The Informatics of the Equity Markets - A Collaborative Approach

Claudiu VINȚE
Opteamsys Solutions, Bucharest, Romania
claudiu.vinte@opteamsys.com

This paper aims to provide a high-level overview upon the information technology that supports the electronic transactions performed on the equity markets. It is meant to offer a succinct introduction to the various technologies tailored to tackle the data transfer between the participants on an equity market, the architectural approaches regarding trading system design, and the communication in a collaborative distributed computing environment. Our intention here is not to provide solutions, or to propose definitive designs, merely to scratch the surface of this vast domain, and open the path for subsequent researches.
Keywords: *securities exchange, stock order flow, trading system architecture, distributed computing, middleware, collaborative system, order-matching algorithm*

1 The role of an equity market and its participants

From an overall economic and social perspective, the capital flows among the economic agents at multiple layers. In this paper, we will focus on the value chain of a securities exchange, and the information technology that nowadays drives this environment as a whole. A securities exchange is an organized and supervised market place where trades are made based on an approved set of rules and regulations [1]. We use the term broadly, referring to dealer networks as well as to traditional exchanges.

A securities exchange is a secondary market. A primary market is where newly issued shares are publically offered through what is commonly known as *initial public offering* (IPO). In trading securities on a stock exchange, members may act either as brokers (agents), or as dealers (principals), or as both (Figure 1).

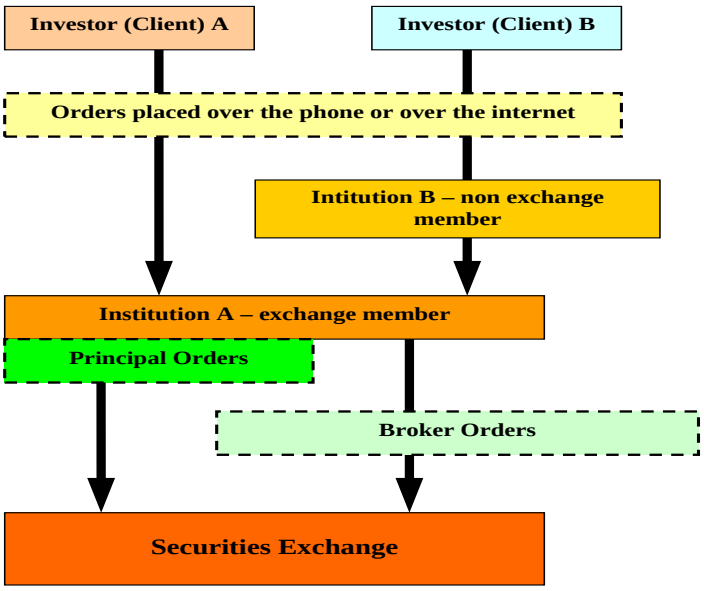

Fig. 1. The order routing, from the investors to the securities exchange

The securities traded include stocks, bonds and warrants. A listed company issues shares of a stock. Orders for trading those shares originate with investors, and are brought to an exchange by intermediaries. The confirmation of each trade by the stock exchange goes both to members and to settlement organizations and central counterparty (CCP). The settlement organizations subsequently confirm settlement of the trades to the stock exchange members, who in turn send confirmation to the investors, the end buyers and end sellers. The settlement organizations, such as Euroc-

lear, Clearstream, and Sega Inter Settle (SIS) in Europe, and Depository Trust & Clearing Corporation (DTCC) in the United States, maintain technical interfaces with the stock exchanges and their members, and they organize the exchange of cash and securities on a delivery-versus-payment basis. The central counter-party is contractually interposed between the trading parties, and it provides post trade anonymity, netting and counterparty risk management services. All these parties and the interactions between them compose the value chain of a securities exchange. Figure 2 provides an overview upon the elements of the value chain and the connections between them.

