

December 2000

On the Development of Intelligent Agents for a Web-based Electronic Market System

Nikos Karacapilidis
University of Cyprus

Pavlos Moraitis
University of Paris-Dauphine

Follow this and additional works at: <http://aisel.aisnet.org/pacis2000>

Recommended Citation

Karacapilidis, Nikos and Moraitis, Pavlos, "On the Development of Intelligent Agents for a Web-based Electronic Market System" (2000). *PACIS 2000 Proceedings*. 52.
<http://aisel.aisnet.org/pacis2000/52>

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

On the Development of Intelligent Agents for a Web-based Electronic Market System

Nikos Karacapilidis

Dept. of Computer Science, University of Cyprus
P.O. Box 20537, 1678 Nicosia, Cyprus, nikosk@ucy.ac.cy

Pavlos Moraitis

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

Abstract

This paper describes a web-based electronic market system in which customers and merchants delegate the related tasks to their personal software agents. Messages passed between these agents can fully encapsulate the associated parties' points of view towards a market transaction. More specifically, an offer request consists of a list of the product attributes the customer wants to know about, a partial order of their importance, and the constraints imposed. On the other side, an offer proposal can be tailored according to the information conveyed in the corresponding offer request. Advanced features of the system include the permanent existence of our agents in the market, thus being able to learn from it, their ability to act proactively in order to initiate a transaction, and the integration of an interactive multiple criteria decision making tool, with which a buying agent performs a comparative evaluation of the proposals in a semi-autonomous way.

Keywords: Software agents in e-commerce, e-marketplace, web-based information system, multiple criteria decision making.

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 (see, for instance, (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 fruitful survey, see (Guttman et al., 1998)). There is already a plethora of systems automating the tasks of *product brokering* (e.g., PersonaLogic, <http://www.personalogic.com>), *merchant brokering* (e.g., Kasbah (Chavez and Maes, 1996), Bargain-Finder (<http://bf.cstar.ac.com>), Jango - Excite Product Finder (<http://www.jango.com>)), and *negotiation* (e.g., AuctionBot (<http://auction.eecs.umich.edu>), eBay AuctionWeb (<http://www.ebay.com/aw>), Tête-à-Tête (<http://ecommerce.media.mit.edu/tete-a-tete/index.html>), Fishmarket (<http://www.iiia.csic.es/Projects/fishmarket/newin-dex.html>)).

However, the above works do not deal with important issues, such as: (i) 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.). (ii) 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 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). (iii) The maintenance of each actor’s profile through the personalization of the agents involved; for instance, a customer 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). (iv) The ability of a seller agent to refine some of the customer purchase criteria, argue in favor or against them, or even bring up new information to persuade him/her to accept its offer. (v) 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. The system is implemented in Java and runs on the world-wide web. It should be noted here that our overall framework is not based on pre-classified ads; instead, the system’s agents cooperate and get the related information in a real-time mode. The remainder of the paper is structured as follows: The proposed electronic market framework is analyzed in Section 2. The architecture of the purchaser and seller agents is illustrated in Section 3. The multiple criteria decision making process developed for a purchaser agent is presented in Section 4. Finally, related work is discussed in Section 5, while concluding remarks and future work directions are outlined in Section 6. It should be made clear here that issues such as ordering, security, payment and delivery, while equally important in commercial transactions, do not fall in the scope of this paper.

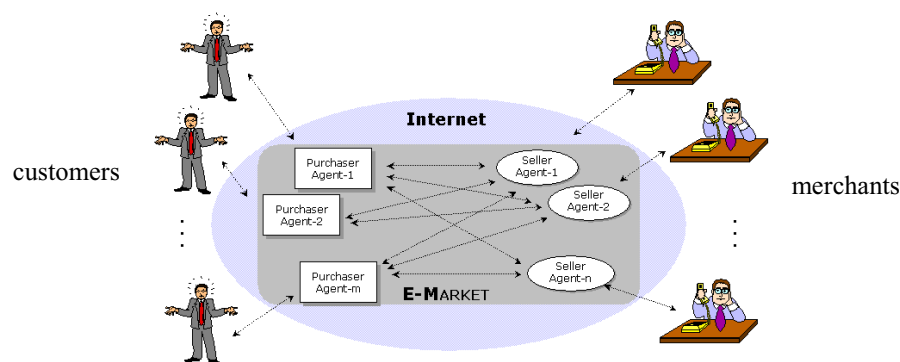


Figure 1. Participants in an E-Market

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 they 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 (Figure 1).

