Agents Capable of Dynamic Negotiations

Garima Parakh, Sandhya Rani, Marcin Paprzycki, Ajith Abraham and Johnson Thomas
Computer Science Department
Oklahoma State University
Tulsa, OK, 74106, USA
gparakh@hotmail.com, ps_rani@yahoo.com, {marcin,jthomas,aa}@cs.okstate.edu

Abstract. Support for negotiation is one of the more important research issues when developing agent systems utilized in e-commerce. While, depending on the type of the transaction, different negotiation procedures need to be utilized, only very few proposed frameworks are generic and flexible enough to handle multiple scenarios. This note presents negotiating agents, which can change their negotiation strategy depending on circumstances. This goal is achieved through dynamic loading of reasoning models and their rule-based selection. We also sketch some details of the initial implementation of such a system.

1. Introduction

The advent of Internet and the rapid development of e-commerce, gave a boost to the agent technology research. While there exist many definitions of agents [GATY], for the purpose of this note we will define them as: encapsulated computer programs, situated in an environment, and capable of flexible, autonomous actions focused on meeting their design objectives [WLD]. Application of such agents is often considered in e-commerce. The number of e-commerce WWW-sites was recently estimated at more than 150000 [GAR] with revenue projections up to $1.5 trillion in 2004 [AND, FOR]. In the context of e-commerce, automated trading using agents are expected to reduce transaction costs, with even better results possible if agents incorporate appropriate "intelligent" trading capabilities. In other words, when agents can reason and negotiate, provide counter-offers and critiques regarding their proposals autonomously and dynamically during the negotiation process.

Most currently existing automated trading systems are not robust enough to become the needed foundation of the next generation of e-commerce. For example, the Kasbah Trading System [MAES] supports buying and selling but does not include auctions; SILKROAD [STB], FENAs [KOW1] and Inter-Market [KOW2] exist as “frameworks” but lack an implementation. This note is a follow up to [PPN] where we proposed a system in which agents can operate according to different business models including auctions, reverse auctions, trading, e-sales etc. This was to be made possible by incorporating a meta-level rule-based support for agents. In addition, each agent was to consist of three independent modules: *protocol*, *strategy* and *communication*, designed as plug-in components loaded remotely on-demand. Here, we will first, outline the rationale and the design of the proposed system and follow with the description of the implementation of the demonstrator system.

2. Negotiations

Negotiation is a method for coordination and conflict resolution. Conflict can be in the form of resolving goal disparities in planning, resolving constraints in resource allocation, and resolving task inconsistencies in determining organizational structure. Overall, research on agent-mediated negotiation can be divided into approaches with their foundation in game theory or in artificial intelligence.

Game-theoretic approaches are based on optimization algorithms (e.g. [ZLT], [AUC]). Their main drawback is that they assume unrealistic properties of the game, e.g. agents that have unbounded knowledge and rationality as well as unlimited computation power and indefinite negotiation time. This makes such approaches impossible to implement. Nevertheless, an extensive research carried out in this field helped develop other theories, e.g. techniques for agents participating in auctions e.g. Dutch, English, Vickery, etc.

Artificial Intelligence based approaches support trading heuristics for different market mechanisms (e.g. [MAES], [JEN1]). AI techniques focus on the negotiation process and utilize techniques such as decision trees, Q-learning and evolutionary algorithms.

When considering the practical aspects of designing multi-agent negotiations, the negotiation protocol, negotiation objects and the reasoning models [JEN2] need to be taken into account.

1. *Negotiation protocol* consists of a set of rules that govern the interaction among agents. Some examples of the rules are: permissible types of participants – negotiators, third parties; negotiation states – accepting bids,

negotiation closed; valid actions of the participant in particular states etc.

2. *Negotiation objects* are issues over which agreement must be reached.
3. *Reasoning model* is the apparatus that participants employ within the negotiation protocol in order to achieve their negotiation objectives, e.g. argumentation, persuasion or heuristics-based. Obviously, the reasoning model depends on both the protocol and the negotiation object [JEN1].

Our architecture is based on managing these three areas: protocol, strategy and negotiation object and we propose an approach to achieving this goal.

