FM96.5 A Java-based Electronic Auction House

Juan A. Rodríguez, Pablo Noriega, Carles Sierra, Julian Padget
IIIA; LANIA-IIIA; IIIA; U. Bath
IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
Vox: +34-3-5809570, Fax: +34-3-5809661
{jar, pablo, sierra}@iiia.csic.es; jap@maths.bath.ac.uk
http://www.iiia.csic.es

March 10, 1997

Abstract

We present an implementation of an electronic auction house inspired by the age old institution of the fish market, where both software and human agents may trade. This implementation supports fair, lively and robust bidder interactions.

FM96.5 is a Java-based multi-agent environment that allows for a real-time concurrent operation of the complete fish market auction process by making use of multi-threading. Agent interactions in this structured environment are modelled through standardized illocutions implemented upon Java Object Serialization.

All market-owned agents are deployed through a simple layered architecture, while buyer and seller agents of arbitrary complexity are confined to the market behavioural conventions through standardized Java agent interface applets.

1 Introduction

Internet is spawning many new markets. One that is particularly attractive for multi-agent technologies is network-based trading. But if that market is to become an effective actual market various non-trivial issues need to be addressed. Three issues appear to be particularly significant:

- *Diversity* , of goods, trading conventions, participants, interests.
- *Dispersion*, of consumers and producers, and also of resources and opportunities.
- *Safety and security* of agent and network-mediated transactions.

Thus it is not surprising that they have been the object of concern and positive attention both by the commercially interested parties as well as the academic community.

We propose to address those issues through a *mimetic* strategy, i.e. by adapting to the new context created by the Information Highway those traditional *institutions* that have proven effective in dealing with those same issues[1].

Traditional trading institutions such as auction houses –and the fish market in particular– have successfully dealt with the issues of diversity and dispersal. For instance, by defining strict trading conventions where goods of specified kinds (e.g. fish of certain quality) are traded under explicit time/location restrictions (e.g. twice a day at fixed times at the fish market building) under strict negotiation protocols (e.g. downward bidding[2]). Participating agents are subject to terms and conditions –involving identity, credit and payment, guarantees, etc.– whereby the soundness of transactions becomes a responsibility of the institution itself, who in turn enforces those terms and conditions on its own behalf. In practice, the auction house upholds the fairness of the negotiation process and the accountability of transactions by defining and enforcing stable conditions on:

- the eligibility requirements for participating buyers and sellers
- the availability, presentation and delivery of goods
- acceptable behaviour of participants within the site
- the satisfaction of public commitments made by participants

We believe that similar functions may advantageously be instituted for multi-agent systems. Be it to address some problems derived from the complexity of multi-agent interactions, or –more practically– to make acceptable some real-world applications of multi-agent technologies. In this spirit we have advocated the implementation of structured environments that allow for the definition and enforcement of explicit constraints on multi-agent interactions [12]. We have also advanced some formal elements for that purpose [11].

In this paper we show how this mimetic strategy can lead to an actual electronic auction house. We present a *proof of concept*-level release, FM96.5, of an electronic auction house that is a rather complete implementation of the trading conventions of the fish market. It includes, among other features, a sound and fair implementation of a real-time downward bidding protocol and secure user interfaces that allow buyer agents of arbitrary complexity to participate in auctions subject to the explicit –and enforceable– behavioural conventions of the fish market. In Section 2 we outline what the electronic Fishmarket consists of, in Section 3 we present our design assumptions and in Section 4 how they were implemented. In a final section we discuss related and future work.

[1] We use the term *institution* in the sense proposed by [13] as a "..set of artificial constraints that articulate agent interactions".

[2] The Spanish fish market still uses the traditional *downward bidding* protocol in which boxes of fish are adjudicated to the buyer who *stops* a descending sequence of prices that is called by an auctioneer in front of all registered buyers. This protocol is also called a *Dutch auction* because it is the way flowers have been traditionally traded in Holland. For historical references and the classical economic-theoretical outlook on auctions, cf., [7, 10].

2 FM 96.5 Blueprint

The actual fish market[3] –and other similar auction conventions– can be described as a place where several *scenes* take place simultaneously, at different places, but with some causal continuity. Each scene involves various agents who at that moment perform well-defined functions. These agents are subject to the accepted market conventions, but they also have to adapt to whatever has happened and is happening at the auction house at that time. The principal scene is the auction itself, in which buyers bid for boxes of fish that are presented by an auctioneer who calls prices in descending order –the downward bidding protocol. However, before those boxes of fish may be sold, fishermen have to deliver the fish to the fish market (in the *sellers' admission scene*) and buyers need to register for the market (at the *buyers' admission scene*). Likewise, once a box of fish is sold, the buyer should take it away by passing through a *buyers' settlements scene*, while sellers may collect their payments at the *sellers' settlements scene* once their lot has been sold.

One important aspect of the actual fish market –which can be transferred directly to the electronic version– is the presence of market intermediaries: the auctioneer, a market boss, a receptionist, a credit officer. These intermediaries interact with buyers and sellers on behalf of the fish market, and therefore have authority to request, acknowledge, dismiss or accept all the actions that sellers and buyers need to perform within the fish market. Furthermore, all those interactions between the market intermediaries and external agents (buyers and sellers) can in fact be associated with standardized illocutions, some of which are probably tacit in the actual fish market, but explicitable nonetheless in the computational model.

FM96.5 has been designed to show the full complexity of those interactions while keeping as strong as possible a similarity with the ontological elements of the actual fish market. For instance, we have tried to identify computational agents in FM96.5 with either buyers or sellers or actual market intermediaries –we identify agents not with functions of intermediation, but with actual persons–. We have also tried to mirror all actual fish market illocutions, tacit or explicit, with agent illocutions that are always explicit. Market documents correspond to FM96.5 log inscriptions, and market instruments –boxes, remote control bidders– are implemented as FM96.5 objects and classes (goods record, buyer interface, ...).

In spite of this healthy mimetic intention, previous implementations of the Fishmarket had shown us that a careful consideration was needed to represent computationally some aspects of physical reality[4]. We had previously realized that activation and closing of the market involved evident differences between the actual market and its computational models. The appropriate implementation of collective speech acts –such as broadcasting and multicasting– also required subtle analysis. In this version we decided to address the underlying problems of identity and persistence of entities, subjective and objective time, and causation and effects of activity with a different set of computational tools. We decided to use a more expressive concurrent programming paradigm and more general and

[3] In this paper we will use the (lower-case) expression *fish market* to refer to the actual, real-world, human-based trading institution, and the (upper-case) *Fishmarket* to denote the artificial, formal, multi-agent counterpart. Thus, FM96.5 refers to a particular implementation of the Fishmarket model of the fish market.