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 is updated and enriched with more specific one each time a transaction is taking place. For instance, the time a customer is about to buy a trip, its purchaser agent will *learn* that he/she 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. This knowledge will be reused when the customer will buy his/her next trip. Similarly, a seller agent is able to update its profile each time it performs a transaction (e.g., by categorizing a customer as a regular one the second time a purchase agreement has been made with him/her, in order to send him special offers in the future, or “keeping a note” that this customer is interested in diving trips, thus refraining from sending him ski resort offers).

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 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 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 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 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 alternative solutions. 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*

To establish the cooperation protocol described above, agents communicate by exchanging messages of various types. Each message type conveys certain semantics associated to a particular task of the purchase procedure. Each time an agent (purchaser or seller) receives a message, it immediately knows what reasoning procedure it must activate in order to deploy 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, namely customers, purchaser agents, seller agents, and merchants.

Motivated by a recent marketing firm report, pointing out that “25 per cent of all new-vehicle buyers use the Internet to arm themselves with vehicle product and pricing information during the vehicle-shopping process” (Power, 1998), we consider (throughout the rest of this paper) a scenario where a customer is interested in purchasing a new car.

2.2.1 *The Offer Request Message (offerReqMsg)*

This message is sent by a purchaser to a seller agent and conveys information formulating the related customer specifications for the product to be purchased. The structure of these messages is illustrated in Figure 2. As shown, the receiver field may be “ny” in case of broadcasting or a list of seller agents' IDs in case of selective communication. The purchaseID field refers to a general *purchase category* (vehicles in our case), while purchaseObj refers to the specific good the customer intends to buy (i.e., a new car). The criterion and feature fields provide information about the specific product attributes. In our framework, *criteria* represent broader evaluation metrics (that is, metrics that may stand for more than one purchase categories; e.g. cost, performance, aesthetics, etc.), while *features* concern specific attributes of a criterion which usually depend on the specific product to be bought (for example, other features are attached to the criterion of safety in the case of purchasing a car than those attached when purchasing a motorboat). Fields featureValue and impact are to be completed by a seller agent (the meaning of the former is straightforward; the purpose of the latter is explained in the following message type). The description field for the scenario used in this paper may contain entries such as (cost, purchase_price, ?featureValue, ?impact), (cost, annual_service_cost, ?featureValue, ?impact), (performance, maximum_speed, ?featureValue, ?impact), (performance, acceleration, ?featureValue, ?impact), etc.

<i>OfferReqMsg</i>	
sender	= <the purchaser agent ID>
receiver	= <list of the seller agents' IDs or 'ny'>
msgID	= <the message ID>
purchaseID	= <the specific purchase ID>
purchaseObj	= <the object to be purchased>
description	= <list of tuples of the form (criterion, feature, ?featureValue, ?impact)>
preferences	= <list of tuples of the form (criterion, impRel, criterion), or (feature, impRel, feature)>
constraints	= <list of tuples of the form (feature, relation, featureValue)>

Figure 2. Structure of an offer request message

Preferences provide a means to weigh pairs of criteria or features. They are represented by tuples of the form (criterion, impRel, criterion) OR (feature, impRel, feature), where $impRel \in \{more(less)_important_than, equally_important_to\}$. A preference of the form (criterion-1, impRel-x, criterion-2) implies preferences of the form (feature-i, impRel-x, feature-j), where feature-i and feature-j have been classified under criterion-1 and criterion-2, respectively. Finally, *constraints* provide a way to express a customer wish about the value a certain feature should possess. For the new-car case, (max_speed, more_than, 200), (airbag, at_least, 2) and (price, less_than, 30000) can be such data types.

