FM0.9beta[1]

Users' Guide

Juan A. Rodríguez-Aguilar
Francisco J. Martín
Francisco J. Giménez
David Gutiérrez

Artificial Intelligence Research Institute (IIIA)
Spanish Council for Scientific Research (CSIC)
08193 Bellaterra, Barcelona, Spain
{jar,martin,dochelio,david}@iiia.csic.es
http://www.iiia.csic.es

October 1998

[1]This work has been partially supported by the Spanish CICYT project SMASH, TIC96-1038-C04001, J. A. Rodríguez-Aguilar and F. J. Martín enjoy the CIRIT doctoral scholarships FI-PG/96-8.490 and FI-DT/96-8.472 respectively. It is also available as an IIIIA research report (IIIA-RR98-32) that must be cited when referring to any results obtained with the aid of FM.

Preface

Abstract

Auction-based e-commerce is an increasingly interesting domain for developing trading agents competing in multi-agent electronic markets.

We present a framework for defining trading scenarios based on fish market auctions. In these scenarios, trading (buyer and seller) heterogeneous (human and software) agents of arbitrary complexity participate in auctions under a collection of standardized market conditions and are evaluated against their actual market performance. We argue that such competitive situations constitute convenient problem domains in which to study issues related with agent architectures in general and agent-based trading strategies in particular.

The proposed framework, FM, constitutes a test-bed for trading agents in auction tournament environments.

This users' guide will attempt at illustrating how to generate, activate, and record auction-based tournaments with the aid of FM.

Contents

List of Figures

List of Tables

Chapter 1

What is FM?

1.1 Introduction

Internet is spawning many new markets. In this sense, we observe that the proliferation of on-line auctions in the Internet—such as Auctionline (http://www.auctionline.com), Onsale (http://www.onsale.com), InterAUCTION (http://www.interauction.com), eBay (http://www.eBay.com), and many others — has established auctioning as a main-stream form of electronic commerce. Thus, agent-mediated auctions appear as a convenient mechanism for automated trading, due not only to the simplicity of their conventions for interaction when multi-party negotiations are involved, but also to the fact that on-line auctions may successfully reduce storage, delivery or clearing house costs in many markets. This popularity has spawned research and development in agent-mediated auction houses as well as in trading agents endowed with intelligent auction strategies.

The matter of trading within an auction house appears to be numbingly complex, because of the numerous variables coming into play. The actual conditions for deliberation are not only constantly changing and highly uncertain—new goods become available, buyers come and leave, prices keep on changing; no one really knows for sure what utility functions other agents have, nor what profits might be accrued — but on top of all that, deliberations are significantly time-bounded. Hence there is the intricate matter of providing agent developers with some support to help them face the arduous task of designing, building, and tuning their trading agents before letting them loose in wildly competitive markets.

The FishMarket project[5] conducted at the Artificial Intelligence Research Institute (IIIA-CSIC) attempts to contribute in that direction by developing FM, an agent-mediated electronic auction house which has been evolved into a test-bed for electronic auction markets. The framework, conceived and implemented as an extension of FM96.5[4] (a Java-based version of the Fishmarket auction house), allows to define trading scenarios based on fish market auctions (Dutch auctions). FM provides the framework wherein agent designers can perform *controlled experimentation* in such a way that a multitude of experimental market scenarios—that we regard as *tournament* scenarios due to the competitive nature of the domain— of varying degrees of realism and complexity can be specified, activated, and recorded; and trading (buyer and seller) heterogeneous (human and software) agents compared, tuned and evaluated. We argue that such competitive situations constitute convenient problem domains in which to study issues related with agent architectures in general and auction strategies in particular.

We highly recommend the interested reader to consult [2] for a more detailed, though concise, description of FM, and [3, 1] for more thorough discussions.

The current version of FM, FM0.9beta, is now available and can be downloaded from the FishMarket project web page. Next, we summarize the most salient features of this very first release.

1.2 Features

- FM is completely written in Java.

- The customizability of FM allows for the specification, and subsequent activation, of a large variety of market scenarios: from simple toy scenarios to complex real-world scenarios, i.e., from extremely simple market scenarios in which the same auction is repeated over and over till market scenarios that make FM behave like the actual market. This capability of scenario generation allows the *repeatability* of the experiments (tournaments) to be conducted.

- FM is multi-user. It allows multiple users to spawn their agents in their own machines in order to make these to participate in remote tournaments.

- FM remains *architecturally–neutral* since no particular agent architecture (or language) is assumed or provided for building trading agents. Alternatively, a library of agent templates written in Java, C, and Lisp accompanies this release in order to assist agent programmers to build their agents. In this way, the programming effort narrows down to developing auction strategies. Importantly, these templates handle the connection of the trading agent to the FM interagents: autonomous software agents which intermediate the communication between the trading agents and the institution, the market,enforcing them to follow the rules of the game.

- A built-in agent builder facility allows for the automatic generation of agents with customizable auction strategies, so that families of agents capable of simulating different trading behaviours can be easily created.

- Auctions can be monitored step-by-step thanks to the FM *Monitoring Agent*. This keeps track of every single event taking place during a tournament in order to obtain a visual, global representation of the agents' flow from scene to scene within the market as well as the communication flow (what the agents utter and when).

- The FM database stores the information to be used by trading agents to carry out market analysis and auditing.

- FM has been designed to be as user-friendly as possible. Thus, the

FM GUI allows the whole interaction between the users and FM to be done through graphical interfaces.

Chapter 2

Installing and Running FM

2.1 Where is FM?

Updated versions of FM are available, free for download via the FM project Web site located at:

http://www.iiia.csic.es/Projects/fishmarket

While version 0.9beta of FM is well-tested, newer versions of FM available on our Web site tend to have better performance, improved scalability, more bug fixes, and occasionally, new features over older versions.

2.2 Installation Process

2.2.1 Software requirements

In order to run FM, you need to have JDK 1.1.6 and JFC 1.1 with Swing 1.0.3 installed. These can be downloaded from:

```
JDK1.1.6: http://java.sun.com/products/jdk/1.1/
JFC1.1: http://java.sun.com/products/jfc
```

This is the minimum software required.

Optionally, FM is also capable of storing the market information into a MySQL database. For that purpose both a MySQL database and the MySQL JDBC driver should be installed. Both packages can be downloaded from the following sites:

```
MySQL: http://www.tcx.se/download.html
MySQL JDBC: http://www.gwe.co.uk/mysql/
```

In order to browse the database contents, you can download and install XMySQL, a freely available Xforms based GUI front end to MySQL, from:

```
XMySQL: http://web.wt.net/~dblhack/
```

Finally, FM keeps track and stores the evolution of the performance of the agents involved in each tournament in order for these to be analyzed. To obtain a visual representation of the evolution of the tournament, download gnuplot from the GNU web site (http://www.gnu.org/gnu) and follow the instructions in the manual.

2.2.2 Installation Notes

Unpacking FM

FM0.9beta is available either as a tar compressed file or as a tar gzipped file.

Unpack the software into your chosen installation directory executing either

```
gunzip FM0.9beta.tar.gz
tar xvf FM0.9beta.tar
```

or

```
uncompress FM0.9beta.tar.Z
tar xvf FM0.9beta.tar
```

depending on the type of the downloaded compressed file.

You should end up with the directory structure shown in Figure 2.1 (assume that FM0.9beta stands for your installation directory)

Setting up FM

Next set the PATH and CLASSPATH environment variables as follows:

```
setenv CLASSPATH ${CLASSPATH}:${FM0.9_HOME}:${FM0.9_HOME}/FM0.9beta.jar:
                 ${FM0.9_HOME}/trace:${FM0.9_HOME}/agentlib

setenv PATH ${PATH}:$FM0.9_HOME/scripts
```

These are the minimum settings required to get FM ready to start.

If you intend to connect a MySQL database to FM , then the following settings must be applied instead:

```
setenv CLASSPATH ${CLASSPATH}:${FM0.9_HOME}:${FM0.9_HOME}/FM0.9beta.jar:
                 ${FM0.9_HOME}/trace:${FM0.9_HOME}/agentlib:
                 ${JDBC_MYSQL_HOME}
```

```
                |-- agentlib
                |
                |-- config
                |
                |-- data
                |
                |-- database
                |
                |-- doc
                |
                |-- html --|-- templates
                |
FM0.9beta ------|-- images
                |
                |-- log
                |
                |-- plot
                |
                |-- scripts
                |
                |-- td
                |
                |-- templates
                |
                |-- trace -|-- data
                           |
                           |-- demo
                           |
                           |-- fsm
                           |
                           |-- images
```

Figure 2.1: FM directory structure

```
setenv  PATH ${PATH}:$FM0.9_HOME/scripts:${MYSQL_HOME}:${MYSQL_HOME}/bin
```

where MYSQL_HOME stands for the installation directory of MySQL and JDBC_MYSQL_HOME stands for the installation directory of the MySQL JDBC driver.

