Agent Oriented Programming with Ada'95 :
Application to financial markets

Férial BENACHOUR HAIT
Université Paris XII. Département Maths-informatique.
61, avenue du GAL de Gaulle. 94010 Creteil Cedex
T: (33) 1 45 17 65 96 / F: (33) 1 47 17 16 49
benachour@univ-paris12.fr

Abstract

This paper shows the suitability of Object Oriented Programming language Ada'95 for Agent Oriented Programming. Agent oriented programming relies on the assumption, that a complex distributed software system can be programmed as a set of communicating, interacting, knowledge base entities called (software) agents. In this work we study the financial market case by introducing new modeling and implementation tools in order to simulate and understand some properties of the market evolution process. Object -Oriented modeling of social systems requires the integration of new concepts related to mental processes such as learning, planning, knowledge representation and communication . If one defines an agent as an object combination of mental state to perceive its environment and achieve some goals, a knowledge base to hold its *memory* and an interfacing communication with other entities then all the above properties could be fulfilled. In this paper we describe a method for the use of Ada object Oriented Programming language for the implementation of a MAS plate-form. The results obtained in the experimental part lead to some parameters of the model validation, like the agent's performance in increasing their own wealth and their adapted behavior throughout the evolving market.

1. Introduction

The recently developed Ada programming language [ISO], has sparked a great deal of interest in the software engineering community. Among its interesting properties are its support for concurrency [BUR 98], distribution, inheritance and its power form of object orientation in comparison with C++. Object orientation influences complex model design and execution [Rum 91]. For design and simulation, the key contributing is the mapping between the real system and the model. At an implementation point of view, main contributions are seen as the convenient decomposition of complex systems into individual objects and the isolated description of their behavior [SMI 96]. Since objects are the basic run-time entities of our modeling system, each object will be represented by an encapsulated unit or module. Modularity becomes one of the main characteristics and advantage of OO paradigm. This enable us to consider objects as concurrent actors that communicate asynchronously [LOE 97].
In financial market context, agents are autonomous concurrent entities acting simultaneously in an evolving changing world. This kind of environment is very hard to design practically with classical economical methods. Even the

famous rational expectation theory [PAL 94] is unable to design the rule of thumb that really work in a market place or to provide some quantitative values of the agent's behavior in their environment.
Object -Oriented modeling of social systems requires the integration of new concepts related to mental processes such as learning, planning, knowledge representation and communication [WEI 96]. Potentially valuable techniques derived from Distributed Artificial Intelligence (DAI) have been introduced in simulation of social systems into agents [FER 95] . That is, agent paradigm is also based on the notion of autonomous and reactive entities embedded in changing uncertain worlds. But if agent is not directly an object, s*omething becomes an agent by the fact that one has decided to control and analyze it's behavior in mental terms, e.g., beliefs and goals* [UHR 97].
Recent and advanced research [JEN 98] allow us each year to more understanding and defining the notion of agent. And as D.Kinny [KIN 96] pointed out: "we can easily imagine an extension of the object paradigm to the agent by building upon and adapting well understood techniques of OO design and programming".
Our present work is rooted in this approach and propose the use of multi-agent systems (MAS) for the study of financial market behavior using Ada'95 language programming as an implementation tool. Therefore we consider the market behavior as a distributed market agents who deal autonomously with local task planning. The behavior of the simulating environment as a whole is then an emerging functionality of the individual skills of, and the interactions among the agents [TAN 98].
The agents action in their environment are inductive. It means the agents are allowed to adapt and learn from their environment by recording the success and the failure of their past actions. To realize that, we implemented a process of reinforcement learning with a credit assignment in the agent's internal state [BEN 99b]. It is shown in the paper how each agent adapts his transaction to the evolving market and updates his own data after each period. The results obtained in the experimental part show clearly how the investors can learn from their environment and increase their wealth by adapting their transaction's behavior throughout the evolving market.
The paper road map : Section 2 describes the markets, the participants and the rule of the game. Section 3 introduces the modeling process and the conceptual model design. Section 4 investigates the implementation issues and introduces Ada'95 features for the MAS plate-form. Section 5 discusses the results and some key parameters for the model efficiency. The last section leads to conclusion .