2.2.2 The Offer Proposal Message (*offerPropMsg*)

This message concerns an offer proposal sent by a seller agent in reply to a purchaser agent offer request. Offer proposals are structured by seller agents according to a certain strategy, which serves (as much as possible) the interests of their owners. In general, such a strategy underlines the positive features of an object to be sold (even if these features have not been explicitly included in the corresponding offer request), while simultaneously attempts to underestimate, or even ignore, the negative ones.

<i>OfferPropMsg</i>	
sender	= <the seller agent ID>
receiver	= <the purchaser agent ID>
msgID	= <the message ID>
purchaseID	= <the specific purchase ID>
purchaseObj	= <the object proposed by the seller agent >
description	= <list of tuples of the form (criterion, feature, featureValue, impact)>
preferences	= <list of tuples of the form (criterion, impRel, criterion, listOfArg), or (feature, impRel, feature, listOfArg)>
arguments	= <list of tuples of the form (argId, preference, justification, impact)>

Figure 3. Structure of an offer proposal message

An *offer proposal* message (see Figure 3) has a similar structure to that of an *offer request*. Compared to it, the description field now contains the purchaser agent's answer for the feature requested (i.e., featureValue) and its opinion about the *impact* of this answer regarding the product it suggests. The impact field takes a value from the set {pro, con, neutral}. For example, (performance, acceleration, 8.5, pro) denotes a purchaser agent opinion that an acceleration rate of 8.5 is a plus for the product it offers, while (cost, purchase_price, 25000, neutral) denotes its inability to decide whether a price of 25,000 is a plus or a minus for its offer. It should be noted here that the list of tuples in the description and preferences fields of an offer proposal may not necessarily

be identical to those of the corresponding offer request, in that the seller agent may (upon the strategy it follows for a specific setting) tailor its offer (by not filling in all entries included in an offer request, while adding new ones). For instance, a merchant of very expensive cars may have his/her seller agent not providing any information about prices, while trying to promote other features such as safety, image, etc.

Furthermore, an offer proposal may contain arguments about the purchaser agent preferences. Such arguments may either be stored in the related database or come up after the interaction of a seller agent with its human agent. The purpose of the *justification* provided is to further validate or challenge the customers' choices before the comparative evaluation of the offer proposals received.

2.2.3 The Customer-PurchaserAgent Messages

There are five message types used for the communication between a customer and his/her purchaser agent, namely *custSpecMsg*, *custSpecUpdReqMsg*, *custSpecUpdAnsMsg*, *purchInitReqMsg* and *purchInitAnsMsg*. Using the first one, a customer is able to inform his/her assistant agent about the product(s) he/she would like to buy together with a related specification (i.e., criteria, features, preferences, constraints, etc.). The next two concern an update of this specification (at any time of a purchase transaction). They may request and provide, respectively, supplementary information regarding the customers' opinion for a feature value, a preference, or some arguments introduced by a seller agent. Finally, the last two message types concern the interaction that is proactively initiated by a purchaser agent request about whether the customer is interested in purchasing a new product (which has just been launched by a seller agent, see Section 2.2.5). As it will be described in more detail in the next sections, a customer may either accept or reject certain items of an offer proposal; such messages are sent to a purchaser agent during the comparative evaluation of the offerings received and help in determining the best alternative.

2.2.4 The Merchant-SellerAgent Messages

There are also three message types serving the communication between a merchant and his/her seller agent, namely *merSpecMsg*, *merSpecUpdReqMsg* and *merSpecUpdAnsMsg*. Using the first one, a seller agent becomes aware of the products it can sell, by getting a specification of them that reflects the corresponding merchant policy (it may promote the advantages of these products, and so on). In addition, a seller can use a *merSpecUpdReqMsg* message to ask its owner about additional information (e.g., when new criteria or preferences, not currently available in its selling database, appear in an offer request) or his/her opinion on which strategy to be followed in a particular setting. The merchant reaction is represented by a *merSpecUpdAnsMsg*.

2.2.5 A new product announcement (*newProdMsg*)

Each time the database of a seller is updated (either when a new product is introduced or an existing product is updated due to promotion reasons), the seller uses this type of message to broadcast its characteristics to the possibly interested purchaser agents of the E-Market.