Moreover, notice that if you intend to use the FM database option of the main menu in order to browse the database contents, then you must have previously installed XMySQL, and then set the environment variables as follows:

```
setenv CLASSPATH ${CLASSPATH}:${FM0.9_HOME}:${FM0.9_HOME}/FM0.9beta.jar:
                 ${FM0.9_HOME}/trace:${FM0.9_HOME}/agentlib:
                 ${JDBC_MYSQL_HOME}

setenv  PATH ${PATH}:$FM0.9_HOME/scripts:${MYSQL_HOME}:${MYSQL_HOME}/bin:
             ${XMYSQL_HOME}
```

Finally, and very importantly, you should execute

```
cd $FM0.9_HOME (goes to the FM installation directory)
java setup
```

2.2.3 Running FM

After installing the FM software, you are done. For example, if FM is installed at /home/fmuser, then:

```
    Type:  /home/fmuser/FM0.9beta/fm
```

Figure 2.2: FM Main Menu

or

```
Add  /home/fmuser/FM0.9beta to your path statement
Type:  fm
```

Then the FM main menu of Figure 2.2 will be displayed.

The following chapter will help you get familiarized with the features of the system by taking you on a tour through FM.

Chapter 3

Getting Started

The purpose of this chapter is to get you familiarized with the FM environment by taking you on an easy, quick, though significant enough, tour. During this tour, you will learn about the capabilities of FM by turning yourself into a contender who competes against some of the agents built out from the FM agent library.

First of all, you must start FM following the guidelines in section 2.2.3.

3.1 Configuring the Experimental Environment

By means of the FM configuration panel (see Figure 3.1), you, the tournament designer, can specify some general features:

- where (machine and port) the auction house is to be running,
- whether the tournament is expected to be monitored, and

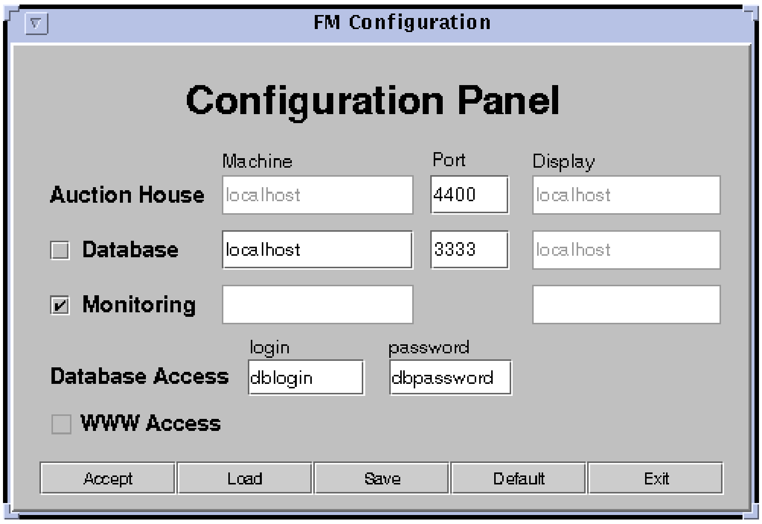


Figure 3.1: Tournament Configuration Panel

- whether the tournament data are to be stored in a database and, if so, where (machine and port) that database will be running[1].

For this very first tournament, simply press *Accept* to validate the default configuration.

3.2 Tournament Scenario Definition

Once FM has been appropriately configured, you must engage in the definition of tournament scenarios. For this purpose, start selecting the *Parameter Setting* option of the FM main menu. Figure 3.2 shows the default tournament definition that is expected to be deployed on your screen.

Next we highlight the most remarkable features of this default tournament scenario:

[1]Section A.1 discusses the current restrictions of this capability.

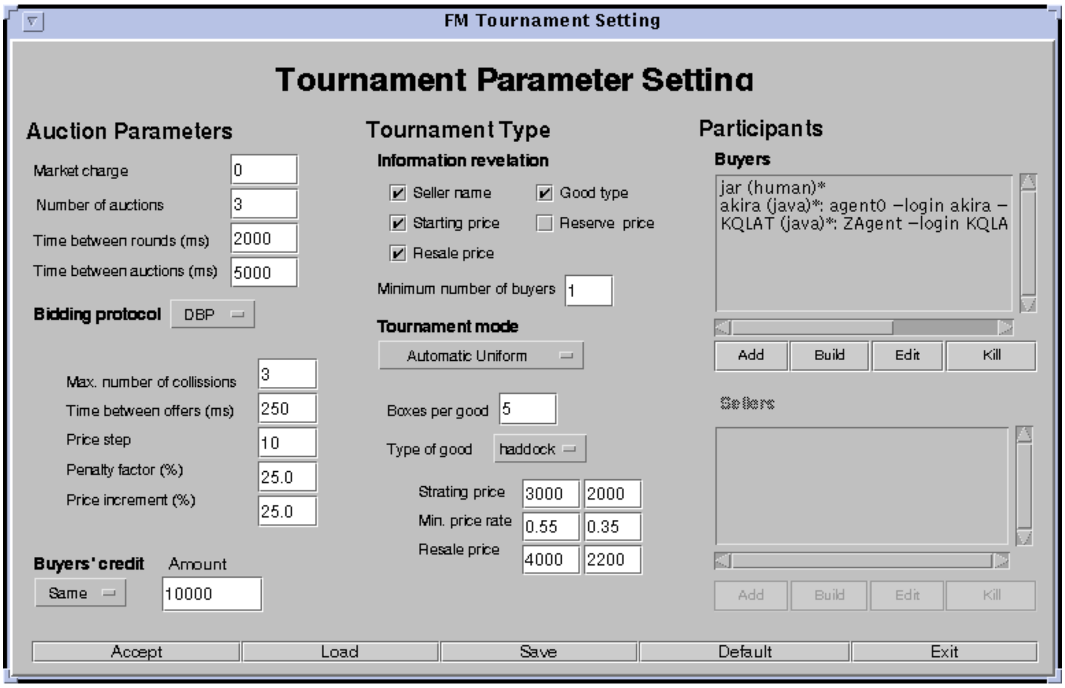


Figure 3.2: Tournament Parameter Setting

- **Auction parameters**. This scenario specifies a 3-auction tournament in which participants are not charged for participating in each bidding round (the *market charge* is 0). At least 5000 milliseconds separate two consecutive auctions, while 2000 milliseconds separate two consecutive bidding rounds, and 250 milliseconds two consecutive price quotations (prices called by the auctioneer). With a chosen good the auctioneer will open a bidding round employing the *Downward Bidding Protocol (DBP)* to auction it, thus quoting offers downward (decreasing by 10 units from offer to offer) from the good's starting price down to the good's reserve price. During the round, the auctioneer will not allow more than 3 consecutive collisions[2] to come about. In this case the good will be randomly sold to the buyers implied in the collision. Finally, the values of the *penalty factor* and *price increase* will determine the penalty imposed on those agents submitting unsupported bids, and the new starting price of the round when either a penalty, expulsion or

[2]A collision occurs when the auctioneer receives more than one bid at the same price.

type	*#Boxes*	*starting*	*resale*	*reserve*
cod	$U[1,5]$	$U[1200,2000]$	$U[2100,2400]$	$U[0.4,0.5]$
tunafish	$U[1,5]$	$U[800,1500]$	$U[1200,2500]$	$U[0.3,0.45]$
prawns	$U[1,5]$	$U[4000,5000]$	$U[4500,9000]$	$U[0.35,0.45]$
halibut	$U[1,5]$	$U[1000,2000]$	$U[1500,3500]$	$U[0.4,0.6]$
haddock	$U[1,5]$	$U[2000,3000]$	$U[2200,4000]$	$U[0.35,0.55]$

Table 3.1: Automatic Uniform Mode Lot Generation

collision comes about.

- **Buyers' Credit**. All buyers have the same amount of money ($10000 ptas.$) at the beginning of each auction (the same per auction credit) composing the tournament.

- **Information revelation**. During each bidding round, FM will convey to the buyers information about the seller, the type, the starting price, and the expected resale price of the good in auction. Besides, the identity of the participants will be also made public.

- **Tournament Mode**.

  The *Automatic Uniform* tournament mode indicates that the lots of goods to be auctioned during each auction are generated by the auction house itself according to uniform distributions that the tournament designer must define for each type of good. Table 3.1 shows the default uniform distributions used by FM to construct lots of goods

  In this way, the composition of each lot will vary from auction to auction. In section 4.1 we introduce an alternative tournament mode, *One auction* which allows for a same lot of goods to be repeatedly auctioned.

- **Participants**. Looking at the *Buyers* window within the *Participants* area, we observe that three agents are defined by this tournament scenario: two software, Java, agents(*akira* and *KQLAT*) and a human agent (*jar*). All of them will be spawned automatically by FM (denoted by the $*$ symbol accompanying each agent definition in the *Buyers* windown) in the same machine where the auction house will be started off.

  No sellers are involved in this scenario.

Press on the *Accept* button to confirm this initial setting.

The following chapter will deal with more advanced features such as:

- how to build and incorporate your own agents,
- how to build agents out of the FM agent library,
- how to distribute the participants in different machines, and
- how to make FM work in multi-user mode.