2. The Financial Market

- Using the inductive approach [PAL 94], we consider the market behavior exactly at the opposite of the RE theory with agents having little knowledge or reasoning ability at the starting game. Agent's activity is involved by his action which is provided by an internal decision. A credit assignment is affected to it's performance (increasing, decreasing) through a process of reinforcement learning that compute the feedback

returned by the environment [BEN 99 a,b] . The agents (investors) trade with a trader on the basis of internal models or rules which are initially very poor and perform their actions by observing the success or failure of their past transactions. A board data price is there to display the current prices and transactions made by the agents. Agents are risk neutral and may be homogenous or not (in term of dumb or smart agents) in their interactions.

- ***Rules of the game and the market specification:***

We consider a financial market where a share is exchanged [BIA 97]. The process follows the next protocol :

1. The display of the first price (fp) for all agents.
2. Each investor, if he can propose simultaneously an order (Q) which is a bid to buy or an offer to sell to the trader for the stock shared on the market.
3. The trader revises his evaluation on the value of the share he holds.
4. The trader gives his new quotations and displays the price transactions to the investors and the board price.
5. A new public price or the current price (Cp) which is the average of the best prices of offers and bids is displayed on the board price.

3. The modeling Process

The modeling process is rooted in a tightly coupled and iterative four activities process composed of :

- system analysis. A guide line has been given in [FIS 95] to build this system consisting in building first a conceptual model of the system and then breaking this model into a hierarchy of abstractions and then choosing models to represent those abstraction levels;
- Simulation model design, since knowledge models must be converted to algorithms to run on computers;
- Interpretation of the results at the experimentation.
- Validation and verification of the simulation process.

3.1. The conceptual Model

A methodology for analyzing and designing a solution to a problem using Object-Oriented techniques is Fusion. Fusion is a second generation Object-Oriented methodology designed by Coelman [COE 96] which builds on many of the first generation Object-Oriented methodologies. We used for the conceptual model a simplified version of the analysis step. Figure 1 displays the class hierarchy for the conceptual model using Fusion. Instantiating this hierarchy leads to a MAS involving three kind of agents : the trader, the investors and the board price.

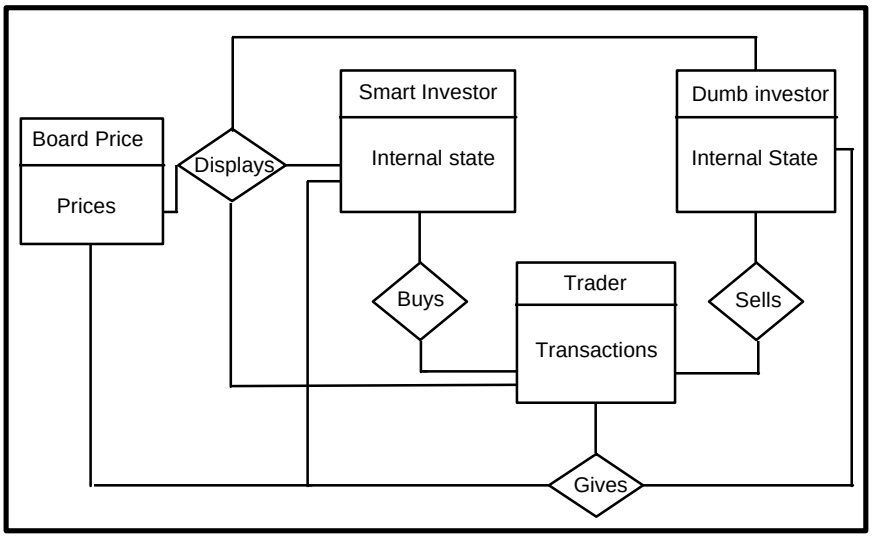

Figure 1. Agent model of the financial market

Only the investors are allowed to learn and provide a cognitive behavior. The trader and the board price are reactive agents. The trader systematically computes the transactions prices and sends them to the investors and the board price computes and displays updating prices.

3.2 The Agent architecture

An investor agent consists of components of perceiving its environment (sensors); for keeping an internal state (the knowledge base); for communicating and interacting with other agents; for establishing and performing actions, based on the current internal state and the environment to achieve some goals [KUH 97].

