Interface Agents' Design for a DRT Transportation System using PASSI

Claudio Cubillos and Sandra Gaete

Pontificia Universidad Católica de Valparaíso
Escuela de Ingeniería Informática
Av. Brasil 2241, Valparaíso, Chile.
claudio.cubillos@ucv.cl, sandra.gaete@gmail.com

Abstract. The present work continues a longer research in the field of flexible transportation services and the design of an agent system devoted to the planning, scheduling and control of trips under such a domain. In particular, this paper focuses in the design and development of the interface agents present in the system by following an agent development methodology named PASSI. The interface agent devoted to interaction with the customers is explained in detail and its prototype is shown.

1 Introduction

In the last two decades, the mobility needs of European citizens have radically changed. The need to cover more diffuse travel patterns, varying periods of low demand, city-peripheral journeys, as well as commuting trips has leveraged the need of Demand-Responsive Transport services (DRTS) in which routes, departure times, vehicles and even operators, can be matched to the identified demand allows a more user-oriented and cost effective approach to service provision.

Software agents are defined as autonomous entities capable of flexible behavior denoted by reactiveness, pro-activeness and social ability [1]. Multiagent systems (MAS) consist of diverse agents that communicate and coordinate generating synergy to pursue a common goal. In this context the present work describes the design of a multiagent system using the agent development methodology called PASSI [2] for modeling a passenger transportation under a flexible approach. In this way, it gives continuity to our past research [3] [4] on heuristics for solving scheduling of passenger trips. In particular the paper focuses in describing the design of the interface agents for the main actors; Customer and Driver, for then detailing the interface agent prototype devoted to Customers.

The paper scope moves toward the design description of interface agents using PASSI withing the context of a complete system. Although literature is plenty of agent systems, agent software engineering (AOSE) is less common. In the particular case of PASSI, finding complete designs in addition to the examples developed by the own creators is not an easy task. Furthermore, a practical design with PASSI devoted to interface agents, showing how interface events are tackled and mapped in terns of tasks, roles, the granularity required, etc. is not

present in literature (at least at the best of our knowledge). Therefore our aim is to somehow cover this lack.

2 Related Work

The research on Multi-Agent Systems (MAS) has deserved an increasing interest in the Intelligent Transportation Systems (ITS) domain. One ITS area of MAS development has been Urban Traffic Control (UTC) systems. In 2000, Ou [6] presented a UTC, which adopted MAS technology based on recursive modeling method (RMM) and Bayesian learning. Ferreira et al. [7] presented a multi-agent decentralized strategy where each agent was in charge of managing the signals of an intersection and optimized an index based on its local state and "opinions" coming from adjacent agents.

In the Advanced Transportation Information System (ATIS) field, Kase and Hattori [8] proposed the InfoMirror application that provides agent-based information assistance to drivers through car navigation systems or on-board PCs. Adorni [9] presented a distributing route guidance system, which allowed dynamic route searching using the coordination capabilities of MAS. Bus-holding control tackles the coordination of multiple lines of fixed-route buses and the different stops, seeking the global optimality. In 2001, Jiamin et al. [10] proposed a distributed bus-holding control approach in which a MAS negotiation between a Bus Agent and a Stop Agent was conducted based on marginal cost calculations.

3 Flexible Public Transport Services

Demand Responsive Transport (DRT) services aim to meet the needs of different users for additional transport supply. The use of flexible transport services, where routes, departure times, vehicles and even operators, can be matched to the identified demand allows a more user-oriented and cost effective approach to service provision. The adaptation of the transport services to match actual demand enables cost savings to the operators, society and passengers.

DRT can be seen as an element of a larger intermodal service chain, providing local mobility and complementary to other conventional forms of transportation (e.g. regular buses and trams, regional trains). In this context, DRT provides a range of Intermediate Transport solutions, filling the gap between traditional public bus services and individual taxis.

The final DRT service can be offered through a range of vehicles including regular service bus, mini-bus, maxi-vans, buses and vans adapted for special needs and regular cars. The use of each vehicle type depends on the transport service to offer, the covered area and the target users.

4 PASSI Methodology

The Process for Agent Societies Specification and Implementation (PASSI) is a step-by-step methodology for designing and developing multi-agent societies.

PASSI integrates design models and concepts from both OO software engineering and artificial intelligence approaches using the UML notation. The design process with PASSI is supported by PTK (PASSI Toolkit [16]) to be used as an add-in for Rational Rose.

