Toward An open intelligent market place for mobile agents

L. Esmahi, P. Dini
Centre de Recherche Informatique de Montreal (CRIM)
lesmahi@crim.ca, dini@crim.ca

J.C. Bernard
Department of electrical and computer Engineering,
Ecole Polytechnique de Montreal
Bernard@vlsi.polymtl.ca

Abstract: This paper focuses on the issues involved when multiple mobile agents interact in multiagent systems. The application is an intelligent agent market place, where buyer and seller agents cooperate and compete to process sales transactions for their owners. The market place manager acts as a facilitator by giving necessary information to agents and managing communication between agents, and also as a mediator by proposing solutions to agents or stopping them to get into infinite loops bargaining back and forth. The buyer and seller agents range from using hardcoded logic to rule-based inferencing in their negotiation strategies. However these agents must support some communication skills using KQML or FIPA-ACL.
So in contrast with other approaches to multiagent negotiation, we introduce an explicit mediator (market place manager) into the negotiation, and we propose a negotiation strategy based on dependence theory [1] implemented by our best buyers and best sellers.

Keywords: Intelligent agents, mobile agents, virtual market place, E-commerce, negotiation

1. Introduction

In recent years, many researchers in intelligent agents domain have focused on the design of market architectures for electronic commerce (E-commerce) [2][3][4], and on protocols governing the interaction of self-interested agents [5] engaged in such transactions. While providing support for direct agent negotiation, the existing architectures for multiagent virtual markets usually lack explicit facilities for handling negotiation protocols [6], since they don't provide such protocols as an integrated part of the framework.
Our goal in this research is to design and implement a generalized multiagent market architecture that can provide explicit and integrated support for complex agent interactions, such as contract-net [7], persuasive negotiation [8], as well as other types of negotiation protocols, including auction [9], open-bid or advertised-price buying and selling. Several requirements have been identified for market architecture such as:

- Providing support for a variety of transaction types including simple buying and selling, auctions, and complex multiagent contract negotiation
- Providing language in which the rich array of semantic content about commerce can be expressed.
- Being extensible, by third parties, so providing multiagent contract and dynamic mediation.
- Providing a secure and private credit and payment mechanisms
- Controlling fraud and misrepresentation
- Discouraging counterspeculation
- Interoperating with other new and existing E-commerce service.

In this paper we will focus on the three first requirements that imply interaction and negotiation concepts.

This paper is organized as follows: in section 2, we describe the market place infrastructure. In section 3, we define the manager agent design, its implementation, and its functions. In section 4, we describe the process of mediation and negotiation used by the manager agent. In section 5, we briefly describe some skills used by the buyer and seller agents. In section 6, we define the strategy used by our best buyer and best seller. Finally we conclude with some of our future works.

2. Market place architecture

The market place provides an infrastructure for exchanging offers and requests. Potential business partners can exhibit their services and available resources, search for offers from service providers, or send their own offers. Hence three type of agents acts in the market place: the market place manager, the buyer agents, and the seller agents. In this paper, we use the terms buyer and seller to distinguish the agent's roles. However, a customer agent can either be a buyer and a seller in the same time.

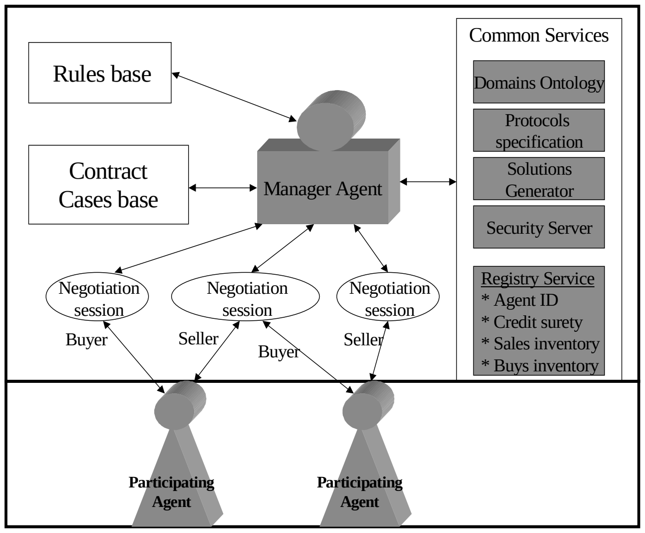

Figure 1: Market place structure

The market place manager acts as mediator between the buyer and seller. All agents must register with the manager before they can have any interaction with other agents in the market place, and all negotiation sessions goes indirectly through the manager. The market place includes a set of domain-specific services and facilities as shown in figure1 that are composed of:

- A registry of market clients who have expressed interest in doing business in the market. Entries in this registry would include the identity of a client, an inventory (or a method for accessing an inventory) of products or services that are offered or requested by that agent. If the agent is a buyer, it must give a credit surety that set his upper limit for payment.
- Protocols specifications that formalize the types of negotiation protocols supported within the market place.
- An ontology[1] specification for each marketing domain supported in the place. Each community of agents adheres to a domain of activity and uses this ontology for their communication and negotiation.
- A solution generator used to construct initial propositions for agents that request this service, and to generate mediated solution for conflicting agents.
- A security service for protection against fraud and misrepresentation.