A formal and abstract view of agents is that an agent consists of an object with a set of properties (values, data) and a set of methods (actions, functions, operations) [FER 95]. We suggest an investor architecture defined by the following 5-tuple (see figure 2). The environment holds the other agents and the board price and the trader.

U: is the set of data received by the agent from his environment like the current stock price, it's dividend, it's ratio or the transaction price sent by the trader.

KB: is his knowledge base according to his individual parameters like the wealth and the risk aversion.

A: is the set of actions or transactions performed by the agent at each period in his environment.

M: is the memory space of the agent. It records for every period all its internal components (instances) or variables.

E: is the evaluation process. It updates systematically all the agent's components, updates his variables for each period and evaluates it's

performance through the process of learning with a credit assignment that computes all the previous state of the agent.

- The perception and action sets design the external state of an agent interfacing with it's environment. It's the *communication* module.

- The memory space and the evaluation process design the internal state of an agent that make him performing his actions to achieve his goal. It's the *evaluation module.*

Looking at these basic properties and for a complete specification of the market system, we need to consider our model at two levels of abstraction:

a) *From the external point of view :* The system is decomposed in modeled and complex agents characterized by their actions, their perceptions and interactions. This is the analysis phase which leads to the object model.

b) *From the internal point of view :* All the elements required by each agent according to his architecture. It means all the different components that underlie his organization and make him acting and learning in his environment. This is the agent structure or architecture.

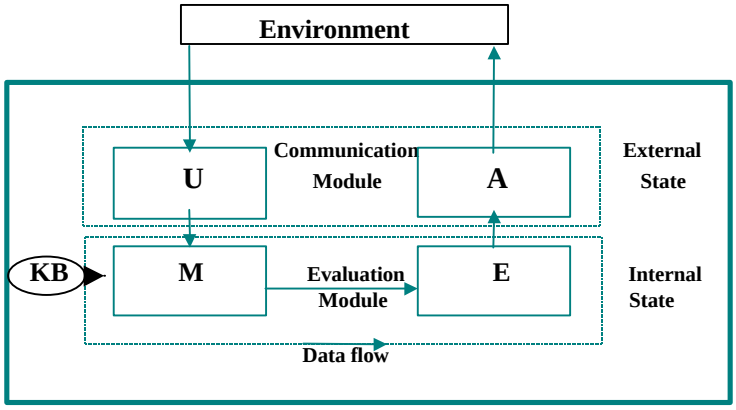

Figure 2. The agent architecture

4. The implementation and running process

A MAS basically refers to a distributed simulation environment, where simulation takes place as a sending of messages between concurrently interacting entities.
From all the above points one can derive requirements to a programming language suitable for Agent Oriented Programming. This language has to provide means to express concurrency, to implement communication between agents, to access sensor data , to represent knowledge, to infer implicit knowledge and to plan actions in order to achieve some goal.
If we look at the Ada'95 programming language then some of the above requirements are already fulfilled.
An agent in our market consists of components for perceiving its environment (sensors), for keeping an internal state (knowledge base), for communicating and interfacing with other agents and performing his actions to achieve his goal. The market it self or the MAS refers to a distributed simulation environment, where simulation takes place as a sending of messages between these concurrently interacting entities. Moreover because investors performs their transactions with the trader meanwhile perceiving the current price from the data board or receiving the transaction price from the trader, our agents system components are intended to run in parallel.

4.1. Implementing Concurrency

In Ada'95 programming language [ISO 95], one can also write programs that perform more than one activity concurrently (during the same period of time). This concurrent processing is calling tasking [NAI 95]. In Ada terminology, a process is represented as a task. Ada's task type mechanism provides support for a modular design method, in which tasks work on encapsulated local data. The communication is done by rendezvous along with corresponding operations (methods) for creating, updating and querying instances of the tasks.
To implement the object classes of the system, we used Ada's package. A package consists of one or more object classes implemented as abstract data type [BUR 98]. Ada packages are divided into two parts, the specification part that gives only an external view of the object and the implementation part or the body which are the internal structure and are hidden from other objects.
Since agents are concurrent entities possessing their own logical processor, a task is attached to each agent as a record component implementing the agent's concurrent dynamical behavior and then encapsulated in a package *class-agent*. A request for performing any method of an object will be transferred to an *entry* call of the corresponding task. The board price is a package displaying and updating the current prices. The trader is a *reactive* agent encapsulated in a package class that systematically computes the quotations for each investor transaction. Real time features like rendezvous are used to realize communication mechanism between agents.