The PASSI methodology is made up of five models containing twelve steps in the process of building multi-agent. The models are: System Requirements Model, Agent Society Model, Agent Implementation Model, Code Model and Deployment Model. Because of space restrictions, the present work will focus in the first model. Please refer to [2] for a more detailed description on the whole PASSI methodology.

The System Requirements Model corresponds to an anthropomorphic model of the system requirements in terms of agency and purpose. It involves 4 steps: 1) a Domain Description (D.D.), which provides a functional description of the system using conventional use-case diagrams, 2) an Agent Identification (A.Id.), leveraging the separation of responsibility concerns into agents, represented as UML packages, 3) a Role Identification (R.Id.), consisting in use of sequence diagrams to explore each agent's responsibilities through role-specific scenarios and 4) a Task Specification (T.Sp.), detailing through activity diagrams the capabilities of each agent.

5 The Agent-based Transportation System

As stated before, the agent system was designed following the PASSI methodology, making use the PTK (Passi Toolkit) add-on for Rational Rose to develop the different models. The system prototype was implemented over the Jade Agent Platform[11], which provides a full environment for agents to work. In the following the general architecture will be explained for then detailing the interface agents in the next sections. For a more detailed description on the agent architecture and the planning & scheduling mechanism please refer to [5] and [3].

As outlined in the PASSI section, the methodology starts capturing the system's requirements through use cases, for then grouping them together to conform the agents. The diagram in Figure 1 shows part of the use cases and agents involved in the system. Due to space restrictions some of the supporting agents are expressed as actors.

The Client is an interface agent with a GUI. Providing the connection between the end user (Customer) and the transportation system. Through it, the Customer can request a trip by giving a description of the desired transportation service through a *Trip Request Profile*. In addition, through a *Client Profile* it is possible to create and manage personalized services profiles with diverse characteristics and preferences common to the different trips requested by the user.

After a service has been contracted, the Customer can also communicate events to the system (e.g. a delay, a change on the agreed service, or simply cancel). In a similar way, the system can communicate with the Customer, informing him about any eventuality that may happen (e.g. a traffic jam or vehicle break

down) which may imply a delay or change in the service to be provided. This agent will be further detailed in the next section. The Trip-Request agent acts as