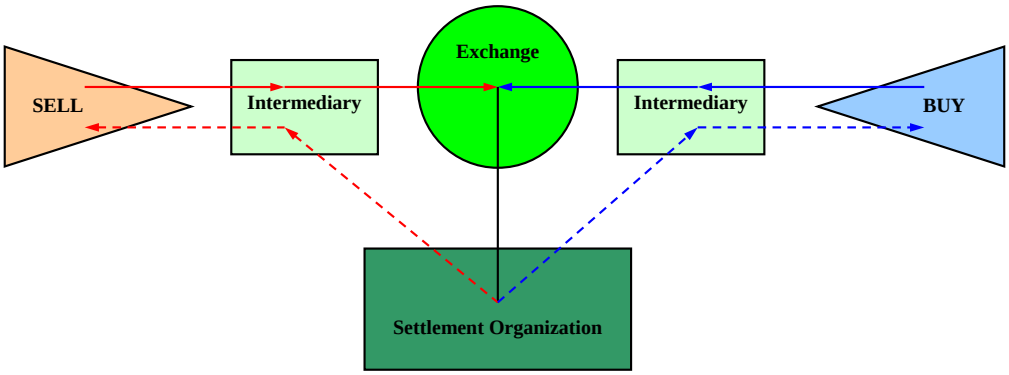

Fig. 2. The elements of an equities market, and the interactions between them

2 The elements of the value chain

The listed companies – the issuers, the companies who are obtaining equity financing through new share issuance, supply the securities traded on a stock exchange. As mentioned earlier, the initial issuance in the case of shares constitutes what is designated as primary market. By listing on an exchange through an *initial public offering*, a company that has previously been in private hands goes public. An IPO involves fixing the issuing price and placing shares with investors, usually via a consortium. The issuing price established by the financial advisors is an indication of the initial price determined at the stock exchange. However, the initial price set by trading after an IPO may differ from the IPO price, depending on the accuracy with which the IPO price was set in the first place, and the distribution of forces (economic interest from the investors, ultimately) in supply and demand on the secondary market [2].

Exchange – a stock exchange is chiefly perceived as a secondary market. Exchanges are interactive, information-driven, and volume-driven marketplaces. An exchange is the formal place where supply meets demand in an organized, regulated environment. The overall objective of an exchange is to attract liquidity, to execute orders with reasonable speed and at minimal cost, and to find appropriate prices for customers (members and investors). Official exchange prices are set, following approved rules and regulations, under the surveillance of special regulators. The main responsibility of an exchange is fair and orderly price discovery for already issued securities, thus facilitating their exchange.

Investors – the investors include individuals (retail customers) and institutions (including mutual funds, pension funds, and insurance companies) who are interested in buying and selling securities, and who, in order to achieve this, hold cash and securities accounts with intermediaries.

Intermediaries – the intermediaries are exchange members who are either brokers acting on behalf of investors as agent or dealers acting on their own accounts as principal. Generally, the intermediaries are supervised banks and financial institutions that have to meet specific regulatory requirements in order to operate. A broker is an exchange member who acts as an intermediary between an investor and an exchange. Brokers do not take orders on their own books, but merely route them to a market. A broker acts as an agent, on his or her name, on the client's account. Some intermediaries also supply marketability services to public investors by trading in their own names on their own accounts. When an intermediary plays this role, he or she is termed a *market maker* or *dealer*. The *market maker* is contractually committed to provide liquidity to a marketplace, by putting up quotes on both buy and sell sides of the market. The market maker is also required to maintain a fair and orderly market. A dealer does not have such obligation. In most of the markets, there is a separate intermediary with whom the broker-dealer entities interact to effect settlement, and who acts as custodian for maintaining the in-

vestors' cash and securities accounts.
Settlement organizations – the settlement organizations ensure delivery versus payment of the traded securities and payment of the money within a predefined period of working days, usually two or three working days (T+2, T+3), in both United States and Europe. A physical exchange of cash and products rarely take place anymore, as the securities exchanged are all standardized and dematerialized [3] [4].
Central counterparty – as we have mentioned previously, a CCP is contractually interposed between the trading parties, and provides post trade anonymity, netting and counterparty risk management services.

3 The Information Technology perspective

Having given the diversity and the sheer number of the investors, the multitude of financial instruments and the volume of transactions, the modern equity markets rely greatly on information technology in order to fulfill their economic and social role. Essentially, there is no aspect of the value chain, succinctly introduced above, which does not involve electronic data communication, at the very basic.
At this point, we would prefer to make a distinction, and identify two generic areas, not necessarily disjointed, where the intrinsic models of the information technology sustain the equity market:
- *transactional area* - concerning the whole flow of order routing, order management, order matching, trading executions, allocations, trade generation, client confirmations and settlement;
- *valuation and analysis area* - involving models for equity data mining, price and volume analysis, derivatives pricing, portfolio risk modeling etc.

Having given the space allocated to this paper, we will focus in the following pages on the Informatics related to the *transactional area*.
Therefore, in this delimited context, Figure 3 depicts, at very high-level, from the information technology perspective, the dataflow related to the trading activities that interconnect all the participants on the equity market. At the broker's level, the functional components of the trading system consist of:
- client interfaces;
- Order Management System (OMS);
- dedicated lines to various securities exchanges;
- trade generation;
- client confirmations;
- settlement and clearing.

