowering prices of the specific itineraries, thus hoping to attract the interest of potential customers and gain extra profit). On the other hand, customers may put in their requests concerning the purchase of transportation services. Such requests may also be as abstract or detailed as they wish; obviously, one should expect to find more carriers satisfying an abstract request, while a very detailed request is expected to get matched with a small number of responses.

As noted in the previous section, the management of all required transactions is based on the exploitation of the software agents technology. Our agents highly reflect the profile of each user (carrier or customer), which are dynamically updated through the transactions taken place. For instance, an agent acting for a certain customer *X* will be able to know that its customer is interested in achieving the lowest cost, independently of the delivery time, for any transportation request he puts in. At the same time, another agent acting for a customer *Y* is aware that its customer is mostly interested in the delivery times and the insurance of the goods to be transported. On the other hand, agents representing transport companies may also build up their profiles, through which they will specify the

services offered (destinations, itineraries, capacity, particular characteristics etc.). The maintenance of agents' profiles will better serve the matching of requests and offers as well as the decision-making capabilities offered. Finally, agents may remain in the e-market as long as their actors wish (i.e., permanently or until a certain request is fulfilled).

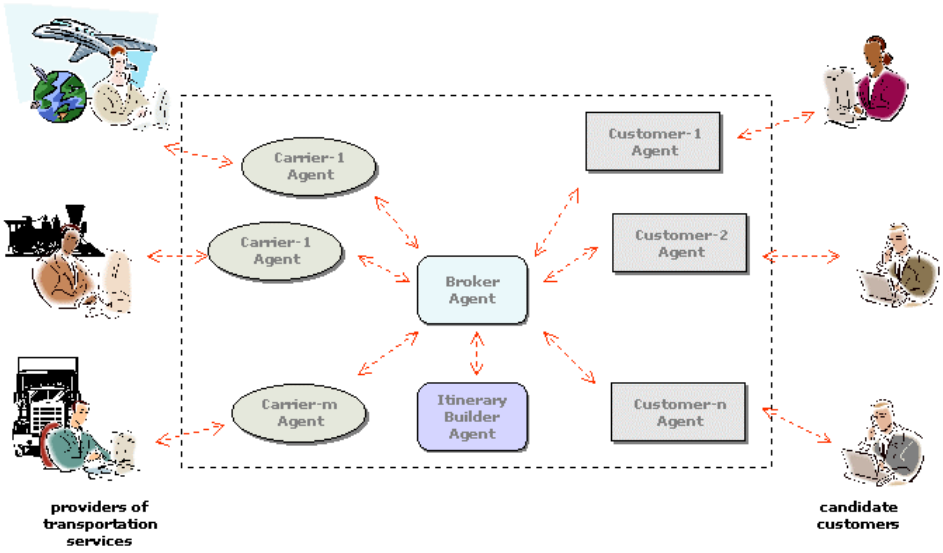

Figure 1: The proposed e-market.

The proposed system (see Fig. 1) will offer efficient mechanisms for the matching of requests and offers for transport services, as well as for the construction of feasible solutions (the exploitation of Operations Research algorithms and techniques is inherent here). As it will be discussed in more detail in the sequel, the above is performed with the aid of two other types of agents (namely, the broker and the itinerary builder agents). After a matching has been found, the system will dynamically inform the related users with the alternative possibilities found to cover their requests. An important innovation of our system is the development of a multi-criteria decision support tool, through which buyers may evaluate alternative solutions. More precisely, a candidate customer will be able to evaluate, through a set of criteria (delivery time, cost, insurance etc.), the alternative offers found, to express preferences and constraints concerning these criteria, and to weigh them accordingly. Moreover, customers may amend some of the features of the offers received, the aim being to further negotiate on them with the carriers. Through continuous interaction with the users, the system will propose the best solution according to the existing information.

The next two sections discuss in detail important analysis and design issues emerged during the development of our agent-mediated transportation market. Following the approach described in (Moraitis et al., 2002), this discussion is based on concepts coming from the Gaia methodology and the JADE framework. To briefly introduce these approaches here, we only mention that the Gaia methodology is specifically tailored to the analysis and design of multi-agent systems (MASs), while it supports both the levels of the individual agent structure and the agent society in the MAS development process. According to Gaia, MASs are viewed as being composed of a number of autonomous interactive agents that live in an organized society, in which each agent plays one or more specific roles, while their interaction protocols are well defined (for more details, see (Wooldridge et al., 2000)).