4.2 The agent's implementation

The purpose of the *update* method in the investor package is firstly to instantiate the investor agent internal state at the first running time and then to update his past data after each cycle of transaction. All the agent's internal state components like his wealth (W_i), risk aversion (ra), his utility function (U_i), his estimator state (E_i) and his evaluation function (V_i) are recorded and stored in the record part held by *event_array* . the specification for the ninth investor package class is given in figure.3. The package defines a task to update the agent's internal state (private component). The *rendezvous* update is used to pass data to the task with the input variables and used to pass the result back with the output variables.

```
package investor_package9 is
type store_data;
type store_data is record
ra: float;
q_1: float;
w_i: float;
r_i: float;
u_i: float;
E_i: float;
v_i: float;
end record;
type event_array is array (1..100)of store_data;
event:event_array;
task type t_task_investor_9 is
entry update(k:in integer;q_1,fp:out float);
end t_task_investor_9;
end investor_package9;
```

Figure.3. Package specification for an investor

All the other investors follow the same principles of building.
A task is created using the normal Ada mechanism. To create an instance of the task *t_task_investor_9*, the following declaration is used:

Thread_9: *t_task_investor_9;*

Figure.4 shows how agents tasks start executing as soon as the begin of the block in which they are elaborated, is entered. The *rendezvous* point *update* is used to control this wayward behavior.
Once started, each of the threads will execute concurrently until the last rendezvous is encountered.
In fact a thread in Ada'95 is a single flow of control, running in parallel to other threads using the same address space. That's why we can have more than one active object at the time during runtime.

```
procedure Marksim9 is
N:constant :=100; --nombre de cycles;
q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,fp,cp:float;     res9_name: constant
string:="res9.txt";
rs9:ada.text_io.file_type;--file descriptor of res9.txt
thread_1:t_Task_Investor_1;
thread_2:t_Task_Investor_2;
thread_3:T_Task_Investor_3;
thread_4:T_Task_Investor_4;
thread_5:T_Task_Investor_5;
thread_6:T_Task_Investor_6;
thread_7:T_Task_Investor_7;
thread_8:T_Task_Investor_8;
begin
 create(file=>rs9,mode=>out_file,name=>res9_name);
  for k in 1..N loop
   thread_1.update(k,q_1,fp);
   thread_2.update(k,q_2,fp);
   thread_3.update(k,q_3,fp);
   thread_4.update(k,q_4,fp);
   thread_5.update(k,q_5,fp);
   thread_6.update(k,q_6,fp);
   thread_7.update(k,q_7,fp);
   thread_8.update(k,q_8,fp);
display_data.data_transac(q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,fp,cp);
put(rs9,cp);new_line(rs9);
end loop;
close(rs9);
Put("Market simulation is closed. A bientot");
end Marksim9;
```