[4] Some references to previous work are reported in the last section of this paper,in [5, 11, 12] and in [19] , where in particular, FM96.5 is described in detail.

abstract computational constructs in the interest of achieving a realistic –i.e., robust, thorough, lively and sound– computational model. In particular, three basic implementation decisions were adopted from the start:

- All agent interactions were to be performed on a *reliable* network[5] .
- Multithreading would be used to implement concurrency[6] ; and
- Object encapsulation and strong typing would allow for layering and modularisation of the specification of agents and environment.

In this version we chose to build internal (market) agents that correspond with actual fish market intermediaries. Thus our agents should be able to perform several functions –sometimes even in different scenes– but should be able to manage precedence conditions and keep track of pending actions and obligations towards other agents. Although our emphasis in their construction has been functionality and performance, a certain degree of *layering* was brought to their design but no abstract reasoning was implemented[7].

The market boss, in FM96.5, thus, fulfils the prosaic function of a *name server* as well as the more anthropomorphic ones of auction supervisor and ultimate authority in the auction house. An auctioneer takes care of the bidding process. Two buyer intermediaries are deployed in this version: a buyer admitter who handles identity and acceptability conditions of buyer candidates, and a buyer manager who takes care of the financial dealings and physical location of buyers. There are also two seller intermediaries, a seller admitter who registers sellers and goods, and a seller manager who settles the sellers' accounts after an auction.

External (non-market) agents may be agents of arbitrary complexity, even human users, but they participate in the fish market always and exclusively through a standardized communication interface. Buyers in this version are handled through software incarnations of a *remote control device* which receives all the (significant) market illocutions, and transmits to the market only those illocutions that the buyer may express; always in a standardized form and only in scenes and moments when these illocutions are acceptable. Sellers, likewise, are always handled through similar nomadic interface-programs.

Three market activities deserve special comment, for their treatment in FM96.5 has been significantly different from what we had done in previous versions: activation, closing and bidding rounds.

In FM96.5, activation of the market is started by the market boss agent who *opens* the market place and establishes the identity of market intermediaries who are enabled by it to perform their intended functions [8]. Once these intermediaries are activated, buyers and sellers may start entering those *rooms* where they would conduct business, but always subject to the fish market behaviour and illocutory constraints. In fact, as

[5]A network is said to be reliable if messages transmitted on it are never lost or duplicated, nor message sequencing altered (e.g. TCP/IP) [4].

[6]In fact we used Java *threads* with their priority operators –aware of their implicit limitations. Cf. [6].

[7]We use the term *layering* (as in [11]) to indicate that the internal architecture of agents involves various units that represent crisply differentiated attitudes.

[8]In FM96.5 we still have a human *user* who triggers an activation command through which the market boss agent is spawned and starting conditions for an auction –including number of sellers, products and product characteristics– are passed.

soon as the market intermediaries are activated, they set up an *agenda* of pending actions that will correspond to sequential or concurrent actions (threads) they have the obligation to perform. These agendas are constantly updated since obligations are fulfilled by the market agents and new actions may be inscribed in the agenda by a directive of the market boss –for example: *open a bidding round*–, by a request from an external agent –e.g. *update my credit line*– or by a delegation from another market intermediary –e.g. *check buyer's credit status*. In this way activity is propagated to different scenes through events that are triggered sometimes by the market boss, sometimes by other market agents, but many times by sellers or buyers as well.

Market closing involves, also, some artificiality in FM96.5. The market boss may stop an auction through a *closing declaration* –whose triggering conditions are explicit, albeit varied– but actual closing requires that all pending actions of market agents be properly terminated. Depending on the prevalent situation of the market at the time of the closing declaration, the termination process may be more or less involved. In order to avoid anomalous conditions, some careful bookkeeping of delegation of execution control and of action flow had to be implemented[9].

Likewise, the implementation of the fish market's downward bidding protocol (as shown in Fig. 2 and Fig. 1) has been revised considerably. In this new version, synchronism is achieved not within each price quote –as in the actual fish market room– but within the sequence of price quotations that are needed to sell one good[10]. By doing so, and thanks to the fact that a reliable network is assumed, fairness conditions are preserved. Thus, premature bids, foot-dragging, and spoofing are adequately avoided directly by the protocol implementation, while malicious suplantation and snooping are dealt with through *ad-hoc* identity devices.

3 Design Issues

For FM96.5 we had two complementary objectives in mind. First of all we wanted a robust, stable version of the fish market that we could expand or refine in a modular fashion in order to develop and test, systematically, our theoretical proposals on agent architecture, agent models, interaction protocols and structured environments. But we also wanted a realistic example of an electronic auction house that could eventually be developed into a commercially interesting product.

Therefore, the guiding principles had to do with transparency, modularity, reusability and standardization on one hand, and, on the other, robustness, functionality and performance. Evidently, the choice of tools and programming methodology was strongly determined by these principles.

First, there was the matter of computing paradigm: Illocutions can be regarded as the basic unit of analysis in the Fishmarket. In the actual-world fish market, these illocutions are performed by humans with some intention in mind and eventually change the state of the world in a way analogous to the way physical actions do (cf. e.g. [15]). In the electronic Fishmarket, an agent performing an illocution can be computationally modelled as a *client* (speaker) contacting a *server* (receiver) and sending a *message* (illocution). In the same way, an agent listening to an illocution (message) can be seen as a server (receiver)

[9]FM96.5 is not provably fault-tolerant yet, but significant security, integrity and failure-recovery features are built-in for that purpose.

[10]That (complete) sequence is called a *bidding round* in this paper.

waiting for incoming communication requests from a client, performing the necessary computation (which eventually changes the state of the world) and perhaps returning an answer to the client. Note that this client/server model is a computational model and is independent of the type of illocution. The fact that every illocution changes in some way or another the state of the receiver justifies this very convenient implementational simplification[11].

Among the competing paradigms (and technologies) for developing new client-server applications, distributed objects seem to be the leading force. Nonetheless, even though we do like the many benefits of distributed objects, we do not believe that such a paradigm is the most suitable for modelling the type of agent interactions present within the Fish-market[12]. In the actual fish market, buyers, sellers and market intermediaries utter illocutions that trigger actions on the hearers. But it is important to notice that the different behaviours exhibited by the hearers are exclusively determined by themselves as a response to incoming messages. From a computational point of view, all we need, then, is to bundle clients (*speakers*) messages and send them out, and it should be up to servers (*hearers*) to determine how to handle incoming messages. We see no benefit from endowing agents with the capability to invoke methods on remote objects since we do not intend that clients trigger actions on the servers' side but only that they *provoke* actions to be triggered. Therefore, we prefer the model of clients' illocutions triggering actions in the server, in contrast to clients invoking those actions directly.