All of the agents are running their own threads that awaken at specified intervals. The first negotiation session takes place when one of the buyer agents wakes up, takes an item from its shopping list and asks the manager to recommend a seller agent who has advertised its ability to sell that item. If more than one seller has advertised an item, the manager uses his rules base to select the best agent that fits to the request. Some criteria of selection are offers similarities, difference between prices, and agent dependencies. When the manager returns the name of this seller to the buyer, the communication starts between the two agents and they try to agree on a price and reach a contract. All of the communication between buyers, sellers, and the manager use the event-listener mechanisms. The argument object that is passed with the event is a message object modeled after a KQML [11] or FIPA-ACL [12] message packet.
The state transition diagram used for the negotiation cycle in the market place is described as follows:

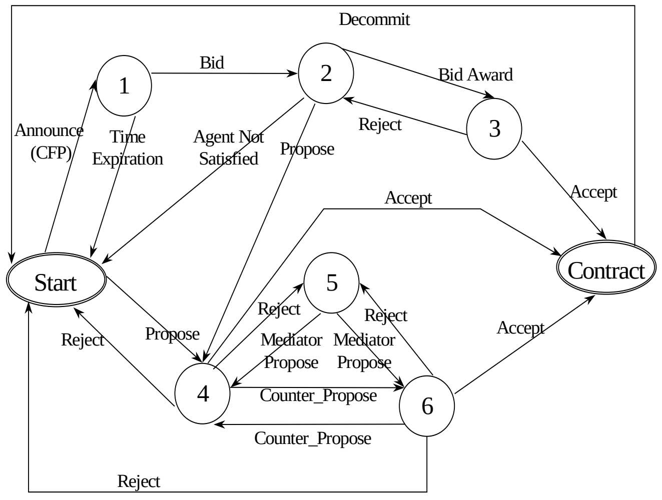

Figure 2: negotiation cycle

3. Market place manager

The manager agent acts as a supervisor of the market place, manages its components and acts as a facilitator or mediator for the negotiating agents. To manage the registry service, the manager handles two hashtables, one that is a registry for all agents in the market place, and one for the communities or domains of interest in the market place. For example, there may be a set of seller/buyers interested in marketing basic connectivity resources, and another set which markets language translation expertise. Seller agents are added to these communities by advertising their willingness or ability to sell a product in that domain.
Negotiation could process between the seller and the buyer, resulting in a sale. The seller must always retract its advertisements when there was no item to sell.
The manager uses the singleton design pattern that insures that only a single instance of it exists at one time. Hence, it runs in the main application thread by synchronously handling agent's events and his process method is used to start its thread. Once there is no event to process; it sleeps for 10 seconds and then wakes up to see if anything exciting is happening.
An alternative design for long-running transaction would be to spin a new thread whenever an event needed to be processed. We would consider this alternative in later version of this application. The manager agent architecture is as follow:

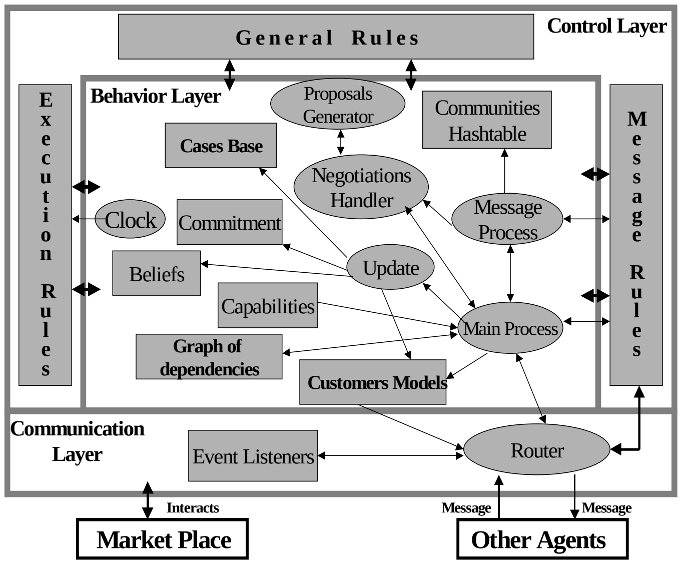

Figure 3: Market place manager architecture

Communication layer: the communication between agents is accomplished by using event handling. Each event object contains two attributes: the source agent that fired the event, and the argument object that is an agent message object, roughly equivalent to a KQML/FIPA-ACL message packet. Once the agent message is extracted from the event, the route method of the router process is called to process it. The route method takes an agent's message as a parameter. The method gets a reference to the sending agent from the hashtable of registered agents.
The manager handles three types of performatives in this method, "Advertise", "Unadvertise", "Recommend-one", in addition to its message routing function.
The "Advertise" and "Unadvertise" performatives either add or remove sending agent from a community. The "Recommend-one" performative first checks to see if there are any agents in the community of interest. If so, the manager determines how many agents have advertised, and then select a seller to recommend to the buyer by using offers similarities and interdependencies between the seller and the buyer (cf. 4.).
The case where a buyer asks for a seller, and none have advertised, results in an undefined state because the buyer is expecting a "tell" response from the manager that never comes. In the present architecture, this situation is under the control of a rule that fires to

[1] In this context we use the term ontology to mean a specification of a conceptualization related to a domain [10].

send a “deny” message back to the buyer. However the buyer agent state logic would have to be updated to deal with this correctly.
The messages exchanged between agents are basically a collection of data that corresponds to the major slots of KQML/FIPA-ACL message. Each message must specify a performative, a content string, the sender name, and the receive name. The “reply with” parameter tells the receiver to use that string in the “ in reply to” slot in the reply message. All of the parameters are publicly visible, although they could have been shielded by making them private or protected and providing accessor methods.

