ELSEVIER

On the development of a web-based system for transportation services

Nikos Karacapilidis [a,*], Alexis Lazanas [a],
George Megalokonomos [a], Pavlos Moraïtis [b]

[a] *Industrial Management and Information Systems Lab, MEAD, University of Patras,
26504 Rio Patras, Greece*
[b] *Department of Computer Science, University of Cyprus, 75 Kallipoleos str., Nicosia, Cyprus*

Abstract

This paper reports on issues raised during the development of an intelligent web-based system for transportation services. Our approach exploits software agents technology to efficiently handle simple or modular transportation requests. The agents involved in the proposed system represent and act on behalf of two basic types of users involved in a transportation scenario, that is, for customers looking for the appropriate way to ship their products, and for transportation companies (carriers) that may—fully or partially—carry out such requests. In addition, they cooperate with two special-purpose agents, namely the Broker and the Itinerary Builder, which act as intermediates between customers and carriers, and facilitate the overall management of transportation transactions by coordinating request and offers, constructing possible alternative solutions and facilitating the required decision-making. We provide a process-oriented description of the proposed system, report on the specification of agents' roles and interactions, and discuss various implementation issues. Particular attention is paid to the construction of modular transportation solutions, that is solutions that fragment the

---

* Corresponding author.
E-mail addresses: nikos@mech.upatras.gr (N. Karacapilidis), alexlas@mech.upatras.gr (A. Lazanas), mego@mech.upatras.gr (G. Megalokonomos), moraitis@ucy.ac.cy (P. Moraïtis).

itinerary requested to a set of subroutes that may involve different transportation means. The corresponding algorithm and tool developed for this process are comprehensively presented.

1. Introduction

In the last few years, multi-agent systems have successfully addressed a variety of business problems, such as business process management [4], supply chain management [2], and manufacturing scheduling [11]. 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 behavior. It is widely argued that such systems become more powerful and may solve complex problems when groups of different kinds of agents start communicating in an efficient and effective way [12].

This paper reports on the exploitation of software agent technology in transportation management. More specifically, it discusses analysis, design and implementation issues raised during the development of an innovative agent-mediated electronic marketplace [1,7], 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 subroutes. It should be also noted here that these carriers might be associated with diverse transportation means, such as trains, trucks, ships and airplanes. Finally, the overall framework proposed will be 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.

The remainder of this paper is structured as follows. Section 2 comments on related work and sketches the novelty of our approach. Section 3 gives a process-oriented description of the proposed e-market, and highlights the analysis and design approaches followed during its development. Section 4 reports on the specification of agents' roles and interactions, while Section 5 discusses design issues concerning models of agents, services, acquaintances and behaviors. Section 6 focuses on the algorithm developed for the construction of modular transportation solutions. Section 7 is then devoted to a series of implementation issues. Finally, concluding remarks and future work directions are listed in Section 8.

2. Related work

Real applications offering freighting and fleet scheduling services in international transportation networks are limited and can be classified under two broad categories. More specifically, the first category comprises 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 and manage efficiently the underlying databases, where information about transportation companies, costs, itineraries, etc. is kept.

The second category includes a limited number of web-based or windows-based transportation management systems (TMSs), which aim at better serving the needs of various parties involved in the related transactions. These systems are often parts of enterprise-wide logistics solutions that consist of modules serving a diversity of associated needs, such as supply chain management, invoicing, and accounting. Representative applications of this category are, among others, Cadence TMS (see http://www.cadretech.com), E.R.T. TMS (see http://www.ertsoftware.com), and LoadMaster (see http://www.mcleodsoftware.com). The first system is web-based, while the other two require installation of the full application at the customer's side

(together with the related databases of carriers). They all offer user-friendly interfaces and tools for the matching of loads with transportation means, while satisfying the customer/business requirements and attempting to maximize revenue and minimize ''empty'' and ''out-of-route'' miles. They also provide load-tracking modules to allow customers obtain their specific load information without speaking with a customer service representative. All loads can be monitored from dispatch to delivery. Systems of this category also integrate GIS (geographical information systems) features in their shipment tracking tools; a customer can view detailed shipment information selecting a specific shipment and clicking at a point on the associated map (the shipment details map may zoom in and out to a certain level). Despite their potential capabilities, these approaches require the involvement of an experienced user in transportation transactions, due to the complexity issues involved in cargo brokering. The basic disadvantage of the systems described above concerns the extensive human interference in the cargo freighting process, in that an employee should be responsible for the management of critical issues, such as carrier selection, delivery date matching, route planning, and transportation pricing. In addition, their route planning procedure does not involve a detailed consideration of modular solutions, as far as alternative carriers or alternative paths are concerned, in that it is based on a customer's request about a specific direct route. Thus, these systems do not guarantee any optimization of critical factors such as the transportation's duration, cost and route selected.

Generally speaking, the field of transportation management requires quick and cost-effective solutions to the customers' demands for distribution and shipping operations. Moreover, due to the multitude of the transportation companies involved, operations offered by a traditional brokerage agency should be continuously provided. Another important factor concerns the ability of the real applications to provide optimal or suboptimal solutions, as far as the transportation's cost and duration are concerned. With respect to these issues, we argue that the role of the real applications described above is *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 (with regard to issues such as cost and duration) and the establishment of an agreement, (iii) dynamic consideration of alternative paths and carriers during the route planning process, aiming at the optimization of the direct or modular route's cost and duration, (iv) a multi-criteria evaluation of alternative solutions integrated with automated assistance for the negotiation of these solutions between customers and carriers, and (v) dynamic maintenance of the profiles of all users involved and exploitation of them, during the evaluation of a certain transaction, in issues such as specification of the right policy, definition of priorities among customers, etc.