3.3 Start

Simply press the *Go* option of the FM main menu. Figure 3.3 shows the tournament panel that is expected to be displayed.

Then it is your turn, as the only human participant, to enter the tournament. Figure 3.4 shows you the GUI that you will be using to interact with FM during the tournament. It is worth spending some time identifying the many areas of information in it so that you do not get lost when the tournament starts. To request for being let in, simply press on the *Auction Room* button, and next type your password (also *jar*, like your login name) into the *Agent Login* window which appears on your interface, and validate it by pressing *Accept*. Ideally you will be accepted in the auction house, and from that moment on you will be receiving the tournament information as illustrated by Figure 3.5. Then you can easily interact with FM through the buyer GUI to either submit bids, or move out into the settlements office, or moving back into the auction room, or update your credit, or finally leave the auction house.

Figure 3.3: Tournament Panel

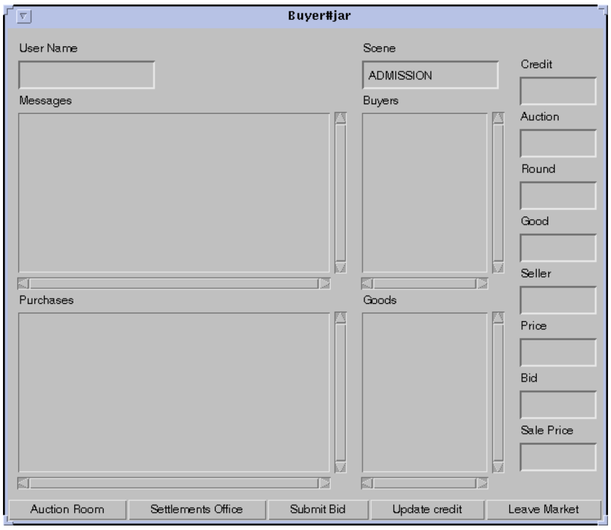


Figure 3.4: Human Buyer GUI

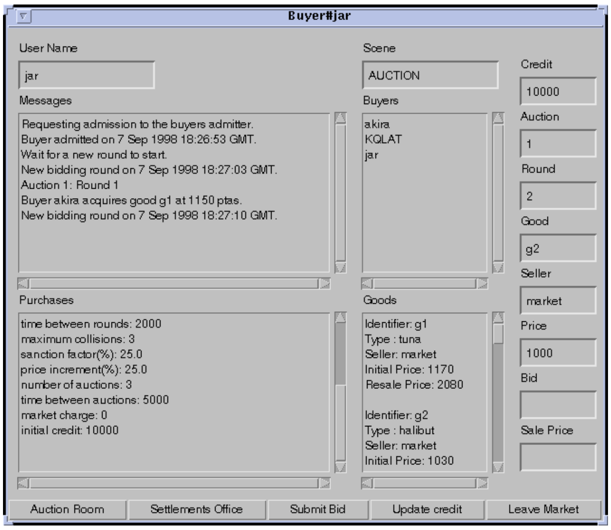


Figure 3.5: Human Buyer GUI. Tournament Information.

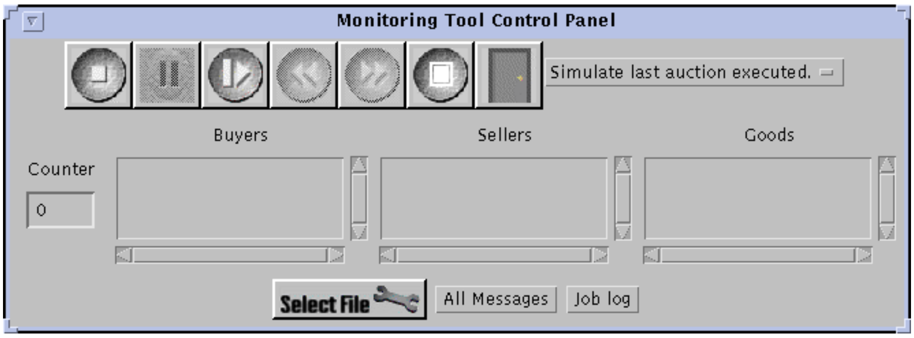


Figure 3.6: *Monitoring Agent Control Panel*

3.4 Auction Monitoring

This section briefly and swiftly sketches out some of the monitoring capabilities offered by FM.

3.4.1 The Monitoring Agent

To start off the FM *Monitoring Agent*, select the *Monitoring* option of the FM main menu. Figure 3.6 displays the video-like panel that will be showed to allow to control the FM *Monitoring Agent*in a fairly easy way.

All you have to do, for the moment, in order to start *playing* the ongoing tournament session is to press the *play* button and watch. Figure 3.7 shows the graphical representation ot the so-called virtual FM. This (market-centered) representation method appears to be convenient for the tournament designer to visualize all the activities occurring distributedly and concurrently in FM in a global manner.

Apart from making this global view of FM available, the *Monitoring Agent*is also capable of displaying individualized (agent-centered) information referred to both the illocutions sent and received by each agent. In order to obtain this local information, simply click on the icon representing