On the other hand, JADE is a software development framework, fully implemented in Java, which aims at the development of multi-agent systems and applications that comply with FIPA standards for intelligent agents (`http://www.fipa.org`). JADE provides standard agent technologies and offers to the developers a number of features in order to simplify the development process. Following JADE, agent tasks and intentions are implemented through the use of behaviours (for more details, see `http://sharon.cselt.it/projects/jade/`).

3. ROLES AND INTERACTIONS

Following the Gaia methodology, the basic objective in the analysis of our framework was the identification of roles that can be instantiated and the definition of their attributes, namely *responsibilities*, *permissions*, *activities* and *protocols*. Responsibilities determine the functionality of each role and are divided into two types: *liveness properties*, describing states of affairs that an agent must carry out, and *safety properties*, stating that an acceptable state of affairs is maintained during the execution cycle. Permissions of a role identify the resources available in order to realise its responsibilities. Activities of a role are tasks that may be fulfilled by an agent without interacting with other agents. Finally, protocols of a role define the way that this role interacts with other ones.

According to the process-oriented description of the proposed e-market, given in the previous section, we have identified six roles, which are described through a set of role schemata (Figures 2-7). In these schemata, activities appear underlined, while the operators used and their interpretations are:

- x . y, meaning "x is followed by y"
- x | y, meaning "x or y occurs"
- x^{ω}, meaning "x occurs infinitely often"
- [x], meaning "x is optional"
- x || y, meaning "x and y interleaved"

More precisely, a customer's behaviour fulfils two distinct roles, that is one acting as an interface to the customer (`CustomerHandler`, Fig. 2), and one aiding the customer in evaluating alternative offers and select one of them (`CustomerDecisionMaker`, Fig. 3). A customer may represent an organization, a company or an individual requiring a transport service, while many of them may simultaneously exist in the e-market. As described in the above two roles, a customer may: (i) Submit a message requesting a transport service, as well as personal details to the e-market (actually, to the broker – see below). Apart from contact information details (company name, address, phone, etc.), a customer may submit additional data concerning the shipment itself (i.e., collection and delivery address, desired delivery dates, mode of transport and terms of shipment, as well as number, weight and dimensions of the freight to be transported). (ii) Receive one or more transport proposals, which actually represent solutions satisfying (fully or partially) its requests; such solutions should have taken into consideration the imposed preferences and constraints. (iii) Determine whether to accept or not a proposal; acceptance or rejection of a proposal should be based on the consideration of a set of criteria that each proposal conveys (criteria taken into account may concern cost, insurance, shipment duration, delivery time, etc.). In our framework, this is performed through a multi-criteria decision making tool that evaluates the proposals at hand and suggests the most suitable one to the specific user. (iv) Amend the initial transport request (having first considered the proposals received), thus getting involved in a negotiation process with a carrier. (v) Inform the related carrier(s) about his decision.

Role Schema: `CustomerHandler (CSH)`

Description: Receives transportation requests from the customer.

Protocols and Activities: AwaitCall, GetCustDetails, GetCustReq, TransportRequest

Permissions:

reads supplied *customerDetails // customer contact information*
supplied *customerRequirements // transport requirements imposed by the customer*
generates *transportRequestQuote // completed quote or nil*

Responsibilities

Liveness:

`CustomerHandler` = AwaitCall. GetCustDetails. $(\text{GenerateQuote})^{\omega}$
`GenerateQuote` = GetCustReq. TransportRequest

Safety:

- *customerDetails = invalid* $\Rightarrow$ *transportRequestQuote =nil*
- *customerRequirements = invalid* $\Rightarrow$ *transportRequestQuote =nil*

Figure 2: The `CustomerHandler` role.

Role Schema: CustomerDecisionMaker (CDM)

Description: Evaluates a set of alternative proposals, according to a set of criteria, and suggests the one outperforming the others.

Protocols and Activities:

TransportProposal, EvaluateProposal, DecisionMade, InformCustomer, AmendedTransportRequest.

Permissions:

reads *proposalDetails* // *information about a transport proposal(sent by the broker)*

generates *suggestedSolution* // *indicates the solution that outperform the others*

generates *amendedProposalDetails* // *amends one or more criteria of a received proposal*

Responsibilities

Liveness:

CustomerDecisionMaker = (TransportProposal. DecisionProcess)$^{\omega}$

DecisionProcess = EvaluateProposal. (DecisionMade. InformCustomer | AmendedTransportRequest. CustomerDecisionMaker)

Safety:

- *true*

Figure 3: The CustomerDecisionMaker role.

