INTELLIGENT AGENTS ACTING AS ARTIFICIAL EMPLOYEES IN AN ELECTRONIC MARKET

Nikos Karacapilidis
Industrial Management Laboratory, Dept. of Mechanical Engineering
University of Patras, GR 26504 Rion Patras, Greece
nikos@mech.upatras.gr

Pavlos Moraïtis
LAMSADE, University of Paris-Dauphine
75775, Paris Cedex 16, France
moraitis@lamsade.dauphine.fr

ABSTRACT

This paper describes an agent-based electronic market system whose underlying communication and cooperation protocols establish an artificial environment with advanced features. Using the system, actors (i.e., customers and merchants) can delegate a variety of tasks to personal software agents that act as their artificial employees. Contrary to other approaches, where a new agent is launched when their associated actors intend to perform a buying or selling transaction and "lives" only while this transaction is processed, our approach builds on a personalization of agents that permanently "live" in the market representing their actors' interests. More specifically, beyond just requesting and proposing an offer, agents in our system maintain a profile of their owners, which is updated upon the actor-agent interaction type, and can proactively ask their owners' permission to initiate a transaction (e.g., when a new product, which match one's profile, appears in the market). Furthermore, the system is enabled with a highly interactive multiple criteria decision making tool that can handle incomplete, inconsistent and conflicting information during a purchase transaction, and perform a progressive synthesis and comparative evaluation of the existing proposals.

1. Introduction

Most basic characteristics of software agents, such as autonomy, proactiveness and "intelligence", together with their ability to cooperate, make them suitable for the delegation of traditional commercial transactions. As many analysts/researchers predict, agent-mediated electronic commerce would further revolutionize Internet Commerce [Nwana et al. 1998]. Work done during the last few years has dealt with a diversity of tasks involved in buying and selling goods and services in an electronic market (for a survey, see [Guttman et al. 1998]). There is already a plethora of systems automating the tasks of *product brokering*, *merchant brokering* (e.g., Kasbah [Chavez and Maes 1996]), and *negotiation* (e.g., AuctionBot [`http://auction.eecs.umich.edu/`], eBay's AuctionWeb [`http://www.ebay.com`], Fishmarket [`http://www.iiia.csic.es/Projects/fishmarket/newindex.html`]). However, the above works do not deal with important issues, such as:

- The permanent existence of agents in the electronic market (e-market); that is, agents that do not "live" only during a specific transaction but much longer, upon the subscription paid by their owners at the time they were launched (an actor may "hire" an agent for a month, a year, etc.).
- The proactiveness and semi-autonomy of agents; that is, agents that take the initiative to contact their actors in order to start a transaction that seems "interesting" to them (e.g., when a new product, which matches one's profile, appears in the market). Semi-autonomy of agents assures the right level of control for the actions they could take (a fully autonomous agent could cause problems).
- The maintenance of each actor's *profile* through the personalization of the agents involved; for instance, a customer's agent is supplied with a number of general interests (e.g., classical music, cruises) and preferences (e.g., one may dislike the black color on any product) of its actor, which can be enriched with more detailed ones each time the customer initiates a transaction. This gets extra value when agents "live" permanently in the market (see first issue above).
- The ability of a seller agent to refine some of the customer's purchase criteria, argue in favor or against them, or even bring up new information to persuade him/her to accept its offer.
- The ability to handle incomplete, inconsistent and conflicting information during a purchase transaction, and perform a progressive synthesis and comparative evaluation (across a set of attributes) of the existing

proposals. This requires a highly interactive tool, based on multiple criteria decision theory, that enables customers easily examine alternative scenarios (by selecting which of the proposals' attributes to be taken into account) and recommends the best solution according to the information at hand.

This paper describes an agent-based e-market system that addresses the above issues. It should be noted here that our approach is not based on pre-classified ads; instead, the system's agents cooperate in a real-time mode. The reader should also keep in mind that issues such as ordering, security, payment and delivery, while equally important in commercial transactions, do not fall in the scope of this paper. The remainder of the paper is structured as follows: The proposed e-market framework is described in Section 2. The architecture of the system's agents is illustrated in Section 3. The multiple criteria decision making process for a purchase transaction is presented in Section 4. Finally, related work is considered in Section 5, while concluding remarks are given in Section 6.

2. The E-Market Framework

The proposed e-market system is based on a network of communicating agents that act as *artificial employees* of the related actors (i.e., customers and merchants), in that agents perform a series of tasks for them. Actors are logged in the system, "hire" their personal agents (by paying a subscription depending on the time they want them to "live"), create a *profile* for them, and launch them in the e-market.

2.1. Cooperation protocol

An agent's profile serves its *personalization*, that is, the process with which an actor supplies his/her agent with the necessary information to sketch himself/herself. Initially, this information may concern general interests, preferences and constraints, when speaking about a customer, or the market area and set of services offered, when speaking about a merchant. In such a way, a purchaser agent may be aware that its actor is generally interested in classical music and philosophical books while a seller agent that its actor commercializes music CDs and permanently makes offers to its regular customers.