3. Conceptual model of the system

It is often suggested that agent mobility is a source of an important advantage of agent systems. Mobile agents support users that are off-line (while continuously working for them). Since transactions carried through mobile devices involve high cost of sending and receiving data, mobile agents can autonomously carry out negotiations and thus reduce cost. However, there is no free lunch. Mobile agents can either be loaded with all the necessary reasoning power or be lightweight. Our approach attempts at designing agents that will remain relatively lightweight, while being able to be well equipped with reasoning powers. To this effect we utilize the concept of dynamically loaded modules. This allows us also make the proposed system more accommodating, by providing agents with a flexibility of selecting negotiating approaches. In our system agents (buyers and sellers) will be composed of a "static core" and plug-in modules that can be remotely loaded when the need arises.

An example, somewhat similar to our proposal, is the Inter-market system [KOW2]. It comprises of mobile agents and intelligent decision-making agents offered as add-on components to the commercial e-marketplace platform Inter-Shop. In the Inter-market, there are two types of agents. Stationary agents run automated processes interacting with other agents or with users. They are a part of the Inter-Market system and their built-in functions cannot be extended or modified by its users. They are parameterized to perform trading tasks according to the users instructions. Mobile agents on the other hand, are used as means of communication when exchanging information between mobile devices and the Inter-Market system.

4. Design of the system

As mentioned above, an agent in our system is composed of a static core and plug-in components (for more details see [PPN]). To keep the agent lightweight it will carry only a part of a table, similar to Table 1 below, containing information about potential sellers (or buyers), resulting from past transactions and/or additional information sources. Only information about "sites" most often negotiated with will be carried, while the remaining part of the table will be kept on the user's machine. In case of dealing with an unknown site or when user's machine is off-line a default strategy will be used (since this will involve only sites that are visited rarely, it should not be detrimental to the performance of the system).

SELLER	PRODUCT	PROTOCOL	STRATEGY	SUCCESS RATE
Seller 1	Used Cars	Offer-Counter Offer	Tit-For-Tat	0
Seller 2	Used Cars	Offer-Counter Offer	Tit-For-Tat	60
Seller 3	Used Appliances	Argumentation	Persuade/Critique	90
Seller 4	Used Appliances	Auction	Heuristics	70
Seller 5	Travel package	Offer-Counter offer	Boulware + Time dependent	40
Seller 6	Travel package	Bidding		
Seller 7	Air tickets	Auction		

Table 1: Sample matchmaking table for negotiation initialization

Each agent (buyer or seller) will consist of three basic modules (the sketch of the design of the agent is depicted in Figure 1).

1. **Communication module** – responsible for communication between the agents; we assume that this is a static module supporting the FIPA ACL language [FIPA]; (while it is possible that this module could be also downloadable and support other communication mechanisms, such a scenario is outside the scope of this note).
2. **Protocol module** – contains general rules of negotiation; when an agent initiates negotiation, on the basis of negotiator table and/or meta-negotiations it finds out which negotiation protocol can be used and dynamically

loads the correct module (from the user's local machine or any agent server, e.g. the nearest one).

3. **Strategy module** – contains policies: set of goals, actions and action rules (triggers); selected strategy depends on an earlier success rate (see Table 1 last column) and the negotiation protocol selected (for example, a strategy used for argumentation cannot be used in case of an auction protocol). In the case, when no information is available a default strategy module is used.