2.2.6 A new seller agent announcement (*newSellAgMsg*)

Each time a new seller is created and uploaded, it broadcasts this type of message to announce its presence to the purchaser agents of the E-Market. This message contains information about its coordinates and the products it can sell.

2.2.7 A new offer announcement (*newOfferAnnMsg*)

Each time a merchant launches a new product or an offer promoting one of his/her already existing products, he/she sends this message to his/her seller agent.

3. The E-Market Software Agents

The development of the software agents proposed in our E-Market framework is based on a generic and reusable architecture, which has been conceived after examining the pros and cons of various existing approaches (see, for instance, Matsatsinis et al., 1999; Sycara and Zeng, 1996; Witting, 1992). Even if the two agent types involved do not have the same functionality, they are built on the same basic architecture principles (see Figure 4). However, their constituent modules are tailored, according to their specific type (e.g., the decision making module of a seller agent is usually much simpler than that of a purchaser agent). The architecture of each agent type is described in detail below.

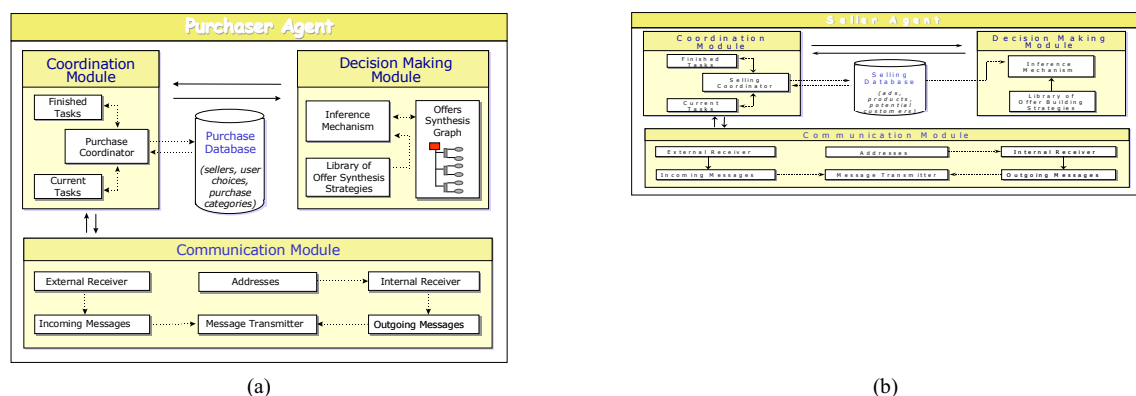


Figure 4. Architecture of (a) a purchaser agent and (b) a seller agent

3.1 The Purchaser agent

A purchaser agent is composed of three, 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 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 or point-to-point communication, it also adds the receiving agent's address. An *external receiver* handles the opposite, 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. The communication module described here is an adaptation of that appearing in (Matsatsinis et al., 1999).

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 (see previous section) 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 4(a), 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, for instance, the *newProdMsg*, and *newSellAgMsg* message types), 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 constraints for products the customer is interested in) and finally, purchase categories. It is a relational database, where all the above elements are structured in tables. 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.

3.1.3 The Decision Making Module

The decision making module is composed of three components (Figure 4a), namely an *inference mechanism*, a *library of offer synthesis strategies*, and the *offers synthesis graph* (see Section 4). 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 *seller* agents), 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, according to the customer 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 decision making process of the *purchaser* agent is described in detail in Section 4.