And then there is the matter of concurrence: One of the main features of the fish market is that it is composed of several, isolated scenes whose activities happen in a concurrent way. But, notably, market intermediaries may be involved in tasks that happen (simultaneously) in different scenes, (as depicted in Fig. 2). We modelled scenes as sets of distributed processes and gave to our market agents a multi-threaded architecture so they are capable of both servicing requests and delegating tasks concurrently. For instance, the buyers' manager may be active enrolling several buyers in its list of buyers while at the same time be involved in verifying whether a bid made in the current bidding round should be regarded as legal. Hence, we in fact model two levels of concurrence. On one hand, that corresponding to the concurrent activity of isolated scenes, modelled as a set of distributed processes. And on the other hand, the inner activity of each market agent, modelled as a multi-threaded process.

Consequently, action-flow in the Fishmarket is non-trivial. One should distinguish an agent-flow corresponding to buyers and sellers moving from scene to scene, and a communication-flow caused by illocutions exchanged between agents. In order to model the mobility of buyers and sellers, we designed our scenes as virtual scenes made up of processes that might physically be running at different sites but which are always virtually situated within the same scene. Buyers and sellers in FM96.5 have therefore the impression of moving among scenes (e.g. from the admission office to the auction

[11]Two technicalities may be worth noting. First, this simplification imposes processing costs: A cost is paid in the interpretation of the illocution on the server side –different illocutions trigger possibly different actions in the server– and another cost had to be paid at the client side in order to produce the utterance of the illocution. Second, a true client/server model usually implies an explicit response from the server to every request from a client. In FM96.5, for performance and transparency reasons, we actually build in a few illocution/action sequences in which servers give no explicit replies. But, these are all reifyable as true client/server interactions.

[12]It will be evident when we discuss the external agents' *nomadic agent interfaces* that the distributed object approach is indeed quite useful for those devices.

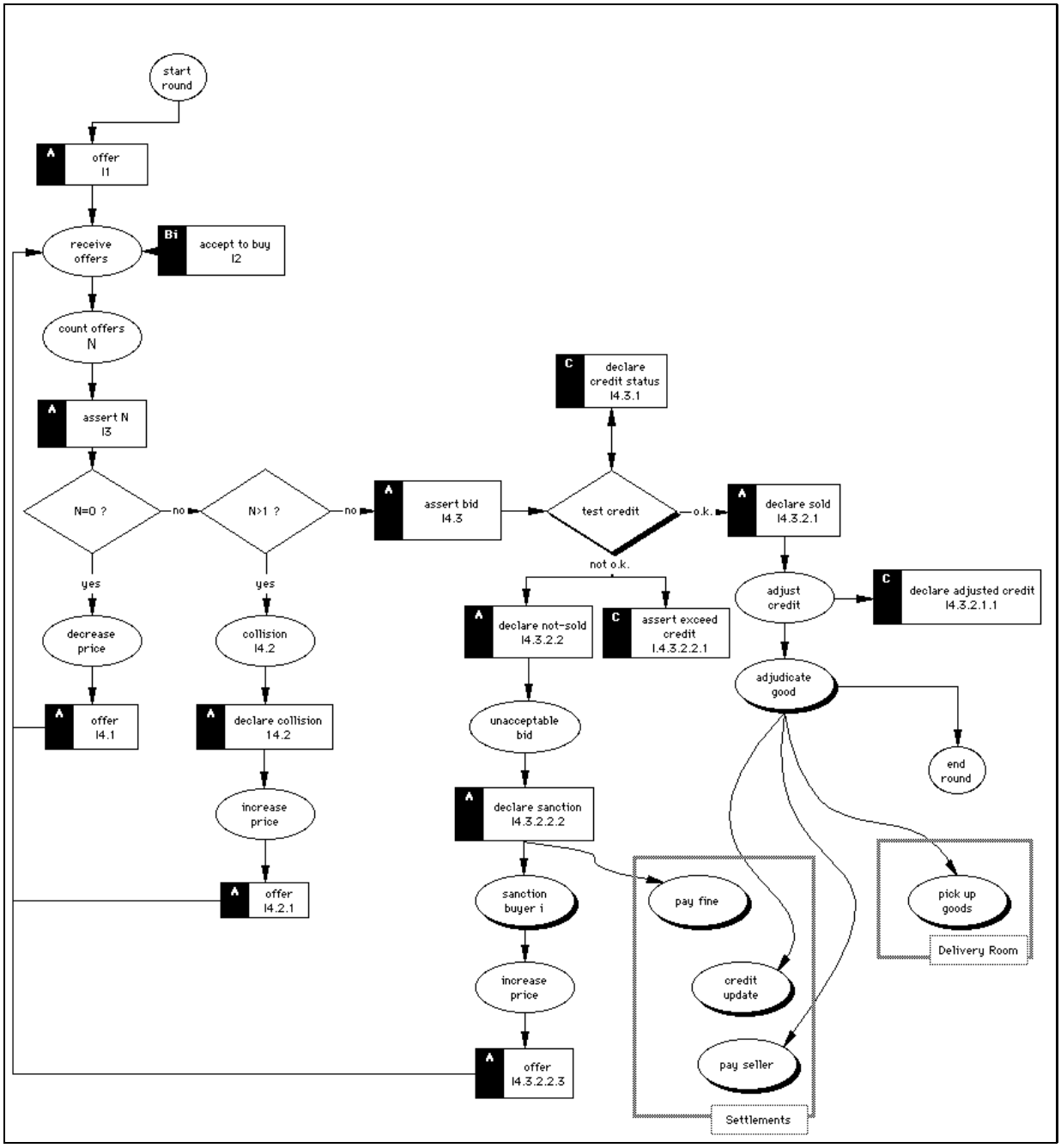
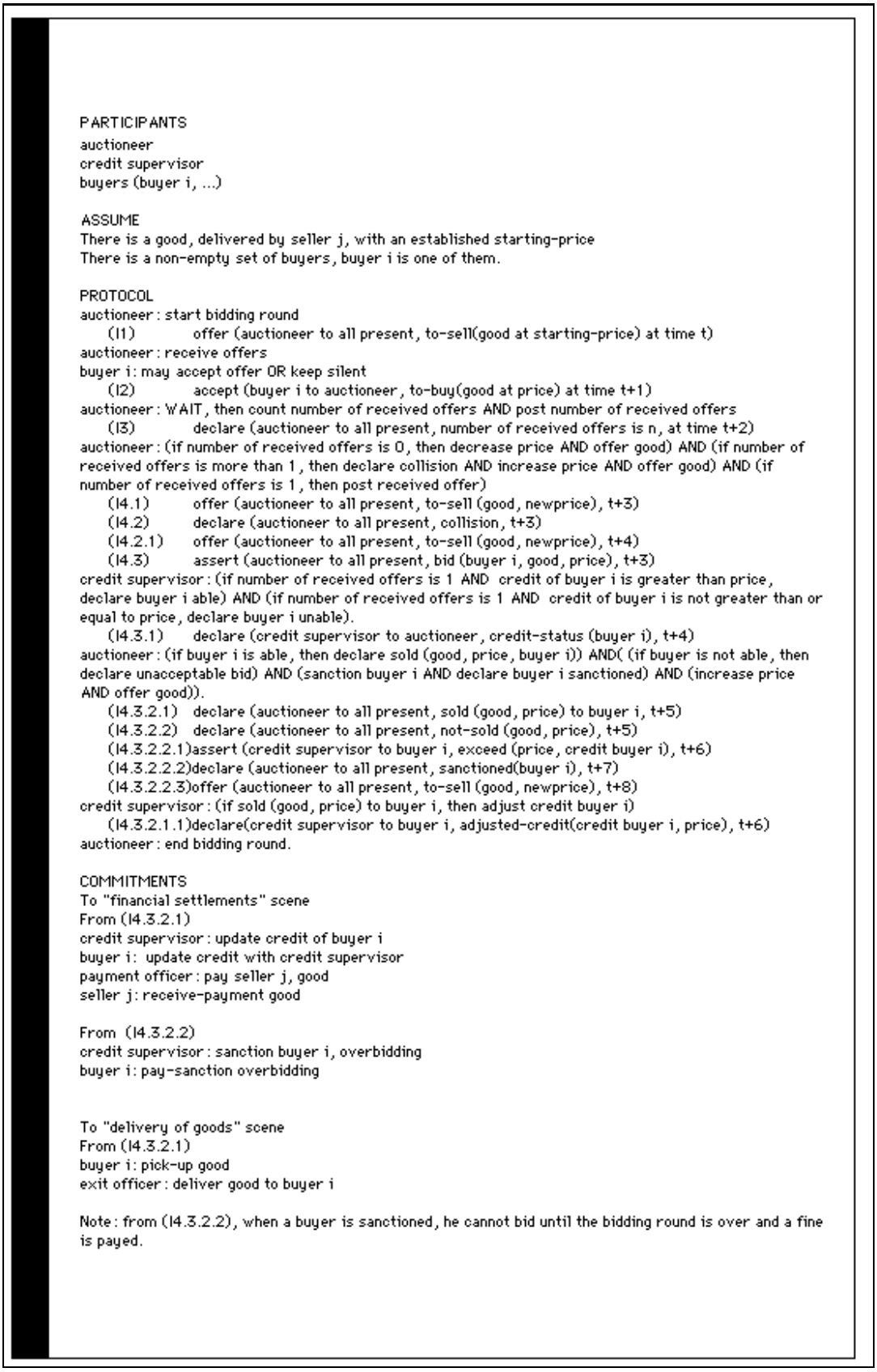