3. The proposed e-market for transportation services

Through user-friendly interfaces, the proposed system will initially offer users the ability to get connected to the transportation 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 is 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).

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—see Section 6). 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 about the alternative possibilities found to (fully or partially) satisfy their requests. An already implemented multi-criteria decision support tool, through which customers may evaluate alternative solutions and interact with the carriers, has been integrated in the system. By launching this application, a
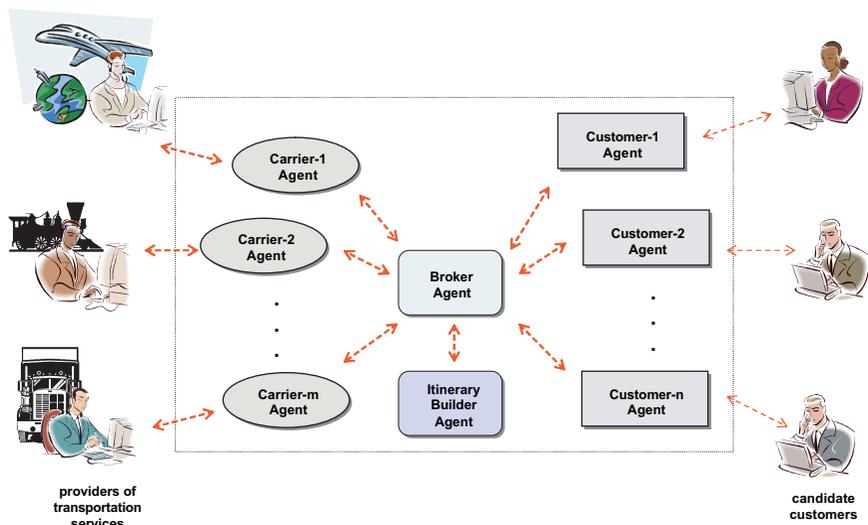
Fig. 1. The proposed e-market for transportation services.

candidate customer will be able to evaluate, through a set of criteria (such as 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 tool will propose the best solution according to the existing information. It is an agent-based tool that has been developed in a generic way, to support diverse multi-criteria decision-making processes involved in e-commerce and e-business activities. The tool has been thoroughly tested and validated, while it has been successfully employed in a real application. A detailed presentation of the tool's features and functionalities can be found in [6].

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 [9], this discussion is based on concepts coming from the *Gaia* methodology. To briefly introduce this approach 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 clearly defined (for more details, see [13]).

4. Roles and interactions

The basic objective in the analysis phase 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 realize 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 the others. According to the process-oriented description of the proposed e-market, given in the previous section, we have identified six roles, which are formally described through a set of role schemata in Appendix A (see Figs. 7–12).

More specifically, a customer's behavior fulfils two distinct roles, that is one acting as an interface to the customer (`CustomerHandler`, Appendix A—Fig. 7), and one aiding the customer in evaluating alternative offers and select one of them (`CustomerDecisionMaker`, Appendix A—Fig. 8). 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:

- 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).
- 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.
- 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.
- Amend the initial transport request (having first considered the proposals received), thus getting involved in a negotiation process with a carrier.
- Inform the related carrier(s) about his decision.

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	amended ProposalDetails	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_termmal, delivery _terminal]) to a transport request
PotentialRoutes	ITB	BRK	itineraryDetails, carrierDB	potentialRoutesDetails	Consider the details of an itinerary (which is related to a transport request) and generates (through the Routes Builder Tool) a set of alternative solutions to sent to the `Broker`

A carrier's behavior also falls into two distinct roles, one acting as an interface to the carrier (`CarrierHandler`, Appendix A—Fig. 9), and one enabling the carrier in making proposals to a specific request, according to his policy (`CarrierPolicyManager`, Appendix A—Fig. 10). 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:

- Submit a transport offer providing company details, number and type of transport means, spatial information about loading and delivery terminals, cargo capability, etc.
- Receive a transport request from a customer.
- 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.
- Respond to the customer (via the broker, see below), by sending a proposal.
- Inform the e-market about changes in his profile (concerning capacity, itineraries, pricing policy, etc.).

Two additional roles, namely `Broker` and `ItineraryBuilder` (Appendix A—Figs. 11 and 12, 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:

- Receive the transport request and proposal quotes from customers and carriers, respectively, and forwards them accordingly.
- 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 routes (paths) that successfully address a customer's request. 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.

5. 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 does not need
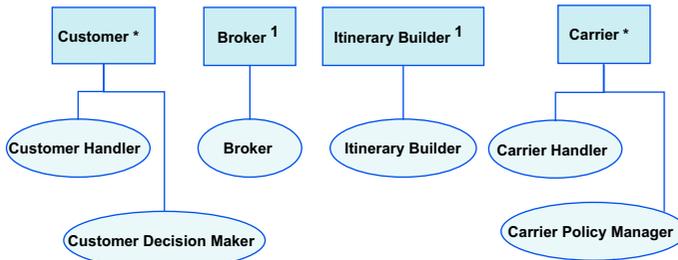
Fig. 2. The agent model showing the correspondence between agents and their roles.

to be the case [13]. 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 `CustomerDecision-Maker` 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. 2 (agent types appear in boxes, whereas agent roles in ovals).

5.1. Services model

The services model illustrates 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.

5.2. Acquaintances model

Adopting the modification of Gaia methodology proposed in [9], 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 do not broadcast it to the full list of them).