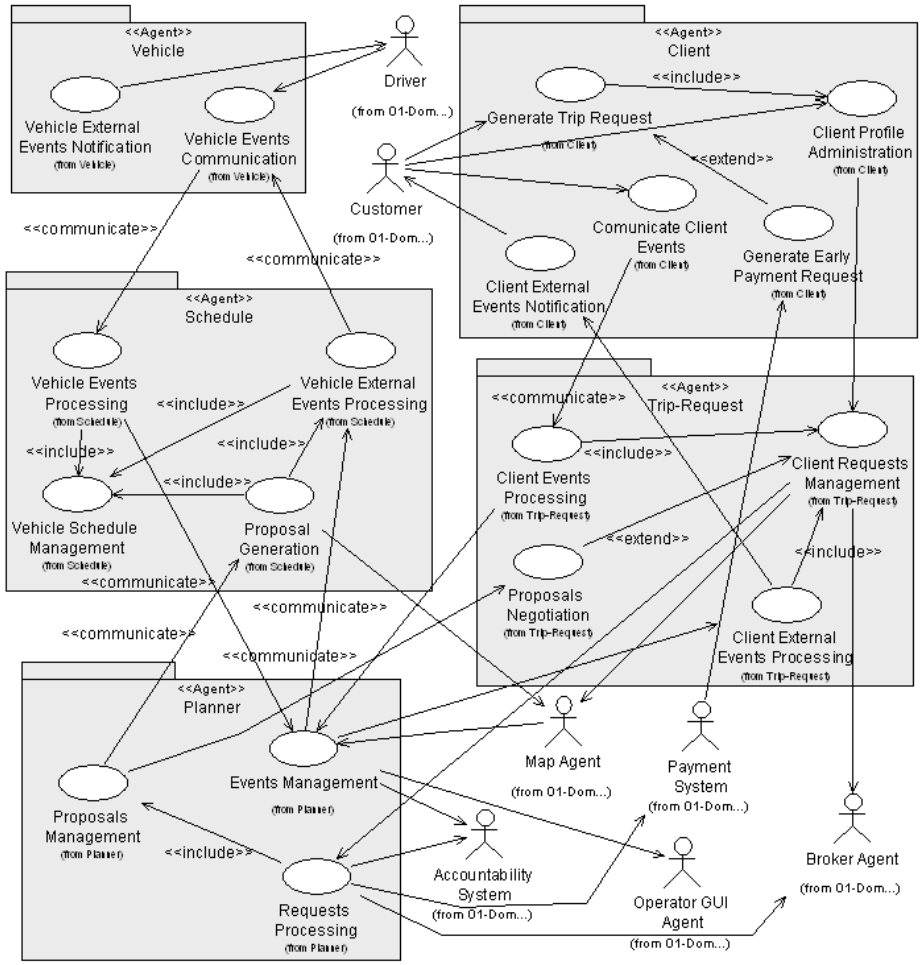

Fig. 1. Portion of the Agents' Identification Diagram

a proxy, representing the Customer in the process of contracting a transportation service. In fact, the trip-request agent is involved in all the interaction of the Customer (through the interface agent) with the transportation system. Its activities regard the management of the client transportation requests, including any negotiation or selection of proposals coming from the Planner, together with processing any events generated by the Customer or by the system. As residing on a device with more processing power (such as a PC), this agent may have

diverse degrees of autonomy for taking decisions on the trip proposal to choose and how to react when faced to eventualities.

The Vehicle is also an interface agent (with a GUI) in charge of providing a monitoring of the route-schedule planned for the vehicle. In addition, it can inform the Driver about any changes to the initial schedule and can be used by him to inform any eventuality (e.g client no show, delay, detour, etc) that may happen regarding the trip and the customers. In particular its interface has been designed to work on-board the vehicle through a touch screen.

The Schedule agent is the one in charge of managing the trip plan (work-schedule) of the vehicle. In addition, the agent is also responsible of making trip proposals upon Planner request and in case of winning will have to update its actual plan to include the new trip. Upon changes (due to vehicle or client events) informed either by the Vehicle or the Planner agent, the Schedule agent will update the plan and reschedule the remaining requests.

Finally, the Planner agent processes all the customers' requests coming through their Trip-request agents. It initiates a contract-net (CNP) [17] with the Schedule agents and manages all the arrived proposals. It is also in charge of managing events that may affect the trip services already contracted and scheduled.

The rest of the actors correspond to supporting-service agents or systems that interact with the diverse agents already detailed, such as the broker, responsible for the initial service matching, the map, providing times and distances, and the payment, responsible for a secure and reliable payment transaction, among others.

5.1 The Agent Interaction

In PASSI the agent interaction is modeled through sequence diagrams that show the diverse scenarios in which agents communicate. The following Figure 2 shows part of the scenario in which a Customer requests a trip service. Each object in the diagram is described following the $\langle role \rangle : \langle agent \rangle$ convention. Therefore, this scenario involves the Customer, Map and Broker actors plus the Client, Trip-request, Planner and Schedule agents.

The scenario starts with the Customer initiating the request of a trip through the client interface by clicking in the top menu. The Customer fills-in the information requested in the form and the interface stores a corresponding *Request Profile*. Then, the *Trip Request Generator* role of the Client generates the Trip-request agent providing the *Request Profile* as argument. This agent initiates the request with the Planner through its *Client Requests Manager* role. The Planner receives the request and queries the Broker for registered vehicles fulfilling the *Request Profile*. The Broker returns a list of possible vehicles and the Planner starts analyzing the alternatives through its *Proposals Manager* role. In fact, this role performs the Manager activities of the Contract-Net protocol. Then, it initiates a call for proposals to the corresponding Schedule agents according to the vehicles' list.

