UNIVERSITY OF CRAIOVA
FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION
DOCTORAL SCHOOL OF SOCIAL - HUMAN SCIENCES
FIELD: CYBERNETICS AND STATISTICAL ECONOMICS

DOCTORAL DISSERTATION ABSTRACT

MULTI AGENT SYSTEMS APPLIED IN E-BUSINESS

Scientific supervisor,
Prof. univ. dr. Vasile GEORGESCU

Ph. D. Candidate,
Adrian Victor SĂNDIȚĂ

Craiova
2015

TABLE OF CONTENTS

1. CONTENTS OF DOCTORAL DISSERTATION

2. KEYWORDS

- Intelligent Agents;
- Bucharest Stock Exchange (BSE);
- Foundation for Intelligent Physical Agents (FIPA)
- Supervised training;
- Computational intelligence;
- Market Maker;
- Support Vector Machines (SVM);
- Agent orientated platforms;
- Neural Networks (NN);
- Multi-agent systems (MaS);
- Automated trading.

3. REASONS, OBJECTIVES AND IMPORTANCE OF RESEARCH

Electronic exchanges play an increasing role within the financial markets, and the study of the trading mechanisms that are used for financial securities represent the key to their understanding. Most important stock exchanges use the market makers model (Market Maker) to assure transactions liquidity and to offer a higher quality and more efficient market (Kim A.J., 2002). Every issuer has one or more market makers that act to provide liquidity for the stocks. The responsibility of the market maker is to establish the volumes and prices at which it is willing to sell or buy, respectively (Stone and Sherstov, 2005). The market makers represent a major advantage for their acting market. The traders that wish to buy or sell the stocks of an issuer do not need to wait one another in order to achieve desired transactions. Moreover, the existence of a Marker Maker discourages traders that try to manipulate the market by introducing orders to determine price fluctuations towards the desired direction (Nevmyvaka et al., 2005).

Market makers beneficiate from the position they hold. Although not directly, a market maker has the advantage because the activity it undertakes allows it to buy at a low price and sell at a higher one (Feng et al., 2004). The market maker introduces two quotations: one to sell and one to buy so that other traders may buy or sell at prices fixed by the market maker. Obviously, the selling price is higher than the buying price and the difference between the two prices is called the spread. To increase market liquidity, spreads should be as small as possible. On the other hand, in order to increase the winning chances of the market maker, the spread value has to be high.

The benefits of the automation of this activity are: a machine properly programmed can always react more quickly to market changes than a human trader and the decisions that the machine takes are not affected by the inconstancy that characterizes in many cases the human decisions.

The main objective of the thesis was necessary study for the design, development and implementation of a real life multi-agent trading system that can automatize the work done by a Market Maker within BSE. Automated trading decisions of the developed multi agent system are based on the trend evolution predictions issued by intelligent agents that use for analysis methods from the computational intelligence domain: neural networks and support vector machines (Săndiță and Durac, 2015).

4. SUMMARY OF THE CHAPTERS INCLUDED IN THE THESIS

The thesis is composed of introduction, four chapters, a separate part dedicated to conclusions and future developments and ends with the bibliography.

Chapter 1. MULTI AGENT SYSTEMS

The term agent has multiple meanings depending on the context in which it is used. Wooldridge (1997) defines an intelligent agent as a system that has the following four properties: autonomy (agents operate without direct intervention from users and can control their internal actions and states), social skills (agents are able to cooperate with users or other agents for the completion of tasks), responsiveness (agents perceive the environment and are able to give a timely response to changes) and pro-activity (agents do not act simply as a response to the environment they are in, they are able to behave objectively by taking initiatives).

An agent is a software entity actively seeking ways to complete tasks. Intelligent agents have the ability to acquire knowledge through problem solving processes. Software agents focus on interactions and collaborations to achieve their objectives in a changing context in a usually unexpected manner. The agents are used when the complexity of the classic large software systems raises design issues that conventional technology fails to address, or when modularity, execution speed, endurance, coordination and communication between components are characteristic to the system that will be implemented. In a distributed dynamic system, agents capable of self-regulation can simplify the architectural design of the system, which can be extremely complicated in the traditional architectures of object oriented modeling.