Figure 1: The Fishmarket *naive* downward bidding protocol.



PARTICIPANTS
auctioneer
credit supervisor
buyers (buyer i, ...)

ASSUME
There is a good, delivered by seller j, with an established starting-price
There is a non-empty set of buyers, buyer i is one of them.

PROTOCOL
auctioneer : start bidding round
(I1) offer (auctioneer to all present, to-sell(good at starting-price) at time t)
auctioneer : receive offers
buyer i: may accept offer OR keep silent
(I2) accept (buyer i to auctioneer, to-buy(good at price) at time t+1)
auctioneer : WAIT, then count number of received offers AND post number of received offers
(I3) declare (auctioneer to all present, number of received offers is n, at time t+2)
auctioneer : (if number of received offers is 0, then decrease price AND offer good) AND (if number of received offers is more than 1, then declare collision AND increase price AND offer good) AND (if number of received offers is 1, then post received offer)
(I4.1) offer (auctioneer to all present, to-sell (good, newprice), t+3)
(I4.2) declare (auctioneer to all present, collision, t+3)
(I4.2.1) offer (auctioneer to all present, to-sell (good, newprice), t+4)
(I4.3) assert (auctioneer to all present, bid (buyer i, good, price), t+3)
credit supervisor : (if number of received offers is 1 AND credit of buyer i is greater than price, declare buyer i able) AND (if number of received offers is 1 AND credit of buyer i is not greater than or equal to price, declare buyer i unable).
(I4.3.1) declare (credit supervisor to auctioneer, credit-status (buyer i), t+4)
auctioneer : (if buyer i is able, then declare sold (good, price, buyer i)) AND((if buyer is not able, then declare unacceptable bid) AND (sanction buyer i AND declare buyer i sanctioned) AND (increase price AND offer good)).
(I4.3.2.1) declare (auctioneer to all present, sold (good, price) to buyer i, t+5)
(I4.3.2.2) declare (auctioneer to all present, not-sold (good, price), t+5)
(I4.3.2.2.1)assert (credit supervisor to buyer i, exceed (price, credit buyer i), t+6)
(I4.3.2.2.2)declare (auctioneer to all present, sanctioned(buyer i), t+7)
(I4.3.2.2.3)offer (auctioneer to all present, to-sell (good, newprice), t+8)
credit supervisor : (if sold (good, price) to buyer i, then adjust credit buyer i)
(I4.3.2.1.1)declare(credit supervisor to buyer i, adjusted-credit(credit buyer i, price), t+6)
auctioneer : end bidding round.

COMMITMENTS
To "financial settlements" scene
From (I4.3.2.1)
credit supervisor : update credit of buyer i
buyer i: update credit with credit supervisor
payment officer : pay seller j, good
seller j: receive-payment good

From (I4.3.2.2)
credit supervisor : sanction buyer i, overbidding
buyer i: pay-sanction overbidding

To "delivery of goods" scene
From (I4.3.2.1)
buyer i: pick-up good
exit officer : deliver good to buyer i

Note : from (I4.3.2.2), when a buyer is sanctioned, he cannot bid until the bidding round is over and a fine is payed.