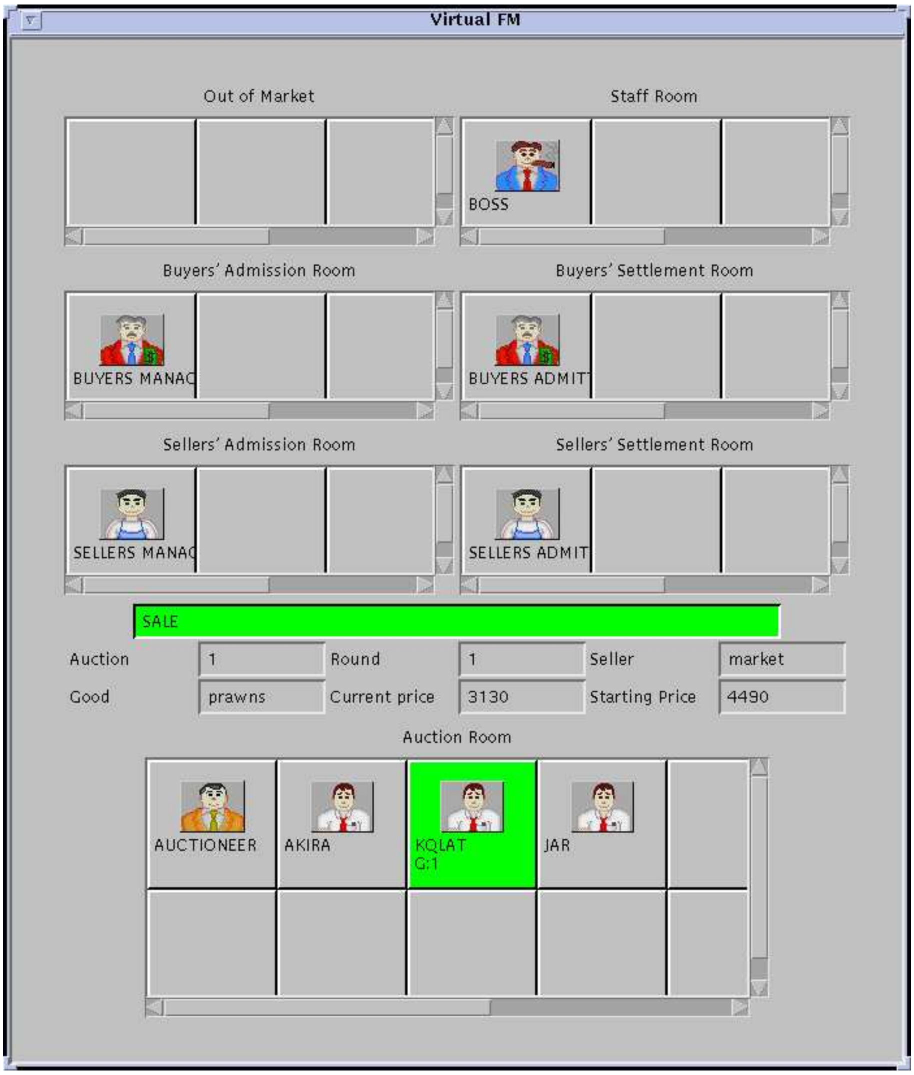


Figure 3.7: Virtual FM

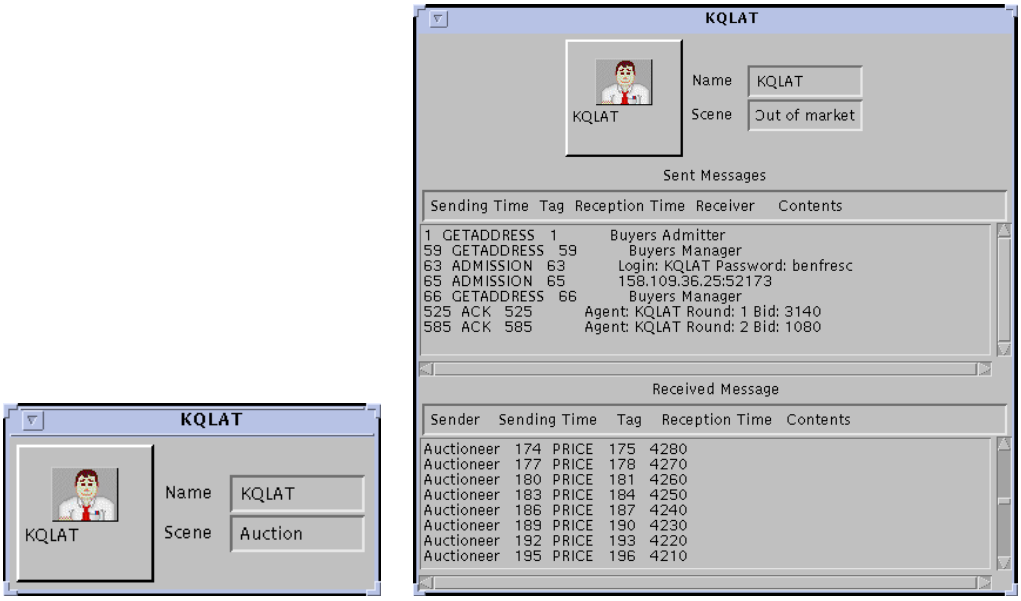


Table 3.2: Agent Card and Agent Illocutions

the agent to be observed within the virtual FM, and next a window similar to that on the left in Table 3.2 will be ideally showed.

Clicking again on the agent's icon lets us browse all the messages being sent and received by the selected agent as shown by the figure on the right in Table 3.2.

Chapter 4 deeply analizes the full functionality of the *Monitoring Agent*.

3.4.2 Displaying the Tournament Evolution

As to obtain the visual representation of the evolution of the competitors along the tournament offered by *Gnuplot*[3], you must proceed as follows:

```
cd $FM0.9_HOME/plot (goes to the plot directory)
```

[3]**Caution:** You must have previoulsy installed *GnuPlot*.

```
gnuplot
```

Once started *Gnuplot* type the following command

```
gnuplot> load ``evaluation.dem''
```

This is expected to bring about a display of the tournament evolution similar to that of Table 3.3. In this way you can conveniently keep track of the agents' performance during the tournament.

Refer to section 4.3.3 for more details about the plotting capabilities of FM.

3.5 Starting Over

FM allows to run a tournament once defined over and over. All you need to do is to close the *Tournament Pannel* window, and then click on the *Go* button of the FM main menu.

Caution: The closing of the *Tournament Pannel* window before either the tournament is over or the tournament has been closed[4] will surely cause an abnormal termination of the current tournament session.

[4]This is done manually by the tournament designer by pressing the *Close Market* button in the upper right corner of the *Tournament Pannel* window.

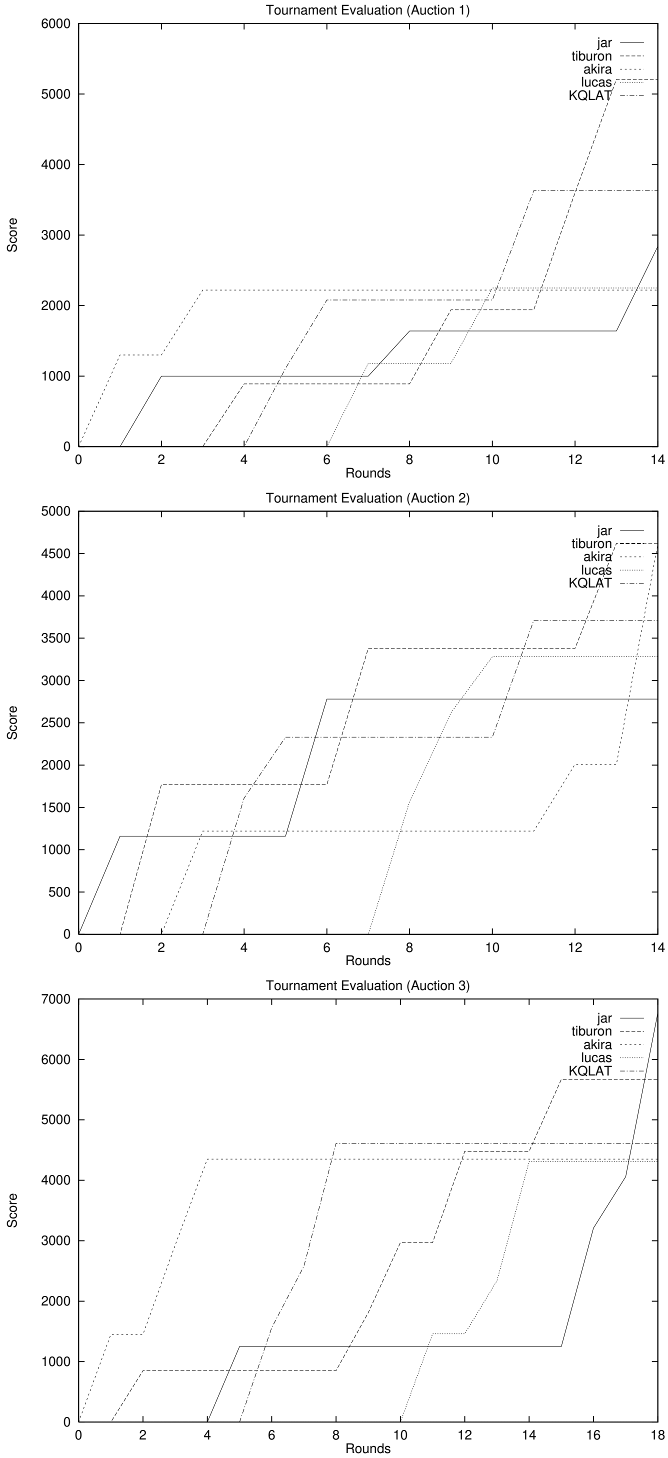


Table 3.3: Example of Tournament Evolution

Chapter 4

Advanced Features

4.1 Tournament Modes

In chapter 3 the *Automatic Uniform* tournament mode was utilized to illustrate the workings of FM. Alternatively, in the current version the tournament designer migh opt for the *One Auction (data set)* mode, which allows for the repeated auctioning of a unique set of goods. Such option must be enabled in the *Tournament Parameter Setting* panel. Next you will be requested to specify a file containing the description of the composition of the lot on sale. Its typical format is exemplified by the contents of the built-in *goods.txt* file[1] provided by Figure 4.1.

Once defined the lot (data set) file, only the related matter of specifying the information revelation level to be applied on the lot remains.

[1] This file is situated in the *data* directory within the FM installation directory. This is the default file showed when choosing the *One Auction (data set)* tournament mode.

```
##################################################################
#                                                                #
# FM                                                             #
# Version:     0.9beta                                           #
# File:        goods.txt                                         #
# Description: This file contains the description of the goods  #
#              composing a lot.                                  #
#                                                                #
##################################################################

# Good descriptor:
# <goodId,goodType,sellerName,startingPrice,reservePrice,resalePrice>

g1,tuna,David,2300,1500,2500
g2,cod,David,2300,1500,2500
g3,haddock,David,2300,1500,2500
g4,halibut,David,2300,1500,2500
g5,tuna,David,2100,1200,2500
```

Figure 4.1: Example of a lot file

4.2 How to Build Agents

4.2.1 Using the Agent Library

FM has got a built-in agent library which contains agents with simple, parametrizable auction strategies. In order to add agents from the agent library to a tournament scenario, select the *Build* option in the *Buyers* area lying down on the *Tournament Parameter Setting* panel. This will display the GUI in Figure 4.7 that will allow you to select the type of agent to be included and its parameters (if needed). Notice that it is not compulsory to set the agent's parameters since all agents are designed to use default parameters.

In the following subsections we list the currently available types of agents along with their bidding strategies:

Agent0

Agent0 calculates his bid bid_0 for each round as follows

$$bid_0 = \frac{\sum\limits_{b \in \mathcal{B}} \kappa(b)}{\sum\limits_{g \in \mathcal{UG}} p_{rsl}(g)} p_{rsl}(g_r)$$

where

- $\sum\limits_{b \in \mathcal{B}} \kappa(b)$ market money (sum of the credits of each buyer b, $\kappa(b)$)
- $p_{rsl}(g_r)$ resale price p_{rsl} of the good in auction g_r
- $\sum\limits_{g \in \mathcal{UG}} p_{rsl}(g)$ market benefits (sum of resale prices of the goods remaining to be auctioned $\mathcal{UG}$)

Though utilizing a very simple strategy, this type of agent showed an excellent performance w.r.t the contenders in the second FM tournament [7].

NAgent

Let us define

$$k = sin(r\frac{\pi}{n}) - \gamma$$

where r and n stand respectively for the current round index and current auction length(total of bidding rounds). Then the bid submitted by the agent during the next bidding round will be calculated as

$$bid_N = \begin{cases} 1 + \frac{k\beta}{\gamma} bid_0 & k \leq 0 \\ bid_0 + \alpha(p_0(g_r) - bid_0)\frac{k}{\gamma} & \text{otherwise} \end{cases}$$

where $p_0(g_r)$ stands for the starting price of good g_r, and α, β, and γ are free variables that must become instantiated.