The above "general" knowledge can be updated and/or enriched with more "specific" one each time a transaction is taking place. For instance, at the time a customer is about to buy a certain trip, its purchaser agent will *learn* that he/she currently considers some adventure issues (e.g., the option to dive and rock-climb), prefers exotic destinations, and is not willing to pay more than a certain amount for it. In other words, the purchase database of a purchaser agent consists of a permanent set of preferences, which is taken into account in all future transactions (it can, however, be updated at any time), and the history of all previous transactions, which maintain the extra preferences set by the customer at each time (these can be reused, upon the will of the customer).

A seller agent is also able to update its profile each time it performs a transaction. Moreover, it is able to keep track of and statistically process the history of all previous transactions taken place with each customer. For instance, the second time a purchase agreement has been made with a customer, the latter can be categorized as a *regular* one in order for the seller to send him/her special offers in the future. In general, results of the above statistical processing can be exploited by the offer building strategies of a seller agent (see next section).

E-market transactions in our system are initiated either by an actor or an agent. In the first case, a customer looking for a certain good or service contacts his/her purchaser agent and provides it with all the necessary information; in turn, the purchaser agent requests (from all or some seller agents) offers that may fulfill its actor's interests (standardization issues, emerged here, are not addressed in this paper). Similarly, a merchant may ask his/her seller agent to broadcast or selectively send an offer for a certain product. This leads to the second case above, which is related to the capabilities of our system's agents to be proactive and semi-autonomous. More specifically, a purchaser agent whose profile matches to a merchant's offer takes the initiative to contact its actor and ask his/her opinion to go ahead. In case that the customer is interested in the certain product, the purchaser agent is capable to take action in order to retrieve related offers and evaluate them all. In a similar way, a seller agent, when noticing that the market's purchaser agents continuously discard its offers due to their prices, can suggest its actor to lower them.

Whenever a match between a purchaser and a seller agent is established, the latter gets information about the customer's buying criteria, preferences that may hold among them, as well as constraints explicitly imposed. By getting the above information, a seller agent is able to build an offer that is as close as possible to the purchase request. Having collected a bunch of such offers, a purchaser agent has to consider and evaluate them all, the aim being to eventually recommend the best one to its user. Conflicts among the different seller agents' points of view are usually inevitable; before responding to a purchaser agent's request, each seller agent would have tailored its offer according to the range of goods at hand. Moreover, each seller agent may adopt its own *strategy* and, subsequently, propose an offer that fulfills (some of) the purchaser agent's goals at a certain level. Offers may also differ about the relative values of criteria. In addition, the purchaser and the seller agents may

have arguments supporting or against their preferences. Finally, before making a decision, the purchaser agent may have to confront the existence of insufficient information; that is, information that would be useful for making a decision is missing.

2.2. Communication protocol

Due to the diversity and complexity of the above transactions and cooperation stages, a proper definition of the interactions between humans and software agents, as well as a provision of procedures for data processing automation, are of high importance. Agents interact by exchanging messages of various types. Each message type conveys certain semantics associated to a particular task of an e-market transaction. Each time an agent receives a message, it immediately knows what reasoning procedure it must activate in order to set up the most appropriate answer or action (if any) to the message received, or what kind of update it has to perform in its domain specific knowledge. In other words, each message type concerns a specific kind of interaction between the different kinds of participants in our framework (i.e., customers, purchaser agents, merchants and seller agents). A list of all message types provided in our framework, together with a description of them, appears in Table 1.

Table 1: E-market messages.

Message Type	Sender	Receiver	Description
offerReqMsg	purchaser	seller	It conveys information about the customer's specifications for a product to be purchased
offerPropMsg	seller	purchaser	It concerns an offer proposal; it is actually a reply to an offer request (see previous message type)
custSpecMsg	customer	purchaser	Used to describe customer's criteria, preferences and constraints for a certain product to be purchased
custSpecUpdReqMsg	purchaser	customer	Used to request and provide, respectively, supplementary information regarding the customer's opinion about features, preferences and arguments introduced by a seller agent
custSpecUpdAnsMsg	customer	purchaser	
purchInitReqMsg	purchaser	customer	They concern the interaction that is proactively initiated by a purchaser agent's request about whether the customer is interested in purchasing a (new or promoted) product that matches his/her interests
purchInitAnsMsg	customer	purchaser	
merSpecMsg	merchant	seller	Used to describe a merchant's products; it conveys their specification (similar to custSpecMsg above)
merSpecUpdReqMsg	seller	merchant	These concern the interaction that is proactively initiated by a seller to get extra information (e.g., when new features appear in a related offer request) or up-date existing one (e.g., when its offers get discarded)
merSpecUpdAnsMsg	merchant	seller	
newProdMsg	seller	purchaser	Sent whenever the database of a seller agent is updated with a new or promoted product
newSellAgMsg	seller	purchaser	Sent whenever a new seller agent is uploaded in the market (info about its coordinates and products)
newOfferAnnMsg	merchant	seller	Sent whenever a merchant wants to launch a new offer (for a specific product, under a certain strategy)

3. The E-Market Software Agents

The development of the software agents proposed in our system is based on a generic and reusable architecture, inspired by [Matsatsinis et al. 1999] and [Sycara and Zeng 1996]. Even if the two agent types involved do not have the same functionality, they are built on the same basic architecture principles (see Figures 1 and 2); their constituent modules are tailored according to their specific type (e.g., the decision making module of a seller agent is usually simpler than that of a purchaser agent). The architecture of each agent type is described in detail below.