5.3. Inter-agent communication

In order to integrate communication between agents we have to implement a number of predefined methods. For instance, agents have to identify themselves

Table 2
The services model

Name of service	Inputs	Outputs	Pre-condition	Post-condition
Obtain customer transport requirements	customerDetails, customer Requirements	transportRequestQuote	true	transportRequest Quote ≠ nil
Evaluate alternative proposals	proposalDetails	suggestedSolution, amendedProposalDetails	∃ proposals	true
Inform e-market about carrier	carrierRegistrationDetails, carrier SpecificationDetails	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

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		
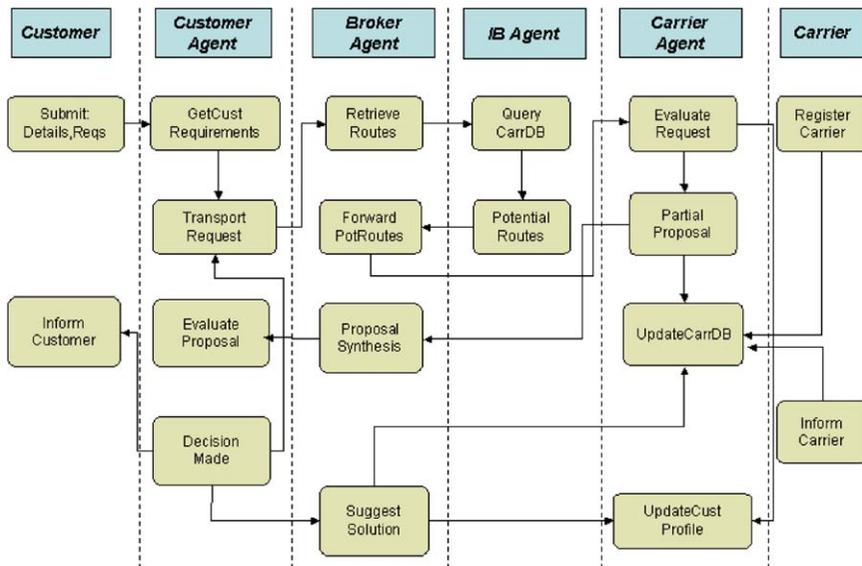
Fig. 3. The agent–human activity diagram.

with a unique name, get registered to the e-market platform, communicate by sending and receiving ACL (agent communication language) messages, establish connection to SQL servers, and submit queries to the databases. All messages exchanged between agents are FIPA (Foundation for Intelligent Physical Agents) 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. Based on the roles and services defined above, Figs. 3 and 4 illustrate the corresponding agent–human activity and the agent–human interaction diagrams, respectively.

5.4. Behaviors

Defining an agent's behavior is a critical point in the design phase of the system. As in human relationships, a behavior declares the way one responds to its
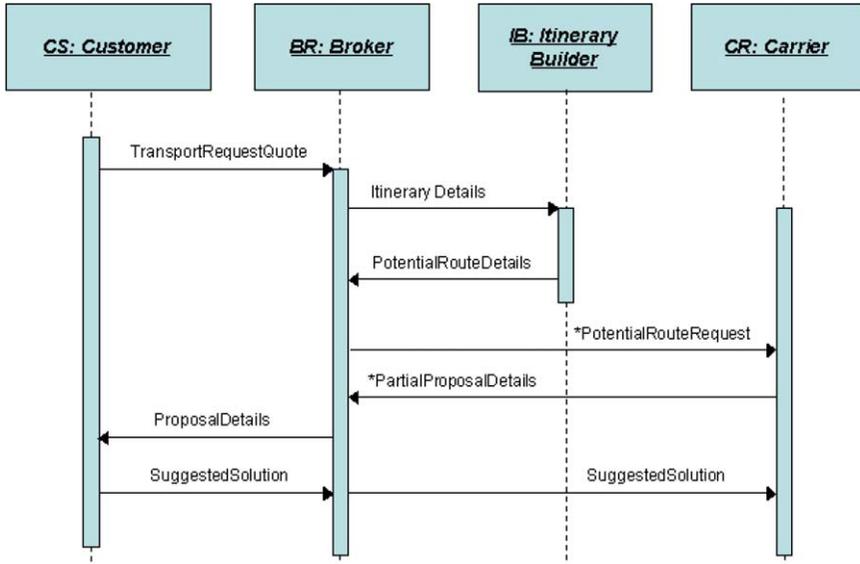
Fig. 4. The agent–human interaction diagram.

environment (that is, the particular way of acting during processing of 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. In order to simulate all these different responds to incoming events, we should define a number of abstract classes that should be responsible for modeling agent behaviors and facilitate the implementation of responding acts and state transitions. Behaviors in our e-market framework result from the liveness expressions of the role schemata defined in Section 4.

6. Construction of modular transportation solutions

As discussed above, our system is able to provide either direct or modular transportation solutions. Formally speaking, the problem can be expressed as follows: Let a graph $G = \{V, A\}$, where V is the set of vertices (loading/delivery terminals) and A the set of arcs joining them. If each arc a_{ij} is associated with a transportation cost c_{ij} and a time distance d_{ij}, find the minimum total cost C_{opt}^{SE} for the transportation of products from a vertex S to a vertex E within a given time frame (up to $T_{\max}$). Exploiting well-established operations research approaches for the solution of the shortest route problem [10], the following algorithm has been developed for the needs of the proposed system:

Step 1: Retrieve all direct *Routes* (*Loading_term*, *Delivery_term*, c_{ij}, d_{ij}) from vertex S to vertex E;

Sort *Routes* by c_{ij} then by d_{ij};

Set $Route_{\mathrm{opt}} = \{\min(c_{SE}), d_{SE} \leqslant T_{\max}\}$;

If $\exists$ route with $d_{ij} \leqslant T_{\max}$ then set $T_{\max} = \min(d_{SE})$;

Go to Step 2.