Figure 2: The Fishmarket downward bidding protocol. *Naive* communication flow.

room, from there to the settlements office and so on) in the same way human buyers and human sellers would in the actual fish market. As to the communication flow, we opted for standardizing the structure of the messages being exchanged between agents. Each message is regarded as a Java object containing a tag, information about the sender and the contents of the message, which is in turn a Java object. The use of Java Object Serialization [25] allowed for serializing each message at the sender side and deserializing it at the receiver side in a straightforward way.

And finally, external agent interfaces: In order to achieve the most realistic implementation of the auction house activity, we decided to standardize as much as possible all conceivable external agent interactions with the market. We took advantage of the highly structured negotiation convention of auctions, and of the fact that in actual fish markets all bidding round interactions can be mediated through a remote control device. Thus, we built *nomadic agent interfaces* –a sort of electronic remote control devices– that could be used as universal interfaces by buyer and seller agents. This nomadic interface is installed in the external agent's computer and becomes the only channel through which messages can pass between external agents and market (internal) agents. Since the Fishmarket interactions are all linked to illocutions, this interface is all that is needed, in principle, to participate effectively in the electronic auction house. But in fact, these interfaces fulfil other necessary duties as well: they sustain the identity of participants, validate illocution emission and reception, and, generally speaking, enforce the auction-house rules –including the bidding protocol[13]. It should be noted, then, that in FM96.5 there are really no buyer or seller agents, only their nomadic interfaces. But through these nomadic interfaces buyer and seller agents –developed and owned elsewhere or even human buyers or sellers– can participate in electronic auctions. This is, we think, the kind of design Castelfranchi was suggesting :

> "...artificial agents can be designed, and they should be designed as rational. But they should also be integrated into human organisations, and be capable to interact with their human partners in an adaptive and understandable way." [2].

In our choice of tools, we profited from our previous experiences too. Having already developed prototypes using PVM and MPI for internetworking and C and EU-Lisp for other features, Java suggested relevant advantages [6] that were worth testing in the Fishmarket implementation:

- Java provides the advantages of object-oriented languages (reusability, modularity, encapsulation, etc.) and claims to be designed for maximum portability.
- Its ease of programming and safety features help produce debugged code quickly.
- Java is reported to be valuable for distributed network environments.

[13] Obviously, this interface permits to address the security issues that would arise when arbitrary foreign agents (i.e. whose code we do not know) are admitted into the Fishmarket. In fact the nomadic quality of the interface makes it possible for other external agents –and necessary for the agent who uses it– to prove a *zero-information* property, *i.e.* that through the interface no information of the market, nor any information of the external agent can be transferred outside the interface, except for the one that is explicitly stated by the interface. Note also that our nomadic interfaces are akin to the payment and service cassettes used in the construction of the Java Wallet (cf. [24]), and can in fact be readily connected to them.

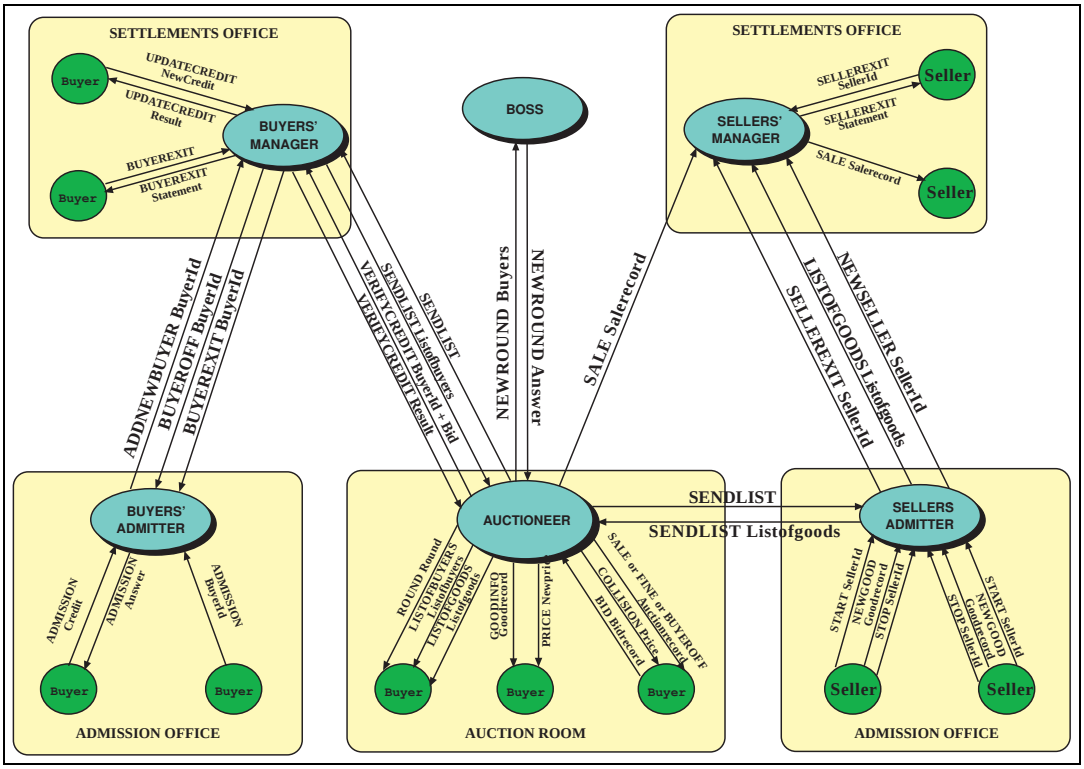


Figure 3: A simplified diagram of the communication-flow between and within market scenes

- There is available an increasingly large collection of specialized packages which allow the programmer to count on powerful tools to tackle distributed computing [25, 26, 21], database connectivity [22], security [27], etc.

Besides, given the recent industrial commitment and investment as well as generalized commercial activity around Java, it seems to us there is strong indication that Java may become a *de-facto* standard, therefore having permanence and complementary developments that would facilitate taking FM96.5 to a product-level stage.

4 Implementation

FM96.5 was developed as an object-oriented client/server distributed application which is actually made up of a collection of Java applications that can run as both applets[14] or standalone applications. There is in fact a Java application for each of the agents depicted in Fig. 2. In addition, one separate package groups those classes defining data structures while another package contains those classes referring to client and server connections capable of reading and writing whole objects. The mechanisms of exception handling that ease the task of dealing with network error conditions are encapsulated within this last package. We utilised JDK 1.0.2, and Java Object Serialization [25]; on a LAN composed of a SUN SPARC/20, several SUN SPARC/5 and a few Macintosh and PCs.