3.1. The purchaser agent

A purchaser agent is composed of three modules (namely, the *communication, coordination* and *decision making* modules), which run concurrently and intercommunicate by exchanging internal (i.e., *intra-agent*) messages. A purchaser agent remains idle while no messages arrive at its communication module. As soon as a message arrives, the communication module (after transforming it to an intra-agent message) sends it to the coordination module using a message queuing mechanism. All modules adopt this behavior and remain idle

while no messages to be processed are available. The same intra-agent queuing mechanism facilitates all three modules.

3.1.1. The Communication Module

The *communication module* of a purchaser agent is responsible for the agent's interaction with its environment, that is the seller agents and the human user it assists. It sends and receives messages, while internally interacts with the *coordination module*. Its functionality is as follows: an *internal receiver* transforms each internally queued message (produced by the coordination module) to an inter-agent message and, in the sequel, stores it to the *outgoing messages* queue. In the case of selective communication, it also adds the receiving agents' addresses. An *external receiver* handles the opposite case, by transforming each external message received to an internal one and adding it to the *incoming messages* queue. Finally, a *message transmitter* monitors the incoming and outgoing queues, sending the queued messages to its coordination module, to another agent or to its human user, accordingly.

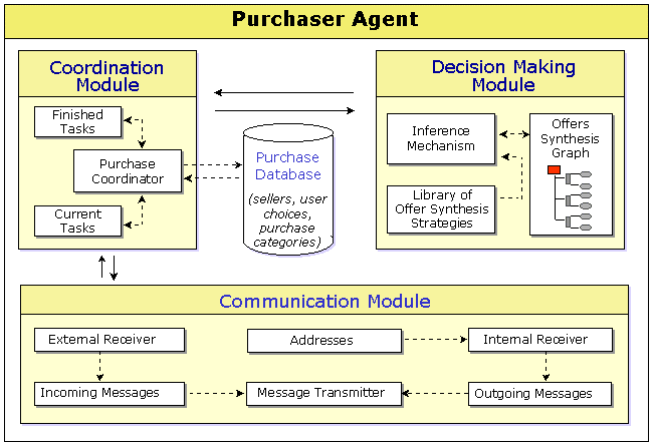

Figure 1: Architecture of a purchaser agent.

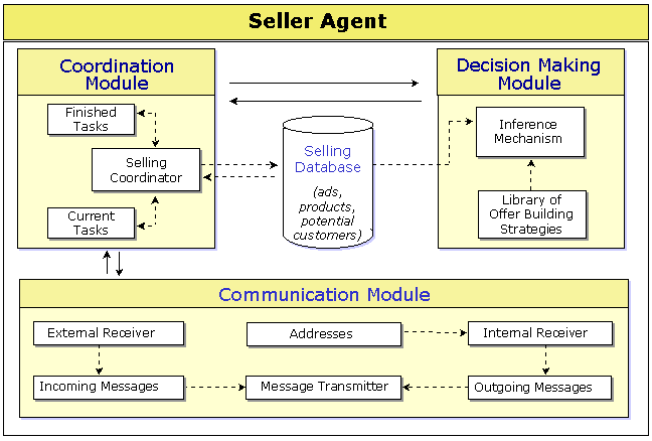

Figure 2: Architecture of a seller agent.

3.1.2. The Coordination Module

This module handles the parts of the cooperation protocol that concern any type of interaction between (i) the purchaser and the seller agents, and (ii) the purchaser agent and the customer it assists. The related message types pass through the *purchase coordinator* component. In addition, the coordination module keeps track of the agent's *finished tasks* (that is, the *purchase history*) and the tasks under evolution (e.g., a purchase evaluation for a specific product can be momentarily suspended due to searching for supplementary information).

As shown in Figure 1, the module interacts with both the communication and the decision making modules. For instance, each time the decision making module needs to interact with the customer, it first sends a message to the coordination module which, in turn, attaches additional information (if required) and forwards it to the communication module. Similarly, the coordination module may filter the content of a received message before forwarding the related data to the decision making module.

In many cases (see Table 1), the purchaser agent has to access its *purchase database*, which contains all necessary information about the sellers (e.g., the products each seller agent provides), user choices (e.g., criteria,

features, preferences and constrains for products the customer is interested in) and finally, purchase categories (the mSQL relational database is used). Retrieving the appropriate data, a purchaser agent is aware of which sellers it can buy a product from, the products each seller agent provides, and the customer choices about a specific product. Update of such a purchaser agent's product database can occur asynchronously, in that the customer may add, remove or refine items of the corresponding lists at any time, independently of the current buying transaction.