Agent oriented modeling is an unconventional approach to system design, in defining the components and in system integration inclusively. Autonomy is a distinctive property of an agent and it implies the ability of an agent to survive in a changing environment. An agent has the ability to perceive environmental factors and make decisions about how to react accordingly. Adaptability assumes the existence of the required learning capacity to adapt decisions based on past experience. Moreover, an agent-oriented design should develop methods and techniques which enable it to react correctly when unexpected events occur.

Distributed Artificial Intelligence is a subdomain of artificial intelligence that deals with solving problems that involve agent interactions in order to solve a common problem (Green et al. 1997). Agents can inherit potential benefits both from DAI: modularity, speed, reliability, and from artificial intelligence: operating at the levels of knowledge maintenance, reusable platform, independence (Nwana , 1996).

In the design of large and/or complex systems, an agent is an abstraction that helps the design of components that cooperate in resolving various aspects of a problem. Each agent is designed in the best paradigm to solve its part of a problem. A multi-agent system is used to solve a complex problem that cannot be solved by a single entity system. Agents have the ability to efficiently process local data and communicate with other agents when necessary, if the tasks they face are beyond their knowledge.

Multi-agent systems are used in a wide range of applications such as e-commerce, e-learning, communication, data mining, simulation, robotics, transportation systems and grid computing systems. There were also theoretical studies initiated in the reasoning and specifications of multi-agent systems, in representation and processing of knowledge, and in cognitive sciences.

The study of multi-agent systems was the basis of our research. The implementation methodology of agents, their defining properties, the communication method and the architectures of multi- agent systems have enabled us to reach the completion of the multi-agent system introduced in the fourth chapter of the thesis.

Chapter 2. MaS orientated platforms

Multi-agent platforms and development tools are important components affecting dissemination and the use of technologies in various agent oriented fields. In fact, the successful implementation of multi agent systems largely depends on the availability of the appropriate technology that enables the implementation of concepts and techniques underlying multi-agent systems (Ricordel and Demazeau, 2000).

Software platforms and frameworks are key enabling resources to develop multi-agent systems (Odell et al., 2002). Most offer a means to implement multi-agent systems on different hardware systems and under different operating systems, typically providing tools and techniques that support running them and allow smooth implementation of critical operations, such as communication and coordination. Some of these platforms and frameworks have a common goal of providing standardized functionality to support interoperability between different multi-agent systems. Moreover, some also aim to support various types of hardware, communication networks and agent architectures (JADE, Cougaar etc.) and others aim to implement only special types of agents, such as, for example, mobile agents (Lange and Oshima, 1998).

Agent-oriented languages and platforms allow researchers and application developers to focus more intensely on the tasks that agents must carry and consume fewer resources for effective implementation of systems of agents.

Foundation for Intelligent Physical Agents (FIPA) is an international organization dedicated to promoting standards of design, implementation and use of agents and agent-based systems (FIPA website). FIPA standards represent a benchmark for developers of agent-based systems. Respecting FIPA specifications offers several arguments that could be decisive in the selection of platforms which will be implemented on a certain class of agents. This architecture ensures compatibility with FIPA standards and system performance based on strong and fully tested protocols that enable communication between agents, cooperation and interoperability. Therefore, it is preferred to respect FIPA standards when the systems that are supposed to be implemented are dynamic, flexible and reconfigurable. In addition, the research community for agent systems extensively uses FIPA compliant platforms and so there is a large number of open source implementations of Java classes, thoroughly tested and well documented that may be useful in application development. However, the diversity of applications and areas where agents are used inherently determine the variety of mechanisms through which implementation problems are approached and resolved, for many of these standards representing a constraint and not an advantage.

The Cougaar platform described in this chapter implements agents in particular, the tasks that they can perform and the method of communication between entities operating in the system. The Cougaar model adds to the traditional system the Binders service, part of a strong mechanism for containment and encapsulation of all components (Helsinger et al., 2004). Standardization of such mechanisms would be expensive relative to the scope.

However, for many directions in the line of research on agent systems, compatibility, portability and easy programming environment are extremely important arguments in favor of choosing FIPA compliant platforms, such as JADE .