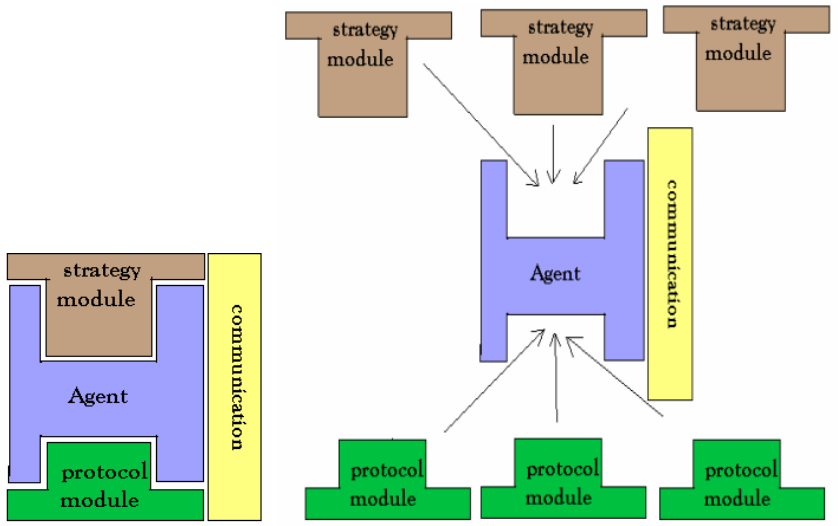

Figure 1. An agent, its static core, and its plug-in components

5. Initial implementation

The initial implementation of the proposed system has been attempted utilizing the JADE agent system [JADE]. The main reason for this selection was the fact that JADE is one of the newest agent environments, it is open-source software and it is FIPA compliant. Before we proceed we have to stress that our objective was to implement dynamically loaded protocols and reasoning modules working with in mobile agents. Our intention was not developing and implementing sophisticated AI algorithms, which will be required in the latter stages of this project. Therefore we have implemented very simple logic that governs the bargaining process. We have assumed that e-commerce agent system involves car buying and selling (but, for all practical purposes, this choice is inconsequential for our implementation).

In our test-implementation we have started with a *personal* agent that is responsible for creating *buyer* and *seller* agents (which are instances of *negotiator* agents). In Figure 2 we present the GUI interface to the *personal* agent. To initiate the process the required fields have to be entered (e.g. the price and the protocol). When the *Start Negotiation* menu item is clicked, the negotiation starts. Let us present it from the point of view of a *seller* agent. Let us assume that a *seller* agent (e.g. Honda car dealer) is created. It then enters a marketplace (we will omit all details related to the organization of the marketplace, or a *Negotiation* Server; environment where the negotiations take place; as they are outside of the scope of this note) and tries to find prospective *buyer* agents by searching for all agents of the *Type=Buyer*. Then, it determines which negotiation protocol is favored by the majority of *buyer* agents (at this stage we assume that a given agent will negotiate alone; it is however possible that a number of agents can work together as a team; here, the *seller* agent could clone itself for each encountered protocol). Once the preferred protocol is determined, the seller agent loads the corresponding protocol module as well as an appropriate strategy module and initiates the negotiation process (in the case of multiple protocols, each *seller* agent clone loads its negotiation protocol and strategy protocol). *Buyer* agents respond and the negotiation process continues until a buyer is found or no buyers are found (the case of multiple agents, the final result of negotiation would have to be further meta-negotiated between the *seller* agent clones).

Figure 2. GUI interface of the *personal* agent

5.1. Negotiation protocols

We have restricted our attention to auctions and implemented two auction protocols that are defined by FIPA for agent negotiations. In accordance with FIPA specifications we created a ***myFipaEnglishAuctionInitiatorBehavior*** subclass and a ***myFipaDutchAuctionInitiatorBehaviour*** subclass of the ***EnglishAuctionInitiator*** class and the ***DutchAuctionInitiator*** classes in the ***jade.proto*** package. Let us now look briefly at the English auction and the Dutch auction protocols.

A. **FIPA English Auction Interaction Protocol:** the auctioneer seeks to find the market price of a good by initially proposing a price below that of the supposed market value and then gradually raising the price. Each time the price is announced, the auctioneer waits to see if any buyers will signal their willingness to pay the proposed price. As soon as one buyer indicates that it will accept the price, the auctioneer issues a new call for bids with an incremented price. The auction continues until no buyers are prepared to pay the proposed price, at which point the auction ends. If the last price that was accepted by a buyer exceeds the auctioneer's (privately known) reservation price, the good is sold to that buyer for the agreed price. If the last accepted price is less than the reservation price, the good is not sold.

B. **FIPA Dutch Auction Interaction Protocol:** the auctioneer attempts to find the market price for a good by starting bidding at a price much higher than the expected market value, then progressively reducing the price until one of the buyers accepts the price. The rate of reduction of the price is up to the auctioneer and they usually have a reserve price below which not to go. If the auction reduces the price to the reserve price with no buyers, then the auction terminates.

5.2. Negotiation strategies

Since the aim of our work was to test-implement an agent system, not to deal with specific negotiation mechanisms we have implemented two simple negotiation strategies. The Seller increments the price by 10 in, what we named, the Heuristics Reasoning module and by 20 in, what we named, the Argumentation Reasoning module (obviously, these names only indicate negotiation strategy that could be used here). Currently we load these two modules randomly as a proof of concept.