[14]These applets can be activated from browsers such as Netscape and HotJava.

Each market agent works as a multi-threaded process with a graphical user interface. This multi-threaded architecture allows market agents to service several message-shaped requests concurrently. Nevertheless not all requests are handled in the same way. There are requests that are regarded to be more important than others. Threads servicing different types of requests are started off with different priorities. Therefore, a market agent would give the highest priority to what it contemplates as the most important tasks, then to requests made by other market agents, requests made by buyers and sellers and, finally, the closing request emitted by the market boss.

Perhaps the major challenge from a technical point of view was the design of the protocols involved in the main activities in the market - activation, bidding round and closing - since they implied the co-ordination of the activities of sets of distributed processes:

(i) **Activation:** The market boss agent acts as a central name server. Once the market boss agent starts, the rest of market agents are forced to check in at the boss' office by identifying themselves and later on ask for the addresses of those market agents that they will work with. Although an auction cannot take place until all the market agents check in, the market does not remain inactive until that condition occurs. In fact, buyers and sellers might start entering the market as soon as the admitters are ready.

(ii) **Bidding protocol**: We can identify several situations that may arise during each bidding round in the fish market:

- *Proper sale.* When a single buyer submits a bid that his credit can support, it is turned into a sale.
- *Unsupported bid.* When a buyer submits a bid that his credit cannot guarantee. The buyers' manager fines this bidder and the round is restarted by the auctioneer who calculates the new starting price by increasing 25% the price within the bid.
- *Collision.* When two or more buyers simultaneously submit the same bid. The auctioneer declares a collision and restarts the round. Again, the new starting price is calculated by increasing 25% the collision price.

FM 96.5 implements faithfully all theses cases, plus two more:

- *Expulsion.* When a buyer is overdrawn and cannot back up a fine, he is sent off the market and the round is restarted as usual
- *Minimum price.* Each good is assigned a minimum price when passing through the sellers' admitter office. If minimum prices are reached, the round is restarted as usual.

Finally, it is worth noticing that FM 96.5 will start or restart a round only when the boss authorises the auctioneer. We implemented minimality conditions on the number of buyers present and the goods to be auctioned.

Our main concern when implementing the downward bidding protocol has been to ensure fairness while preserving realistic response time[15]. In FM96.5 we achieve it

[15]In the fish market this corresponds to time delays between prices that are short enough to be imperceptible to human buyers but long enough to allow for collisions (i.e. one or two seconds between successive prices).

–without supposing common fixed delay intervals– through a clever alternative to common clocks.

In FM 96.5 we regard the termination of a bidding round as the synchronisation point of the round participants. All buyers receive syncopated price sequences. If a buyer is going to submit a bid, he will signal this as soon as the price quotation reaches his target bid. The signal sent back from the remote control device to the auctioneer includes the price at which the buyer signalled his intention and the time stamp. As soon as the auctioneer receives a bid, he multicasts to the buyers' remote control devices the information that a bid is in, which these devices must acknowledge. Since we can assume a reliable network, the order in which messages are transmitted is never altered, thus the auctioneer must receive any delayed bids before we receive the corresponding acknowledgements from these bidders. Hence we have the standard two cases:

- *Proper sale.* One bidder
- *Collision.* Multiple bidders at the same price

and a new case:

- *Multiple bidders at different prices.* In this case, the highest price bid wins if there is just one, or we restart as usual.

To understand our current implementation of the bidding protocol, it may prove helpful to regard downward bidding rounds as competitions for a unique resource (the good being auctioned). This turns out to be similar to the problem of *distributed mutual exclusion* ([3]) with a slight difference: distributed mutual exclusion involves a single process being given the right to access shared resources temporarily before another is granted it, whereas a bidding round involves a single process being given the right to acquire resources that will not be shared with the rest of the processes. Therefore it is not surprising the many features our bidding protocol has in common with Ricart and Agrawala's distributed algorithm using logical clocks to implement mutual exclusion [14]. In fact, it is possible to prove that a given implementation of the downward bidding protocol is *correct* using an adaptation of Lynch's formalism for I/O Automaton correctness proofs [9].

(iii) **Closing:** We could briefly summarise the termination of the market in the following steps:

1. The boss sends a closing signal to the auctioneer (the boss has the right to close the market at any time).
2. The auctioneer acknowledges this closing signal as soon as the current bidding round is over.
3. The boss multicasts a closing signal to the rest of market agents.
4. Buyers' admitter and sellers' admitter close the doors of the market to buyers and sellers intending to enter by rejecting their requests.

5. Lastly all the market agents engage in a co-ordinated activity which basically consists in finishing off all pending activities before definitively terminating. In this sense it is important to highlight that any market agent cannot properly finish until all pending requests have been dealt with, all pending services have been provided and all potential clients have indicated no further need for services. Each market agent will keep track in its agenda of pending obligations and active agents so as to determine what remains to be done when a closing signal is received from the market boss.

Let us, finally, draw attention to buyers' and sellers' nomadic interfaces. These devices are versatile. First, they allow the user to determine the scene (or virtual location) where it wants to be active (external agents can only act –or more properly, engage in *dialogue* with market agents– at one place at a time). Secondly, depending on the specific location, and the prevalent market conditions, it displays market information and habilitates dynamic interface windows –and buttons– through which the external agent receives and transmits the pertinent standardized illocutions. Finally, since these devices are market-owned, some accounting, liveness and security functions are performed in the background and transmitted to the market agents. Note that these remote control devices are coupled with a graphic interface when dealing with human agents, while when interacting with external software agents they transmit and receive message-shaped illocutions between these external and the internal market agents.

It is important to notice that these FM96.5 nomadic interfaces convey to buyers and sellers other information that human buyers and sellers in the fish market would have available *in situ*. For instance, buyers receive the list of participating buyers (which would be seen by a human buyer taking part in the auction), the list of auctionable goods (which are scattered over the floor in the auction room), details of the next good to be auctioned, and his own current credit and the list of purchases.

5 Final Remarks

As commercial use of the Internet grows, new services that combine traditional auction techniques with web-based technology have started to appear leading to a number of electronic auctions of a *naive* type. These are being used to trade different goods: vintage records [31], computers and electronics [32], art [34] objects in general [29, 30, 33][16].

Buyers and sellers interactions are in most of these cases quite natural and simple:

- Goods –which may be inscribed directly by external sellers [29, 30, 33], or otherwise obtained by the auction house– are catalogued and even sometimes displayed –electronically and/or physically– before and during an auction.
- After registering in a given auction –usually a simple e-mail inscription– a buyer can submit his bids either by e-mail ([31, 32, 33]), by fax ([31, 32]), by submitting a web-form ([29, 30, 34]) or even by post ([31]).