JADE architecture is based on the existence of containers in which agents live and that provide logistics of the executed tasks that agents must meet. There is a main container that holds all other containers addresses and characteristics and manages the table that retains information on all platform agents (JADE website).

The containers contain a table which retains descriptions of locally registered entities and the communication between the containers is achieved through a specific protocol, implemented locally. Being used solely for communication between agents from the same platform without exceeding its borders, the used protocol is not bound by FIPA standards. In fact, JADE uses this protocol to also launch specific commands to the distributed platform and to monitor from a distance the condition of the platform containers.

Unlike the protocol used in the platform, for ensuring communication with entities located outside, *Message Transport Protocol* meets the FIPA standards. JADE also supports multiple parallel complex interactions and conversations and provides a set of frameworks of interaction models for specific tasks. By using these frames (implemented as Java abstract classes), programmers can relieve from the problems caused by synchronizing the activity of agents.

Furthermore, to increase scalability or to meet environmental constraints with limited resources, JADE enables parallel execution of multiple tasks in the same Java thread.

Although our designed multi-agent system has not been implemented on a dedicated platform, the study of platforms and languages oriented towards agents allowed us to identify and integrate in our project more elements crucial to the functioning of the system: inter-agent asynchronous communication, the White and Yellow Pages mechanism, and not least, the coordination and supervision mechanism of agents.

Chapter 3. Computational intelligence techniques

Computational Intelligence techniques are increasingly being used to solve problems that cannot be addressed by traditional techniques or when the information is insufficient to create a model on which we are able to develop a solving algorithm.

This chapter presents two of the techniques specific to the domain, techniques used in the development of the achieved multi- agent system, namely artificial neural networks and support vector machines.

The main quality of neural networks is that they can store knowledge that they can later use in new situations. Acquisition of knowledge in neural networks is made by storing values in the synaptic weights, values that depend as much on the network architecture that includes them. The process of assigning values to synaptic weights in a neural network is called learning process or training process .

As we have seen, there are two fundamental types of learning: supervised learning and unsupervised learning. In the case of supervised learning, the network is presented with plenty of examples of learning, represented by input-output vector pairs, characteristic to the knowledge that is to be acquired. The values of the input vectors are introduced into calculations, affecting current network weights and causing the output values to compare themselves to expected results. Depending on the deviation between the observed and expected values, the weights are adjusted so that the differences become zero or as minimal as possible.

The whole process is repeated until one considers that the network has been fully trained. The learning algorithm is based on the fact that if the network is behaving in a certain way in familiar situations, it will retain its behavior in new and unfamiliar situations.

The data used for learning are called training data, and those used to track the network behavior in new situations are called test data .

In the case of unsupervised learning, the learning data set consists only of a lot of input vectors that, as in the case of supervised learning, affect the weights stored in the network. The difference to supervised learning is that the adjustment of weights in this case is made such that for close input vectors the same output vector or output vectors with very similar values are obtained.

A problem that arises frequently in networks with back propagation and dramatically decreases the effectiveness of learning is local minima. Because the training process stops when the network reaches a global minimum or a local minimum of the objective function, a method that determines the exit of the network from local minimum points has to be implemented, the goal being the achievement of a global minimum. Usually, methods require change of initial weights, of the training constant or of the number of hidden units.

However, if the solution obtained as a result of network training is acceptable in terms of rate error, the fact that the network has reached a local minimum or global minimum is no longer relevant.

Support vector machines (SVM) represent an effective method in designing a feedforward network with a single hidden layer of linear units. As the name suggests, this type of neural network design is based on extracting a subset of training data that serves as support vectors and represent a stable characteristic of the data (Vapnik and Chervonenkis, 1971).

In literature, SVM were imposed as the most used algorithm due to good performance of generalization, to rigorous theoretical foundations, to their relatively easy implementation and the ability to provide outstanding performance in pattern recognition problems and regression (Yom-Tov, 2007).

In the completion of the system introduced in the next chapter the supervised learning was used, training data and the test data taken from the trading system of the Bucharest Stock Exchange.