The investors initiate the flow by placing buy or sell orders with their brokers. The orders are placed electronically, by the means of a client application, which is usually supplied to the investor by its broker. The client application may be either a secured webpage, which facilitates order data transfer to the web server side hosted by the broker, or a dedicated client, supplied by the broker, that connects using TCP sockets, tunneling data through a proprietary API, or an industry standard like FIX (Financial Information eXchange Protocol) to the broker's server.
Once the client orders reach the broker's trading system, they may be introduced into the Order Management System (OMS) or channeled directly to the exchange (Direct Market Access – DMA), depending of the contractual agreement between the investor and the broker. In both cases, the investor's orders are sent by the broker to the securities exchange through dedicated communication lines that employ proprietary APIs, normally over TCP/IP connections.
Without entering into too many details for the purpose of this paper, there may be distinct lines supported by the exchange for receiving buy/sell orders from the brokers/dealers, and for sending back executions (order match results). The execution data, received by the broker/dealer back from the exchange, it is put in connection with the original order data within the Trade Management System, and the actual trade data is generated. In case the investor has multiple accounts opened with the broker, in order to have an appropriate trade generation the investor has to supply to the broker what is known as allocation schema. That is commonly the case with an institutional investor. The allocation data consists in instructions upon how the execution results are to be distributed by the broker on the investor's accounts. The investor sends the allocation instructions to the broker by similar means as the buy/sell orders. There are cases where the allocation instructions may apply to the entire trading activity of the investor for a certain day, and not attached to each individual order. In this situation the allocation data flow, between investor and broker, may be arranged in a simplified manner (upload a file containing the allocation instructions on the broker's website, or running a FTP based batch, for instance).
After the trade is made, within the broker/dealers trading system, there is subsequent data processing in order to generate settlement instructions that have to be sent to the settlement organization. The settlement organization will then ensure delivery versus payment and pave the

way for clearing between the participants on the securities marketplace.

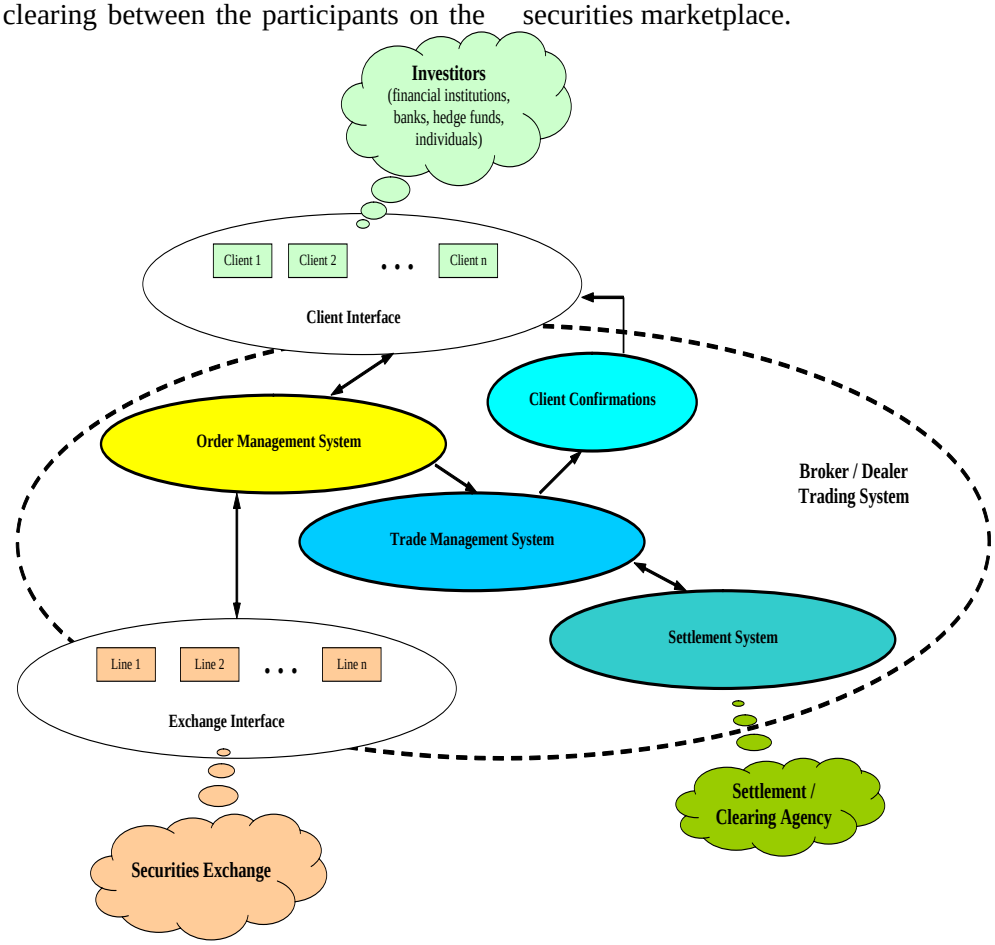

Fig. 3. The general data flow between the participants on the equity market

Figure 4 illustrates the data communication technologies employed for transferring messages with financial content between the participants on the equities market.

It is worth pointing out that the technology differs considerably, depending on the factors like:

- data volume, and frequency of message exchanges;
- security measures required for privacy protection of the sensitive data;
- data transfer speed requested for orders to reach the exchange, and the response time for acknowledgments;
- novelty and accuracy of the market data (see aspects related to price discovery);
- available industry standards and common practices among the participants.