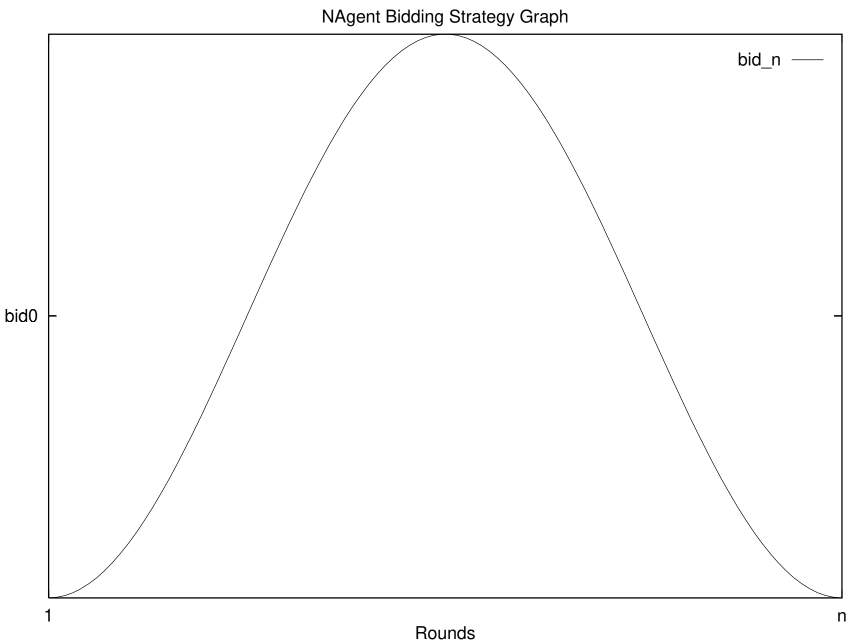


Figure 4.2: NAgent Bidding Strategy Profile

SAgent

SAgents calculate their bid bid_S for each round as follows

$$bid_S = \begin{cases} bid_0 + \alpha(p_0(g_r) - bid_0)\frac{atan(\frac{r-\frac{n}{\gamma}}{\frac{n}{\gamma}})}{\frac{\pi}{\gamma}} & r \geq \frac{n}{\gamma} \\ (1 + \beta\frac{atan(\frac{r-\frac{n}{\gamma}}{\frac{n}{\gamma}})}{\frac{\pi}{\gamma}})bid_0 & \text{otherwise} \end{cases}$$

where α, β, and γ are free variables that must become instantiated.

PAgent

PAgents calculate their bid bid_P for each round as follows

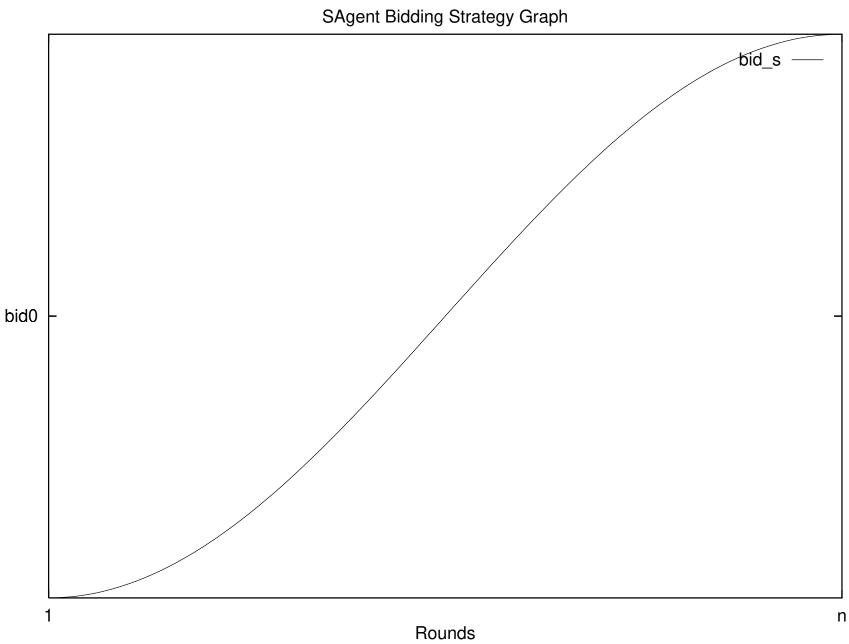


Figure 4.3: SAgent Bidding Strategy Profile

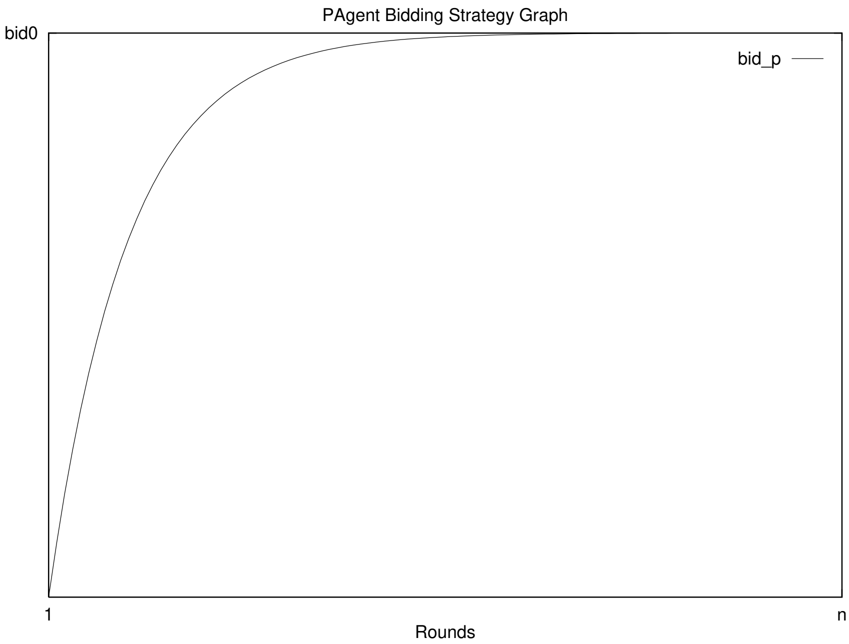


Figure 4.4: PAgent Bidding Strategy Profile

$$bid_P = \begin{cases} (1 - e^{\gamma \frac{r}{n} + log(1-\alpha)}) bid_0 & r \leq \frac{n}{\beta} \\ bid_0 & \text{otherwise} \end{cases}$$

where α, β, and γ are free variables that must become instantiated.

FAgent

FAgents calculate their bid bid_F for each round as follows

$$bid_F = \begin{cases} bid_0 + \alpha(p_0(g_r) - bid_0) \frac{atan(\frac{r - \frac{n}{\beta}}{\frac{n}{\beta}})}{\frac{\pi}{\beta}} & r > \frac{n}{\beta} \\ bid_0 & \text{otherwise} \end{cases}$$

where $\alpha and \beta$, are free variables that must become instantiated.

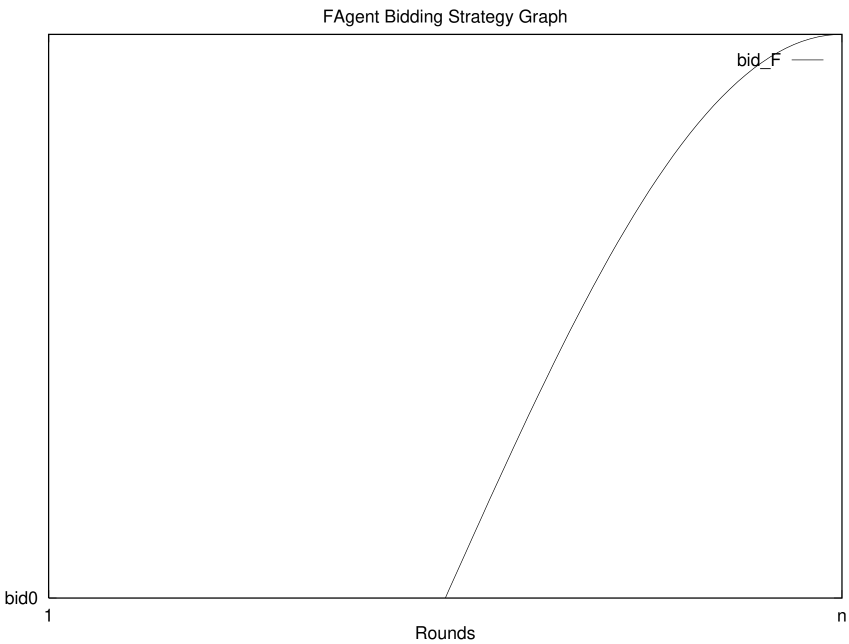


Figure 4.5: FAgent Bidding Strategy Profile

ZAgent

ZAgents calculate their bid bid_Z for each round as follows

$$bid_Z = \begin{cases} bid_0 - \alpha(p_0(g_r) - bid_0)\frac{atan(\frac{r-\frac{n}{\gamma}}{\frac{n}{\gamma}})}{\frac{\pi}{\gamma}} & r \le \frac{n}{\gamma} \\ (1 - \beta\frac{atan(\frac{r-\frac{n}{\gamma}}{\frac{n}{\gamma}})}{\frac{\pi}{\gamma}})bid_0 & \text{otherwise} \end{cases}$$

where α, β, and γ are free variables that must become instantiated.