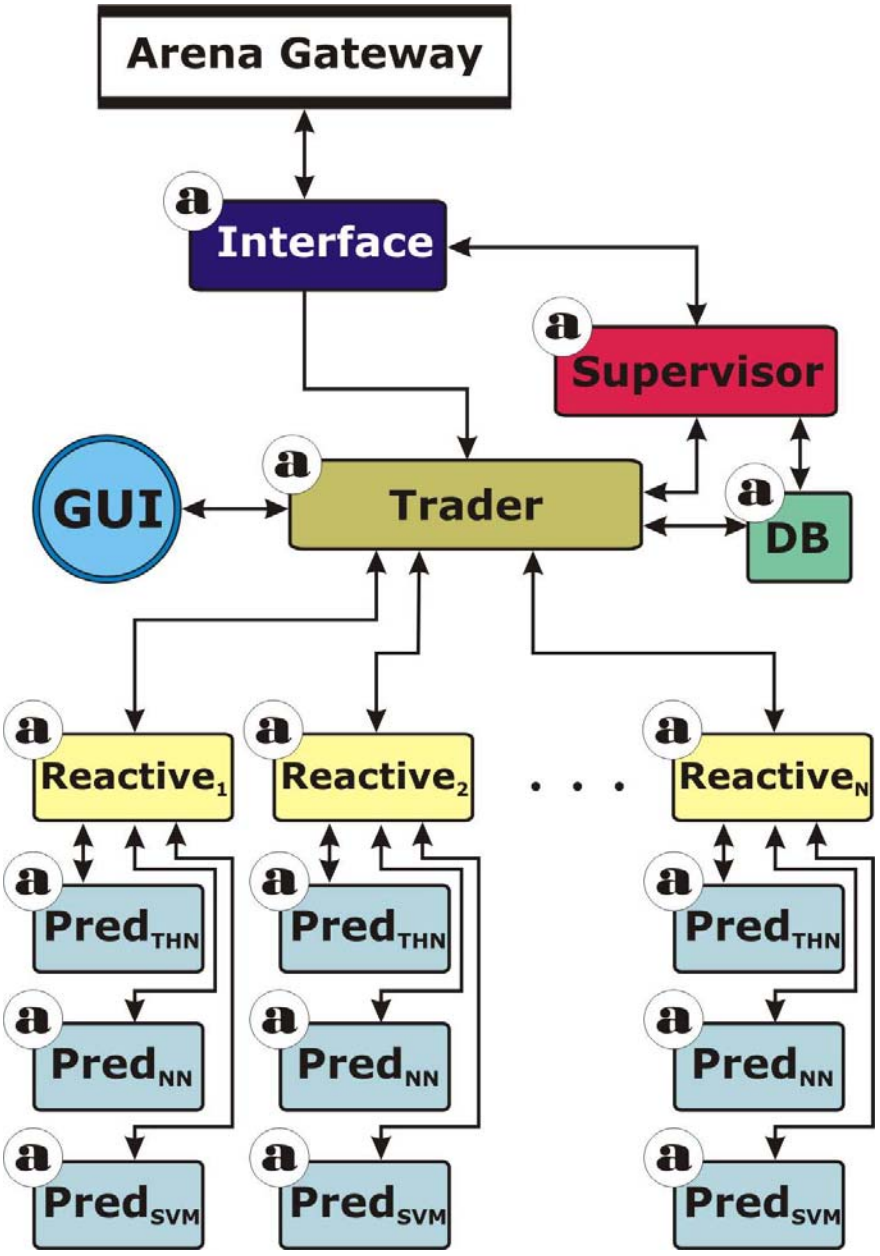
Chapter 4. Multi-agent system for automated trading using computational intelligence techniques

This chapter describes in detail the architecture, the implementation and operation of the multi- agent system that was in fact the subject of research. This is the first and currently the only functional system of automated trading conducted in Romania, dedicated to the work carried out by Market Makers on the Bucharest Stock Exchange. The system operates in real time, is connected to the Bucharest Stock Exchange through a dedicated interface, a Gateway Arena and allows bidirectional communication via a continuous data flow with trading servers.

The chapter describes the activity of a market maker, the rules by which it is conducted, the system constraints and the global mechanism through which the connection and communication is made with servers of the Bucharest Stock Exchange.