Having retrieved the necessary information (loading and delivery terminals, cost and duration) about the direct routes (if such routes exist), these routes are sorted according to their cost (primary criterion) and duration (secondary criterion). Based on this sorting, we identify the best solution (optimum route—$Route_{\mathrm{opt}}$), and we set its associated duration as the time upper bound ($T_{\max}$) of the system's transportation solution (the initial cost and time upper bounds are set by the user).

Step 2: Retrieve all *RouteArc$_{ij}$* (i, j, c_{ij}, d_{ij}) in *Route$_{SE}$*;

Sort *RouteArc$_{ij}$* by c_{ij} then by d_{ij};

Set $c_{\mathrm{opt}} = \min(c'_{ij}, c''_{ij}, \ldots)$;

Go to Step 3.

In this step, we start looking at modular solutions, the aim being to eventually find the best modular solution which either is better than the direct one found in Step 1, or satisfies the customer's requests (in case that a direct solution has not been found in Step 1). Our algorithm retrieves information (cost and duration) about the solutions (called "route arcs") offered by carriers for each segment of a route (in a modular solution, the route consists of multiple segments). For each segment, we sort the solutions retrieved (by cost and duration, as in Step 1), and we set the cost of the best solution as optimal (c_{opt}).

Step 3: Use *shortest route algorithm*[1] to find the minimum c_{ij};

if ($c_{SE} \leqslant C_{\max} \land d_{SE} \leqslant T_{\max}$) then {

   $Route_{\mathrm{opt}} = Route_{SE}$;

   exit;}

else if ($c_{SE} > C_{\max} \lor d_{SE} > T_{\max}$) then $c_{\mathrm{opt}} = C_{\max}$;

Go to Step 4.

In this step, we try to construct an optimum solution (if it exists) for the complete route. Using the shortest route algorithm, and by taking into account the best solutions found in the previous step for each segment of the route, we calculate the modular solution's total cost and duration. We first check whether both its duration and cost are less than the time and cost upper bounds ($T_{\max}$ and $C_{\max}$, respectively).

---

[1] See [10, pp. 110–111].

In such a case, an optimal solution has been found and the algorithm terminates. If the solution's cost or duration is greater than the cost or duration upper bound, respectively, we set the optimal cost to be equal to $C_{\max}$ and we move forward to the next step.

Step 4: $\forall \, RouteArc_{ij} \in Route_{\mathrm{opt}}^{\mathrm{SE}}$

find the cost increment $dc'_{ij} = c''_{ij} - c'_{ij}$ for every d_{ij} reduction;

Sort $RouteArc_{ij}$ by dc'_{ij};

Go to Step 5.

In this step, we replace each route arc found in the modular route constructed above with alternatives having a shorter duration. For each such replacement, we calculate the associated cost increment and we sort the route arcs accordingly.

Step 5: For each segment

$\{\forall \, c_{ij} \in RouteArc_{ij};$

set $c_{ij} = \min(dc'_{ij})\}$;

if $(d'_{SE} \leqslant T_{\max} \wedge c'_{SE} \leqslant C_{\max})$ then {

$Route_{\mathrm{opt}}^{SE}$ found;

exit;}

else if $(c'_{SE} \leqslant C_{\max} \wedge d'_{SE} > T_{\max})$ then go to Step 6.

In this step, considering the sorted route arcs list for each segment (taking into account the solution with the less cost increment for each segment), we calculate the total cost of the corresponding modular solution. If both its duration and cost are less than the time and cost upper bounds, then an optimal solution ($Route_{\mathrm{opt}}^{SE}$) has been found and the algorithm terminates. Otherwise, in case that the solution's total duration exceeds the duration upper bound, we move to the next step.

Step 6: For each segment

{repeat

{consider the next $\min(dc_{ij})$ increment;

if $(d'_{SE} \leqslant T_{\max} \wedge c'_{SE} \leqslant C_{\max})$ then {

$Route_{\mathrm{opt}}^{SE}$ found;

exit;}}

until $(d'_{SE} \leqslant T_{\max} \wedge c'_{ij} \leqslant C_{\max})\}$;

if $(c'_{SE} > C_{\max} \wedge d'_{SE} > T_{\max})$ then no solution.

We repeat the cost substitution procedure performed in the previous step by considering alternative route arcs for each segment (successive solutions from the sorted route arcs lists), until both the duration and cost of the corresponding modular solutions are less than the time and cost upper bounds, respectively. In case that the above cannot be satisfied, there is no solution to the problem.

The detailed behavior of the `ItineraryBuilder` agent, which is based on the above algorithm, is fully illustrated (in pseudo-code) in Appendix B.

7. Implementation issues

Access to the services of the proposed e-marketplace is provided through dedicated Web servers, where multithreading applications act as personal agents for each user involved. In our implementation, we have exploited a series of technologies supported by the Microsoft's. NET platform, such as ADO.NET, XML Web Services, and Visual J#.NET (http://www.micro-soft.com/net/). The system's architecture is shown in Fig. 5. According to the communication protocols specified, agents exchange each time the appropriate ACL messages. The population of customers and carriers databases is performed through user-friendly interfaces. The Broker and the ItineraryBuilder agents are responsible for establishing communication with the corresponding SQL Servers, where transportation details and user profiles are maintained. It should be noted that, in order for a user to access the services provided, a secure connection with the Web server has first to be established. Generally speaking, the agent-based implementation of the system gives it an active role and yields the benefits mentioned in Section 2.