Considering the above, there can be identified various data flows channeled through specific technologies.

The flow between *investors and intermediaries*:

- for fast connections and *direct market access* (DMA): TCP sockets and industry standard (FIX) or broker proprietary protocols **[5]**; client-server architectures over TCP/IP;
- client application running in a web browser, on the investor's side (HTML, SOAP-stub, HTTP GET/POST, JavaScript, Java Applets, Java RMI (stub) etc.), having as correspondent functionalities offered through a web server on the broker's side (PHP, SOAP-skeleton, Java Servlets, Java RMI (skeleton) etc.).

Flow between *broker/dealer and securities exchange*:

- dedicated and highly secured communication lines, employing an exchange specific protocol over TCP/IP.

Flows between *exchange, broker/dealer and institutional investor, on one side, and settlement, clearing organization, on the other side*:

- industry standard protocols like FIX protocol or the financial messaging network offered by SWIFT (Society for Worldwide Interbank Financial Telecommunication) [5], both over TCP/IP.

Price data flow between a securities exchange and the investors:

- real-time quotes distributed via proprietary mechanisms like NASDAQ ITCH, normally using UDP sockets;
- delayed price data fetched by clients using

SOAP, HTTP GET/POST.

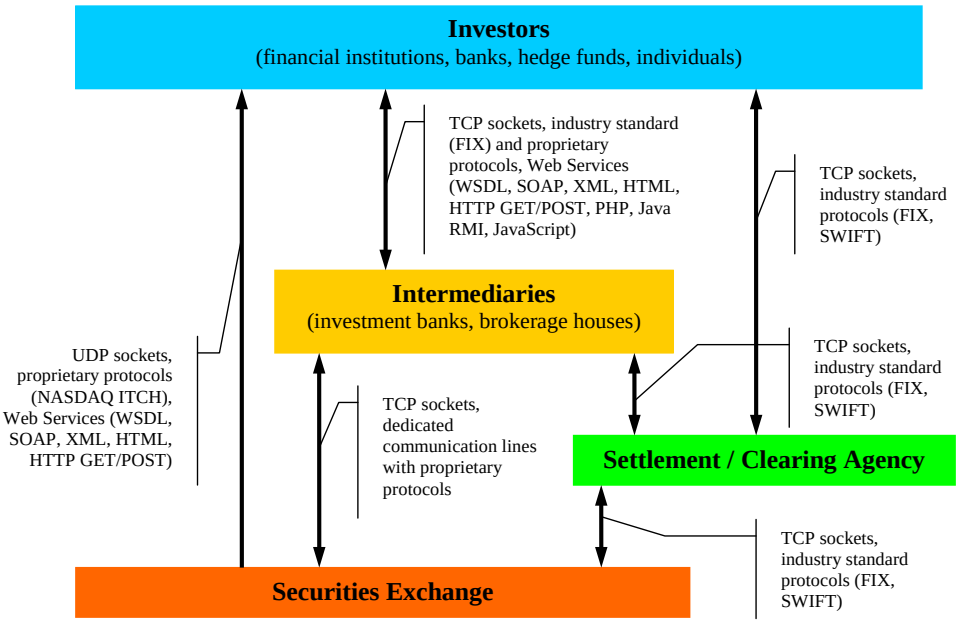

Fig. 4. The technologies employed for data communication between parties

4 Upon a collaborative architecture of a trading system

We have presented briefly the data flow and the technologies commonly employed for transferring data between the participants. Now, we will take a closer look on how the data processing is organized within a brokerage house, the place where the investor orders are put in connection with the exchange executions and where the electronic trades are generated.

We will focus on the order management system (OMS), since this area influences decisively the interface between the investors and the brokerage house. The literature regards it as the *Front Office* [6]. There are various architectural approaches for handling the investors' buy/sell orders, sending them to the market, and processing the executions received back from the exchange. Each component can supply its functionality within a cascade type of processing model, where each module enrich the data from a certain perspective and then pass it to the next module along the overall processing chain [7]. Within such architecture, the system modules are tightly coupled, and a malfunction in one of the chain's components may bring down the whole processing flow. Over the years, the tendency has been oriented toward architectures with loosely coupled modules, where the functionality is provided based on *publisher-consumer* paradigm. That potentially involves a multitude of service providers and consumers, all interconnected through a communication platform called *middleware*. In this architectural approach, the *middleware* handles the entire communication layer, which, otherwise, would have had to be embedded in each of the system's components [8] [9] [10]. The communication layer provides a common *application programming interface* (API) that allows for a standardize way of exchanging messages between the functional modules of the system. In this manner, the application programmer can freely focus on the functionality deliverable by the application, rather than on how to interconnect it with other components of the trading system. Figure 5 shows such a generic architecture for a trading system, each component being plugged in system through a *network daemon* (ND) which ensures the creation of a virtual network dedicated to the trading system. The ND is a middleware component, and commonly serves as a general purpose platform for process communications in a distributed computing environment. In the larger family of distributed applications, collaborative systems are distinguished by the fact that the agents in the system are working together towards a common goal and have a critical need to interact closely with each other: sharing information, exchanging requests with each other, and checking in with each other on their status. The applications that compose the system may be pure client applications, or play, simultaneously, the role of client and server. The *network daemon* has to be present and running on each physical machine part of the system virtual network, in order to allow the applications resid-