3.1.3. The Decision Making Module

The decision making module is composed of three components, namely an *inference mechanism*, a *library of offer synthesis strategies*, and the *offers synthesis graph*. It actually deploys the agent's reasoning mechanism that (i) implements the proactive behavior of the agent by using appropriate rules (which are activated upon the reception of a specific message, sent by a seller agent – see Table 1), and (ii) performs a synthesis and a comparative evaluation of the offers proposed by the seller agents; this process ultimately aims at finding the best offer (to be then recommended to the customer), according to the customer's choices and the information at hand. The inference mechanism is supplied with the necessary knowledge to perform the above tasks.

A *strategy* encapsulates the appropriate information in order for the agent to perform the above comparative evaluation. It prescribes the algorithms to be followed in: (i) conflicting or inconsistent cases; for instance, stating whether the purchaser agent should alert its master in case of a conflict, or simply ignore it and conclude the issue with the consistent parts of the existing information (semi-autonomy of the agent), (ii) the sequencing of the evaluation process, that is specifying when to interact with the customer, whether iterations are allowed, etc., and (iii) the underlying multiple criteria decision making process. Due to the variety of the information and mechanisms involved, the processes of offer synthesis and evaluation are described in detail in Section 4.

3.2. The seller agent

The architecture of a seller agent is similar to that of a purchaser (Figure 2). The communication module has exactly the same functionality with the homonymous module of the purchaser agent.

3.2.1. The Coordination Module

This module is responsible for the cooperation between (i) the seller and the purchaser agents, and (ii) the seller and its merchant. A *selling coordinator* manages the exchange of the related messages (see Table 1). The *selling database* keeps records of the products specification, and potential or regular customers to be informed about the release of a new or promoted product. Merchants may update the related databases of their software agents at any time.

3.2.2. The Decision Making Module

This module consists of an *inference mechanism* and a *library of offer building strategies*. As in a purchaser agent, the inference mechanism of a seller implements its proactive behavior (see Section 2.1). Furthermore, it uses the appropriate strategies to build offers for a requested or promoted product. Each such strategy comprises a set of rules reflecting the selling policy to be followed by the seller agent, and may depend on the specific customer, product to be sold, merchant status, and so on (for instance, a different policy may be adopted when selling a new than a second-hand car).

4. The Decision Making Process

Having defined the architecture of the purchaser and seller agents, as well as their cooperation and communication protocols, this section focuses on the multiple criteria decision making process *per se*. This process takes place in the decision making module of the purchaser agent. The tool implemented for the automation of this process is an extension of the work presented in [Karacapilidis and Papadias 1998a] and [Karacapilidis and Papadias 1998b]. Throughout the rest of this paper, we consider the following example scenario: A customer has uploaded in the e-market his/her assistant purchaser agent and is interested in purchasing a new car. Before asking the agent to initiate a purchase transaction, the customer has well shaped in his/her mind that the criteria of *performance*, *cost*, and *safety* are critical for the buying decision (we assume that, for the moment, he/she ignores the criterion of the *firm's image*). He/she has also ranked the relative importance of performance and safety as the former being more important than the latter. Moreover, he/she intends to pay less than 30,000 Euros. He/she could also desire the maximum speed of the car to be more than 200 km/h. Note that, according to the purchaser agent's strategy, only pieces of that knowledge can be made transparent to the seller agents, the rationale being that the less clear the specification of the customer's intentions are, the more offers will be finally submitted by the seller agents.

4.1. Synthesis of offer proposals

As soon as a purchaser agent gets a new offer proposal, it integrates it with the ones already arrived and constructs an *offers synthesis graph*, which is presented to the customer through the web interface shown in Figure 3 (there is an optional *time limit*, set by the purchaser, after which no more offers for the specific purchase transaction are accepted). In the instance illustrated in Figure 3, there are three proposals arrived so far,

namely offer-12: car-6.1, offer-29: car-A25 and offer-16: car-XY-34, submitted by sellerAgent-33, sellerAgent-12 and sellerAgent-3, respectively. As shown, an offer may consist of:

- *Criteria* (e.g., criterion-22.5: safety; criterion-22.8: cost), together with their associated *features*. Each such feature has a *value* and an *impact* that reflects the opinion of a seller agent about the feature value it provides (e.g., (feature-22.5.3: airbag, 2), (feature-22.8.1: purchase_price, 25000)). Impact can be pro (+), con (-), or neutral (this is depicted through the buttons that accommodate each feature, see Figures 3 and 4); for instance, (feature-33.4.2: sidebars, double) is accommodated by a "f+" button, indicating that its impact is positive, while (feature-22.5.3: airbag, 2) by a "f" button, indicating that its impact is neutral.
- *Preferences*, brought up either by the purchaser agent when requesting an offer (e.g., preference-22.13: (performance, more_important, safety)), or the seller agent itself when replying to such a request (e.g., preference-33.3: (safety, more_important, performance)).
- *Arguments* in favor (e.g., argument-33.13: (report12: "safety was the big issue in 1998 car sales")) or against (e.g., argument-33.11: (if (maximum_speed > 200) then (accident_risk, high))) a preference. Once again, the buttons that accommodate each argument indicate whether an argument speaks in favor or against a preference (see Figures 3 and 4).