Behavior layer: this layer implements the reaction behavior of the agents, by calling methods at specified times or when messages are received. This layer includes a real time clock to control the execution of scheduled actions (communication actions, and local actions), and the methods to update the mental state composed of: beliefs, decisions, commitments, and capabilities.
The belief component contains the model of the world. Each belief consists of a temporal fact that the manager believes as being true or false at a given time. Commitments refer to a schedule of actions, which will be executed by the agent. The capabilities component represents the set of methods corresponding to the facilities that the manager can give to the customers of the market place (i.e. Giving pertinent indices of the market, constructing an initial proposition, recommending a partner based on some criteria, or mediating a conflict). The customers models represent knowledge about each agent customers of the market place (Agent name, buys inventory, sales inventory, domain of activity). The default behavior is defined by the type of the message received by the manager. Hence, “advertise” messages update the customers models, and if the message type is “tell”, the agent will try to incorporate the message content to its beliefs. If the message is a request for information (“stream-about”, “ask-if”, etc.) the manager will try to unify the message content with its beliefs and take the decision to reply the result to the sender agent. Finally, if the message is an "achieve" and the action/service specified in the message content is within the capabilities of the agent, it will schedule the action as a commitment and execute it accordingly.
The cases base component usually used for generating initial solutions or for mediating is composed of a set of past contracts or unsuccessful negotiation cases. Each case is represented by a set of impasses; each impasse represents the proposals of the agent, the feedback of other (acceptance or reject), the cause of the reject if it exists. Cases of success are indexed according to relevant characteristics of the domain, cases of fail possess three others supplementary index: the type of failure, the cause, and the item that has entailed the failure. Cases are organized in hierarchies around important concepts of the domain, and their research is made according to conceptual similarities. The highest level of the hierarchy is called generalized episode [13], and each node is constituted of an individual case or a generalized episode.

The graph of dependencies describes all possible transactions between the agents, which have advertised themselves as buyers or sellers. A such graph is represented as a directed graph where each node represents an agents, and arcs linking nodes represents the dependencies between agents in term of potential transactions on items. With each edge is associated the item object of the potential transaction and the price requested by the seller agent. The arcs are oriented and start from the buyer to the seller. The following figures describe a situation of the market place and its corresponding graph of dependencies.

Items	**Agent A1**		**Agent A2**		**Agent A3**		**Agent A4**	
	Selling Price	**Purchase Price**	**Selling Price**	**Purchase Price**	**Selling Price**	**Purchase Price**	**Selling Price**	**Purchase Price**
I1		X	P1					X
I2		X	P2					
I3	P3			X	P6			
I4		X					P4	
I5		X			P5			
I6				X			P7	
I7	P8					X		
I8					P9			X
I9					P10			X
I8+I9					P11			X

Table1: a market place situation

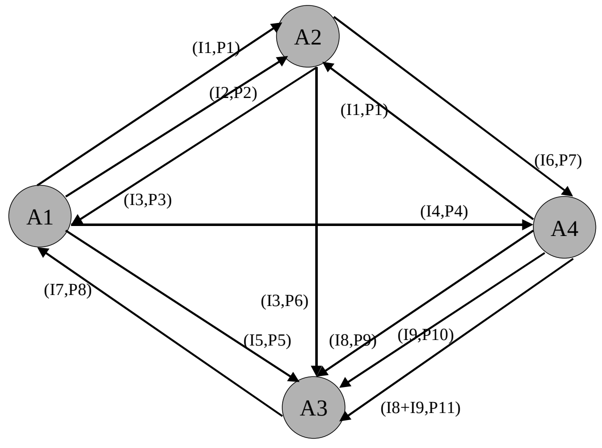

Figure 4: Graph of dependencies.

Control layer: this layer implements the deliberative functionality of the manager in which his behavior is controlled by rules. The rules encoded in this layer are of three sets: (1) message rules handle incoming messages; (2) execution rules control the fulfilling of actions; and (3) the general rule set is used to make general deductions based on the mental state of the manager.
Messages coming from the router are matched to the rules in the message rule set. If a match succeeds, a rule will be selected and executed, otherwise the default message-processing behavior will occur. When a scheduled action is up for execution, the execution rule set will be invoked, and the actions associated with matching execution rules will be performed. The action will fail when no matching execution rule is found. Finally, when there are no pending messages to process or actions to execute, the manager will focus on applying its general rules. It will fire rules from the general rule set until one of the following conditions is met: (1) a new message arrives in the router; (2) an action needs to be executed; and/or (3) there are no more general rules that can be fired. Finally, if there's any general reasoning rule to fire, no message, and no commitment to execute at the present time, the agent sleeps for 10 seconds and then wakes up.

4. Market place mediation process

The manager plays a facilitator role and a mediation role in the market place. In the previous section we have discussed some of the services provided in the facilitator role (registration, communication, responding to the customer’s request). Hence this section details the generation of initial solution for agents, the selection and recommendation of buyers partners, and the mediation process for resolving conflicts and persuading agents to accept the solution proposed.

4.1. Generating initial solution.

For constructing an initial solution, the proposal generator use the attributes specified in the buyer's message, the cases base, and the market place indices referring to the product or service in the buyer's request. First the generator retrieves similar cases from the cases base using similarity metrics[2], sort them according to their closeness to the current request, and select the best case.

The initial solution is constructed by adjusting the case to the scale of the request; and is evaluated in order to avoid potential reject.