The Schedule agent encapsulates the underlying optimization algorithm for scheduling the trips of the vehicle. In our implementation Schedule agents im-

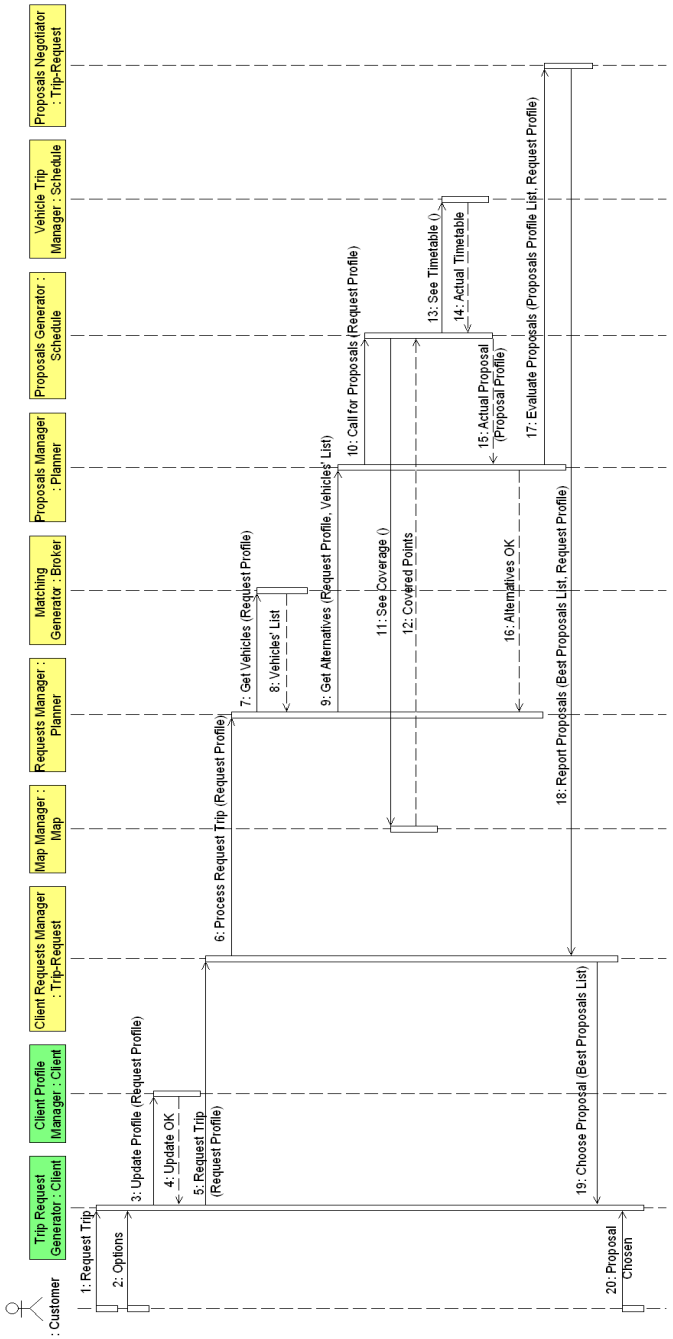

Fig. 2. Roles Identification: Part of the "Customer Requests a Trip Service" scenario

plement a distributed version of a well known greedy insertion algorithm called ADARTW (Please refer to [3] for further details).

The *Proposals Generator* role of the Schedule agent analyzes the *Request Profile*, first by requesting the Map for the coordinates of the pickup and delivery addresses and some paths and distances to evaluate incorporation alternatives to the actual schedule of the vehicle. The Schedule agent turns back a valid proposal or a refuse performative. The Planner waits for proposals until a deadline or until receiving all answers back, asking the Trip-request to evaluate the proposals. The *Proposals Negotiator* role of the Trip-request will process the alternatives and depending on its degree of autonomy, can decide on behalf of the Customer or can report the list of proposals to the Client agent for the Customer to choose.

6 The Client Agent

As stated before, the Client is an interface agent devoted to the Customer-System interaction. In principle, this trip-client assistant may reside on diverse devices (e.g PC, PDA, mobile phone) in order to allow a more flexible and pervasive access to the transportation system. In our prototype, has been developed a Client agent for PC, remaining the versions for more restricted devices as future work. In this sense, it is important to highlight that all the complex processing or decision-making (if delegated by the Customer) has been attached to the Trip-request agent in order to lightweight the Client (the interface agent).

In the following Figure 3 a screenshot of the Client agent GUI is shown, detailing the tab that appears when initiating the request of a trip. In the "Request Data" area, on the left, is asked all the information necessary to detail a transport service request under the demand-responsive considered scenario. This regards the date, the pickup and delivery points (addresses), the corresponding times and other specific information such as the required seats and diverse vehicle characteristics.

It is important to mention that all the concepts involved in the specification of the services make part of a Domain Ontology specific for this transportation domain (for further details on the ontology please refer to [5]).