3.2 The Seller agent

The architecture of a *seller* agent is similar to that of a *purchaser* (see Figure 4b). 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, for instance, the message types *offerReqMsg*, *offerPropMsg*, *merSpecMsg*, *newSellAgMsg* and *newProdMsg* presented in Section 2). The coordination module interacts with the communication and the decision making modules, as in a *purchaser* agent. The *selling database* basically keeps records of the products' specification together with an indication whether the value of a specific feature is a strong, indifferent or weak point. This information is represented by tuples of the form (product, criterion, feature, featureValue, impact), where impact may take a value from the set {pro, con, neutral}. The database may also contain information about potential customers (*purchaser* agents) to be informed about the release of a new product.

begin

1. get *offerReqMsg*
2. search the selling database for *objectType* = *purchaseObj*
3. if not exists then *exit*, else for each *objectType.objToBeSold* found *do*
 - if all *offerReqMsg.constraints* are satisfied
 - 3.1. create a new *offerPropMsg*
 - 3.2. *offerPropMsg.sender* ← *agentID*; *offerPropMsg.receiver* ← *offerReqMsg.sender*;
offerPropMsg.msgID ← *offerPropMsg.msgID* + 1;
offerPropMsg.purchaseID ← *offerReqMsg.purchaseID*;
offerPropMsg.purchaseObj ← *objectType.objToBeSold*
 - 3.3. search in *objectType.objToBeSold* for criteria and associated features appear in *offerReqMsg.description*;
for each match (criterion and associated feature) found *do*

```

        add in offerPropMsg.description a new item (cr, f, fv, imp) where, cr and f correspond to the match
        found, while fv and imp the values of featureValue and impact, respectively, of objToBeSold
    end for
3.4. for each additional pair of criterion and feature (that not exists in offerReqMsg.description)
    if the respective impact is pro (positive)
        add in offerPropMsg.description a new item (cr, f, fv, imp) where, cr and f correspond to the
        additional match, while fv and imp the corresponding values of featureValue and impact,
        respectively, of objToBeSold
    end if
end for
3.5. search in objectType.objSold for offerReqMsg.preferences;
    for each found do
        if there exist arguments referring to it
            add in offerPropMsg.preferences a new item (cr1, impRel, cr2, listOfArg) where, cr1, impRel and cr2
            correspond to the match found, while listOfArg are links referring to this preference of
            objToBeSold;
            add in offerPropMsg.arguments a new item (argID,pref,just,imp) where, argID,
            pref, just and imp the corresponding values of objToBeSold
        end if
    end for
3.6. for each additional preference of objToBeSold (that not exists in offerReqMsg.preferences) do
    if there exist arguments referring to this preference
        add in offerPropMsg.preferences a new item (cr1, impRel, cr2, listOfArg) where, cr1, impRel, cr2 and
        listOfArg the corresponding values of objToBeSold;
        add in offerPropMsg.arguments a new item (argID,pref,just,imp) where, argID,
        pref, just and imp the corresponding values of objToBeSold
    end if
end for
end if
end for
end

```

Figure 5. An offer building strategy

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, that is, actions taken upon the reception of a specific message sent by its merchant. 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).

Concerning the building of an offer, the inference mechanism gets as input an offerReqMsg, consults the *selling database* and the library of strategies to retrieve the appropriate data and algorithms, respectively, and provides a set of offerPropMsg as output. The offer building strategy used for our example scenario is sketched in Figure 5 above.

4. The E-Market Multiple Criteria 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, 1998). For the scenario used in this paper, let the customer have well shaped in his/her mind that the criteria of *performance*, *cost*, and *safety* (let him/her ignoring for the moment that of *firm* image) are critical for the buying decision, while 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 for the car.

4.1 Synthesis of offer proposals

For each specific offer request, the *purchase coordinator* of the purchaser agent receives offer proposals, which are progressively sent by a subset of the E-Market seller agents. Figure 6(a) illustrates a proposal (offer-12: car-6.1), made by sellerAgent-33. This offer consists of criteria (e.g., criterion-22.5: safety, criterion-22.8: cost) together with the associated features, feature values and impact (e.g., (feature-22.5.3: airbag, 2, neutral), (feature-22.8.1: purchase_price, 25000, pro)), 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)) and, finally, 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.