Figure.4. The main program of the running system

In the body part of each investor package, there is calling task to trader in order to make a transaction. When the trader has treated all the transactions, orders (q_1,q_2,…q_n) are transferred to the board_price package in order to compute the new current price (cp). This is done by the calling task display_data.data_transac in the board price package.

```
package body investor_package9 is
  ……..;
      task body t_task_investor_9 is
          begin
             create(file=>ws8a,mode=>out_file,name=>wes8a_name);
            loop
               open(file=>fd,mode=>in_file,name=>file_name);
             accept update(k: in integer;q_1,fp:out float) do
              if k=1 then
               ……..;
               event(k).ra:=ra;
               event(k).q_1:=0.0;
               event(k).w_i:=w_i;
               event(k).r_i:=0.0;
               event(k).u_i:=-exp(-(w_i)*ra);
               event(k).e_i:=event(1).u_i;
               event(k).v_i:=event(1).u_i;
                ……..;
                var:=fl;
                ……..;
                 q_1:=j+fp/(Ra*Var);
```

```
                ……….;
               div:=fp/10.0;
                if fp>div/int then put("buy");new_line;
                 else
                   q_1:=(-q_1);put("sell");new_line;
                end if;
                trader.investor(q_1,fp,ap);
              else
                ……..;
                event(k).ra:=ra;
                event(k).q_1:=q_1;
                event(k).w_i:=(event(k-1).w_i)+(event(k).q_1)*fp;
                event(k).r_i:=(event(k).w_i)-(event(k-1).w_i);
                w_i:=event(k).w_i;
                event(k).u_i:=-exp(-ra*(w_i));
                  if event(k).r_i>=0.0 then g:=-2.5;
                   else g:=3.5;
                  end if;
                event(k).e_i:=(event(k).r_i)+(g*(event(k).u_i));
                event(k).v_i:=(event(k).e_i)+(event(k-1).e_i);
                ……..;
                j:=fl;
                q_1:=j+fp/(ra*Var);
                 div:=fp/10.0;
                  if fp>div/Int then Put ("buy");New_Line;
                   else
                    q_1:=(-q_1);Put("sell");New_Line;
                  end if;
                  bool:=0.0;
                  if event(k).v_i > bool then
                   trader.investor(q_1,fp,ap);
                    else
                   put("non execut");new_line;delay 1.0;
                  end if;
            end if;
          end update ;
        end loop;
      end t_task_investor_9;
    end investor_package9;
```

figure.5. The body package of an investor

5. The simulation model and experimentation

The fore going has described how the micro-level behavior of the modeled entities is constructed. All model outputs are the result of their complex interactions. We used 10 agents 8 investors, a trader during and a board price running during 200 or 300 periods. All the investors are not homogenous and have individual wealth risk aversion and learning ability for their own transactions.

- **Heterogeneity of the market :**

we start the description of the results by the study of the dynamic of market prices when investors acting are either totally smart (CPSA) (with a process of learning), totally dumb (CPDA) (with no process of learning) or mixed (CPMA) (a market with smart and dumb investors). Figure 6 shows the evolution of the market price.

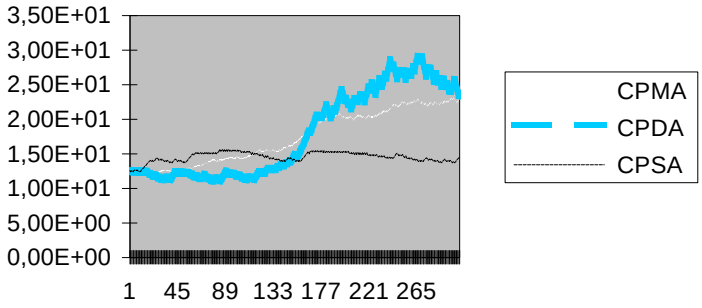

Figure 6. Dynamics of prices for different agents

When the agents are smart or mixed, the variability of the market price is low. We interpret these results as saying that the market converges to an equilibrium fairly quickly when agents learn from their experiences. When we increase the dumb agents we obtain a market price which is more volatile than before. At the beginning of the market, prices show a quite hesitation until the middle of the period where the prices become more volatile. This illustrates that the agents cannot adapt their actions to the environment.

- **The speed of performance in a homogenous market**

Figure 7, shows the speed of agent's performance gaining wealth when acting in a homogenous market. We can see that the dumb agents in a dumb market (WDADM) have more chances than before to maintain their actions in the market and raise their wealth even quite slowly. However the smart agents in a smart market (WSASM) become less rapidly richer than they were in the mixed market because they are confronted to other agents concurrency.