On the right hand, the available transport services are deployed, showing for each selected service the covered area in terms of street intersections. The services' list can be imported from the system (on line) or from a local file. At the bottom, the Customer can send the trip request and save the services' list.

The PASSI methodology used for the modeling considers a *Task Specification* step. In this activity the scope is to focus on each agents behavior, decomposing it into tasks which usually capture some functionality that conforms a logical unit of work. Therefore for each agent an activity diagram is developed, containing what that agent is capable of along the diverse roles it performs. In general terms, an agent will be requiring one task for handling each incoming and outgoing message.

In the following Figure 4 a portion of the Task Specification Model for the Client Agent is depicted. The diagram shows six tasks that constitute the main

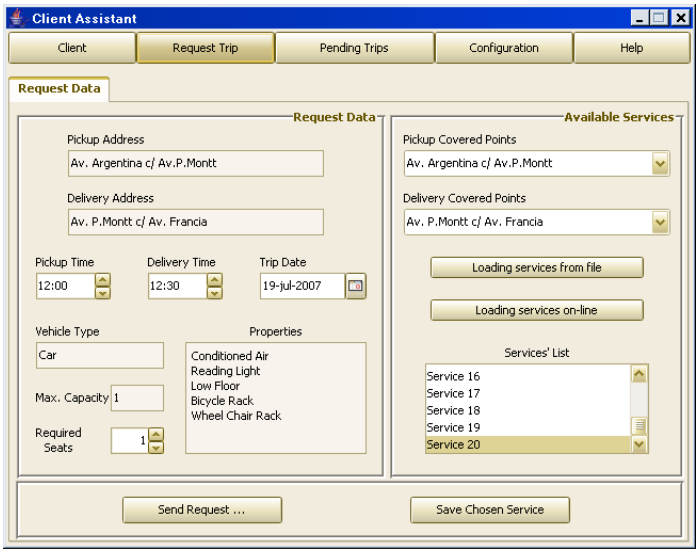

Fig. 3. Client agent GUI showing the "Request Data" tab in the "Request Trip" menu

Client agent capabilities devoted to the process of requesting a transportation service. The *SendQueryAvailableService* task handles the request from the Customer to search for available services and triggers the *ManageClientQuery* task of the Trip-Request agent which is in charge of requesting the Broker for possible transportations services available. These are returned by the *SendActualAvailableService* task of the Trip-request and is received by the *ReceivingAvailableService* task of the Client which processes and decodes the ACL message and forwards the services' list to the *ShowAvailableService* task responsible for displaying the list in the proper form.

The Customer, when making a trip Request Profile (see Figure 3), can browse on the available services (after loading them) in the right-hand area calling to the *ShowAvailableService* task or can send the request (by pressing the button) after filling the left-hand information, calling the *SendTripRequest* task. This Client's task is responsible for sending the *Request Profile* to the Trip-Request, being handled by its *ManageClientQuery* task, which on its turn will forward the request to the Planner.

As explained in the "Customer Requests a Trip Service" scenario of section 5.1, the Trip-request agent will receive from the Planner the trip proposals coming from the different vehicles' Schedule agents and its *SendTripProposals* task will send them to the Client. On its turn, the Client will receive and handle the proposals through its *ShowTripProposals* task, also responsible for displaying them on an appropriate form. Finally, the Customer will be able to select the best alternative calling to the *SendChosenProposal* task of the Client.

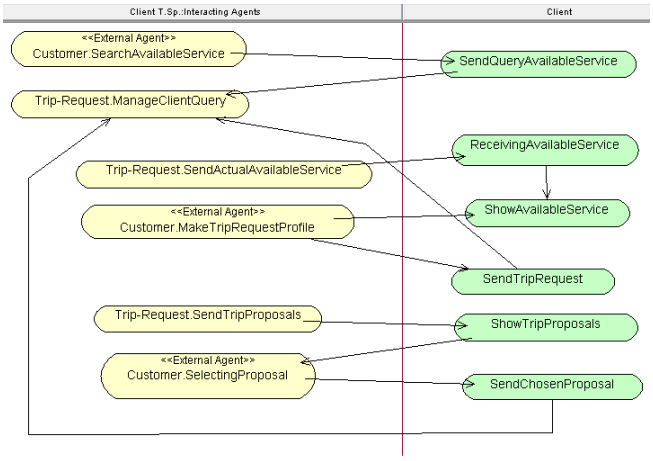

Fig. 4. Part of the Task Specification for the Client Agent, showing the flow of tasks involved in the trip request processing