ing on the machine to access other resources within the system. Each application has to subscribe to a set of messages that is interested in and/or publish its availability in offering specific services. The messages between applications are channeled *point-to-point* using TCP sockets, broadcasted to all applications within the system via UDP sockets, or delivered to a subset of applications through a multicast implementation [8].

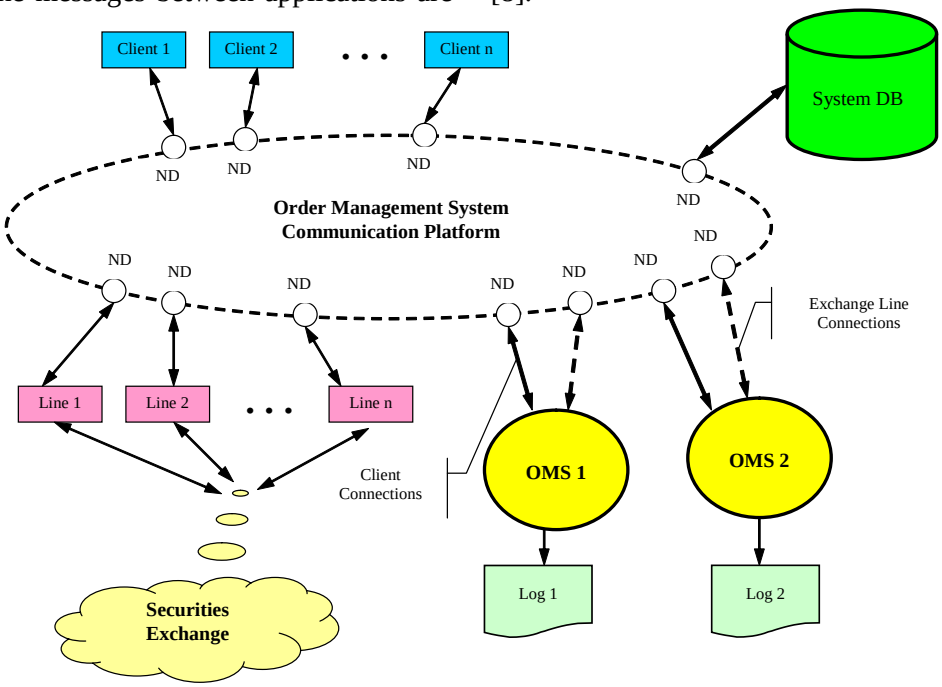

Fig. 5. A generic distributed architecture of a trading system based on publisher-consumer paradigm

There are commercial solutions on the market that offer this kind of *middleware* functionally, such as TIBCO Rendezvous. It offers APIs for Java, C, C++, C#, Perl, and COM. Opting for such a generic platform may be costly.

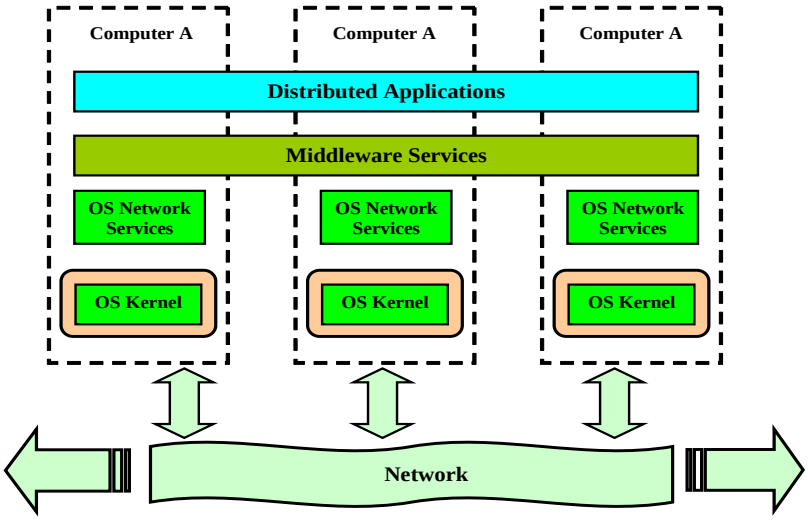

Fig. 6. The location of a middleware component in distributed computing environment

Many trading system providers or brokerage houses though, are tempted to opt for developing an in-house solution, better tailored for their specific environment [9] [10]. Figure 6 illustrates a middleware platform that conceals the communication specific aspects from the application layer. The *middleware* isolates the applications from the operating system specific network services, and encapsulates all the necessary functionality for delivering a transparent messaging service [11].