The picture below shows the multi-agent system architecture. The system is connected to the BVB trading system through a Gateway Arena. Bidirectional coupling is achieved through the agent interface that forwards data received to the Trader agent and to the supervisor agent. The Trader Agent and the Supervisor Agent are connected to a MySQL database through a specialized agent, the DB agent. For each issuer for which actions operate the Market Maker, the Trader Agent is connected to a reactive agent that receives from this market information, and communicates to the subordinated Pred agents

and decides based on the analysis that it accomplishes if the orders placed on the symbol should be canceled or not. The chapter contains a detailed overview of the features and workflow of each agent of the MaS. It also addressed the hierarchical structure of communication and how agents communicate in the system.



Agent system

The chapter also contains a section dedicated to the system behavior and performance, recorded under high volatility conditions caused by unpredictable events.

The system includes three agents specialized in trend analysis and prediction: one who uses technical analysis algorithms, one which uses neural networks and a third using support vector machines. At the end of the chapter we show the ways of how predictive agents are trained and details on data used for their training are given. In the previous sections we have exemplified the use of two of the most powerful computational intelligence techniques in terms of predictive accuracy: neural networks and support vector machines. Their implementation was based on two different software tools: MATLAB language and RapidMiner package.

The chapter ends with a brief overview of the benefits that can be obtained by combining at a functional level two complementary software architectures, both designed in Java: the JADE multi- agent platform and the RapidMiner package, which provides a wide quasi-complete pallet of data-mining methods, including those based on computational intelligence.

5. CONCLUSIONS OF THE SCIENTIFIC RESEARCH

The results achieved in the implementation of multi- agent system for automated trading shown in the fourth chapter clearly indicate that agent-based architectures can form the basis of achieving reliable applications in an extremely dynamic field as the financial transactions one.

Intelligent agents have the ability to acquire knowledge through problem solving processes. The study of the social behavior of agents in the cognitive science is an important part of the intelligent agents.

Software agents focus on interactions and collaborations to achieve their objectives in a randomly changing context. The features that suggest the use of agents in complex systems are primarily adaptability, autonomy and collaboration. Autonomy is a distinctive property of an agent and it implies the agent's ability to survive in a changing environment. An agent has the ability to perceive environmental factors and make decisions on how to react accordingly. Adaptability assumes the existence of the required learning capacity to adapt decisions based on past experience. Collaboration between agents makes them so they can be designed to address various aspects of solving a problem, each agent being designed in the most appropriate paradigm to solve its part of the overall system tasks. The coordination of the behavior of independent agents represents a central part of multi-agent system design as well.

Agents have the ability to efficiently process local data and communicate with other agents when necessary and when the tasks that they are facing are beyond their scope of knowledge. Multi-agent systems are used in a wide range of applications such as e-commerce, e-learning, communication, data mining, simulation, robotics, transportation systems and grid computing .

Produced by the Bucharest Stock Exchange and made available to intermediaries operating on the capital market in Romania, Gateway Arena is a complex application that facilitates the transfer of messages between the central system and the applications dedicated to the exchange of the participants. BSE offers intermediaries a system of servers connected through secured transactions based on fixed addresses, at access points called Gateways. Via sockets, a Gateway connects to the user's applications, negotiates connection to the system and provides a two-way flow of data that is required for trading. It offers service request/response, event-based services, and connectivity. Using a system of XML messages transmitted over the network, Gateway Arena receives commands and requests from dedicated applications belonging to participants, sends them to the central stock market system and provides answers and market data from the applicant.

The multi- agent system connects to BSE via Arena Gateway allowing bidirectional transfer of data between specialized agents and trading servers.

In the case of agents, the message body is interpreted as an XML formatted text. The XML message structure is fully described using an XML schema file provided by the stock exchange that allows connection with the application. Each outgoing message will be reviewed and validated using the schema file. If it detects a message that does not comply

with the scheme constraints, Arena Gateway sends an error message to the multi-agent system and ignores the message.

The implemented multi- agent system architecture has a complex structure consisting of several types of agents performing specific tasks and interactions. The system enables easy scaling and adding of new methods of data analysis in real time. Although the agent authentication within the system is performed using the IP address of the host computer, for privacy reasons its implementation was done on a local network.