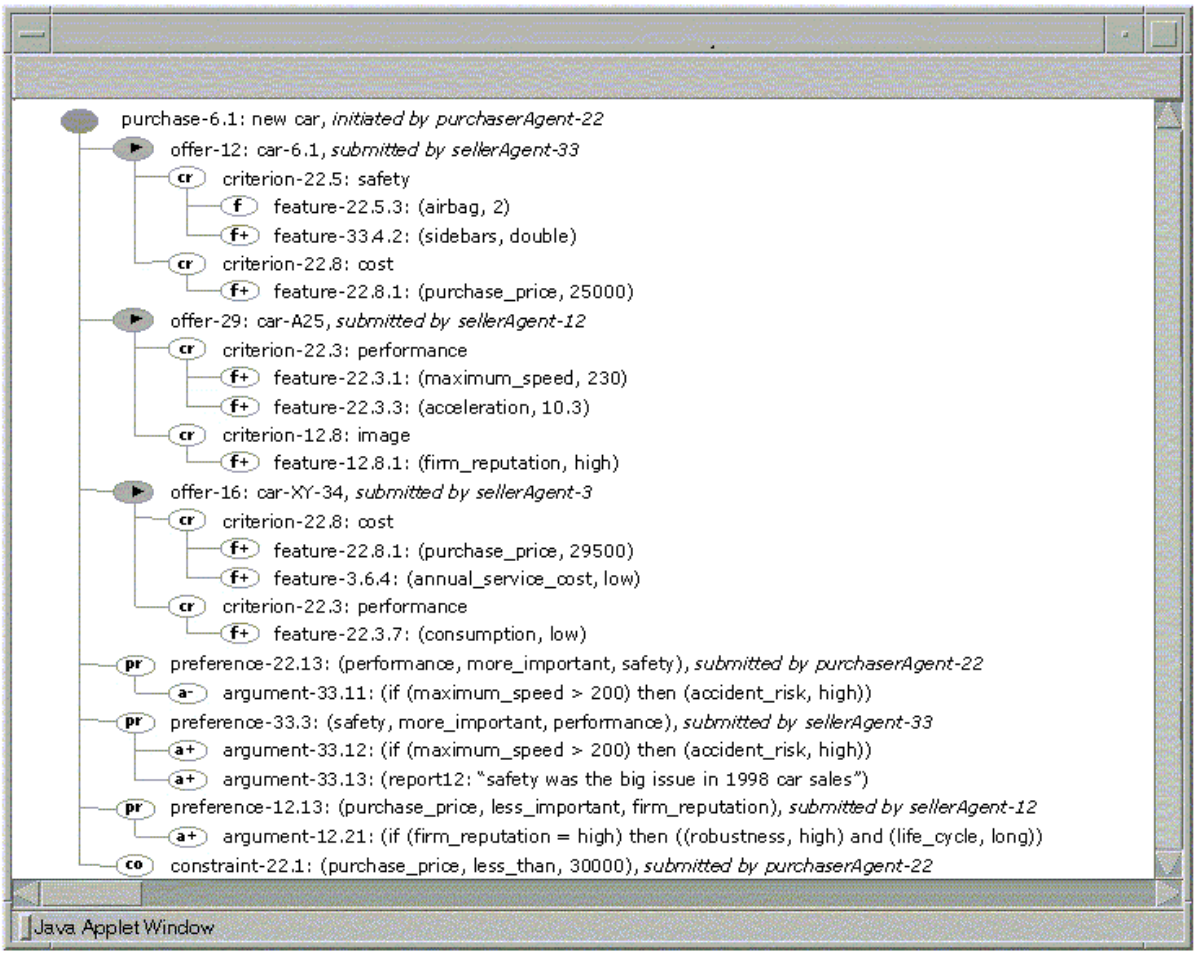

Figure 3: The offers synthesis graph.