Exploiting the algorithm described in the previous section, the ItineraryBuilder agent is able to construct direct or modular transportation
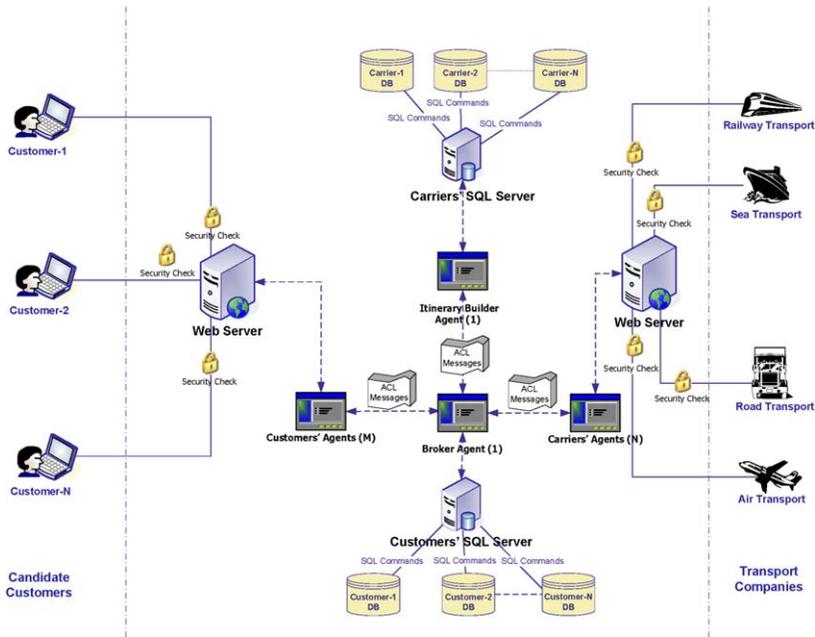
Fig. 5. The proposed system architecture.
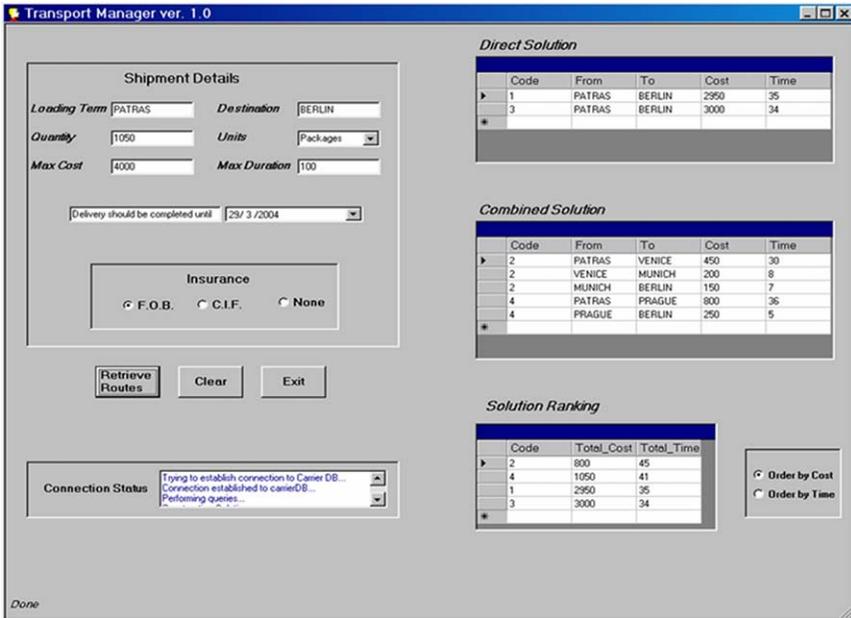
Fig. 6. The Transport Manager tool.

solutions whenever there is a related request from the `Broker` agent (the inter-communication of these two types of agents has been explicitly defined in their roles—see Section 4). A special-purpose tool, namely the *Transport Manager* (see Fig. 6) has been implemented for the above process. This tool is responsible for the establishment of connections with the appropriate SQL servers (of the carriers), the transmission of the corresponding SQL queries, as well as the retrieval and processing of the results found. In our system, a remote SQL server connection with the carriers' databases is achieved through the available and well-tried OLEDbControls of .NET platform. After a connection has been established, the agent's methods submit the appropriate SQL queries to each carrier's database, while the results obtained constitute a dataset (table) to be then processed by the agent's built-in algorithm described in the previous section.

The solutions constructed upon each request are then forwarded (through the `suggestedSolution` messages) to the customer for further evaluation. The instance shown in Fig. 6 corresponds to a customer's request for the transportation of some goods from Patras to Berlin. The customer has asserted constraints regarding the maximum cost and maximum duration of this transportation, as well as the desired maximum date for the completion

of the corresponding delivery ("delivery should be completed until 26/2/2004") and insurance type (F.O.B. stands for "Free-On-Board", while C.I.F. for "Cost-Insurance-Frame"). The potential routes satisfying the customer's request are then retrieved (in case of direct solutions) or constructed (in case of combined, modular solutions) by the tool (it should be noted that this tool provides the full list of potential solutions). In our example, two direct and two combined solutions are provided (top and the middle panes at the right part of the window). As shown, the combined solutions are Patras → Venice → Munich → Berlin and Patras → Prague → Berlin (pieces of a combined solution are associated with the same ID). The tool also performs a (primary) solution ranking, ordering them by their total cost or by their total duration.