There are alternatives supported by components created in the open-source community, based on Java Message Service, to commercial products

like TIBCO Rendezvous [12] [13]. Sun Java System Message Queue is an affordable, standards-based messaging solution based on the open-source Open Message Queue project. It dramatically reduces the time needed to develop distributed applications because it handles all the inter-process communication aspects of an n-tier architecture. Normally, a *middleware* does not need to provide persistence for the messages it handles, but sometimes persistence is necessary or even required. This aspect is still debated in the scientific community; since persistence seems to not be a functionality to implement at the *middleware* level and may impair the overall response time. On the other hand, there are situations when, due to unexpected failures, the entire system has to be brought back to the state prior the temporary suspension. Trading systems tend to need coverage for the latter scenario.

5 Collaborative developments employing a simulation matching engine

Most of the modules within a trading system have to be designed, implemented and tested in an environment that reproduces faithfully the real-time connectivity to a securities exchange.

Usually, the stock exchanges offer to their members testing environments available in non-working days. There can be offered availability, for limited testing activity, even during the working days, in specially insolated environments. In order to satisfy their needs for conducting collaborative tests, financial software providers and the brokerage houses require a dedicated, readily available environment for trading simulation. Figure 7 shows how the components of the simulation environment can *replace* the functionality of a real stock exchange order-matching engine.

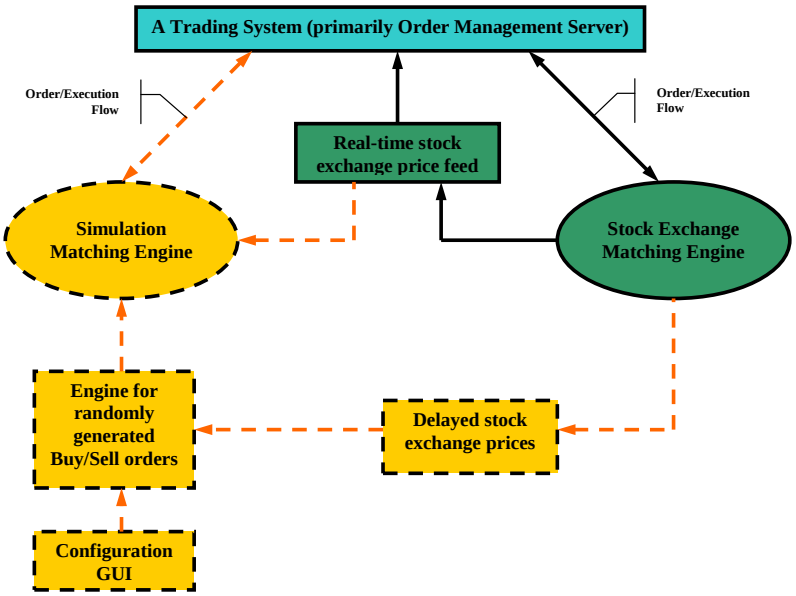

Fig. 7. A simulation order matching system, in connection with a real OMS

The essential data that has to be send to an order-matching engine consists of:

- financial instrument identification code (stock symbol), notated herein with **ID**; which follows a stock exchange specific coding rules;
- targeted stock exchange or section of a stock exchange, **E**; represents the market that the buy/sell order is intended to reach;
- the actual side of the order, buy (bid) or sell (offer) intention, notated with **B** or **S** respectively;
- the price at which the order is desired to be executed, if it is a limit order, or zero in case the order is to be executed at the market price (**p**);
- order quantity, which designates the number or shares that are intended to be bought or sold, normally a integer number, multiple of a stock specific lot size (**q**).

The actual messages, transferred between broker/dealer and exchange, have in fact a much more complex structure (data for identifying the client, other trading conditions and control data), but the above is the minimal data set necessary for the order-matching engine to accomplish its role.

The simulation order-matching engine accepts orders placed from the trading system connected to it, and try to match them with orders generated randomly, within certain parameters of price and quantity, and with a configured frequency. The

parameterization can be realized from a configuration setting *graphic user interface* (GUI). In addition to the configuration ability, a feed with delayed prices from e real exchanged (this data can be also stored and used to reproduce various trading scenarios) can provide certain limits to for the orders generated randomly. Alternatively, the simulation order-matching engine can be fed with real-time prices coming from a real stock exchange.

The order-matching engine has to maintain, for each traded symbol, an order book structured as is illustrated in Figure 8.