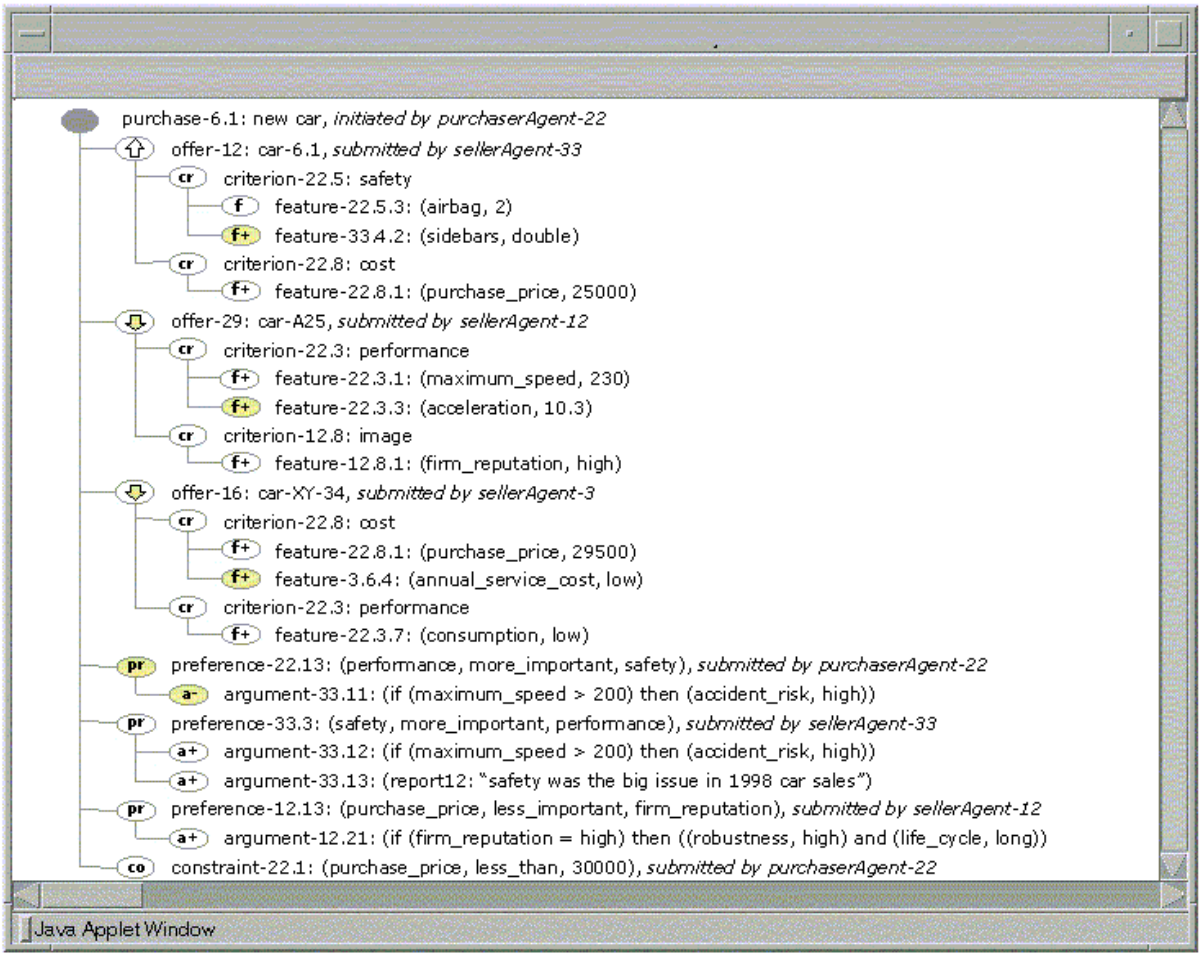

Figure 4: Intervention of customer and purchase suggestion.

An offer synthesis graph also includes the *constraints* asserted by the customer. Constraints are of the form (feature, relation, featureValue), where the desired value may fall into a numerical range (e.g., constraint-22.1: purchase_price, less_than, 30000), a set of discrete values, or a list of predicates.

4.2. Interacting with the user

Each graph entry has an *activation label* (it can be *active* or *inactive*), indicating its current status. By default, all entries are initially active. Viewing the graph through a standard web browser, the customer is able to *inactivate* any of its nodes, the rationale being that their corresponding data types do not suit to his/her interests. Each offer's feature value is checked against the existing constraints. In the example presented here, offer proposals do not violate the constraints imposed so far (this is due to the offer building strategy followed); however, since the strategy followed by a seller agent is not known to the purchaser, this check is always performed at the latter's side. Nodes that violate a constraint become automatically inactive. At any time, the customer may activate again any node and test again the outcome of the decision making procedure. In general, the manual activation/inactivation of the graph nodes enables the customer to further elaborate the problem, by examining various alternative scenarios. Figure 4 shows the status of the offers synthesis graph after the customer's intervention. Inactivation of a node renders all of its children nodes inactive; for instance, inactivation of preference-22.13: (performance, more_important, safety) also inactivates argument-33.11: (if (maximum_speed > 200) then (accident_risk, high)).

4.3. Detection of conflicts and inconsistencies

Apart from an activation label, each preference has a *consistency label*, which can be *consistent* or *inconsistent*. Each time a new preference is inserted in the offer synthesis graph, a mechanism checks if both of its constituent features or criteria exist in another, already inserted, preference. If yes, we distinguish two cases:

- If the new preference also has the same importance relation, it is considered as being *redundant*; thus, it is ignored and not inserted in the graph;
- Otherwise, it is considered as being *conflicting* and it is put next to the preference it was checked against. In such a way, conflicting preferences are gathered together and the system stimulates the user to contemplate on them (that is, to select which one to inactivate), until only one becomes active. Such

an instance is illustrated in Figure 4, with the entries preference-22.13: (performance, more_important, safety) and preference-33.3: (safety, more_important, performance).