As mentioned above, the solutions found are then forwarded (by the `Broker` agent) to the corresponding customer for further evaluation. Such an evaluation can be performed through a multi-criteria comparative consideration of the solutions in hand and is facilitated with the aid of a decision making tool. Towards this aim, we have integrated in the proposed system an already implemented tool for handling the decision making processes required at an electronic marketplace [6]. To only give an overview here (see also the related discussion in Section 3), the decision making tool allows customers to specify the relative importance of various criteria associated with the selection of a transportation solution, as well as additional constraints to be taken into account (other than those handled by the Transport Manager tool). It also enables customers to dynamically modify their personal preferences and constraints and contemplate upon alternative solutions. Finally, the tool provides to both the customer and carriers the ability to argue in favor or against the preferences and constraints expressed so far.

8. Discussion and conclusion

This paper reports on issues emerged during the development of an electronic marketplace for advanced transportation transactions. Table 4 summarizes the characteristics of each software agent type deployed in our approach and provides a short description of the process where each characteristic is involved. Exploiting characteristics of software agents, such as autonomy, proactiveness and intelligence, together with their ability to cooperate, the proposed system is suitable for the delegation of diverse traditional transportation transactions, while these are expected to be significantly faster and cheaper.

The above remarks came out from the testing and evaluation of the system's basic modules, as well as the overall approach followed so far during the

Table 4
Characteristics of agents involved in our approach

	Autonomy	Proactiveness	Intelligence
Carrier	Submits transport proposals to the broker	After each request, updates customers' profiles for future exploitation	Determines the overall offering strategy of the carrier, based upon the customers' profile
Customer	Receives proposal details through the Broker and amends (some of) them in order to further negotiate	Remains active and performs continuous look up for transport solutions	Evaluates alternative proposals and suggests the one outperforming the others
Broker	Remains active in order to coordinate communication between different agent types	Determines which carrier should be involved in each transaction	Performs synthesis of partial proposals sent by the Itinerary Builder
Itinerary Builder			Able to provide optimal or suboptimal modular transportation solutions

development of the system. More specifically, the Transport Manager tool, which is a built-in component of the `ItineraryBuilder` agent, has been thoroughly tested without any technical and implementation problems, as far as the establishment of connection with the remote carriers' databases and the efficiency of the underlying algorithm are concerned. We have conducted experiments with diverse queries (corresponding to a customer's transportation request) and up to eight remote databases populated on average with more than 70 entries each (these entries correspond to itineraries served by a carrier). In all cases, the time needed for the extraction of optimal or suboptimal routes was negligible compared to that needed for the establishment of connection and the retrieval of the corresponding data from the remote databases. Second, the integrated in our approach multiple-criteria decision support tool was actually an "off-the-shelf" solution, which had been already validated and successfully employed in a real e-commerce application. Its integration in the proposed system was straightforward, basically due to its adaptable and expandable implementation. This tool has been also tested in the transportation context (by five users with a considerable experience in performing various e-business transactions), while the feedback received concerning the easiness to use it, the "look-and-feel" of its interfaces, the appropriateness of the features provided, and the clarity of the functions it supports was very positive. Third, the system's design closely followed a comprehensive identification of the requirements of the users involved (both carriers and customers). Such requirements arose primarily through interviews conducted with seven representatives of national and international transportation companies, and secondarily

through questionnaires filled in by people with experience in seeking transportation solutions. Finally, the platform adopted for the implementation of our system is admittedly a robust solution, which can improve and accelerate business operations carried out through the Internet.

Generally speaking, we argue that full exploitation of software agents technology may further enhance the solution of the overall transportation problem, fully satisfying requirements such as continuous look-up for the construction of feasible solutions, dynamic updating of the users involved for the feasible (optimum or suboptimum) solutions, dynamic exploration of the market in order to satisfy a service request or offer, dynamically triggered initiatives in order to establish a deal, ability of negotiation and multi-criteria decision support, and dynamic maintenance of the users profiles for the evaluation of the priority of their requests and their overall "attitude" in the e-market [3,6].

Our primary future work direction is the full implementation of the electronic marketplace presented in this paper, according to the above directions. Issues to be also addressed concern the enhancement of the Transport Manager tool in order to take into account additional constraints (capacity of carriers, transshipment times) in the synthesis of modular solutions, the interoperability of our approach with legacy systems running on diverse carriers' platforms, and the dynamic formulation of the agents' strategy, which will be based on the exploitation of data from previous transactions (for instance, according to such an approach, a carrier's agent will be able to apply different pricing and scheduling policies to each customer's request). Another direction concerns research on alternative ways of coordination among customers and carriers in the proposed framework. This is associated with the consideration of various types of middle-agents, such as *mediators*, *match-makers*, etc. [8,5]. Finally, further work needs to be done on the definition of semantics and syntax of data and knowledge conveyed among the agents of our marketplace.

Appendix A. Formal description of agents' role schemata

In the following 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".

Role Schema: CustomerHandler (CSH)

Description: Receives transportation requests from the customer.

Protocols and Activities: <u>AwaitCall,</u> <u>GetCustDetails,</u> <u>GetCustReq</u>, 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 = <u>AwaitCall.</u> <u>GetCustDetails</u>. (GenerateQuote)$^{\omega}$

        GenerateQuote = <u>GetCustReq</u>. TransportRequest

Safety:

        • *customerDetails = invalid ⇒ transportRequestQuote =nil*

        • *customerRequirements = invalid ⇒ transportRequestQuote =nil*

Fig. 7. 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, <u>EvaluateProposal</u>, DecisionMade, <u>InformCustomer,</u> 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 = <u>EvaluateProposal</u>. (DecisionMade. <u>InformCustomer</u> | AmendedTransportRequest.

                           CustomerDecisionMaker)

Safety:

        • *true*

Fig. 8. The CustomerDecisionMaker role.