A carrier's behaviour also falls into two distinct roles, one acting as an interface to the carrier (CarrierHandler, Fig. 4), and one enabling the carrier in making proposals to a specific request, according to his policy (CarrierPolicyManager, Fig. 5). As noted above, a carrier may represent a transport company that owns a number of transportation means. Many carriers may simultaneously exist in the e-market. A carrier may: (i) Submit a transport offer providing company details, number and type of transport means, spatial information about loading and delivery terminals, cargo capability etc. (ii) Receive a transport request from a customer. (iii) Evaluate a transport request, taking into consideration freight-dependant parameters (freight type, dimensions, destination), the already reserved capacity of his transportation means (LTL – Less than Truck Load vs. FTL – Full Truck Load), his customer policy, etc. (iv) Respond to the customer (via the broker, see below), by sending a proposal. (v) Inform the e-market about changes in his profile (concerning capacity, itineraries, pricing policy, etc.).

Role Schema: CarrierHandler (CRH)

Description: Acts as the interface to a transport company; it actually informs the e-market about a transport company's contact information, facilities, itineraries, timetables etc.

Protocols and Activities: AwaitCall, RegisterCarrier, GetCarrInfo, GetCarrUpdInfo, UpdateCarrDB

Permissions:

reads supplied *carrierRegistrationDetails // carrier contact information*

reads supplied *carrierSpecificationDetails // information about facilities, itineraries, timetables etc. of a carrier*

updates *carrierDB // registers a new carrier in the DB or updates existing information*

Responsibilities

Liveness:

CarrierHandler = AwaitCall. RegisterCarrier. (([GetCarrInfo]. [(GetCarrUpdInfo)$^{\omega}$]) || UpdateCarrDB)

Safety:

- *carrierRegistrationDetails = invalid ⇒ quote =nil*
- *carrierSpecificationDetails = invalid ⇒ quote =nil*
- *successful_Connection(CarrierDB) // a successful connection with the CarrierDB is established*

Figure 4: The CarrierHandler role.

Role Schema: `CarrierPolicyManager (CPM)`

Description: Determines the overall strategy of a carrier and, accordingly, submits transport proposals to the broker. Each strategy takes into account parameters such as the profile of the customer (including information about criteria that a transport proposal should satisfy), the frequency of a customer's requests, the low/high season pricing policy, the potential discounts and special offers, etc.

Protocols and Activities:

ForwardPotRoutes, EvaluateRequest, PartialTransportProposal, DecisionMade, UpdateCarrDB, UpdateCustProfile, InformCarrier

Permissions:

reads *potentialRouteRequest*
reads *carrierDB, customerDB*
updates *carrierDB, customerDB*
generates *partialProposalDetails* // *information about a partial transport proposal(sent to the broker)*

Responsibilities

Liveness:

`CarrierPolicyManager` = ((ProposalPreparation)$^{\omega}$ | ProposalImpact)
`ProposalPreparation` = ForwardPotRoutes. MakeProposal
`MakeProposal` = EvaluateRequest. UpdateCustProfile. PartialTransportProposal
`ProposalImpact` = DecisionMade. UpdateCustProfile. UpdateCarrDB. InformCarrier

Safety:

- *successful_Connection(CarrierDB) // a successful connection with the CarrierDB is established*
- *successful_Connection(CustomerDB) // a successful connection with the CustomerDB is established*

Figure 5: The `CarrierPolicyManager` role.

Two additional roles, namely `Broker` and `ItineraryBuilder` (Figures 6 and 7, respectively), correspond to two other parties involved in the proposed e-market framework. More precisely, a broker acts as an intermediate between customers and carriers. Such a party may: (i) Receive the transport request and proposal quotes from customers and carriers, respectively, and forwards them accordingly. (ii) Interact with another party, namely the Itinerary Builder (see below), in order to get potential route schemata (these schemata may involve more than one carrier), to be then forwarded to the associated carriers. Finally, the Itinerary Builder consults the carriers' profiles in order to define route paths that successfully address a customer's request.

Role Schema: `Broker (BRK)`

Description: Acts as the intermediate between customers and carriers. It also cooperates with the Itinerary Builder to obtain alternative routes (note that each route is associated with a set of carriers).