If both features (or criteria) of a new preference do not exist in a previously inserted preference, its *consistency* is checked against the already existing active and consistent preferences. For instance, consider the case in which there exist two preferences (feature-x, more_important, feature-y) and (feature-y, more_important, feature-z). This implies for the system that the (not yet inserted) preference (feature-x, more_important, feature-z) also stands. When a new preference (feature-z, more_important, feature-x) is inserted, the system considers it as being *inconsistent* with respect to the existing ones, although it is not directly conflicting with any of them.

Inconsistency checking is performed through a polynomial ($O(N^3)$, N the number of the associated features) *path consistency* algorithm [Mackworth and Freuder 1985]. Although the algorithm interacts with the database where the offers synthesis graph is stored (the public-domain *mSQL* has been integrated in the module), the algorithm is efficient; even for cases involving offers with numerous criteria and associated features, execution time is negligible.

4.4. The weighting schema

Active and consistent preferences participate in the weighting scheme. Note that, for the example illustrated in Figure 4, only preference-33.3 and preference-22.13 are active and consistent. In order to demonstrate how the algorithm for assigning weights to an offer's features works, we use the example shown in Figure 5. There exist five features and four preferences that relate them as illustrated in Figure 5(a). Each feature has a weight=(max_weight + min_weight)/2. Max_weight and min_weight are initialized to some predefined values (in the following, we assume that initially min_weight=0 and max_weight=10). The arrowed lines correspond to the "more important" relation (e.g., (f_1, more_important, f_2)), while the dotted line to the "equally important" relation (e.g., (f_3, equally_important, f_4)). Path consistency explicates all the "more important" relations. More specifically, *topological sort* [Knuth 1973] is applied twice to compute the possible maximum and minimum weights for each feature (Figure 5(b)). The weight is the average of the new max_weight and min_weight; that is, weight(f_1)=6, weight(f_2)=4.5, weight(f_3)=5, weight(f_4)=5 and weight(f_5)=4. Note that weights always have a positive value ($0 \leq$ weight ≤ 10), independently of whether they concern a feature with a positive or a negative impact.

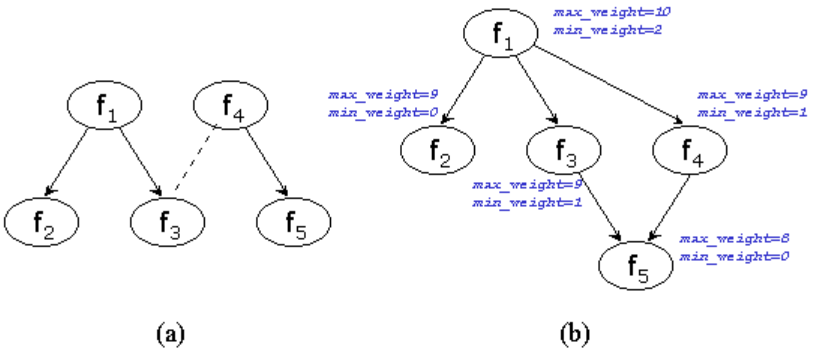

Figure 5: The weighting mechanism

The basic idea of the above schema is that the weight of a feature (or a criterion) is increased every time it is more important than another one (and decreased when is less important), the final aim being to extract a total order of offers. Since only partial information may be given, the choice of the initial maximum and minimum weights may affect the purchaser agent's recommendation. However, the above weighting scheme is not the only solution; alternative schemes, based on different algorithms, have been also implemented.

The *score* of each offer is calculated from the weights of the active features the offer consists of, according to the formula:

$$\text{score}(\text{offer}_i) = \sum \text{weight}(\text{feature}_j) - \sum \text{weight}(\text{feature}_k), \text{ where:}$$

feature_j an active feature which refers to offer_i having a positive impact, and feature_k an active feature which refers to offer_i having a negative impact.

It should be noted here that an offer which includes no product features at all gets score = 0. The scores of offer-12, offer-29 and offer-16 in Figure 4 are 11, 9 and 10, respectively; concerning the first one, both feature-22.5.3 and feature-22.8.1 have score 5.5, while feature-33.4.2 is inactive; similarly, for the other two offers, it is score(feature-22.3.1) = score(feature-12.8.1) = score(feature-22.3.7) = 4.5, while feature-22.3.3 and

feature-3.6.4 are inactive. Therefore, offer-12 is the one recommended by the purchaser agent (as shown in Figure 4, the best proposal is accompanied by an "up-arrow" button, while the rest by a "down-arrow" one). Once again, this may change in the future upon a different configuration (through activation/inactivation) of the offers' features by the customer, or the receipt of a new offer.

5. Discussion

Today's success of agent-mediated electronic commerce in creating new markets and reducing various transaction costs will arguably be boosted by the displacement of the current buying and selling processes with new business models. As concluded in a recent survey [Guttman et al. 1998], such a change will take place as software agent technologies come to better deal with issues such as the ambiguity of content, personalization of preferences, complexity of goals, and dynamics of the related environments.