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: <u>AwaitCall,</u> <u>RegisterCarrier,</u> <u>GetCarrInfo,</u> <u>GetCarrUpdInfo,</u> <u>UpdateCarrDB</u>

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 = <u>AwaitCall.</u> <u>RegisterCarrier</u>. (([<u>GetCarrInfo</u>]. [(<u>GetCarrUpdInfo</u>)$^{\omega}$]) || <u>UpdateCarrDB</u>)

Safety:

        • *carrierRegistrationDetails = invalid ⇒ quote =nil*

        • *carrierSpecificationDetails = invalid ⇒ quote =nil*

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

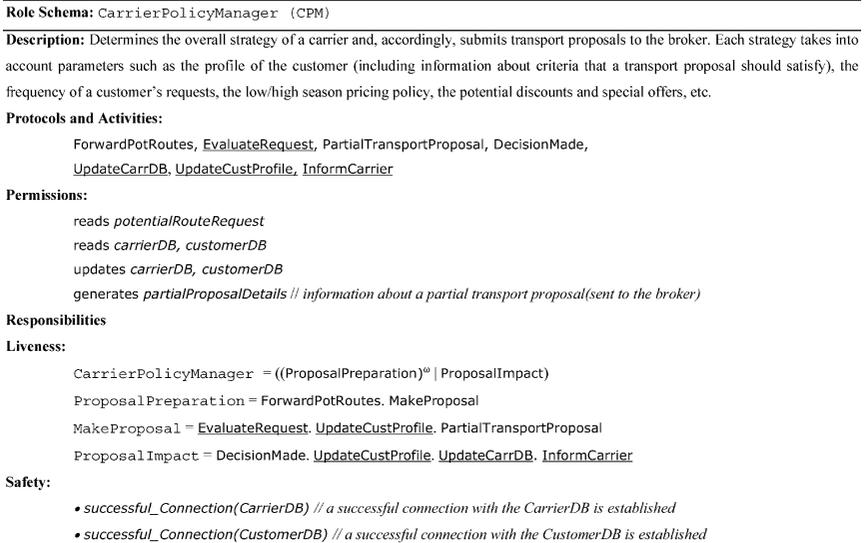---

Fig. 9. 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*

Fig. 10. The `CarrierPolicyManager` role.

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
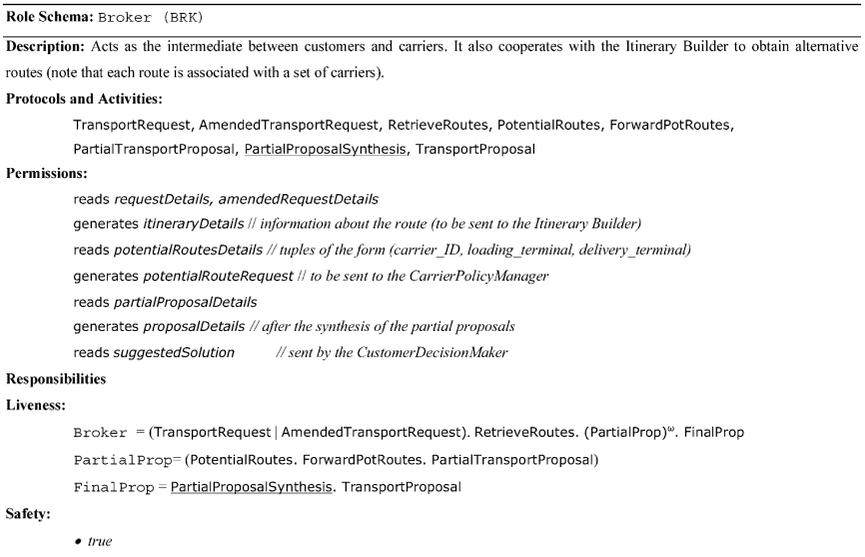
Safety:

• *true*

Fig. 11. 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*

Fig. 12. The `ItineraryBuilder` role.