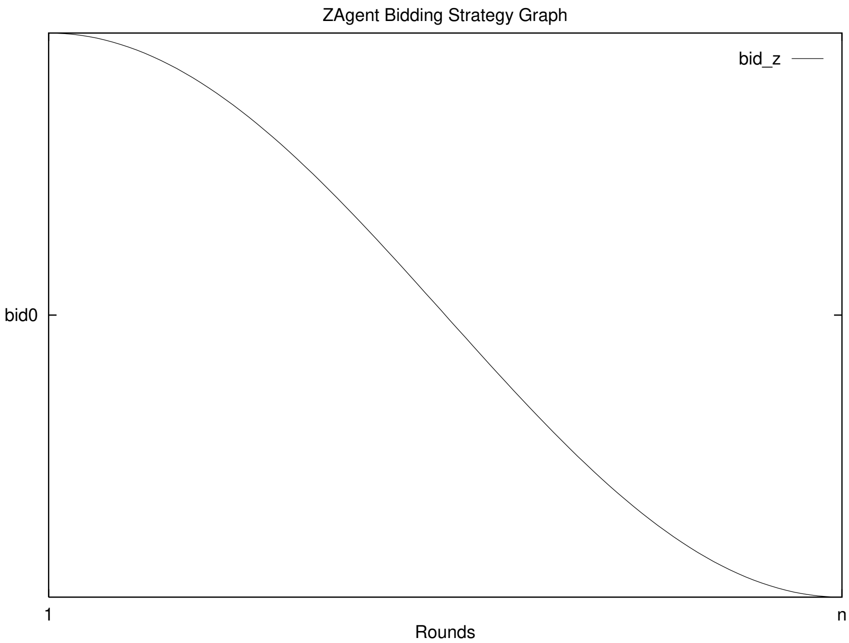


Figure 4.6: ZAgent Bidding Strategy Profile

LAgent

This type of agent simply calculates his bid as

$$bid_L = \alpha p_0(g_r)$$

where $0 \leq \alpha \leq 1$

RAgent

RAgent is a random agent. It obtains its bid randomly from the starting price.

TFTAgent

This is a kind of *Tit For Tat* agent that calculates his bid as

$$bid_{TFT} = \frac{p_{rsl}(g_r)}{1 + \frac{p_{rsl}(g_{r-1}) - p_\omega(g_{r-1})}{p_\omega(g_{r-1})}}$$

where $p_\omega(g_{r-1})$ stands for the sale price of the good sold during the $r-1$ bidding round.

4.2.2 Using the Agent Templates

FM has been equipped with a library of agent templates intended to help agent programmers to construct their own agents in either *Java, C,* or *Lisp.* The templates' documented code is available in the *templates* directory, and very interestingly, a set of agents' examples constructed out of these templates can also be found in the *agentlib* directory. These examples will surely constitute a good starting point for those who dare to program their agents. We must also mention that, in the particular case of the *Java* template, apart

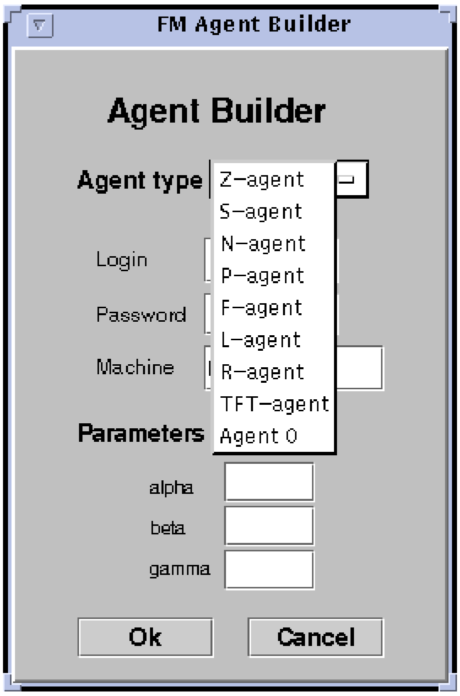


Figure 4.7: Agent Builder

from the source code, the *javadoc*-generated documentation of its package is also available in the *templates* subdirectory of the *html* directory[2].

Before proceeding any further, we must stress the fact that the main goal pursued when designing the FM templates was to ease the task of programming new agents as much as possible. From this follows that agent builders can almost exclusively concentrate on the design of bidding strategies.

Next we go through the steps to be followed for adding a new agent once constructed.

First of all, open the *Tournament Parameter Setting* panel, and select the *Add* option in the *Buyers* area. That will display the GUI in Table 4.1.

[2]This documentation can be accessed through a browser by opening the *packages.html* file.

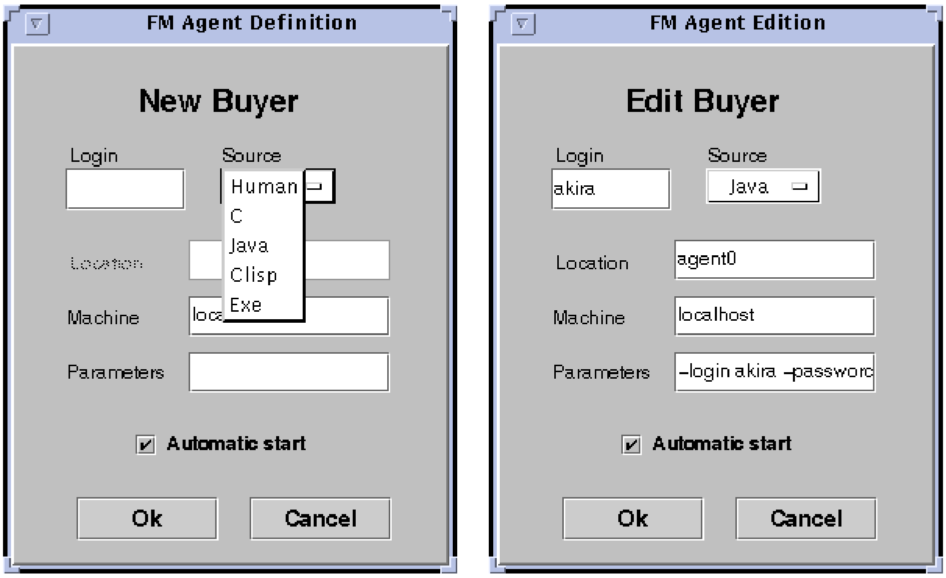


Table 4.1: Addition and Edition of Buyers.

At this point you must specify

- **Source.** Whether the agent is human (i.e. an agent that will participate through the buyer GUI) or it is a software agent (*Java, Lisp, C, Exe*[3]).
- **Login.**
- **Location.**
  - Unnecessary for human agents.
  - Full path for Lisp, C and Exe agents (f.i. /home/fmuser/agentlib/CBuyer).
  - Only class name for *Java* agents, but remember that the class name path must be added to the CLASSPATH environment variable.
- **Machine.** The machine where the agent will be spawned. This is unnecessary for human agents too, but quite useful for sotware agents

[3] *Exe* stands for executable. In practice C and Exe agents are handled in the same way by FM.

because it allows to distribute your agents. Keep in mind, however, that in order to succeed in a remote connection you must have been guaranteed access to the remote machine (being a user), and, moreover, your environment variables must be set as explained through Section 2.2 . Notice also that solely a single user can profit from this feature to distribute his agents. In Section 4.4 we give the details about how to carry out a multi-user distribution.

- **Parameters** needed by the software agent.
- **Automatic/Manual Start.** Tick this option to request FM for automatically starting the agent. In order to guarantee that Lisp and *Java* agents are correctly spawned, your path environment variable must include the path to your *Java* and Clisp[4] executable interpreters.

  The manual start of agents is specially convenient for tournaments involving humans and software agents owned by other users[5] or simply agents that cannot be started automatically by FM(f.i. agents developed with the aid of an interpreted language). Anyhow the tournament (experiment) designer needs only to specify the participants' logins so that FM knows which agents are expected to request for participation. Complementarily, refer to Section 4.4 for a description of how different users' agents can remotely participate in tournaments.

4.2.3 Editing and Removing

The features of each one of the participants can be conveniently redefined by the tournament designer. For that purpose, select with your mouse the line containing the participant to be reconfigured, and then click on the *Edit* button in the *Buyers* area of the *Tournament Parameter Setting* panel. That will show the window in Figure 4.1.

[4]The FM Lisp template is Clisp-compliant, and furthermore FM explicitly calls Clisp when automatically starting Lisp agents.

[5]We consider that other users –apart from the tournament designer– are to run their agents in their accounts, in their machines, with their resources.