Protocols and Activities:

TransportRequest, AmendedTransportRequest, RetrieveRoutes, PotentialRoutes, ForwardPotRoutes, PartialTransportProposal, PartialProposalSynthesis, TransportProposal

Permissions:

reads *requestDetails, amendedRequestDetails*
generates *itineraryDetails* // *information about the route (to be sent to the Itinerary Builder)*
reads *potentialRoutesDetails // tuples of the form (carrier_ID, loading_terminal, delivery_terminal)*
generates *potentialRouteRequest* // *to be sent to the CarrierPolicyManager*
reads *partialProposalDetails*
generates *proposalDetails // after the synthesis of the partial proposals*
reads *suggestedSolution // sent by the CustomerDecisionMaker*

Responsibilities

Liveness:

`Broker` = (TransportRequest | AmendedTransportRequest). RetrieveRoutes. (PartialProp)$^{\omega}$. FinalProp
`PartialProp`= (PotentialRoutes. ForwardPotRoutes. PartialTransportProposal)
`FinalProp` = PartialProposalSynthesis. TransportProposal

Safety:

- *true*

Figure 6: The `Broker` role.

Role Schema: `ItineraryBuilder (ITB)`

Description: Responsible for generating alternative combinations of routes that satisfy a certain itinerary.

Protocols and Activities: RetrieveRoutes, PotentialRoutes, QueryCarrDB

Permissions:

reads *itineraryDetails* // *information about the route (sent by the Broker)*

reads *carrierDB*

generates *potentialRoutesDetails* // *tuples of the form [carrier_ID, loading_terminal, delivery_terminal]*

Responsibilities

Liveness:

`ItineraryBuilder` = RetrieveRoutes. QueryCarrDB. PotentialRoutes

Safety:

• *successful_Connection(CarrierDB)* // *a successful connection with the CarrierDB is established*

Figure 7: The `ItineraryBuilder` role.

Having identified roles, the next step in the development of our framework was to define the model of interactions taken place. The identification of the system's interactions provides a detailed description about the information interchange that occurs between different roles. As far as our framework is concerned, such information is given in Table 1.

Table 1. The Interactions Model

Protocol	Initiator	Responder	Inputs	Outputs	Purpose/ Processing
TransportRequest	CSH	BRK	customerDetails, customerRequirements	transportRequestQuote	Receives the customer's details (information about his profile) and requirements for a certain transport service and produces a quote to be forwarded to the *Broker*
TransportProposal	BRK	CDM	partialProposalDetails	proposalDetails	Performs a synthesis of the partial proposals and forwards the output of this synthesis to the *CustomerDecisionMaker*
DecisionMade	CDM	BRK, CPM	proposalDetails	suggestedSolution	Informs the *Broker* and, in the sequel, the *CarrierPolicyManager* about the decision made by the *CustomerDecisionMaker*
AmendedTransportRequest	CDM	BRK	proposalDetails	amendedProposalDetails	Receives a proposal sent by a *CarrierPolicyManager* (through the *Broker*), evaluates it (through the Decision Making Tool) and amends its features, aiming at achieving a better proposal
ForwardPotRoutes	BRK	CPM	potentialRoutesDetails	potentialRouteRequest	Forwards the constituent parts of the desired route (obtained from the *ItineraryBuilder*) to the associated *CarrierPolicyManagers*
PartialTransportProposal	CPM	BRK	carrierDB, customerDB, potentialRouteRequest	partialProposalDetails	Considers a request sent by the *Broker* together with the associated customer's profile and the database of a carrier, and constructs a (partial) proposal
RetrieveRoutes	BRK	ITB	requestDetails, amendedRequestDetails	itineraryDetails	Requests the *ItineraryBuilder* to generate potential solutions (tuples of the form [carrier_ID, loading_terminal, delivery_terminal]) to a transport request
PotentialRoutes	ITB	BRK	itineraryDetails, carrierDB	potentialRoutesDetails	Considers the details of an itinerary (which is related to a transport request) and generates (through the Route Builder Tool) a set of alternative solutions to be sent to the *Broker*

4. DESIGN ISSUES

The mapping between roles and agent types is usually one-to-one, in that each role corresponds to a different agent type. However, this need not be the case (Wooldridge et al., 2000). A designer could easily aggregate a number of roles into the same agent type, thus generating a more complex behavior for this agent. In our framework, the `CustomerHandler` and the `CustomerDecisionMaker` roles have been aggregated into a single agent type, called *`Customer`*. Similarly, the `CarrierHandler` and `CarrierPolicyManager` roles into another single agent type, called *`Carrier`*. The correspondence

between roles and agent types (known as Agent Model) in our framework is illustrated in Fig. 8 (agent types appear in boxes, whereas agent roles in ovals).