Appendix B. The behavior of the `ItineraryBuilder` agent

```
begin
get  transport_details  (Code,  Cust_ID,  Loading_Term,
Destination, Max_Cost, Max_Time) from the Customer;
Route_opt ← NULL;
  // Set algorithm variables into initial values
Cmax ← Max_Cost;
Tmax ← Max_Time;
Direct_Transport ← FALSE;
    // Assume that no efficient solution will be found
Modular_Transport ← FALSE;
for (each Carrier registered in the E-Market) do
  // Examine all Carriers' DBs
{
   try to
   {
   establish connection to Carrier (ID) DB;
     // Connect to Carrier's SQL Server
   Query Carrier's DB (Table = "Itineraries");
     // Find the Itineraries Table
   Generate RoutesList DataSet;
     // Create an empty Table
   }     // END TRY TO
   for (each RoutesList in Dataset)
     // for each set of routes in carrier's DB
   {
      for (each DataRow in RouteList) do
        // Try to construct a Direct Solution
```

```
{
if  ((Loading_Term  =  Transport_details.Loading_
Term)  AND  (Destination  =  transport_details.
Destination)
{
  if  ((Cmax  ⩽  Datarow.Cost)  AND  (Tmax  ⩽  Data-
  row.Time))     // Check cost and duration
  {
  Direct_Transport ← TRUE;   // A direct solution
  is found
  Route_opt ← Datarow;
  insert (Datarow) into Table (Direct_Solution);
  SORT  Table  (Direct_Solution)  BY  Cost  THEN  BY
  Time;     // Sort the Table
  next DataRow;     // Search the next set of routes
  }
  next RouteList;
  }     // END IF
}     // END FOR
else     // Try to construct a modular solution
{
  Step 1
  search  in  "Itineraries"  Dataset  for  RouteLists.
  Code;     // Search all different route codes
  for each N different RouteList.Code found do
  {
    if there exist Datarows with
      // Check the starting and ending point of each
    route
    (DataRow[l].Loading_Term  =  Transport_details.
    Loading_Term AND DataRow[N].Destination = Trans-
    port_details.Destination)
    Step 2
    {
      Modular_Transport ← TRUE;
        // try to construct a modular solution
      Route_Arc_ij ← {code, i (Loading), j (Destina-
      tion), C_ij, D_ij};
      Insert  Route_Arc_ij  into  Table  (Modular_
      Solution);
      SORT Table (Modular_Solution) BY C_ij;
        // Sort the table by cost
      for each Route_Arc_ij in {Modular_Solution} do
```

```
Route_Arc_ij ← min {C'_ij, C''_ij, ... with D'_ij = D''_ij =
...};
  // for route arcs with the same duration find
the
  // route arc with minimum cost
Step 3
use Shapiro's Algorithm to find Route_ij;
if (D_ij ⩽ Tmax AND C_ij ⩽ Cmax) C_opt ← Route_ij;
  // an optimum route is found
else if (C_ij ⩽ Cmax AND D_ij > Tmax)
  // Try to substitute every duration that
exceeds
  // the upper limit
Step 4
{
  for each Route_Arc_ij ∈ {min (Route_ij)} do
  {
    find the Cost increasement ∀ new D_ij;
      // Calculate the cost increment for each
      // duration substitution
    dC'_ij ← C'_ij − C''_ij;
    Sort (Route_Arc_ij) BY dC_ij;
      // Sort all route arcs by cost increment
    next Route_Arc_ij;
  }    // end for
  Step 5
    find min (dC_ij);
      // Find the minimum cost increment
    Route_Arc_ij.Cost ← C_ij;
    if (C_ij ⩽ Cmax AND D_ij ⩽ Tmax) Route_opt ←
    Route_ij;
    // If the cost and the duration of the
    route
    // don't exceed the upper limits then an
    // optimum solution is found, else go to
    step 6
    Step 6
    else if (C_ij ⩽ Cmax AND D_ij > Tmax)
    {
      Repeat
        // Substitute the cost C_ij until D_ij ⩽
      Tmax
```

```
                {
                for each Route_Arc_ij ∈ {min (Route_ij)}
                do
                find next (C''_ij > C'_ij);
                Route_ij ← Route_ij + Route_Arc_ij (C''_ij);
                if (C_ij ≤ Cmax AND D_ij ≤ Tmax)
                Route_opt ← Route_ij;
                  // An optimum solution is found
                }
              until ((C_ij ≤ Cmax AND D_ij ≤ Tmax) OR (C'_ij >
              C_ij))
            }     // Step 6
          if (C'_ij > C_ij) Modular_Transport ← FALSE;
            // the algorithm cannot construct a modular
            // solution
        }     // Step 4
      }     // CONSTRUCT MODULAR SOLUTION
    next Carrier     // search next carrier's database
  }     // END FOR
  if  exist  (Direct_Transport  OR  Modular_Transport)
  perform potential solution ranking;
      // Sort all potential routes by cost and then by
    // duration inform the customer;
  end     // end of algorithm
```

References

[1] Y. Bakos, Towards friction-free markets: the emerging role of electronic marketplaces on the internet, Communications of the ACM 41 (8) (1998) 35–42.

[2] M.S. Fox, M. Barbuceanu, R. Teigen, Agent-oriented supply-chain management, International Journal of Flexible Manufacturing Systems 12 (2/3) (2000) 165–188.

[3] G. Froehlich, H.J. Hoover, W. Liew, P. Sorenson, Application framework issues when evolving business applications for electronic commerce, Information Systems 24 (6) (1999) 457–473.

[4] N. Jennings, P. Faratin, T.J. Norman, P. O'Brien, B. Odgers, Autonomous agents for business process management, International Journal of Applied Artificial Intelligence 14 (2) (2000) 145–189.

[5] R. Guttman, A. Moukas, P. Maes, Agents as mediators in electronic commerce, Electronic Markets 8 (1) (1998) 22–27.

[6] N. Karacapilidis, P. Moraitis, Building an agent-mediated electronic commerce system with decision analysis features, Decision Support Systems 32 (1) (2001) 53–69.

[7] N. Karacapilidis, P. Moraitis, Intelligent agents for an artificial market system, in: Proceedings of the 5th International Conference on Autonomous Agents (Agents-2001), ACM Press, Montreal, 2001, pp. 592–599.

 [8] M. Klusch, K. Sycara, Brokering and matchmaking for coordination of agent societies: a survey, in: A. Omicini et al. (Eds.), Coordination of Internet Agents, Springer, Berlin, 2001, pp. 197–224.
 [9] P. Moraitis, E. Petraki, N. Spanoudakis, Engineering JADE agents with the Gaia methodology. In: Proceedings of the International Workshop on Agent Technology and Software Engineering (AgeS'02), 2002.
[10] F.J. Shapiro, Mathematical Programming: Structures and Algorithms, John Wiley and Sons, New York, 1979.
[11] W. Shen, D.H. Norrie, An agent-based approach for dynamic manufacturing scheduling. In Proceedings of Autonomous Agents'98 Workshop on Agent-Based Manufacturing, Minneapolis/St Paul, MN, 1998, pp. 117–128.
[12] K. Sycara, D. Zeng, Coordination of multiple intelligent software agents, International Journal of Cooperative Information Systems 5 (2–3) (1996).
[13] M. Wooldridge, N.R. Jennings, D. Kinny, The Gaia methodology for agent-oriented analysis and design, Journal of Autonomous Agents and Multi-Agent Systems 3 (3) (2000) 285–312.