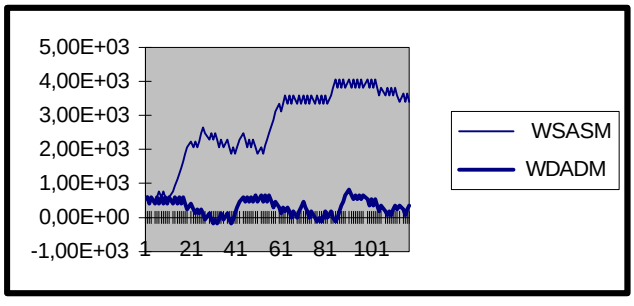

Figure.7. Performance of agents in a homogenous market

This result is very interesting because it's very close-fitting to the reality. In fact in any market many irrational agents make transactions for a simple

reason of liquidity, and it's very hard for the "smart" agents to maintain a coherent trend of learning in this kind of environment. However it seems much easier to increase our richness in a more homogenous market were the agents follow roughly the same behavior.

- **Some emergent features**

Figure 8 shows how the price is gaining in volatility when the number of agents is gradually increasing from 3 agents (Res3) to (Res9). In fact more the number grows, more the market is dynamic and the prices evolving. This shows an interesting feature explaining the market efficiency and the evolution trend of the market through the number of agents acting together.

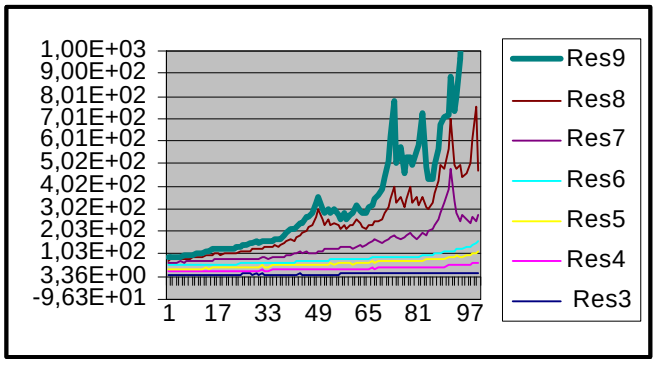

Figure 8. Prices variation with increasing number of agents

Figures 9 and 10 show what we called the *mass effect* in terms of conditions of rapid enrichment in a market. Here we are not surprised to see that more the number of agents is increasing in a market, less the agents have chance to become quickly much richer. Even if they are *smart* in majority, the speed of performance seems deeply correlated with the agents weight. Thus even if an agent has an important wealth when starting the game, the concurrency with other agents plays a non neglected role in the speed of his enrichment.

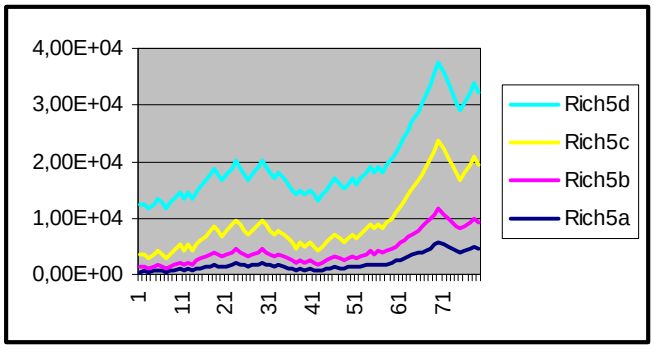

Figure.9. agents enrichment in a market with 5 agents

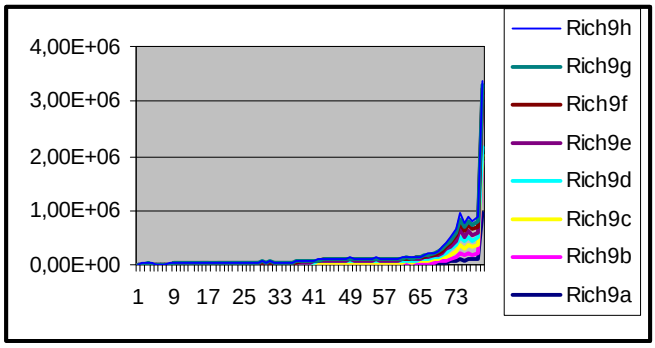

Figure.10. agents enrichment in a market with 9 agents

6. Some Validation Keys and conclusions

Our first objective in studying financial markets was to enhance understanding of how OO design might be used and applied to improve MAS. Here we believe that our results go towards this approach suggesting that MAS could be suitable tools for understanding complex mechanisms like social systems.