The general algorithm of solution generation is as follows:

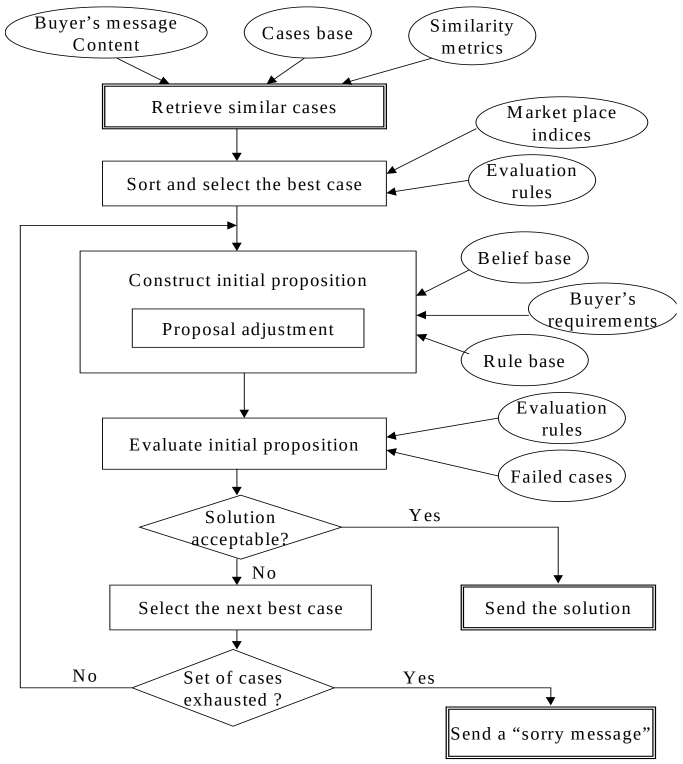

Figure 5: Solution generation using case based reasoning

4.2. Conflict mediation :

Conflicts in the market place arise for several reasons (conflict on price, conflict on product /service characteristics, etc). In this paragraph we limit our self to conflict on price between buyer and seller. However other conflicts in the market place are subject of our future works.

The conflict may be either signaled by à customer or detected by the manager when negotiating agents get into infinite loop bargaining back and forth.

When the manager has a price conflict to resolve three cases can arise: first, the manager has an other seller to recommend to the buyer. Second, there's no other seller to recommend but the manager has detected a dependence on some items between the buyer and the seller, so the manager can construct a solution by grouping the items causing the dependence. Finally, the first and the second conditions are not satisfied, Hence the agent use case based reasoning to generate a solution (cf. 4.1). The general algorithm for resolving conflict is as follow.

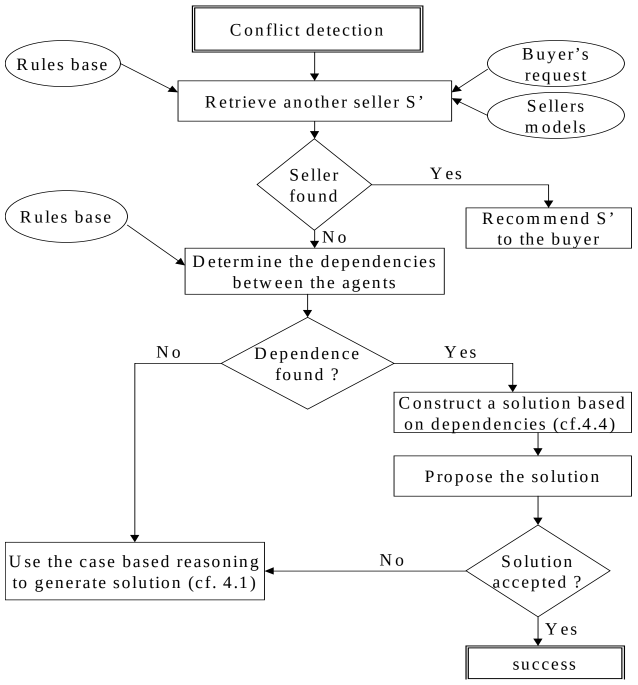
Conflict detection
Rules base
Buyer's request
Retrieve another seller S'
Sellers models
Seller found
Yes
Recommend S' to the buyer
Rules base
No
Determine the dependencies between the agents
No
Dependence found ?
Yes
Construct a solution based on dependencies (cf.4.4)
Propose the solution
Use the case based reasoning to generate solution (cf. 4.1)
No
Solution accepted ?
Yes
success

Figure 6: Mediation process

4.3. Types of dependencies and contracts

Before describing the generation process we have to coin same definitions that we will use.

Strong dependence: a strong dependence occurs between a buyer agent A1 and a seller agent A2 on an item I (S-Dep (A1,A2,I)). If agent A1 has requested to buy I, and agent A2 is the unique agent in the market place that has advertised himself to sell item I.

Weak Dependence: a weak dependence occurs between a buyer agent A1 and a seller agent A2 on an item I (W-Dep (A1,A2,I)). If agent A1 has requested to buy I, and the agent A2 is one of the group of agents in the market place that have advertised themselves to sell item I.