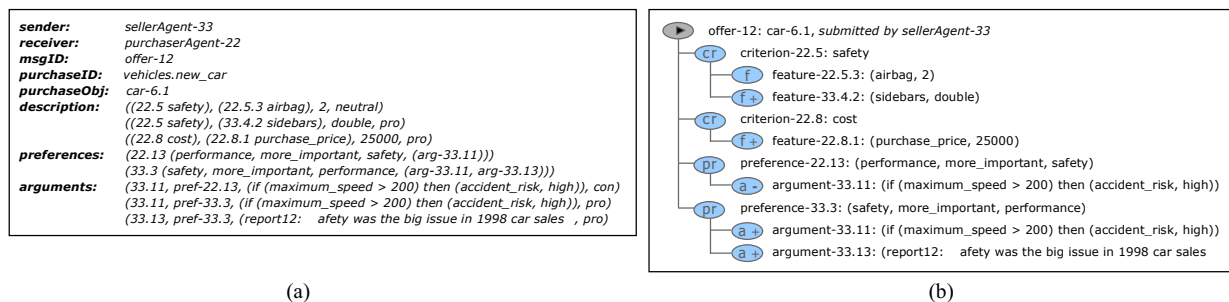


Figure 6. A seller agent offer proposal

Appropriate mechanisms in the decision making module of the purchaser agent transform an offerPropMsg, such that appearing in Figure 6(a), into a graph of the form shown in Figure 6(b). As soon as a purchaser agent gets a new offer proposal, it integrates it with the ones already arrived, constructing an offers synthesis graph. We assume that there is a time limit, set by the purchaser, after which no more offers for the specific purchase transaction are accepted.

4.2 Interacting with the user

The offers synthesis graph is presented to the customer through a web interface (see Figure 7). Note that preferences and constraints have been kept together at its bottom part (that is, after the offer proposals), since these refer to the overall purchase transaction. Also, note that the constraints appearing in this graph come from the related offerReqMsg message (or possibly from updates of the initial specification, initiated by the customer); an offerPropMsg does not contain constraints. Each entry in this graph has an *activation label* indicating its current

status (it can be *active* or *inactive*). By default, all entries are initially active. Viewing the graph through a standard web browser, the customer is able to *inactivate* any of the graph nodes (by using the mouse and clicking on them), the rationale being that their corresponding data type do not suit to his/her interests.

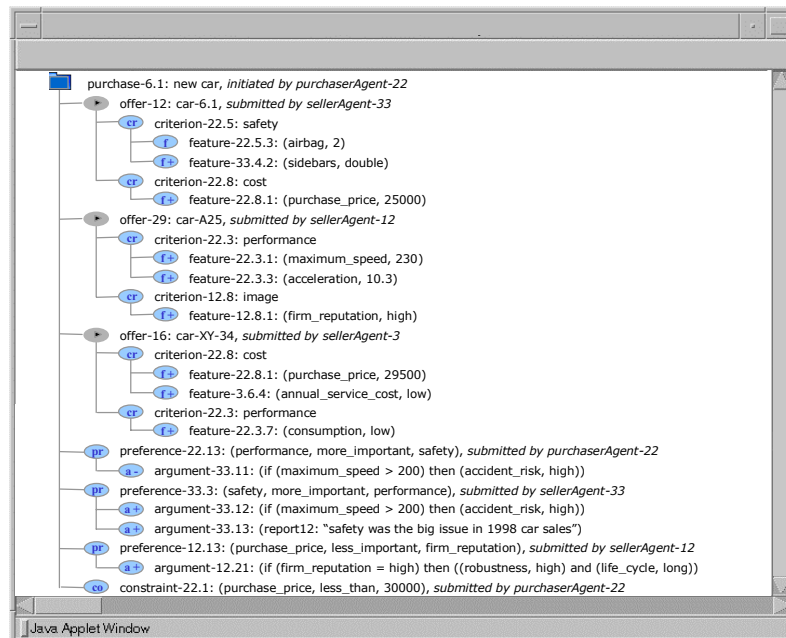


Figure 7. The offers synthesis graph

Figure 8 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)). Upon inactivation, the color of the associated button for each node changes (note the darker gray color) denoting that these nodes will not be taken further into account in the decision making process.