Our second interest was to conceive and build an adapted model to understand some properties of the stock market. In this I believe we were successful, designing and implementing an artificial stock market with different type of agents. Even if the stock market is much simpler than the real thing, it still contains considerable complexity; Because investors both create and exploit the prices series, the agents are essentially *coevolving* by making mental models or hypotheses, based on past experiences and training.

We showed in the experimentation part that a market populated by agents acting according to some simple rules and learning, can generate rich dynamics in market price. Agents make or loose money according to their ability for learning from their environment. They adapt gradually their transactions to the market behavior. Besides that the criterion of

homogeneity gave an interesting result concerning the agents behavior and their impact in the whole market dynamics.
An other important thing to point out with a multi-agent simulation is the possibility to give a quantitative answers to the problem of the qualitative aspects of social science like the design of the behavior which is very hard to formalize in theory, and seems more tangible with this kind of systems where the actions of agents are well designed.
The last point concerns agent oriented simulation in social systems where various issues can be investigated to study the emergent dynamic of the system and understand more about the relationship between micro-levels specification of individual entities and the macro level emergent behavior of the whole system. The many emergent features go towards that and can lead to many others characteristics of the studied system.

Acknowledgement

I would like to thank Konstantin Verchinine, Brahim Chaib-Drâa, Suzanne Pinson Jean-Pierre Briot and Steve Lai for their encouragement and helpful remarks .

7. References

[BEN 99a] F.Benachour Hait, « Méthodologie et développement d'agents artificiels adaptés à un marché financier », *Technical Report :*TR-19, 1999.

[BEN 99 b] F.Benachour Hait,, «A multi-agent system integrating a learning process applied to Financial market », To appear in L'Objet/ Avril 2000.

[BIA 97] B.BIAIS, *Microstructure des marchés financiers*, Puf, Paris, 1997.

[BUR 98] A.Burns and A.Wellings. *Concurrecy in Ada*. Cambridge University Press, 1998.

[COE 96] D.Coelman et al., *Fusion La méthode orientée objet de 2 ème Génération*, Masson, 1996.

[FER 95] J.Ferber. *Les systèmes multi-agents*. Interéditions, Paris, 1995.

[FIS 95] P.Fishwick. *Simulation model design and execution: building digital worlds*. Printice-Hall, 1995.

[HIL 96] D.RC.Hill, *Object-Oriented Analysis and Simulation*, Addison-Wesley, 1996.

[ISO 95] International Standardisation Organisation , « Information Technology-Programming Languages », *Ada International Standard ISO/IEC 8652*, 1995.

[JEN 98] N.R.Jennings and M.J.Wooldridge, *Agent Technology : Foundations applications and Markets*, M.J. Wooldridge (ed), Springer, 1998.

[KIN 96] D.Kinny, M.Georgeff and A.Rao, « A methodology and modelling technique for systems of BDI Agents », in *MAAMMAW'96*, in *LNAI* , Agents breaking away, pp 56-71.

[KUH 97] R.Kuhnel, «Agent oriented programming with java» in *Proceeding of the search international conference on artificial intelligence and information control system of robots*, 1997.

[LOE 97] H.Loeper and al., « Concurrent Objects in Ada'95 », in *Ada Letters* VOL XVII n°6, 1997.

[NAI 95] D.J.Naiditch, Rendezvous with Ada'95, Naiditch edition, 1995.

[PAL 94] R.G.Palmer, W.Brian Arthur, J.H.Holland and al, « Artificial economic life: a simple model of a stockmarket», in *Physica D*, n°74, 1994, pp 264-275.

[RUM 91] J.Rumbaugh et al., *Object Oriented Modeling and Design*, Printice Hall international, 1991.

[SMI 96] M.A.Smith, *Object oriented software in Ada'95* , In. Thomson. Com. Press, 1996.

[TAN 98] G.S.H.Tan, « Applying intelligent agent technology as the platform for simulation », in *Simulation Symposium*, 1998.

[WEI 96] G.Weiss, «Adaptation and learning in multi-agent systems: some remarks and a bibliography, G.Weiss and S.Sen (Eds), pp(1-21), LNAI , Vol 1042, 1996.