Several interesting works have been already proposed in the area of agent-mediated electronic commerce. For instance, *BargainFinder* from Andersen Consulting was the first "shop-bot" to provide comparison shopping on the Internet, allowing users to specify the name of a CD or music group and then search on-line stores for the lowest prices available. Excite's *Jango* operated in a similar way. Using *PersonaLogic* customers could impose constraints, to be then exploited by a constraint satisfaction engine in order to prune alternatives that do not satisfy them. Even more sophisticated (in that it provides a set of pre-determined, user-profile based specifications for the requirements of each product category, and multi-attribute utility theory to rank merchant offerings), *Tête-à-Tête* also followed the above approach. Finally, *Kasbah* [Chavez and Maes 1996] helps users creating agents to negotiate the buying and selling of goods on their behalf, also allowing the specification of parameters to guide and constrain an agent's overall behavior.

Compared to the above works, both agent types involved in our approach are more sophisticated. This comes out with the following facts: First, they are conceived as being completely personalized, playing the role of an *artificial employee*. More specifically, a user profile is permanently maintained by his/her corresponding agent and can be updated at any time upon the kind of the user-agent interaction. Second, they act proactively and semi-autonomously, upon the specific setting.

Speaking about the overall e-market framework, our differences are:

- Our agents are not launched when their associated actors intend to perform a buying or selling transaction and do not "live" only while this transaction is processed; instead, they "live" permanently, representing their actors' interests, while monitoring and learning from the e-market environment;
- Our approach is not based on classified ads; instead, offer proposals are created through a real-time cooperation between purchaser and seller agents;
- Our system provides a powerful multiple criteria decision making tool, allowing the efficient synthesis and evaluation of the offer proposals.

Due to all these reasons, we believe that our approach is "closer" to a real business model, in that its communication and cooperation protocols allow the implementation of an artificial setting that better represents a real market.

6 Conclusion

We have described an agent-based electronic market system whose underlying communication and cooperation protocols establish an artificial environment with advanced features. The system is fully implemented in Java (using the "Java Shared Data Toolkit", see http://java.sun.com) and runs on Windows NT. It is based on previous well-tried work concerning intra-agent control [Matsatsinis et al. 1999], inter-agent communication [Matsatsinis et al. 1999], and automation of multiple criteria decision making [Karacapilidis and Papadias 1998a,b]. Agents communicate using TCP/IP, while actors interact with them through web interfaces. All transactions carried out use information encoded in an XML/EDI format. All messages described in the paper were based on the FIPA's (Foundation for Intelligent Physical Agents) Agent Communication Language (see http://www.fipa.org). A rough estimation of the system's total development duration is 8 months, involving two senior researchers and two post-graduate students on a part-time basis. The system has been deployed on the World-Wide Web, while a start-up company is currently testing the system's first fully integrated version.

Our next work direction concerns the integration of a negotiation stage; negotiation of a product feature, for instance, between a purchaser and a seller agent will enable the latter to better tailor its offer proposal and, eventually, work more efficiently for its merchant. Work towards the above will exploit a multiple criteria based negotiation model that has first been presented in [Moraitis and Tsoukias 1996] and will be built taking into account issues emerging in recent related studies (see, for instance, [Lo and Kersten 1999]).

REFERENCES

Chavez, A. and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods", Proceedings of PAAM-96 Conference. London, UK, 1996.

Guttman, R., A. Moukas and P. Maes, "Agent-Mediated Electronic Commerce: A Survey", *Knowledge Engineering Review* 13 (3), 1998.

Karacapilidis, N. and D. Papadias, "A Computational Approach for Argumentative Discourse in Multi-Agent Decision Making Environments", *AI Communications Journal* 11(1), pp. 21-33, 1998a.

Karacapilidis, N. and D. Papadias, "Hermes: Supporting Argumentative Discourse in Multi-Agent Decision Making", Proceedings of AAAI-98 Conference, AAAI/MIT Press, pp. 827-832, 1998b.

Knuth, D.E. *Fundamental Algorithms.* Art of Computer Programming, Vol. 1, Addison-Wesley 1973 (2nd edition).

Lo, G. and G. E. Kersten, "Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies", Proceedings of the 29th Atlantic Schools of Business Conference, Halifax, NS, 1999 (it appears at: `http://interneg.carleton.ca/interneg/research/papers/1999/03.html`).

Mackworth, A., and E. Freuder, "The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems", *Artificial Intelligence* 25, pp. 65-74, 1985.

Matsatsinis, N., P. Moraïtis, V. Psomatakis and N. Spanoudakis, "Intelligent Software Agents for Products Penetration Strategy Selection", Proceedings of MAAMAW-99, Valencia, Spain, June 1999.

Moraïtis, P. and A. Tsoukias, "A Multi-Criteria Approach for Distributed Planning and Conflict Resolution for Multi-Agent Systems", Proceedings of ICMAS-96 Conference, MIT Press, pp. 213-219, 1996.

Nwana, H., J. Rosenschein, T. Sandholm, C. Sierra, P. Maes and R. Guttmann, "Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints", Proceedings of Autonomous Agents 1998 Conference, ACM Press, pp. 189-196, 1998.

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