6. Running the system

In order to demonstrate the dynamic loading of modules, we need to set-up JADE using multiple containers. There is one *Main* container and the container attached to the *Main* container is called *Container*-1. The *personal* agent resides in *Main* container and all other *negotiating* agents migrate to *Container*-1 after they receive the *preferences* from the *personal* agent. The *Container*-1 may (but does not have to) exist on a remote machine and represents the marketplace. The host name of our machine is *inspiron*. This set up can be seen from the screen shots (Figure 3). Here we present a step-by-step account of running of our system:

1. Start JADE and create the *personal* agent and a *sniffer* agent (to monitor the transaction messages exchange; see Figure 2 and 3)

2. Create a satellite container that acts as the *marketplace*
3. A screen with default values comes up. The default mode is *buyer*. Since this is an auction, create two or three *buyer* agents and then create a *seller* agent
4. Create a *buyer* agent (e.g. Garima), give it a name and enter a reserved price $1000. Select the protocol e.g. English Auction, Click on Submit button.
5. Create a second *buyer* agent (e.g. Sandhya), give it a name and enter a reserved price $1500. Select the protocol e.g. English Auction and Submit.
6. Create a *seller* agent (e.g. Honda Dealer), give it a name and enter a reserved price $900. Change the protocol to Dutch Auction and Submit. (although Dutch auction is selected, the *seller* will actually load English Auction since the two *buyers* prefer the latter).
7. Initiate the *sniffer* agent for all agents created
8. Click on Negotiation Server → Start Negotiation. A call for proposal (containing the proposed selling price of the car) will be sent by the Honda Dealer to the buyers Garima and Sandhya. Buyers will reply by a REFUSE or PROPOSE message and negotiation will continue until a single buyer remains.
9. When the auction stops, we check the last negotiating message. Honda Dealer will choose the *buyer*: Sandhya and the selling price will be $1500.

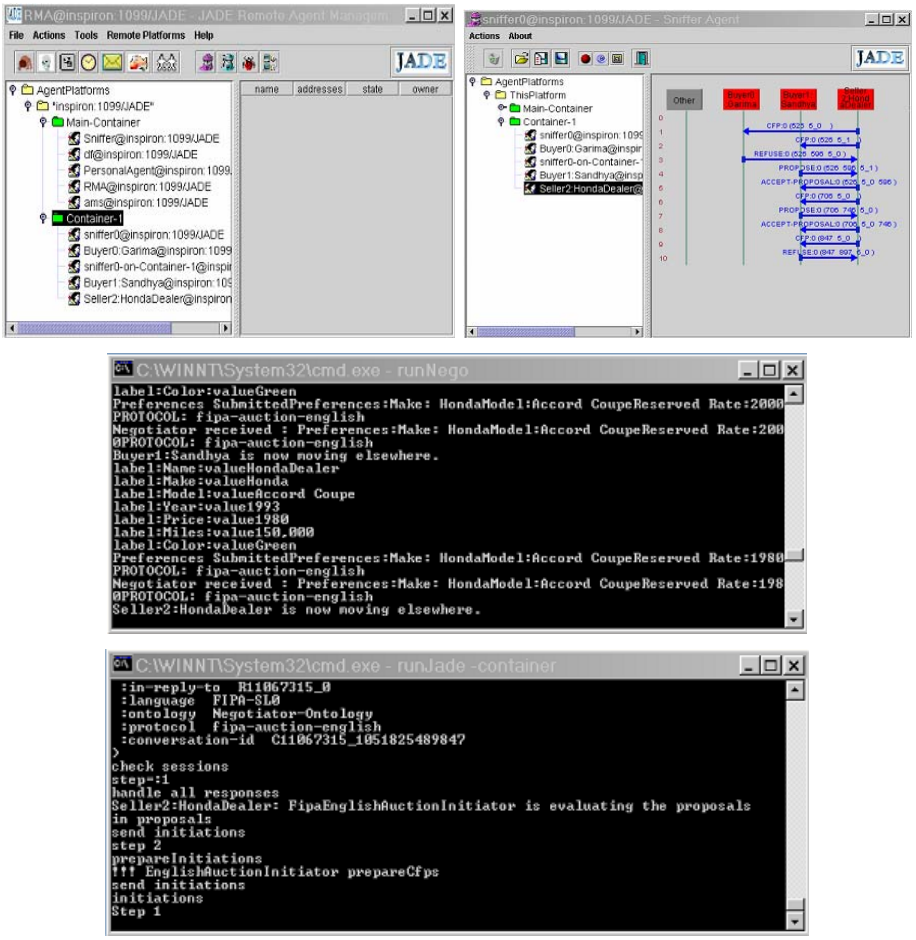

Figure 3. Screen captures of the developed system in action (left top: JADE environment with two containers, right top: sniffer agent, remaining two: agent mobility and dynamic loading of reasoning modules)

7. Conclusions

In this note we presented an agent framework capable of dynamic negotiations depending on the circumstances. In addition, we have discussed details of a JADE based implementation of an actual demonstrator system. We have shown that the proposed framework, using rule based selection and dynamic loading of reasoning

modules is capable of facilitating dynamic negotiations scenario.