The bi-directional coupling to Gateway Arena is achieved through the agent interface that forwards data received via the Trader agent and supervisor agent. The Trader Agent and Agent Supervisor are connected to a MySQL database through a specialized agent, the DB agent. The Trader agent is connected to a reactive agent, which receives the information from the trader agents and passes them on to the Pred subordinated agents and then decides based on their analysis if the orders are to be canceled or not.

The graphical interface that comes with the multi-agent system allows the tracking of international trends, setup modification and the overall behavior of the system. Usually placed on the human trader 's workstation, it facilitates the supervision of the entire process. Critical events are reported through text and auditory messages. Trader agent knows the exact state of the market through the constant updating of the lists containing the active orders in the market for issuers selected by the user. Orders list is automatically updated based on data received from the agent interface, the list being loaded only when the interface agent interrupts signal communication with the trading server and sends the data. Based on signals received from Reactive agents, the Trader agent manages operations for the Market Maker symbols.

Three ways in which the agent acts as Market Maker Trader have been defined: Real, Fake and Undercover. For each managed symbol either one of the three work modes can be established. In the Real mode, the automatically issued orders are entered into the BSE, with all the consequences of the portfolio and the funds arising from this action. To work in real mode, the trading system participant must request and receive permission from the BSE, following the signing of a contract and assuming obligations. In real terms, the pairs of orders are entered into the system through a single command.

In the Undercover mode, their capacity to act as Market Maker can be tested, without being registered as such on BSE and without being obliged to comply with conditions of exchange in terms of spread, the time for which orders are active in market and the quantity from the order. This mode has been used in the second phase of the test. The procedure has been used extensively in the Fake mode in the first phase of tests because it does not involve financial risk, and does not require the existence in the shareholder’s account of the stocks that can be sold. The major difference from the first two modes is the fact that in this case the orders are not placed in the real market, their introduction is only simulated. The agents behave as if these were real orders, with the difference that the potential transactions are not real but simulated. Thus, a purchase transaction is deemed to be carried out if the lowest sale price has reached or fallen below the price of the purchase order or whether it was a transaction in the real market at a price less than or equal to the order. Similarly, a sale order shall be made if there was a transaction in the real market at a greater or equal price with the one of the order or if the best buy price has reached or surpassed the market price of the sale.

As stated above, the designed and implemented agent system functions independently and requires no operator intervention unless there are serious hardware malfunctions present (communication failures, power failure etc.). However, to minimize

the risks inherent to the use of an automated trading application and to meet legal risk management procedures, the interface agent has a complex system of displaying information and communication operations carried out to the human trader. If desired, the human trader can invalidate decisions taken by the system and can modify its parameters in real time for the system to act in the desired direction. However, the system has proven to be very stable so far and the operator's intervention was never needed .

Reactive agents are decision agents who continuously receive information through the trader agent about transactions on the market and make the decisions of canceling orders for the symbols managed by the agent system. For each symbol in the list, the agent allocates a Trader agent and its subordinate structure, made up of prediction agents. At the moment there are three types of such agents: Pred-THN, which uses a specific technical analysis predictive model, Pred-NN which uses a prediction model generated using neural networks and Pred-SVM that relies on vector support machines. Prediction agents are fed with real-time market data from the reactive agent to whom they are subordinated. Prediction models upon which agents PRED-NN and PRED-SVM provide predictions are updated at the end of each week with the last period's trading data and are loaded into the system when it is not connected to BSE in XML format. A further development of the system could be achieved by expanding the number of prediction agents to reason based on other models.

Right now the system operates on five of the most important issuers listed on the Bucharest Stock Exchange: Fondul Proprietatea SA, SIF Oltenia SA, OMV Petrom SA, CNTEE Transelectrica S.A. and Banca Transilvania S.A. Adding or removing an issuer is done through a simple selection from the list, the only constraint being that the system must be disconnected from trading servers during the time of the selection. Obviously, adding a new issuer must be preceded by training the prediction agents based on the trading data of the new issuer. During the tests the simultaneous monitoring of ten issuers was simulated, without this leading to a decrease in system performance. Moreover, none of the intermediaries who act as Market Makers can do that for more than five issuers, due to the restrictive conditions which they must fulfill as regards the financial capacity and the minimum portfolio which it must have.