Once selected, an agent can be removed by the participants list simply by clicking on the *Kill* button in the *Buyers* area of the *Tournament Parameter Setting* panel

4.2.4 Registering Participants

The *password.txt* file in the data directory contains the list of logins and passwords that FM will admit as valid during a tournament session. This is an editable file that the tournament designer can congigure as needed in order to include or exclude agents. Importantly note that those lines beginning with the # character will be regarded as comments and therefore ignored.

4.3 Auction Monitoring

4.3.1 Database Access

If you intend to connect a *MySQL* database to FM, the tournament database must be firstly created. For this purpose, start *MySQL* and type

```
mysqladmin create tournament
```

After this, your tournament database will have been created. There only remains to create the tables composing such database so as to get it ready for FM connections. This operation can be easily achieved using the SQL table creator script in the *database* directory. Simply typing

```
mysql tournament < CreateTournamentTables.sql
```

from your command line will do the job without even needing to start mysql.

Finally, if you decide to eliminate the database, executing

```
mysql tournament < DeleteAllData.sql
```

will delete all the tournament tables (*DeleteAllData.sql* can be also found in the *database* directory). And complementarily the dropping of the database by executing

```
mysql > DROP DATABASE tournament;
```

will end up with the database removal.

Finally, we must say that FM offers the *Database* option in its main menu. This allows the connection of FM to *XMySQL* for the visualization of the tournament database. Evidently, *XMySQL* must have previosuly installed following the guidelines in Section 2.2.

4.3.2 The Monitoring Agent

The FM *Monitoring Agent* GUI is composed of two parts:

- **Virtual FM.** It offers a global, visual representation of both the agent flow (from scene to scene) and the (intra-scene and inter-scene) agent communication flow during a tournament session taking place in the auction house.
- **Control panel.** It allows an easy interaction with the *Monitoring Agent* thanks to its design, highly inspired on a video player remote controller so as to help the user identify the available functions by analogy.

The next sections will attempt at presenting both components in some more detail.

The Virtual FM

Following [1], the fish market can be described as a place where several *scenes* take place simultaneously, at different locations, but with some causal continuity. The main 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, at 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.

The FM electronic auction house is based on this traditional fish market metaphor. In a highly mimetic way, the workings of FM also involve the concurrency of several scenes governed by the market intermediaries identified in the fish market (*boss, buyer admitter, buyer manager, auctioneer, seller admitter, seller manager*). Therefore, seller agents register their goods with a seller admitter agent, and can get their earnings (from a seller manager) once the auctioneer has sold these goods in the auction room. Buyers, on the other hand, register with a buyer admitter, and bid for goods which they pay through a credit line that is set up and updated by a buyer manager. Buyer and seller agents can trade goods as long as they comply with the *institutional* —auction market— conventions.

The *Virtual FM* attempts at offering a visual representation of the distributed workings of FM. Hence it has been divided into a set of scenes (*out of market, buyers' admission, sellers' admission, auction room, buyers' settlements,* and *sellers' settlements*) corresponding to their counterparts in

FM. Within this setting, two representation levels can be differentiated:

- **Global (Macro).** Represents the agent flow (i.e., the flow of agents from scene to scene) and the events taking place during a bidding round.
- **Local (Micro).** Represents two levels of agent communication:
  - *Intra-scene.* Illocutions exchanged between the traders (buyers or sellers) placed in a scene and the market intermediary governing that scene.
  - *Inter-scene.* Illocutions exchanged between market intermediaries situated within different scenes.

  Anyhow, each agent's local information can be easily displayed by clicking on the icon representing the agent to be observed, and then a window similar to that on the left in Table 3.2 in page 24 will be ideally showed. Clicking again on the icon lets us browse all the messages sent and received by the selected agent as shown by the figure on the right in Table 3.2.

During each bidding round composing an auction of a tournament session, several events may come about: bids, collisions, sales, etc. These events are graphically represented by illuminating with different colours the icons of the agents involved in a given event. Find below the list of colours employed and their associated meanings:

1. **Purple.**: Bid submission. The agent's icon is labelled with a message of the type "B: price" accounting for the bid already submitted;
2. **Red.**: Penalty caused by an unsupported bid submission (the agent did not have enough credit to support his bid). The agent's icon is labelled with a message of the type "F: penalty;
3. **Green.** Sale. The agent acquires the good in auction and his icon is labelled with the amount of accumulated purchases "G: purchases".

4. **Yellow.**The agent takes part in a collision.

Also note that no buyer is illuminated when the auctioneer declares the good on sale out of market (because its reserve price has been reached). Nonetheless, the *Monitoring Agent* does signal this event.

In what follows, the interaction through the *Monitoring Agent* configuration panel with the *Virtual FM* is explained in detail.

Control Panel

Before starting using the *Monitoring Agent*, it should be configured. So firstly select the *Monitoring* option of the FM main menu in order to activate the *Monitoring Agent Control Panel*(see Figure 3.6 in page 22).

Next press the *Select File* button on the *Control Panel* to display the *Monitoring Agent Control Panel* in Figure 4.8.

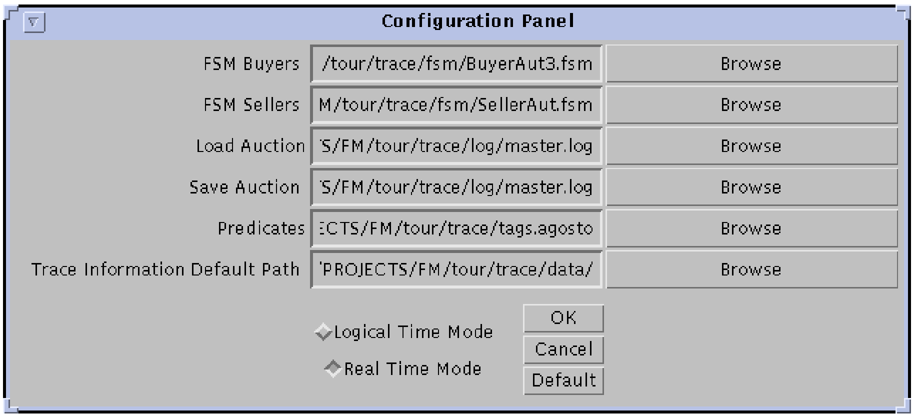


Figure 4.8: *Monitoring Agent* Configuration Panel.

Among the many configurable parameters, for the current version only two of them are worth considering:

- **Trace Mode.** The tournament designer can choose between the *logical time mode*, for finished market sessions, and the *real time mode*, for ongoing market sessions[6].

- **Trace Information Location.** The trace information produced by both the market agents and interagents is stored, by default, in the subdirectoy *data* of the *trace* directory. However, you could opt for keeping the trace information corresponding to different tournament sessions separated, in different directories. Then you must specify the location of the trace informacion that you intend to trace. For instance, the subdirectory *demo* of the *trace* directory contains the trace information of a tournament session that displays all the possible events that might take place[7].

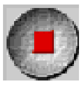
Once the *Monitoring Agent* has been configured, you can start playing about with it by means of the buttons lying down on the *Monitoring Agent Control Panel*. The overall functionality provided by these buttons is highly similar to that of a video player remote controller. Next, we make such functions explicit:

[6]Actually the subsequent versions of FM will incorporate a more sophisticated trace mode— currently being tested— that sorts out the inherent drawbacks of logical time-based tracing methods.

[7]Jump up to the previous section for a complete account of these events.

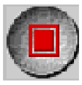
 *Play* the auction tournament.

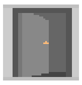
 Pause. The monitoring process pauses till the *Play* button is pressed again.

 Stop Monitoring. When restarting the monitoring process, this runs right from its very beginning.

 Forward. Disabled in the current version.

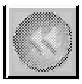
 Rewind. Disabled in the current version.

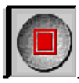
 Record the auction tournament to be played.

 Exit.

These *Control Panel* buttons will show the following possible states:

 Active. The button is available and so can be pressed.

 Inactive. The button is unavaiable.

 On. The action is actually taking place. This example illustrates how the *Record* button looks when recording a tournament.

Also notice that not all buttons can be active at the same time because of incompatibility reasons:

- When pressing the *stop* button, the rest of buttons become active except the *pause* button.
- The *record* button can be solely active before starting off playing a market session. Therefore, to record a session this button must be pressed beforehand.

- Pressing *pause* temporarily blocks the ongoing market session reproduction. Simply press *play* to continue.

Observe that nex to the *Control Panel* buttons there is a choice menu with the *Simulate last auction executed* option as the default choice. This menu allows you to select the auction tournament to be played out of the list of recorded tournaments and the last tournament executed (see Figure 4.9 for a sample of list of selectable auctions).