7 Conclusions

The design of an agent system devoted to passenger transportation under a demand-responsive approach was described. The PASSI methodology used allowed an appropriate level of specification along its diverse phases. The present work focused in the specification of the interface agents for the main actors involved with the system: Customers and Drivers. The Client interface Agent prototype was detailed providing an in-depth example of agent design & implementation using the AOSE PASSI. Future work considers enable the system openness with the implementation of a high-level communication mechanism in order to provide a dynamic participation in such a system.

8 Acknowledgement

This work is part of Project No. 209.746/2007 entitled "Coordinación en una sociedad multiagente dedicada a la programación y control bajo ambiente dinámico", funded by the Pontifical Catholic University of Valparaíso (www.pucv.cl).

References

1. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Massachusetts, USA. 1999.
2. Burrafato, P., and Cossentino, M.: Designing a multiagent solution for a bookstore with the passi methodology. In Fourth International Bi-Conference Workshop on AgentOriented Information Systems (AOIS-2002).

3. Cubillos, C., Crawford, B., Rodríguez, N.: Distributed Planning for the On-Line Dial-a-Ride Problem. F.P. Preparata and Q. Fang (Eds.): FAW 2007. Springer Heidelberg LNCS, vol. 4613,2007, pp. 124-135.
4. Cubillos, C., Rodríguez, N., Crawford, B.: A Study on Genetic Algorithms for the DARP Problem. Mira, J., Alvarez, J.R. (eds.) IWINAC 2007, Part I. Springer Heidelberg LNCS, Vol. 4527,2007. pp. 498-507.
5. Cubillos, C., Gaete, S.: Design of an Agent-Based System for Passenger Transportation using PASSI. Mira, J., Alvarez, J.R. (eds.) IWINAC 2007, Part II. Springer Heidelberg LNCS, Vol. 4528,2007. pp. 531-540.
6. Ou, H. T.: Urban Traffic Multi-Agent System based on RMM and Bayesian Learning. Proc. American Control Conference 2000, pp. 2782-2783.
7. Ferreira, E. D.; Subrahmanian E.: Intelligent Agents in Decentralized Traffic Control. IEEE Intelligent Transportation Systems Conference Proceedings, August 2001, USA, pp. 705 -709.
8. Kase N.; Hattori M.: InfoMirror - Agent-based Information Assistance to Drivers. IEEE\IEEJ\JSAI Intelligent Transportation Systems Conference Proceedings. 1999, pp. 734 -739.
9. Adorni G. "Route Guidance as a Just-In-Time Multiagent Task". Journal of Applied Artificial Intelligence, 1996, 10(2), pp. 95-120.
10. Jiamin Zhao; Dessouky, M.; Bukkapatnam, S.: Distributed Holding Control of Bus Transit Operations. IEEE Intelligent Transportation Systems Conference Proceedings, Oakland - USA, August 2001, pp. 976 - 981.
11. Bellifemine, F. et al: JADE - A FIPA Compliant Agent Framework. C SELT Internal Technical Report, 1999.
12. Brckert, H; Fischer, K.; et al.: TeleTruck: A Holonic Fleet Management System. 14th European Meeting on Cybernetics and Systems Research, 1998, pp. 695–700.
13. Fischer, K.; Mller, J.P.; Pischel, M.: Cooperative Transportation Scheduling: An application Domain for DAI. Journal of Applied Artificial Intelligence, Vol. 10, 1996.
14. Kohout, R; Erol, K. Robert C. In-Time Agent-Based Vehicle Routing with a Stochastic Improvement Heuristic. In Proc. Of the AAAI/IAAI Int. Conf. Orlando, Florida, 1999, pp. 864–869.
15. Perugini, D.; Lambert, D.; et al.: A distributed agent approach to global transportation scheduling. IEEE/ WIC Int. Conf. on Intelligent Agent Technology, 2003, pp 18–24.
16. PASSI Toolkit (PTK) Available at http://sourceforge.net/projects/ptk
17. Smith, R. G. and R. Davis: Distributed Problem Solving: The Contract Net Approach. Proceedings of the 2nd National Conference of the Canadian Society for Computational Studies of Intelligence. 1978.