[16] We should acknowledge that other forms of electronic auctions have been developed recently. On one hand, there are actual auction simulation environments like FCC [1, 8], whose purpose is to train bidders or to test innovative bidding protocols and trading mechanisms(*cf. e.g.* [16, 17, 18]). On the other hand, electronic auctions have been used as coordination mechanisms in market-oriented programming. Although these developments have many points of contact with our own project, a full comparison is beyond the scope of this paper.

- Payments are usually through credit cards, and sales are definite up to actual payment, but most of the time the physical transactions (actual payments) are explicitly relinquished by the auction houses. Some sales are defeasible if protested –and properly supported– within a period of time [33, 34].

- Most service providers adopt a rather primitive sealed-bid *English* auction protocol –the item goes to the highest bidder– except for [32], which offers several auction formats (*Yankee auction, Dutch auction, Straight sale, Buy or Bid* and *English Auction*), [29] which has an on-line simplified English auction, and [34], whose only auction format is *Buy or Bid*.

- The evolution of each bidding round is displayed on a browser [29, 30, 32, 33] or sent by e-mail [29, 30, 31, 32, 33, 34]to participating buyers.

- In most cases, single bidding rounds are open for an extended period of time –up to a couple of months; the exception is [29] that allows for more lively bidding rounds– and terminate on a previously announced closing date, though sometimes the auctioneer determines when a bidding round closes.

- Security is an important concern in some of these applications: [29, 32] and handle security by utilising the Netscape Commercial Server which uses the HTTPS protocol to encrypt bids; whereas [30] uses a validation code for bidder identification in each auction.

These *naive* electronic auctions are probably adequate for the type of products they trade. But there are many differences between these *naive* electronic auctions and our Fishmarket implementations, and some may be significant for many types of web-based trading. Apart from those that have to do with their purpose and operation –these *naive* auctions do trade actual goods in the real world, ours is still an *academic prototype*– the most significant differences have to do with performance and accountability.

First, the performance of FM96.5 is significantly better than those of *naive* auctions in two respects: vivaciousness and functionality. Depending on the type of good to be traded and its market conditions, the speed at which bidding takes place may alter significantly the strategy –and decision making process– of buyers. FM96.5 can register buyers and perform auctions in the same frenetic speeds at which non-*naive* traditional auctions take place; something that is impossible in all these *naive* auctions (except possibly for AUCTIONLINE), while FM96.5 can also handle long-term bidding rounds without any difficulty. Similarly, the Fishmarket has a more complex structure than these *naive* auctions: our development has taken special care about modelling and automating the several scenes making up the fish market, however, most *naive* auctions seem to rely upon a centralised service which acts as admitter, auctioneer and, some times, even as the accountant.

Second, accountability. We do identify and take care of multiple conditions that either should or should not happen in auctions: Those that are inherent on the auction house conventions, those that are inherent in the interactions among participants and those that depend on the implementation of these conditions and interactions. For example, a bidding protocol should be *fair* in the sense that all participants may bid under identical conditions, and may be required to be *synchronic* in the sense of assuring equal timing for each price quotation for every bidder or *vivacious* (fast price changes) or *private* (not

Version	Place	Basic Tool	Concerns	Advantages
FM96.0	IIIA	Netscape	Fast development	Demonstrability
FM96.1	IIIA & Naples	PVM	Synchronisation, Bidding protocol	Proof of concept
FM96.2	IIIA-Bath	MPI/C	Open Network	Portability
FM96.3	IIIA-Bath	MPI/C	More agents, Market functionality	Isolated Contexts
FM96.4	IIIA-Bath	EU-Lisp/MPI	Agent interactions	Expressiveness
FM96.5	IIIA-Bath	JAVA	Modularity, concurrency, functionality, fairness, livelihood of protocol	Full functionality Robustness Expandability

Table 1: Implementations of the Fishmarket Environment

revealing bidder identities). Thus *naive* auctions may function adequately for simple one-round sealed-bid protocols or for very quiet bidding rounds. Meanwhile, FM96.5 can handle more elaborate transactions, and numerous lively rounds only assuming TCP/IP communication. Finally, as indicated in section 4, we have not only an empirical way of addressing anomalies, but also a formal framework through which we can characterise and prove that many of these conditions hold, or not, in a given implementation. Thus fundamental issues such as *fairness* or *privacy* can be obscured in *naive* auctions[17], but become clearly guaranteed in FM96.5.

These advantages did not come for free. They are the result of a systematic analysis of a more general problem –that of structured agent interactions– and the empirical testing of the conviction that multi-agent systems can be fruitfully applied to model and automate social interactions, such as the ones present in trading. In [5] we presented a prototype implementation of a simple version of the fish market. FM96.5 is a far more thorough implementation. In between we have addressed different aspects of the problem, and gone through the exercise of exploring specific technical or methodological issues (as shown in Table 1).

We are currently developing a few extensions of FM 96.5. The first one concerns the implementation of alternative bidding protocols (English, FCC, MexTR, etc.), in order to have a general electronic auction platform. Second, the development of intelligent buyer and seller agents capable of exhibiting different trading behaviours. And, as a complementary task, we are also engaged in the development of analysis tools that will serve two main purposes: to help debug future releases of FM 96.5 –and other structured multi-agent environments– and to run simulations that offer the possibility of testing agents aptitudes and studying the emergent behaviour of the market as a whole[18] These developments are part of a research programme addressed at the exploration of agent-mediated institutions, where accountable trading and negotiation interactions are our main focus [35]. This programme shares many concerns with the *market-oriented programming* approach,

[17]All the *naive* auctions cited happen not to be strictly fair. Those bids arriving after the closing date are rejected –irrespective of the reason for that delay– although bidders might have expressed their intention to bid before that time.

[18]Some future developments are more mundane: Among the many features of JDK 1.1, we find particularly useful the Java Database Connectivity modules [22]. As was mentioned above, our market agents keep track of pending obligations through an *agenda*; with JDK 1.1 features we plan to make these devices persistent. This will allow for the implementation of databases of transaction information in a similar fashion to that proposed in the specification of the Java Electronic Commerce Framework [24]. Finally, although our implementation does not address secure transactions issues, we plan to tackle them in the near future with the aid of the SSLava toolkit [27].

(*cf.* Wellman) [18, 36], particularly the desing of automated trading mechanisms, the study of trading features that should be brough to electronic marketplaces, and the development of sound schemes for automated negotiation.