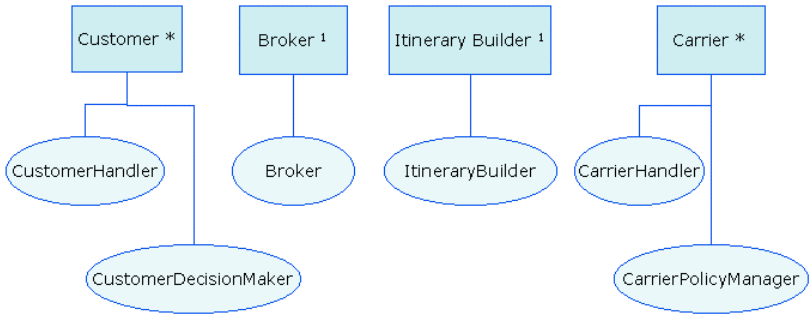

Figure 8. The Agent Model showing the correspondence between Agents and Roles

4.1 Services Model

The services model presents the tasks that each party is responsible for. According to Gaia methodology, each service can be associated with a distinct protocol. Regarding our e-market framework, this model is shown in Table 2. Note that some of these services could be further analyzed; for instance, "evaluate transport requests" also includes services related to the generation of a (partial) proposal and could be split accordingly.

Table 2. The Services Model

Name of Service	**Inputs**	**Outputs**	**Pre-Condition**	**Post-Condition**
Obtain customer transport requirements	customerDetails, customerRequirements	transportRequestQuote	true	transportRequestQuote ≠ nil
Evaluate alternative proposals	proposalDetails	suggestedSolution, amendedProposalDetails	∃ proposals	true
Inform e-market about carrier	carrierRegistrationDetails, carrierSpecificationDetails	carrierDBQuote	true	carrierDBQuote ≠ nil
Evaluate transport requests	potentialRouteRequest, carrierDB, customerDB	partialProposalDetails	customer_credit_history ≥ good	true
Perform synthesis of partial proposals	partialProposalDetails	proposalDetails	∃ partial proposals	true
Provide alternative route schemata	itineraryDetails, carrierDB	potentialRouteDetails	requested *loading_terminal* exists in carrierDB	potentialRoutes ∨ nil

4.2 Acquaintances Model

Adopting the modification of Gaia methodology proposed by Moraitis et al. (2002), our agent acquaintance model not only specifies which agent interacts with whom but also which agent is acquainted with whom (i.e. knows whom). For instance, as shown in Table 3, the *Broker* agent both interacts and is acquainted with the *Carrier* agents; in such a way, he is able to forward a transport request message to a certain subset of them (and not broadcast it to the full list of them).

Table 3. The Acquaintance Model (I: Interacts. A: Is Acquainted)

	Customer	**Broker**	**ItineraryBuilder**	**Carrier**
Customer		I		
Broker	A, I		A, I	A, I
ItineraryBuilder		A, I		
Carrier		I		

4.3 Inter-Agent Communication

Adopting JADE, communication between agents is facilitated through predefined methods. For instance, agents may identify themselves with a unique name (*setName(AID)* method of *DFAgentDescription* class), get registered to the e-market platform (*register*(*this*, *DFAgentDescription*) method of *DFService* class), and communicate using the *send(message)* and *receive(message)* methods of the *ACLMessage* class. All messages exchanged between agents are FIPA compliant and use the ACL Message Structure Specification. The proper definition of the ACL messages is very important, since it determines, among others, the responding actions of an agent's behavior.

4.4 Behaviors

Following the approach described in (Moraitis et al. 2002), the JADE framework considers the Gaia role schemata as agent behaviors (simple or complex). As in human relationships, a behavior declares the way a human being responds to its environment, e.g. acting in a particular way in transactions, answering to questions, choosing among alternatives, etc. The term "agent behavior" explains how an agent responds to an external event, e.g. when the *Broker* agent receives a transportRequest message from a *Customer* agent. JADE provides an abstract class for modeling agent behaviors and facilitates implementation of responding acts and state transitions. It also provides predefined classes that simulate various alternative behaviors. Behaviors in our e-market framework result from the liveness expressions of the role schemata defined above (Section 3). To only give an example in this paper, the behavior of the *ItineraryBuilder* agent is illustrated (in pseudo-code) below:

```
{
        agent = "ItineraryBuilder";
        register(agent) to the e-market platform;
        addNewBehavior to the Itinerary_Builder_Agent;
        receive the obtainRoutes ACLMessage;
        if MessageContent = {loadingTerminal, deliveryTerminal, CustID} then
        {
```

```
                search the CarrierDB for matching terminal points;
                perform Operational Research algorithms for route decomposition;
                construct a routeList; // tuples of the form (loadingTerminal, deliveryTerminal, CarrierID)
                if (routeList_elements >0) then
                {
                        receiver = "Broker";
                        respondRoutes = new ACLMessage.setContent(routeList);
                        send(respondRoutes);
                }
                else
                {
                        receiver = "Broker";
                        send(respondRoutes.setContent(null));
                }
        }
        else block the agent behavior;
}
```