Multi-dependence: a buyer agent A1 is said to be in multi-dependence towards a seller agent A2 on the items I1, I2 (X-Dep (A1,A2,I1,I2)). If one of the following situations occur:

- S-Dep (A1,A2,I1) and S-Dep (A1,A2,I2)
- S-Dep (A1,A2,I1) and W-Dep (A1,A2,I2)
- W-Dep (A1,A2,I1) and W-Dep (A1,A2,I2)

Mutual dependence: two agents are said to be in mutual dependence (M-Dep (A1,A2,I1,I2)). If one of the following situations occur:

- S-Dep (A1,A2,I1) and S-Dep (A2,A1,I2)
- S-Dep (A1,A2,I1) and W-Dep (A2,A1,I2)
- W-Dep (A1,A2,I1) and S-Dep (A2,A1,I2)
- W-Dep (A1,A2,I1) and W-Dep (A2,A1,I2)

Swap contract: sometimes, it is interesting to swap items between agents to enhance the global solution, especially when the two agents have mutual dependence. In swap contract, the first agent subcontracts a product/service to the second agent, while the second agent in its turn subcontracts a product/service to the first agent. The following example shows the beneficial of such a contract.

Items	**Agent A1**		**Agent A2**	
	Selling Price	**Purchase Price**	**Selling Price**	**Purchase Price**
I1	>=4			<=3
I2		<=3	>=4	

[2] The similarity degree depends on several criteria: the appurtenance to the same level of the cases hierarchy, the number of attributes shared by the cases, the scale difference, and the unit prices difference.

In this situation of the market place, no simple contract can be concluded between the two agents. However, with a swap contract all the agents satisfy their goals.

The swap contract becomes more interesting with agents that have mutual dependencies.

A general form of the swap contract is where more than two items are swapped between agents.

<u>Clustering contract:</u> this type of contract is constructed by grouping a set of items in the same contract, in a manner that the global cost of the contract becomes beneficial for the two agents, even if the individual cost of some items is greater than the accepted price. The following example shows the beneficial of such a contract.

Items	**Agent A1**		**Agent A2**	
	Selling Price	**Purchase Price**	**Selling Price**	**Purchase Price**
I1	>=4			<=3
I2	>=4			<=6

In this situation of the market place, only one simple contract can take place (the one implying item I2). However, with a clustering contract regrouping I1 and I2, all the two agents can satisfy their goals.

The clustering contract becomes more interesting with agents that have multi-dependence relation. The interest to this type of contract grow if we remember that in real trading, the price of a group of items is always lower than the sum of their individual prices.

<u>Multiagent contract</u>: in this type of contract, a group of agents take a commitment to buy/sell a set of items. The contract is constructed in such a manner that is beneficial for all participating agents. A multiagent contract may include swapping and clustering of items.

As an example of situation where a multiagent contract is of great benefice consider the scenario bellow in the market place.

Items	**Agent A1**		**Agent A2**		**Agent A3**	
	Selling Price	**Purchase Price**	**Selling Price**	**Purchase Price**	**Selling Price**	**Purchase Price**
I1	>=4			<=3		
I2			>=3			<=2
I3				<=5	>=4	
I4		<=4	>=5			
I5			>=2			<=2
I2+I4			>=6			

The multiagent contract which contains all the three agents items, is the alone contract that satisfy all agents goals without paying any additional cost. This contract take benefice from the price of (I2+I4) which is low than (price(I2) + price(I4)).

4.4. generating solutions based on agents dependencies

In order to achieve the mediation and generate solution for conflicts, the manager use the classes of dependencies and contracts defined here above.

When mediation is requested, the manager search first for mutual dependencies between the two agents, in order to construct a solution using swap contract. If no mutual dependencies were found, the manager searches then for multi-dependence in order to construct a clustering contract. In the case where the manager doesn't find those types of dependencies, it uses the graph of dependence for constructing Multiagent contract.

The general algorithm for mediation is as follow:

```
1- construct and propose a swap contract using mutual dependencies.
   1.2- search the graph for all mutual dependencies of the buyer towards the seller
   1.3- sort the dependencies based on their strength
   1.4- construct a swap contract using the mutual dependence
   1.5- propose the swap contract
   1.6- if the contract is accepted then success, stop
   1.7- else select the next mutual dependence in the list and go to step 1.4
   1.8- if the set of mutual dependencies is exhausted without success then go to step 2
2- construct a clustering contract using multi-dependencies.
   2.1- search the graph for all multi-dependencies of the seller towards the buyer
   2.2- sort the dependencies based on their strength
   2.3- select the first dependence in the list
   2.4- construct a clustering contract based on the multi-dependence relation
   2.5- propose the clustering contract
   2.6- if the contract is accepted then success, stop
   2.7- else select the next multi-dependence in the list and go to step 2.4
   2.8- if the set of multi-dependencies is exhausted without success then go to step 3
3- construct a Multiagent contract using other agents dependencies
   3.1- search the graph for a balanced cycle of dependencies that close the path[3] between the buyer and the seller
   3.2- construct a Multiagent contract using the cycle of dependencies
   3.3- propose the Multiagent contract to the group of agents
   3.4- if all agents agree on the contract then success, stop
   3.5- else go to step 3.1
   3.6- if there is no cycle of dependencies between the buyer and the seller then go to step 4
4- use the case based reasoning algorithm to generate a solution (cf. 4.1.).
```

5. Basic functionality for the buyer and seller

5.1. Buyer Agent

In this section we describe the base functionality that must be supported by all of the buyers. However those Agents may be constructed by different developers using different tools and different architectures.

The major buyer data includes the « wish list », a vector of items the agent wants to buy and desired purchase prices; the inventory, which contains all of the items the agent has purchased; and the negotiations, a hashtable of negotiations in progress. The agent's main process method is used to register the buyer agent with the market place manager, initialize the wish list with instances of basic negotiation objects, and start the agent's thread running. A negotiation takes place, if the agent still has items on its wish list and no negotiation is in progress. Hence it takes the first item of the wish list and kicks off a negotiation by asking the manager to recommend a seller for the item. It does this by first instantiating a message object "Recommend-one" and setting the message slots appropriately, and then using it as the argument for a new Event object. When the buyer agent registered with the manager, this latter added itself to the buyer event listeners list. So, calling the notify event listeners method results in the manager receiving a copy of the message. The manager will reply to the buyer by sending a message. All such messages are received through the event-fired method of the agents.