Each offer's feature value is also checked against the existing constraints (note that in the offer building strategy described in Figure 5, the offer proposals sent do not violate these constraints; 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). Recall that a constraint has the form (feature, relation, featureValue), where the desired value may fall into a numerical range, a set of discrete values, or a list of predicates. Nodes that violate a constraint become automatically inactive, that is upon the presentation of the offer synthesis graph to the customer. He/she may then - at any time - activate again (some or all of) these nodes (again, by using the mouse) and test again the outcome of the decision making procedure. In general, the manual activation and inactivation of the graph nodes enables the customer to further elaborate the problem, by examining various alternative combinations.

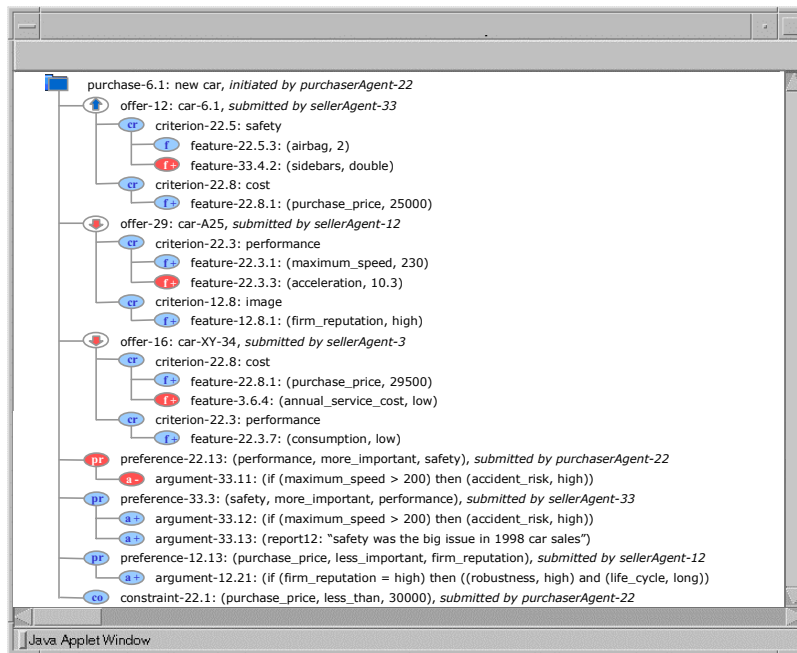


Figure 8. Intervention of customer and purchase suggestion

4.3 Detection of conflicts and inconsistency

Apart from an activation label, each preference has a *consistency label*, which can be *consistent* or *inconsistent*. Each time a preference is inserted in the offer synthesis graph (this follows the receipt of a new offerPropMsg), a mechanism checks if both constituent features or criteria of it exist in another, already inserted, preference. If yes, the new preference is considered either *redundant*, if it also has the same importance relation, or *conflicting*, otherwise. A redundant preference is ignored (not inserted in the graph), while a conflicting one is put next to the previously inserted preference, the rationale being to gather together conflicting preferences and stimulate 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 8, 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 previous active and consistent preferences. Consider, for example, a situation, where there exist two preferences (feature-x, more_important, feature-y) and (feature-y, more_important, feature-z). A new preference (feature-z, more_important, feature-x) is inconsistent with respect to the first two ones, although it is not directly conflicting with either one. 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 (only preference-33.3 and preference-22.13 in the example of Figure 8). In order to demonstrate how the algorithm for assigning weights to an offer features works, we use the example shown in Figure 9. There exist five features and four preferences that relate them as illustrated in Figure 9(a). Each feature has a $\text{weight} = (\text{max_weight} + \text{min_weight})/2$. Max_weight and min_weight are initialized to some predefined values (in the following, we assume that initially $\text{min_weight} = 0$ and $\text{max_weight} = 10$). The arrowed lines correspond to the “more important” relation (e.g., $(f_1, \text{more_important}, f_2)$) and the dotted line to the “equally important” relation (e.g., $(f_3, \text{equally_important}, f_4)$). 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 9(b)). The weight is the average of the new max_weight and min_weight: $\text{weight}(f_1) = 6$, $\text{weight}(f_2) = 4.5$, $\text{weight}(f_3) = 5$, $\text{weight}(f_4) = 5$ and $\text{weight}(f_5) = 4$.