For each issuer, the exogenous variables are the trading volume and the values of the two stock indices representative of the issuer or those in which the issuer's share is the highest (Săndiță and Matei, 2015).

As discussed in chapter four, the supervised training of the agents aims to equip them with the ability to make predictions at the end of a process of supervised training. Training involves learning from examples, through the collection of data from a training data base, the ultimate goal being to inductively get a validated prediction model that can be used later by a specialized agent. The type of the prediction model corresponds to the nonlinear, dynamic and unstationary nature of stochastic processes, otherwise intrinsic nature of financial time series. Therefore, we have considered the NARX discrete dynamical models type to be suitable. Each agent is connected to a set of input variables. When the values of these variables get refreshed, reaching its port of entry, the agent is activated and processes the input signal using the prediction model that it has been provided with. Based on this, it will infer an output value representing the prediction, and then transfer it to the upper level agent who will use it to make a decision.

The model is delivered offline, using the XML format and syntax . Agents read the script file that describes the model and have an interpreter and a runtime module for calculating the prediction based on the model and its use in the decision making process .

Integrating MAS classes and RAPIDMINER operators provide operational phases of data acquisition and burn them in the SQL database system, identifying and estimating the NARX model, using, if necessary, supervised learning techniques based on neural networks or on support vector machines.

Respecting the principle of parsimony, the obtained models should contain only the necessary modeling. To avoid over-adjustment that occurs when a neural network excessively specializes, an early stopping technique was used, involving shared technical data available in three subsets: the training subset, the validation subset and the test subset. The neural network training subset is used to calculate the gradient and update the network weights. The subset of validation is used only in training to decide when it is to be stopped via error monitoring for the validation subset during the training process. The error associated with the validation subset normally decreases during the initial training phase, in the same way as the error for the training subset.

When the error for the validation subset rises for a specific number of iterations, the training process is stopped and the weights that are returned are the ones which correspond to the minimum validation error. This way the network model with the best performance for the validation subset is selected. Obviously, in order to achieve untamed results, the test data subset is selected from a set of input data that has not been used in the training or the validation process. This strategy was implemented using the MATLAB application that has specific functions required to process this strategy.

In the second chapter of the thesis I have analyzed two of the most used platforms for agents: Cougaar and JADE. Cougaar implements a complex model of agents, the tasks that they can perform and the method of communication between entities that operate within the system. The Cougaar implements a specific mechanism, strong insulation and encapsulation for all the components. Major impediments that made me quit using this platform for the implementation of the agents were the ones presented as the platform's essential quality, respectively the over specialisation in error tolerance and generality, elements which would have unacceptably complicated the system.

Regarding the JADE platform, respecting the FIPA standards and mechanism of communication between agents determines slow system reactions to events which happen quickly in the market. In this case, the general nature reduces the performance of the system in terms of processing speed , which is unacceptable in an automatic trading system.

Although the multi- agent automated trading system was not developed on a standardized platform, the study of agent-oriented platforms and of their performance determined the implementation, in the constructed system, of mechanisms similar to those installed on the platforms. We are talking about how to identify agents through a mechanism similar to the White Pages and asynchronous messaging communication between agents. Also, predictive agents use different threads to perform their duties .

Multi-agent systems can benefit from the development and refinement of computational intelligence techniques to extract knowledge from inductive databases, to generate predictive models and automatic instantiation of predictive agents able to use these models to make predictions. As I tried to detail above, the predictive model can be generated at the end of a supervised learning process using neural networks or support vector machines. RapidMiner is perfectly suited to perform this activity. Its essential advantage is that the model obtained by supervised training can be described and saved in XML code. An agent that has a syntactic analyzer can then interpret the XML code and execute the model, thus fulfilling a predictive function. Such an approach enables the automatic instantiation of a whole class of agents within the predictive multi-agent system.

Thus we made predictive agent classes which interpret and execute predictive models based on neural networks and support vector machines. Both the model XML parser and the executive (the prediction module) can be designed to perform general tasks. This way, a predictive agent equipped with an interpreter and an executive oriented towards the neural networks will be capable to syntactically analyze any neural model and make predictions based on it.