There are several conditions that the buyer must detect and handle. The first is when the manager responds to the recommend request. This message is a "tell" performative and the content is the name of a seller agent that has advertised the desired item. In this case, the seller is asked (through the manager) how much it wants for the item. Otherwise the message must contain an "offer" or "counter-offer" from a seller. In this case an instance of an offer is created. An offer contains a triplet of item name, item identifier, and offer price, along with the name of the seller making the offer.

The offer can contain one of three performatives: "make-offer", "accept-offer", or "reject-offer". If the message is an "accept-offer", the sale is acknowledged by sending a "tell" message back to the seller, the item is put in the inventory of purchases, the negotiation is removed from the active list, the money spent is added to the tab and

[3] A path X to Y in the graph of dependence constitutes a causal chain that provides an amount of transaction that must be balanced in the other direction from Y to X, in order to determine a cycle.

the “tell” message is sent. The pending negotiation is also set to null, so that the next time the buyer agent wakes up, it can start a new negotiation if there are still items on its wish list. This behavior is an arbitrary design. We could just as easily have decided to immediately pull another negotiation off the wish list, but we chose this design to allow the transactions to be more spread out in time, and to give some homogeneity to the agents design.

Another alternative is that the seller rejects the last offer. In our design, this ends the negotiation unless the buyer requests the manager for mediation. The buyer agent cleans up the negotiation from the active list and places it back on the wish list, so it can try again by asking the market place manager to recommend another seller for the item.

The third alternative is that the seller makes an offer and the buyer agent needs to negotiate using the negotiate method. The sophistication of the processing logic in this method is the primary point of differentiation between our basic buyer agent and the best buyer agent classes.

a- Basic buyer agent negotiation strategy

The basic buyer agent checks if the seller’s offer price is lower than the strike price, which is the desired maximum price. If it is, it accepts the offer by echoing the message back to the seller, assuming that the seller agent will accept the offer it just made. If the offer price is higher than the buyer is willing to pay, the buyer agent must make a counter offer (remember, in this market place design, only the seller can reject an offer). The basic agent simply set the price to the maximum that the buyer is willing to pay and makes an offer in the hopes that the seller will accept.

b- Enhancement toward the best buyer agent

In this section, we describe improvements to our basic buyer agents deriving into the best buyer agents. These enhanced agents have a more sophisticated negotiate method, and utilize additional information about the negotiation process. They have a rule base that they use to determine their prices and actions during the negotiation, and take profits from the manager knowledge.

In his “recommend-one” message addressed to the manager, the best buyer also ask the manager for the current value or indices of the item in the market place, and use it to calculate its strike price and its utility variable.

The utility variable is the difference between the maximal accepted price and the asking or strike price.

The negotiation strategy used by the buyer is based on conflict risk evaluation. So depending on the willingness of the buyer to risk a conflict, increasing amounts are deduced from the seller’s offer and a counter-offer is made. The detail of the best buyer’s strategy is given in section 6.

5.2. Seller agent

In this section we describe the functionality that must be supported by all of the sellers, and some orientation in their construction. However, no constraint was set on the architecture or implementation tools.

The major seller’s data includes the inventory, a hashtable of items the agent wants to sell and desired sales prices, and the negotiations, a hashtable of negotiations in progress. The main process method of the agent must be able to register the seller agent with the market place manager, initialize the inventory with instances of basic negotiation objects, advertise the items to the manager, and to start the agent’s thread running.

After the seller agent advertises its wares to the manager, it must wait for a prospective buyer agent to contact the manager looking for a recommendation of a likely seller. When the manager passes the seller’s name to the buyer agent, the buyer sends an “ask” message to start the negotiations. Like the buyer agent described earlier, the seller receive any message as an argument from an event through his event-fired method. The agent processes the message and determines the appropriate response.

In the process of the message, the seller must take care of two basic cases. First is to respond to an initial “ask” from a buyer, and the other is to respond to an “offer”. In the first case, there is a lot of housekeeping to do when a buyer asks about an item. It must firstly check to see if it still has any of the desired items in inventory, and if so it then generates a unique item identifier. This item identifier is used as the hashtable key for both the buyer and seller negotiation objects. The item is removed from the inventory list, and the last offer field is initialized. The seller agent then places the negotiation on the active negotiations list, and sent a “make-offer” message back to the buyer agent through the manager.

In the second case, there is a negotiation in progress. The seller agent uses the item Id as the hashtable key retrieving the negotiation from the active negotiations list and instantiating an offer. If there is no basic negotiation object with that item Id, it must have just been sold, so the seller agent sends a “reject-offer” message to the buyer. Otherwise, it updates the basic negotiation with the current offer. Next, it determines whether the buyer is making a “counter-offer” or simply acknowledges a previous “accept-offer” message. If it is a “tell”, it must be the latter case, an item was just sold. The basic negotiation is removed from the active negotiation list, and the purchase price is added to the sales total. If the performative is “make-offer”, the “negotiate” method is called to come up with an appropriate response.

a- Basic seller agent negotiation strategy

In the basic seller agent strategy, if the buyer’s price is above the desired minimum selling price, the offer is immediately accepted and a message is sent back to the buyer agent. The buyer agent must acknowledge the offer with a “tell” message before the item can be removed from inventory. If the buyer agent is offering less than the seller is willing to take for the item, the offer is rejected and the negotiation is closed by placing the item back in inventory.

b- Enhancement toward the best seller agent

Much of the best seller improvements are like the changes to best buyer. So it use a rule base to determine it’s prices and actions during the negotiation. It also use the manager’s knowledge for constructing its first offer.