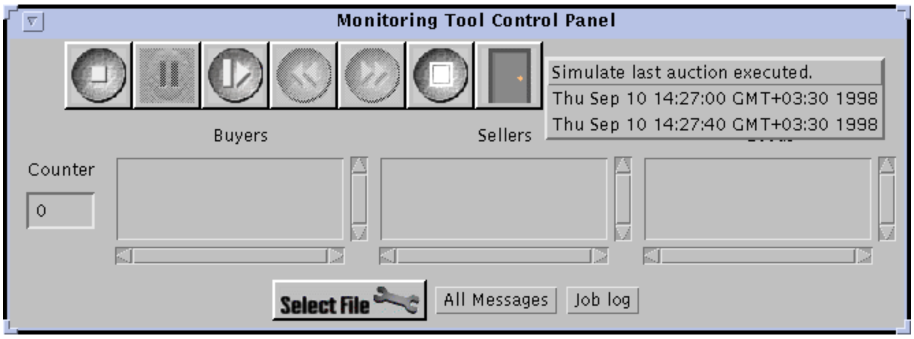


Figure 4.9: *Monitoring Agent* List of Selectable Auction Tournaments.

When choosing a recorded tournament session to be played, the information characterizing the auction will be showed on the windows making up the so-called *information area* of the *Monitoring Agent Control Panel*as shown in Figure 4.10:

- **Counter.** Displays the logical time as the tournament progresses
- **Buyers.** Displays the buyers that took part in the tournament.
- **Sellers.** Displays the sellers that took part in the tournament.
- **Goods.** Displays the goods on sale.

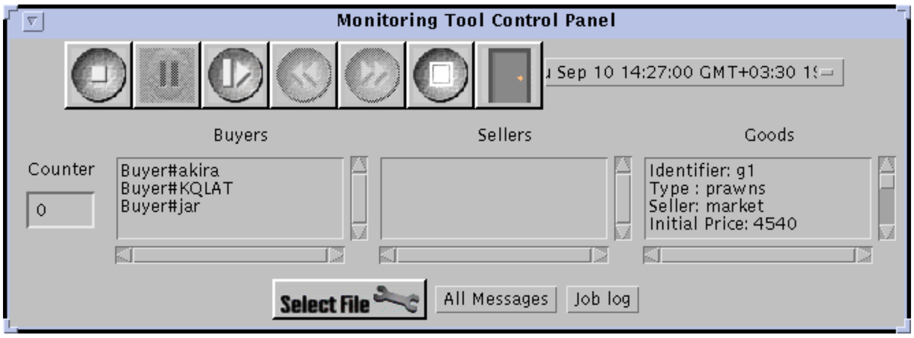


Figure 4.10: *Monitoring Agent Control Panel* Information Area.

4.3.3 Representing the Agents' Performance Evolution

During the tournament, FM keeps track of the performance value (benefits) of each contender. These values are stored in a series of raw data files named *auction#xxx.dat*[8] in the *plot* directory. FM also creates a file named *evaluation.dem* that can be employed by *Gnuplot*[9] in order to produce a visual representation of the evolution of the competitors along a tournament. Needless to say that other tools might alternatively be utilized to interpret these data files.

Interestingly, when using *Gnuplot* you can easily generate postcript representations of the evolution of the competitors by editing *evaluation.dem* and uncommenting some command lines wherever it is indicated.

Section 3.4.2 shows how you must proceed in order to obtain the visual representation offered by *Gnuplot*.

We must remind you that all the relevant data yielded during a tournament session are also available from the FM database whenever this capability has been enabled.

[8] *xxx* stands for a given auction number.

[9] Gnuplot is a command-driven interactive function plotting program that can be downloaded from the GNU web page (see 2.2.1 for more details)

4.4 FM Runtime Environment

FM is multi-user. It allows multiple users to spawn their agents in their own machines in order for these to participate in remote tournaments. Those users intending to connect their agents to FM must have been previously installed FMRE (FM Runtime Environment), the minimum subset of FM required for managing the interaction between a remote agent and FM.

What happens if you intend to run tournaments which involve not only your own agents but other users' agents? People might feel reluctant to hand in their agents for you to snoop about. You could think of letting these other users to participate in your tournaments but, and very importantly, running their agents in their machines, i.e., consuming their own resources. What you must do is to enforce those intending to connect their agents to your tournaments to have previously installed FMRE(the FM Runtime Environment), the minimum subset of FM required for managing the interaction between remote agents and FM. Therefore, the use of FMRE will allow multiple users to spawn their agents in their own machines in order for these to participate in remote tournaments.

4.4.1 How does it work?

As to the participants spawning software agents, they must make explicit where the auction house is by hand. That is simply achieved by executing

```
java prepareConnectionTo aucitonHouseHost auctionHousePort
```

where *auctionHouseHost* and *auctionHousePort* stand respectively for the host and port where the auction house server is said to be running. After this, the remote participants are requested to start their agents by hand.

As to human participants, each one of them must execute

```
java Buyer -machine auctionHouseHost -port auctionHousePort -buyerId myLogin
-trace [on|off] -credit [myCredit]
```

in order to obtain the GUI that allows a human buyer to interact with the auction house as seen in Section 3.3 of Chapter 3.

Appendix A

Miscellanea

A.1 Restrictions and Future Releases

Several restrictions apply to the current version of FM:

- **Platforms**. FM works exclusively under Unix and Linux.
- **"Centralized" Auction House**. Though the auction itsef can be run in a distributed manner (each agent intermediary composing the market institution[1] could be spawned in a different machine), the current version does not exploit this capability and thus runs the whole auction house on the same machine. Notice that this restriction has been solely motivated by our desire to keep things simple when possible.
- **Bidding Protocols**. FM implements the Dutch bidding protocol
- **Tournament Modes**. Those modes involving sellers have not been included in this version.
- **Database**. FM is prepared to be connected to a MySQL database.

[1] The interested reader should refer to [3] for a thorough account of the architecture of FM.

- **Local trace**. FM monitoring capabilities reduce to LAN tournaments.

The FM team is committed to delivering constant improvements to the FM software.

While the current version of FM is well-tested, the newer versions of FM available on our Web site tend to have better performance, improved scalability, more bug fixes, and occasionally, new features over older versions.

Many new features are currently under development for the forthcoming release, FM1.0:

- **Platforms**. Support for Windows.
- **Distributed Auction House**. Possibility of distributing the agent intermediaries composing the auction house.
- **Bidding Protocols**. Implementation of FPBS, Vickrey and English.
- **Tournament Modes**. Inclusion of tournament modes involving sellers.
- **Database**. Possibility of selecting the database to connect to (oracle, mysql, informix, etc.) plus a built-in database browser.
- **WWW Access**. Possibility for human agents to participate through the web. This is currently possible if using a *HotJava* browser, but it is somewhat troublesome for *Netscape* since there is no full support for *JDK1.1*.

A.2 Reporting Bugs

If you have a specific feature request or bug to report, please send an e-mail to:

jar@iiia.csic.es

We are also very eager to receive your comments and suggestions as early as possible. Please feel free to send comments and informal suggestions directly to the email address above.

A.3 Troubleshooting

The participants in the FM tournaments[7, 6] reported the existence of some problems when executing FM on X terminals. The *Sun* people assert that these "might" be fixed by *JDK1.2*, but we have not upgraded yet because *JDK1.2* is currently a beta version.

A.4 Where to Find More Information

More information concerning the FM project can be found at the FM project web page

http://www.iiia.csic.es/Projects/fishmarket

You can also subscribe to the FM mailing list by sending an e-mail to *Majordomo@iiia.csic.es* with the contents

```
SUBSCRIBE fm
```

in the body of the message.

A.5 Acknowledgements

FM has been implemented by Juan A. Rodríguez-Aguilar, Francisco J. Martín, Francisco J. Giménez, and David Gutiérrez.

However, FM is the result of a collective effort in which many more people have been involved. Thus, we would like to acknowledge:

Xavier Márquez manager of the Blanes fishermen's cooperative market (Blanes, Girona) for his illuminating explanations.

Pere Garcia, Pablo Noriega, Carles Sierra, Lluís Godo, Ramon López de Mántaras, Ulises Cortés, Julian Padget, Andreas Kind, Julio García, Maurizio Giordano, Claudia Di Napoli, and Mario Mango for their contributions to the evolution of the project up to the current stage.

Bibliography

[1] Pablo Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. PhD thesis, Universitat Autonoma de Barcelona, 1997. Also to appear in IIIA mongraphy series.

[2] Juan A. Rodríguez-Aguilar, Francisco J. Martín, Pablo Noriega, Pere Garcia, and Carles Sierra. Competitive scenarios for heterogeneous trading agents. In *Proceedings of the Second International Conference on Autonomous Agents (AGENTS'98)*, pages 293–300, 1998.

[3] Juan A. Rodríguez-Aguilar, Francisco J. Martín, Pablo Noriega, Pere Garcia, and Carles Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.

[4] Juan A. Rodríguez-Aguilar, Pablo Noriega, Carles Sierra, and Julian 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.

[5] The FishMarket Project. http://www.iiia.csic.es/Projects/fishmarket.

[6] The First FM Tournament. http://www.iiia.csic.es/Projects/-fishmarket/tournament97.html.

[7] The Second FM Tournament. http://www.iiia.csic.es/Projects/-fishmarket/tournament98.html.