Once a new order reaches the electronic system the securities exchange, the order receives a uniquely identified code for the given trading day (**O**), and a timestamp (**t**), representing the moment when the order was registered in the system. At any given moment, during the trading session, the exchange system has to maintain the whole information related to the outstanding orders: the orders that either were not fully executed, nor entirely canceled by the client, and are still opened on the market for being matched with orders placed on the side. For each symbol, the exchange system has to determine a market price at the beginning of the trading session, based on an auctioning mechanism. Then, during the trading session, the price at which the last transaction took placed becomes the published market price that is to be considered for the orders sent to the exchange to be executed at the market.

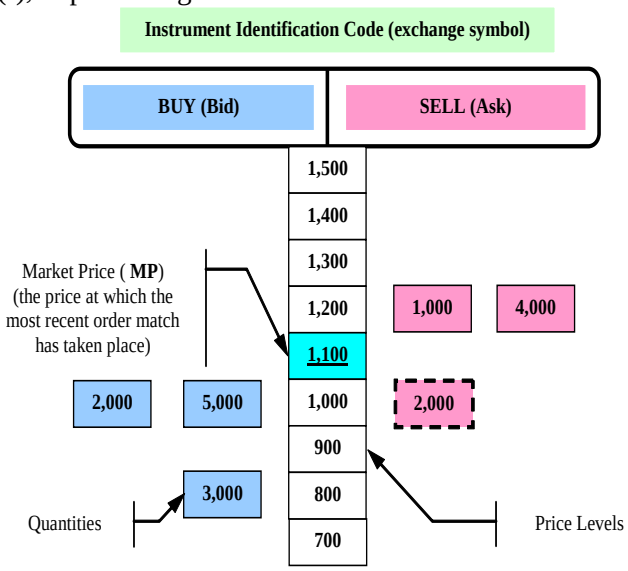

Fig. 8. The stock exchange order book with a new buy order

The order-matching engine has to offer real-time response; therefore, the whole data structure used for storing the orders sent to the market has to be modeled in the memory. The order-matching algorithm has to abide by the following principles:

- best price priority – the highest bid is considered the best price for the buy side, and the lowest ask is considered the best price for the sell side; in Figure 8, the newly arrived sell order of 2,000 shares (dotted line square), at the limit price 1,000 represents a better offer on the market than the exiting outstanding orders and will be executed as soon as the matching algorithm identifies a corresponding buy order;
- time priority – in case there a multiple orders, on the same side, placed at the same price, the orders that reached the exchange system earlier have priority;
- orders placed the market price have to be satisfied first – since the investor who placed an order at the market price takes a higher risk than the one who placed a limit order; that risk has to be rewarded through a swift execution and, as we mentioned earlier, the securities exchange system has available, at any moment, a published market price (**MP**) for each listed stock (symbol).

The data structure that we propose for the simulation order-matching engine is illustrated in Figure 9. There can be identified four levels, suggested by the principles enounced above.

At the first level are stored, for each listed stock on the exchange, the identification code **ID** (exchange symbol), the market (section) code **E**, the current published market price **MP**, and links to the outstanding buy and sell orders.

Level two contains, for each side, a linked structure of prices, sorted upon the best price priority principle enounced earlier: descending order for

the list of bids, and ascending order for the list of asks. Having this in place, assures an orderly and fast searching for the matching algorithm.

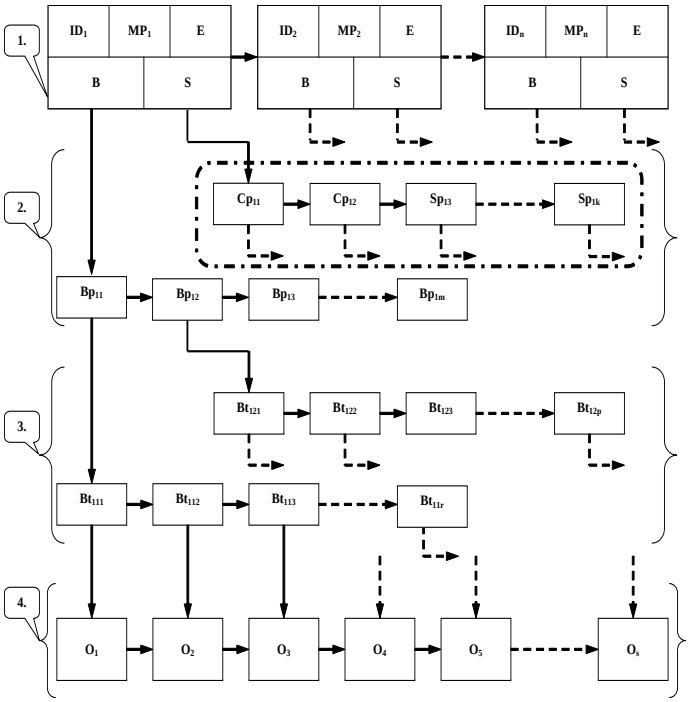

Fig. 9. The proposed data structure for the order-matching algorithm

Level three has a similar role, introducing a hierarchy based on time, among the orders placed at the same price; the earlier orders are placed in the front.