Other developments –like the full formalization and proof of properties and conditions of agent models, roles, protocols and their implementations– are fundamental and open-ended. But all these more fundamental improvements are desirable to achieve realistic market needs –e.g. a true fault-tolerant operation of the market, and full on-line auditing of the auction house– and essential, if agent-based technologies are to be safely and responsibly used in this our Information Society.

6 Acknowledgements

We are gratefully indebted to Xavier Márquez, manager of the Blanes fishermen's cooperative market (Blanes, Girona, Catalunya) for his illuminating explanations. We also want to acknowledge Francisco Martín for his always pertinent remarks and enthusiastic collaboration, as well as Andreas Kind and Julio García contributions.

This work has been partially supported by the European TMR number PL93-0186 VIM, CEC/HCM VIM project, contract CHRX-CT93-0401 (cf. [28]); the Spanish CICYT project SMASH, TIC96-1038-C04001; the Mexican CONACYT grant [69068-7245], and the British EPSRC grant GR/K27957 (http://www.maths.bath.ac.uk/ jap/Denton). Juan Antonio Rodríguez enjoys a CIRIT doctoral scholarship FI-PG/96-8.490 from the Catalan Government, and developed a first version of FM96.5 while on an extended visit supported by VIM/HCM at the University of Bath. Carles Sierra is currently on sabbatical leave at Queen Mary & Westfield College, University of London, thanks to the Spanish Ministry of Education grant PR95-313.

References

[1] Backerman, S. R., Rassenti and Smith, V. (1996). *Efficiency and Income Shares in High Demand Energy Networks: Who Receives the Congestion Rents When a Line is Constrained?.* In UPC Conference on Auctions, Theory and Empirics.

[2] Castelfranchi, C. and Conte, R. (1996). *Distributed Artificial Intelligence and Social Science: Critical Issues.* In Foundations of Distributed Artificial Intelligence. (G.M.P. O'Hare and N.R.Jennings, ed.). John Wiley & Sons, New York.

[3] Coulouris, G., Dollimore, J. and Kindberg, T.(1994). *Distributed Systems.Concepts and Design.* (2nd edition). Addsison Wesley.

[4] Comer, D.E. and Stevens, D.L.(1993). *Internetworking with TCP/IP Volume III: Client-server programming and applications.* Prentice-Hall International, New Jersey.

[5] Di Napoli, C., M. Giordano, M. Mango Furnari, C. Sierra and P. Noriega (1996). *A PVM Implementation of the Fishmarket Multiagent System.* In Proceedings IX International Symposium on Artificial Intelligence, Cancún (México), Nov. 12-15, 1996 (in press).

[6] Gosling, J.(1996). *The Java Programming Language.* Addison-Wesley, Reading.

[7] McAfee, R.P. & McMillan, J. (1987). *Auctions and Bidding.* J. Ec Lit., Vol. XXV, (June 1987), pp. 699-738.

[8] Levin, D. and Smith, J. (1996). *Entry Coordination in Auctions: An Experimental Investigation.* In UPC Conference on Auctions, Theory and Empirics.

[9] Lynch, N. (1996). *Distributed Algorithms.* Morgan Kaufman, S Fco.

[10] Milgrom, P.R. & Webber, R.J. (1982). *A Theory of Auctions and Competitive Bidding.* Econometrica; Vol. 50, N. 5 (Sep 1982), pp. 1089-1122.

[11] Noriega, P. and Sierra, C. (1996). *Towards Layered Dialogical Agents.* Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures and Languages. ATAL'96. Budapest, pp. 69-81.(L.N.A.I. Springer, in press)(IIIA-RR-96-19)

[12] Noriega, P., C. Sierra, M. Giordano, R. López de Mántaras, F. Martín, J.A. Rodríguez (1996). *The Fish Market Metaphor. A proposal for Structured Multi-Agent Negotiation Environments.* Institut d'Investigació en Intel.ligència Artificial, Research Report IIIA-RR-96-17.

[13] North, D. (1990). *Institutions, Institutional Change and Economics Performance.* Cambridge U.P.

[14] Ricart, G. and Agrawala, A. K.(1981). *An optimal algorithm for mutual exclusion in computer networks.* Comms. ACM, vol.24, no.1,pp. 9-17.

[15] Searle, J. R. (1969) *Speech acts.* Cambridge U.P., 1969.

[16] Sandholm, T. (1996) *Limitations of the Vickrey Auction in Computational MultiAgent Systems.* In Proceedings of ICMAS-96, pp. 299-306

[17] Tsvetovatyy, M. and Gini, M. (1969) *Towards a Virtual Marketplace: Architecture and Strategies.* In Proceedings of PAAM96.

[18] Wellman, M. P. (1993) *A market-oriented programming environment and its application to distributed multicommodity flow problems.* Journal of Artificial Intelligence Research, 1:1-22, 1993.

[19] FM. *The Fishmarket Project Webpage.* http://iiia.csic.es/Projects/fishmarket

[20] *HTMLScript.*http://www.htmlscript.com/samples

[21] *IDL. The Java IDL System for Distributed Programming.* http://splash.javasoft.com/JavaIDL/pages/index.html

[22] *JDBC. The JDBC Database Access API.* http://splash.javasoft.com/jdbc/

[23] *JDK. The Java Developers Kit Version 1.0.2.* http://java.sun.com:81/products/JDK/1.0.2/

[24] *JECF. The Java Electronic Commerce Framework White Paper.*http://java.sun.com:81/products/commerce/doc/white_paper.html

[25] *JOS. Java Object Serialization.* http://chatsubo.javasoft.com/current/serial/index.html

[26] *RMI. Java Remote Method Invocation.* http://chatsubo.javasoft.com/current/rmi/

[27] *SSLAVA. SSLava (tm) Information Center.* http://www.phaos.com

[28] *VIM. European Comission TMR project VIM.* http://www.maths.bath.ac.uk/ jap/VIM

[29] *AUCTIONLINE.* http://www.auctionline.com

[30] *InterAUCTION.* http://www.interauction.com

[31] *Nauck's Vintage Records.* http://www.infohwy.com/ nauck/vra19/PROTOCOL.htm

[32] *ONSALE.* http://www.onsale.com

[33] *Phoebus Auction Gallery.* http://www.phoebusauction.com

[34] *Seven Seas Trading Company, Ltd..* http://www.7cs.com

[35] *Emporium Project.* http://www.iiia.csic.es/Projects/Emporium

[36] *AuctionBot.* http://auction.eecs.umich.edu