As described earlier, the seller agent control the negotiation, and can decide to accept an offer, make a counter-offer, or reject the offer and break off negotiations with the buyer agent. Hence, if the offer is not acceptable, the seller has to calculate his willingness to risk a conflict and approximate his partner’s risk. If his risk is smaller or equal to that of his partner, it construct an offer by conceding and send him a counter-offer message. Otherwise a reject message is sent to the buyer and the negotiation is stopped. The strategy used by the best seller’s agent is described in the next section.

6. Best buyer/seller negotiation strategy

Given that our market place adhere to the offer and demand law, and the customer agents are supposed individually rational[4], it is clear that if no one of the two agents give his partner an acceptable offer the negotiation runs into conflict. We have already specified that if both agents stand still during the negotiation, the manager takes the control for mediation. This can happen even when the space of possible deals include profitable deals for both agents.

One way of thinking about which agent should concede at each step is to consider how much each has lose by running into conflict at that point. the core of our strategy can be expressed intuitively by: “The more concession you’ve already made, the less likely that you will make the next concession”. Hence, we use the risk evaluation criteria proposed by Zeuthen [14] to determine who should concede in the next step. Precisely, we have adapted the Zeuthen strategy [6] in a manner that we can deal with the lack of knowledge that each agent

[4] An agent is individually rational if it accepts only deals that give him non-negative utility.

has about his partner's utility[5]. In this adaptation, each agent use the current price of the item in the market place as the minimum offer accepted by his partner.
In fact, if after step t the agent decides not to take a concession, he takes a risk that his partner will also not make a concession, and that they will run into a conflict.
So let's consider a buyer agent A_i and his partner A_j (seller), and consider the following variables :
γ_i : maximal cost acceptable for A_i
γ_j : minimal price acceptable for A_j
γ_m : current price in the market place given by the manager
For each step t, let $\delta_i(t)$, $\delta_j(t)$ be the deal offers made respectively by A_i and A_j.

- **In the buyer's point of view, the utility and risk is defined by:**
  - The utility that agent A_i believes it will have by offering $\delta_i(t)$ is:
    $U_i^i(\delta_i(t)) = \max(\gamma_i - \delta_i(t), 0)$
  - The utility that agent A_i believes it will have by accepting A_j's offer $\delta_j(t)$ is:
    $U_i^i(\delta_j(t)) = \max(\gamma_i - \delta_j(t), 0)$
  - The utility that agent A_i believes that agent A_j will have by offering $\delta_j(t)$ is approximated by:
    $U_i^j(\delta_j(t)) = \max(\delta_j(t) - \gamma_m, 0)$
  - The utility that agent A_i believes agent A_j will have by accepting A_i's offer $\delta_i(t)$ is approximated by:
    $U_i^j(\delta_i(t)) = \max(\delta_i(t) - \gamma_m, 0)$

Hence, we can now define the degree of willingness to risk a conflict as follow:

  - The agent A_i's belief of willingness to risk a conflict is:

$$Risk_i^i(t) = \begin{cases} 1 & \text{if } U_i^i(\delta_i(t)) = 0 \\ (U_i^i(\delta_i(t)) - U_i^i(\delta_j(t))) / U_i^i(\delta_i(t)) & \end{cases}$$

  - The agent A_i's belief about the A_j's willingness to risk a conflict is:

$$Risk_i^j(t) = \begin{cases} 1 & \text{if } U_i^j(\delta_j(t)) = 0 \\ (U_i^j(\delta_j(t)) - U_i^j(\delta_i(t))) / U_i^j(\delta_j(t)) & \end{cases}$$

- **In the seller's point of view, the utility and risk is defined by:**
  - The utility that agent A_j believes it will have by offering $\delta_j(t)$ is: $U_j^j(\delta_j(t)) = \max(\delta_j(t) - \gamma_j, 0)$
  - The utility that agent A_j believes it will have by accepting A_i's offer $\delta_i(t)$ is: $U_j^j(\delta_i(t)) = \max(\delta_i(t) - \gamma_j, 0)$
  - The utility that agent A_j believes that agent A_i will have by offering $\delta_i(t)$ is approximated by:
    $U_j^i(\delta_i(t)) = \max(\gamma_m - \delta_i(t), 0)$
  - The utility that agent A_j believes agent A_i will have by accepting A_j's offer $\delta_j(t)$ is approximated by:
    $U_j^i(\delta_j(t)) = \max(\gamma_m - \delta_j(t), 0)$

Hence, we can now define the degree of willingness to risk a conflict as follow:

  - The agent A_j's belief of willingness to risk a conflict is:

$$Risk_j^j(t) = \begin{cases} 1 & \text{if } U_j^j(\delta_j(t)) = 0 \\ (U_j^j(\delta_j(t)) - U_j^j(\delta_i(t))) / U_j^j(\delta_j(t)) & \end{cases}$$