5. DISCUSSION

This paper addresses issues emerged during the analysis and design of an innovative e-market for transportation transactions. The services to be offered will be by far more advanced to those currently supported by related applications, which mainly concern information seeking abilities. Exploiting characteristics of software agents, such as autonomy, proactiveness and "intelligence", together with their ability to cooperate, our overall framework should be suitable for the delegation of any traditional transportation transaction. Moreover, the transactions performed by the proposed system are expected to be significantly faster and cheaper.

Full exploitation of software agents technology is expected to advance the solution of the overall problem, fully satisfying requirements such as: a) continuous look-up for the construction of feasible solutions, b) dynamic updating of the users involved for the feasible (optimum or sub-optimum) solutions, c) dynamic exploration of the market in order to satisfy a service request or offer, and dynamically triggered initiatives in order to establish a deal, d) ability of negotiation and multi-criteria decision support, and e) dynamic maintenance of the users profiles for the evaluation of the priority of their requests and their overall "attitude" in the e-market (Froehlich et al., 1999; Karacapilidis and Moraitis, 2001a).

Our primary future work direction is certainly the full implementation of the electronic marketplace presented in this paper. Another direction concerns research on alternative ways of coordination of customers and carriers in our framework. The latter involves consideration of various types of middle-agents, such as *mediators*, *match-makers* etc. (Klusch and Sycara, 2001; Guttman et al., 1998). Finally, further work needs to be done on the definition of semantics and syntax of data and knowledge conveyed among our agents.

REFERENCES

Bakos, Y., 1998. Towards Friction-Free Markets: The Emerging Role of Electronic Marketplaces on the Internet. *Communications of the ACM* 41(8), 35-42.

Fox, M.S., Barbuceanu, M., and Teigen, R., 2000. Agent-Oriented Supply-Chain Management. International Journal of Flexible Manufacturing Systems 12(2/3), 165-188.

Froehlich, G., Hoover, H.J., Liew, W., and Sorenson, P., 1999. Application Framework Issues when Evolving Business Applications for Electronic Commerce. *Information Systems* 24(6), 457-473.

Jennings, N., Faratin, P. Norman, T.J., O'Brien, P. and Odgers, B., 2000. Autonomous Agents for Business Process Management. *International Journal of Applied Artificial Intelligence* 14(2), 145-189.

Guttman, R., Moukas, A., and Maes, P., 1998. Agents as Mediators in Electronic Commerce. *Electronic Markets* 8(1), 22-27.

Karacapilidis, N., and Moraitis, P., 2001a. Building an Agent-Mediated Electronic Commerce System with Decision Analysis Features. *Decision Support Systems* 32(1), 53-69.

Karacapilidis, N., and Moraitis, P., 2001b. Intelligent Agents for an Artificial Market System. In Proc. of the *5th International Conference on Autonomous Agents (Agents-2001)*, Montreal, ACM Press, 592-599.

Klusch, M. and Sycara, K., 2001. Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In A. Omicini et al. (eds.), *Coordination of Internet Agents*, Springer, 197-224.

Moraitis, P., Petraki, E., and Spanoudakis, N., 2002. Engineering JADE Agents with the Gaia Methodology. In Proc. of the International Workshop on Agent Technology and Software Engineering (AgeS'02).

Shen W. and Norrie, D.H., 1998. An Agent-Based Approach for Dynamic Manufacturing Scheduling. In Proceedings of Autonomous Agents'98 Workshop on Agent-Based Manufacturing, Minneapolis/St. Paul, MN, pp. 117-128.

Sycara, K., and Zeng, D., 1996. Coordination of Multiple Intelligent Software Agents. *International Journal of Cooperative Information Systems* 5(2-3).

Wooldridge, M., Jennings, N.R., and Kinny, D., 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems* 3(3), 285-312.