In this way, the integration of JADE and RapidMiner systems into a shared platform provides the premise for the implementation of other prediction models in the system. Such an integrated system is flexible by nature, enabling automatic instantiation for the predictive agents and their use in the various decision-making processes. Such an agent may change its built-in predictive function simply by changing the model that processes it, given that the model represents a function exterior to the agent (such as a plug –in, used in the Cougaar platform). Obviously, changing the predictive function by adding another model will have to be accompanied by a potential adjustment of the format for the data structures which are to be transferred for processing to the new model.

Currently, the automated trading system is installed and working for a sole intermediary. We hope that early next year we will implement it in two other locations so we can monitor its behavior when it is competing with itself. Due to the fact that it is equipped since the initial versions with an action logging mechanism, which allows the offline behaviour study of each system agent, we can better evaluate the agents' reactions to outside stimuli.

6. SELECTIVE BIBLIOGRAPHY

Feng Y., Yu R., Stone P., *Two Stock-Trading Agents: Market Making and Technical Analysis*, Agent Mediated Electronic Commerce V: Designing Mechanisms and Systems, Volume 3048 of Lecture Notes in Artificial Intelligence, Springer Verlag, 2004;

FIPA website, *www.fipa.org*, 22 iul. 2015;

Green S., Hurst, L., Nangle, B., Cunningham, D. P., Somers, F., Evans, D. R., *Software agents: A review* (Tech. Rep. No.TCS-CS-1997- 06). Trinity College Dublin, Broadcom Éireann Research Ltd. 1997;

Helsinger A., Thome M., Wright T., *Cougaar: A Scalable, Distributed Multi-Agent Architecture*, Proc. of the IEEE International Conference on Systems, Man and Cybernetics, The Hague, 2004;

JADE website, *http://jade.tilab.com*, 22 iul. 2015;

Kim A.J., Christian R. Shelton C.R., Poggio T., *Modeling Stock Order Flows and Learning Market-Making from Data*, AI Memo 2002-009, 2002;

Lange D., Oshima M., *Programming and Deploying Java™ Mobile Agents with Aglets™*. Addison-Wesley, 1998;

Nevmyvaka Y., Sycara K., Seppi D.J., *Electronic Market Making: Initial Investigation*, Artificial Intelligence in Economics and Finance, World Scientific, 2005;

Nwana H. S., *The potential Benefits of Software Agent Technology to BT,* Internal Technical Report, Poject NOMADS, Intelligent Systems Research, AA&T, BT Labs, UK, 1996;

Odell J. J., Parunak H., Bernhard Bauer B., *Representing Agent Interaction Protocols in UML*, Volume 1957 of the series Lecture Notes in Computer Science, pp 121-140, 2002;

Ricordel P.M., Demazeau Y., *From Analysis to Deployment: a Multi-Agent Platform Survey*, ESAW:pp. 93-105, 2000;

Săndiță A.V., Durac C., *Market Maker şi Sisteme Multi-Agent în cadrul Bursei de Valori Bucureşti*, Finanțe, provocările viitorului, nr. 17, Universitatea din Craiova, 2015;

Săndiță A.V., Matei Gh., *Tranzacționare algoritmică la Bursa de Valori Bucureşti*, ANNALS of the University of Petrosani, ECONOMICS, vol. XV, 2015;

Stone P., Sherstov A.A., *Three Automated Stock-Trading Agents: A Comparative Study*, Agent-Mediated Electronic Commerce VI, Theories for and Engineering of Distributed Mechanisms and Systems, Volume 3435 of the series Lecture Notes in Computer Science, pp 173-187, 2005;

Vapnik, V., Chervonenkis, A., *On the uniform convergence of relative frequencies of events to their probabilities*.Theory of Probability and its Applications, 16(2):264-280, 1971;

Wooldrige M., *Agent – based software engineering,* IEE Transactions of Software engineering, February, 144(1), 26 – 37, 1997;

Yom-Tov, E., *A distributed sequential solver for large-scale SVMs*, L. Bottou, O. Chapelle, D. DeCosta, and J.Weston, editors, Large-Scale Kernel Machines, pp.139–154, Cambridge: MIT Press, 2007.