  - The agent A_j's belief about the A_i's willingness to risk a Conflict is:

$$Risk_j^i(t) = \begin{cases} 1 & \text{if } U_j^i(\delta_i(t)) = 0 \\ (U_j^i(\delta_i(t)) - U_j^i(\delta_j(t))) / U_j^i(\delta_i(t)) & \end{cases}$$

If t is not the last step in the negotiation, then the risk is always between 0 and 1, for both agents.
$Risk_i^i(t)$ is an indication of how much A_i is willing to risk a conflict by sticking to his last offer. As $Risk_i^i(t)$ grows, agent A_i has less to lose from a conflict, and will be more willing to not concede, and risk reaching a conflict. Intuitively, we propose the agent strategy where the agent with a smaller risk will make the next concession.
Let's look at the strategy in detail. The buyer agent start the negotiation by offering the seller the deal that is best for him among all possible deals. Next at every subsequent step t, each agent calculate his risk (ie. $Risk_i^i(t)$) and estimate his partner's risk (ie. $Risk_i^j(t)$). if his risk is smaller or equal to that of his partner, then it must make an offer that involves the minimal[6] sufficient[7] concession from his point of view. Otherwise, it can offer the same deal that it offered previously.
Of course, at every point the agents are only making offers from within the acceptable set.

7. Conclusions and Future work

In this paper we have presented a generalized market architecture that provides support for a variety of transactions types, from simple buying and selling to complex multiagent contract negotiations. We have also presented a negotiation strategy, implemented by our best buyers/sellers, that takes advantage of the services of the market to construct beneficial contracts. Our market architecture is organized around three basic components: the market place and its manager, the customers (buyers/sellers), and the transaction sessions. We have shown that the introduction of an explicit mediator can help resolving conflicts and add value to multiagent contracting, by allowing different types of contracts.
We have begun implementing software agents to interact within our market, and intend to do some empirical measures to analyze the mediation performance. We have also planed to evaluate our negotiation strategy in term of its efficiency, stability, and simplicity.
For empirical evaluation, we eventually would like to open our testbed to outside participation over the internet.
This work raises several interesting questions for future research:

- Other requirements of the virtual market: (1) Providing a secure and private credit and payment mechanisms, (2) Controlling fraud and misrepresentation, (3) Discouraging counterspeculation, (4) Interoperating with other new and existing E-commerce service.
- Penalties in contracts: exploring the role of decommitment penalties to manage the uncertainty about reliability of the agents.
- Complex structure for the bid's cost: extending the algorithm to include time and other qualitative considerations in addition to price, in evaluation of the offers.
- Computational limits of the agents: agents have limited computational resources. Hence they must decide on the amount of time to take for refining an offer, and make a trade off between the time for refining a solution and taking several negotiations simultaneously. The manager has to decide on time for searching the graph of dependencies or the cases base.

[5] In the Zuethen strategy [6] it is assumed that the two agents has complete and correct knowledge about each other, thing that is unrealistic to suppose and implement in a market place.

[6] A sufficient concession is one that changes the balance of risk between the agent and his partner. For example, after a sufficient concession made by A_i we will have $Risk_i^i(t) > Risk_i^j(t)$.
[7] The minimal sufficient concession is the sufficient concession that the agent believes it gives his partner the least utility.

References

[1] C. Castelfranchi, Guarantees for autonomy in cognitive agent architecture. In intelligent agents, ECAI’94 (M. Woolgrige & N. Jennings), pages:56-76, august 1994.

[2] M. Pattie, A. Chavez, Kasbah: An Agent Marketplace for Buying and Selling Goods. In Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology. April 1996.

[3] J.A. Rodriguez, F. Noriega, C. Sierra, J. Padget, FM96.5 – A java based electronic auction house. In second international conference on the practical application of intelligent agents and multi-agent technology, PAAM’97, 1997.

[4] M. Tsvetovatyy, M. Gini, B. Mobasher, Z. Wieckowski, MAGMA: An Agent-Based Virtual Market for Electronic Commerce. In Applied Artificial Intelligence, special issue on Intelligent Agents, No 6, September 1997.

[5] T.W. Sandholm, Negociation among self-interested computationally limited agents. PhD Thesis, University of Massachusetts, 1996.

[6] J. S. Rosenschein, G. Zlotkin, Rules of Encounter. MIT Press, 1994.

[7] R.G. Smith, The Contract Net Protocol : high-level communication and control in a distributed problem solver. In Readings in distributed artificial intelligence (A.H. Bond & L. Gasser), pages : 357-366, Morgan kauffmann publishers 1988.

[8] J.C. Bernard, L. Esmahi, D. Gauvin, and H. Marchal, Un Modèle de coopération pour les agents sociaux. Soumis pour le Colloque International sur les Nouvelles Technologies de la Repartition, NOTER’98. Montréal, Québec, Canada, Octobre 1998.

[9] H. Varian, Electronic mechanism design for computerized agents. In readings of the usenix workshop on electronic commerce, July 1995.

[10] T.R. Gruber, A translation approach to portable ontology specifications. In Knowledge acquisition, vol 5, No 2, pages: 199-220, 1993.

[11] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck, DRAFT Specification of the KQML Agent-Communication language, plus example agent policies and architecture. DARPA Knowledge Sharing Initiative External Interfaces Working Group, June 1993.

[12] Foundation for Intelligent Physical Agents (FIPA), Agent Communication Language. FIPA’98 Draft Specification : Part2, July 1998.

[13] J. Kolodner, Case-Based Reasoning. Morgan Kaufmann Publishers, 1993.

[14] F. Zeuthen, Problems of monopoly and economic welfare, 1930.