Obviously, the implemented system is highly simplified, as our goal was to perform an initial assessment of the feasibility of our approach. In the future we plan to continue developing its capabilities in two main directions. The first one is associated with agent technology itself. Here we have to test the technical limits of JADE as the agent platform. To do this we have to, among others, increase the number of agents and computers; test the above-mentioned possibility of utilizing agent cloning; test the assumption that sending modules over the network improves performance of the system. The second is associated with the negotiation process itself. Here we have to take the existing results and actually implement them in our agents and use such agents in actual e-commerce scenarios.

References

[AND] Andersen Consulting. http://www.andersen.com
[AUC] AuctionBot. http://auction.eecs.umich.edu
[BEAM] P. Beam, M. Segev et. al., On Negotiation and Deal Making in Electronic Markets
[JEN3] A. Beer, M. d'Inverno, P. Luck, P. Jennings (1999), Negotiation in Multi-Agent Systems
[BEN] D. Benameur, A. Chaib-draa, H. Kropf, Multi-item Auctions for Automatic Negotiation
[MAES] V. Chavez, P. Maes, Kasbah: An Agent Marketplace for Buying and Selling Goods
[FIPA] http://www.fipa.org
[FOR] Forrester. http://www.forrester.com
[GATY] V. Galant, J. Tyburcy (2001) Intelligent Software Agent (in Polish), in: A. Baborski (ed.), Knowledge Acquisition in Databases, Wrocław University of Economics
[GAR] Gartner. http://www.gartner.com
[GUT] H. Guttman, P. Maes (1998), Agent-mediated Integrative Negotiation for Retail Electronic
[GRF2] F. Griffel, W. Lamersdorf, M. Merz, A plug-in architecture for providing dynamic negotiation capabilities for mobile agents
[GRF3] F. Griffel, W. Lamersdorf, M. Merz, Interaction-Oriented Rule Management for mobile agent applications
[JADE] http://jade.cselt.it
[JEN2] P. Jennings, S. Parsons (1998), On Argumentation-Based Negotiation
[KOW1] R. Kowalczyk, On Fuzzy e-Negotiation Agents: Autonomous negotiation with incomplete and imprecise information
[KOW2] R. Kowalczyk, B. Franczyk, A. Speck, Inter-Market, towards intelligent mobile agent E-Market places
[KRS2] V. Kraus, et al. (1998), Reaching agreements through argumentation: a logical model
[LES] D. Lesser, Negotiation among computationally bounded self-interested agents
[STB] S. Michael, Design of Roles and Protocols for Electronic Negotiations, Electronic Commerce Research Journal, Special Issue on Market Design, Vol.1
[PPN] G. Parakh, M. Paprzycki, C. E. Nistor, Dynamically Loaded Reasoning Models in Negotiating Agents, Proceedings of the 3rd European E-COMM-LINE 2002 Conference, Bucharest, Romania, 2002, 199-203
[PAR] M. Parunak, Characterizing Multi-Agent Negotiation
[KRS1] H. Sarit, V. Kraus, Negotiation and Cooperation is Multiagent environment
[JEN1] M. Sierra, G. Faratin, Jennings, Service Oriented Negotiation Model between Autonomous Agents
[GRF1] M. T. Tu, F. Griffel, W. Lamersdorf, Integration of Intelligent and Mobile Agents for E-commerce
[TAMBE] X. Qiu, F. Tambe, Flexible Negotiation in Teamwork
[WLD] M. Wooldridge (1997) “Agent-based software engineering” IEEE Proc. on Software Engineering, 144 (1) 26-37.
[ZLT] G. Zlotkin, J. S. Rosenschein, Cooperation and conflict resolution via negotiation among autonomous agents in non-cooperative domains