On the fourth level will be stored the entire order data (**O**), including the quantity that is to be executed or remained to be executed, the state of the order, other conditions in the case or conditional orders (*fill or kill, all or none, stop loss, immediate or canceled* etc.)

When a new order arrives on the market (reaches the simulation order-matching engine), the matching algorithms follows the steps summarized below [14].

MP ← the closing price of the previous trading day

while (trading session is open)
  insert a new order in the system data structure
  compose the searching key: {ID, (B/S), p, t}
  search for a match,
  either in Buy or Sell order space

  if (match found for the new key)
    update the correspondent quantities
    for the mathed orders
    update the status of the matched orders
    (fully executed orders are deleted from the system)
    inform the clients upon the new order match
  else
    continue

The simulation order-matching engine collaboratively interacts with the trading system connected to it and, along with the engine that randomly generates orders to offer matching opportunities in the simulation environment, accomplish the goal of creating a realistic testing environment in which the trading activity can be conducted continuously and smoothly.

6 Conclusions

The information technology employed for sustaining the modern trading activity requires components that interact closely to each other, in a collaborative fashion, for the common goal of executing the clients' orders. Our research is currently directed toward creating a collaborative infrastructure for a trading system, based on Open Message Queue. The main purpose is to provide a *middleware* capable of sustaining future experiments regarding trading strategies, by allowing collaborative interactions between various and distributed components. Through such a communication platform, the modules composing the distributed system are inheritably capable of *knowing* more about each other and, consequently, interacting more efficiently as a whole. In a simulated trading environment, a few human agents compete on resources created by computer algorithms, within a scenario-driven environment. The components that create these scenarios have to *sense* the trading patterns of the human agents, and act accordingly, in a collaborative manner.

References

[1] A. R. Schwartz and R. Francioni, *Equity Markets in Action (The Fundamentals of Liquidity, Market Structure & Trading)*, John Wiley & Sons, 2004.
[2] L. Harris, *Trading and Exchanges*, Oxford University Press, Oxford, 2003.
[3] A. W. Rini, *Fundamentals of the Securities Industry*, New York: McGraw-Hill, 2003.
[4] B. Ghilic-Micu, *Bursa de Valori*, Bucuresti: Editura Economica, 1997.
[5] H. McIntyre, *How the U.S. Securities Industry Works - Updated and Expanded in 2004*, New York: The Summit Group Press, 2004.
[6] H. McIntyre, *Straight Through Processing*, New York: The Summit Group Publishing, 2004.
[7] I. Ivan and C. Ciurea, "Using Very Large Volume Data Sets for Collaborative System Study," *Informatica Economică*, vol. 13, no. 1/2009, Bucharest: INFOREC, 2009.
[8] A. S. Tanenbaum and M. van Steen, *Distributed Systems - Principles and Paradigm*, New Jersey: Prentice Hall, 2002.
[9] C. Vințe, "Aspecte ale Proiectării unui Order Request Broker (ORB) - Partea I," *Informatica Economică*, vol. 5, no. 2 (18)/2001, Bucharest: INFOREC, 2001.
[10] C. Vințe, "Aspecte ale Proiectării unui Order Request Broker (ORB) - Partea a II-a," *Informatica Economică*, vol. 5, no. 3 (19)/2001, Bucharest: INFOREC, 2001.
[11] R. W. Stevens, *UNIX Network Programming – Vol. 1, Networking APIs: Sockets and XTI, Second Edition*, Prentice Hall, 1998.
[12] A. S. Tanenbaum, *Computer Networks - Fourth Edition*, New Jersey: Prentice Hall PTR, 2003.
[13] Sun Microsystems, *Java Message Service*. Available: http://java.sun.com/products/jms/
[14] C. Vințe, *Sisteme distribuite de asistare a tranzacțiilor bursiere – Doctorate thesis*, Library of The Bucharest Academy of Economic Studies, Bucharest, 2006.

Claudiu VINȚE has over eleven years experience in the design and implementation of software for equity trading systems and automatic trade processing. He is currently CEO and founder of Opteamsys Solutions, a software provider in the field of securities trading technology and equity markets analysis tools. Previously he was for six years with Goldman Sachs in Tokyo, Japan, as Senior Analyst Developer in the Trading Technology Department. In March 2006, Goldman Sachs acknowledged the importance of the integer allocation algorithm created by Claudiu, and filed in his name a patent application for *Methods and apparatus for optimizing the distribution of trading executions* with the US Patent Office (USPTO Application 20060224495). Claudiu graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1994, Department of Economic Informatics, within The Bucharest Academy of Economic Studies. He holds a PhD in Economics from The Bucharest Academy of Economic Studies. His domains of interest and research include combinatorial algorithms, middleware components, and web technologies for equity markets analysis.