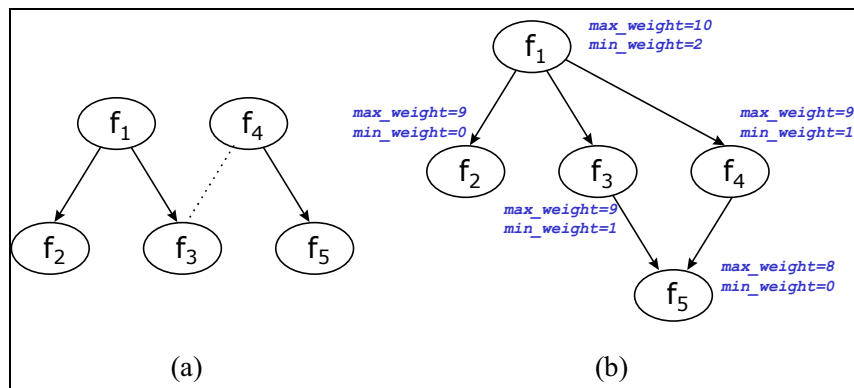


Figure 9. The weighting mechanism

The basic idea behind the above scheme 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 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 (we assume that an offer which includes no product features gets $\text{score} = 0$), according to the formula:

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

where feature_j , (feature_k) an active feature which refers to offer_i having a positive (negative) impact. The scores of offer-12, offer-29 and offer-16 in Figure 8 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 $\text{score}(\text{feature-22.3.1}) = \text{score}(\text{feature-12.8.1}) = \text{score}(\text{feature-22.3.7}) = 4.5$, while feature-22.3.3 and feature-3.6.4 are inactive). Therefore, offer-12 is the one recommended by the system (as shown in Figure 8, 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 (activation/inactivation) of the offers’ features by the customer, or the receipt of a new offer (in case that the time limit given has not been exceeded).

5. Related Work

The 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 "hop-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* operates in a similar way. Using *PersonaLogic*, customers may impose constraints, to be then exploited by a constraint satisfaction engine in order to prune alternatives that do not satisfy them. Even being 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 follows the above approach. Finally, *Kasbah* 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, the agents involved in our approach (both purchaser and seller agents) are more sophisticated. This comes out with the following facts: First, they have all the necessary characteristics enabling them 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. In such a way, our framework allows the agents "live" in an E-Market, learn from it and proactively ask its owner's opinion about the initiation of a transaction, whenever a new or promoted product is launched. Second, our purchaser agent is enabled with an interactive multiple criteria decision making tool, allowing the handling of incomplete, inconsistent and conflicting information, as well as the progressive synthesis and comparative evaluation of the offer proposals. Furthermore, we believe that the overall framework presented in this paper is "loser" to a real business model, in that its communication and cooperation protocols allow the implementation of an artificial environment that better represents a real market.

6. Conclusions

This paper has presented a web-based system for agent-mediated cooperation in an electronic market. The main contribution concerns the personalization of the agents involved, the progressive synthesis and multiple criteria comparative evaluation of the alternative proposals, and the specification of (communication and cooperation) protocols attempting to bring real and artificial markets closer. Future work directions concern the integration of a negotiation stage; negotiation of a product feature, for instance, between a purchaser and a seller agent will enable the latter better tailoring its offer proposal and, eventually, work more

efficiently for its merchant. Work towards the above will exploit a multicriteria-based negotiation model that has first been presented in (Moraitis and Tsoukias, 1996).

References

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

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

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

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

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

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

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

Nwana, H., Rosenschein, J., Sandholm, T., Sierra, C., Maes, P., and Guttmann, R. "Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints," In *Proceedings of Autonomous Agents 98*, ACM Press, 1998, pp. 189-196.

Power, J. D. and Associates. "One Out of Four New Car Buyers Shop the Internet," *Internal report*, 1998. Available at: <http://www.jdpower.com/releases/80914car.html>

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

Witting, T. (Ed.). *ARCHON: An Architecture for Multi-Agent Systems*, Ellis Horwood, 1992.