SOA and Web Technology for Computing the Intrinsic Entropy of BSE Listed Stocks

Claudiu VINȚE[1], Ion SMEUREANU[1], Ionuț-Alexandru LIXANDRU[2]
[1]Bucharest University of Economic Studies
[2]Bucharest Stock Exchange
claudiu.vinte@ie.ase.ro, smeurean@ase.ro, alexandru.lixandru@bvb.ro

Measuring investors' level of interest for a traded equity product can provide, along with the status of the stock market as a whole, a consistent mean for building a hierarchy among the traded equities. The concept of intrinsic entropy associated to stock exchange traded equity has the ability to capture the factual perception of investors regarding the performance of a publically traded company. Our ongoing research has been conducted based on the transactions executed on the Bucharest Stock Exchange (BSE), and aims to prove that price variation weighted entropy can offer a synthetic and readily computed indicator, for evaluating direction and intensity of trading activity. This paper will focus on how trade data is captured, and stock intrinsic entropy is computed and presented within a SOA solution.
Keywords: *Service-Oriented Architecture (SOA), Message-Oriented Middleware (MOM), Java Message Service (JMS), Stock Intrinsic Entropy, Web Services*

1 Introduction

Entropy concept originated in physics, and it plays important roles in numerous other disciplines ranging from logic and statistics to biology and economics. However, there is not an unique or unified interpretation of this concept. Entropy is defined differently in different contexts, and even within the same domain, different notions of entropy are at work. Some of these are defined in terms of probabilities, others are not. Even when entropy is defined in terms of probabilities, these could be either probabilities chances – as physical probabilities - or credence - as degrees of belief [1].

Gibbs defined in 1878 the most general formula for the entropy of a thermodynamic system, after earlier work by Boltzmann (1872). The Gibbs entropy,

$$S = -k_B \sum_i p_i \log p_i \qquad (1)$$

where k_B is the Boltzmann constant, and p_i is the probability of a microstate.

The summation is over all the possible microstates of the system, and p_i is the probability that the system is in the i^{th} microstate [1]. For most practical purposes, this can be taken as the fundamental definition of entropy since all other formulas for S can be mathematically derived from it, but not vice versa.

The most general interpretation of entropy is as a measure of our uncertainty about a system. The equilibrium state of a system maximizes the entropy because we have lost all information about the initial conditions except for the conserved variables. This uncertainty is not of the everyday subjective kind, but rather the uncertainty inherent to the experimental method and interpretative model. The interpretative model has a central role in determining entropy. In other words, the set of macroscopic variables one chooses must include everything that may change in the experiment; otherwise, one might see decreasing entropy [2]. We will see later on that the number of states observed or taken into consideration can have a significant impact on the value of entropy, and in contouring of a certain conclusion or another during analysis.

In information theory, entropy is a measure of the uncertainty associated with a random variable. In this context, the term usually refers to the Shannon entropy [3], which quantifies the expected value of the information contained in a message, usually in units such as bits. In the context of

communication, a message means a specific realization of the random variable.

Equivalently, the Shannon entropy is a measure of the average information content one is missing when one does not know the value of the random variable. The entropy of a message is in a certain sense a measure of how much information it really contains.

In finance and economics literature, entropy and information theory analysis experienced a brief surge in interest during the early 1970' [4].

The fundamentals of employing entropy analysis in economics and finance were later on questioned by Horowitz and Horowitz (1976) [5]. They tried to conclude if entropy analysis measures meaningful information, which would not be otherwise available to standard statistical techniques, such as variance or correlation analysis. On the other hand, there does not seem to be any statistical measure that states whether or how meaningful information is in economic sense. In their papers (1972 [6], 1974 [7]) Philippatos and Wilson argue that entropy is a better statistical measure of risk than variance measure, since entropy is a nonparametric measure – it does not make assumptions concerning the underlying probability distribution. An additional argument was made by White (1974) [8] who states that, since entropy analysis is not integrated into economic theory, including the theory of choice under uncertainty, it should not be used in economic and financial studies [9].

These early interpretations of entropy and information theory, and their applicability in the economic realm were later on left behind, once entropy was integrated into dynamic models of economic behavior. Shackle (1952) [10] was the earliest researcher to apply the concepts of information theory to economics, in the field of business decisions. He splits a decision into two extreme components: a representative gain (*focus gain*) and a representative loss (*focus loss*). Subsequently, he attaches a subjective probability to each extreme case, and measures the potential surprise to the investor. In this context, an occurrence with a low probability contains more surprise than an occurrence with high probability, and therefore having the amount of surprise as a measure of risk on the investment.

2 A brief presentation of the proposed stock intrinsic entropy model

In an earlier paper [11], we proposed an entropic model that intends to measure the investor interests in a given stock, listed and traded on a stock exchange, along with the indication regarding the direction of interest: in buying or in selling the stock.

In the original Gibbs entropy (1), there is $\boldsymbol{k_B}$, the Boltzmann constant, which for obvious reasons was not carried over as such in the information entropy proposed by Shannon. The original information entropy development and the economics literature in general use the convention of having the constant equal to 1.

$$H = -C\sum_{i=1}^{n} p_i \log p_i = -\sum_{i=1}^{n} p_i \log p_i \,, (C = 1) \quad (2)$$

From this general form of computing the entropy, Nawrocki and Harding (1986) [9] introduced the concept of weighted entropy. They noted that there appears to not be any *a priori* reason why $\boldsymbol{C}$ should be constant for all the observed (considered) system states. It had been observed that the generally accepted calculation for the entropy is not a good measure of security risk because it ignores the states values of different frequency classes used in the calculation.

Our model starts from interpreting the following observations relative to intraday transactions executed on a stock exchange. A trade made on the market for a given stock at a certain price means that:

a) the quantity executed through that transaction, relative to the total quantity executed for the considered stock during the day, up to the moment of entropy calculation, represents the degree of credence that the market gives to the price level at which the trade was made;
b) the price at which the match occurred, relative to a selected reference price, offers

an indication for the inclination of the market to (rather) buy or sell the considered stock.

The intraday stock entropy model that we propose has the following formalization, $\boldsymbol{H_X^t}$ being the intraday entropy computed for symbol $\boldsymbol{X}$ at moment $\boldsymbol{t}$.

$$H_X^t = -\sum_{i=1}^{n}\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right)\frac{q_i}{Q} ln\left(\frac{q_i}{Q}\right) \qquad (3)$$

where the components are:

- $\boldsymbol{n}$ - total number of transactions (executions) realized on symbol X from the beginning of trading session up to moment $\boldsymbol{t}$
- $\boldsymbol{i}$ - ordinal execution (trade) number
- $\boldsymbol{q_i}$ - executed quantity (number of shares) of execution $\boldsymbol{i}$ for symbol $\boldsymbol{X}$
- $\boldsymbol{Q}$ - total executed quantity (number of shares) during the day for symbol $\boldsymbol{X}$, up to moment $\boldsymbol{t}$
- $\boldsymbol{p_i}$ - price of execution $\boldsymbol{i}$ for symbol $\boldsymbol{X}$
- $\boldsymbol{\bar{p}_{i-1}}$ - a reference price for symbol $\boldsymbol{X}$

Regarding the reference price to be employed, we experimented with the closing price from the previous day of trading, the opening price of the current day, and the volume weighted average price (VWAP) computed with trading data realized till the most recent considered transaction. The latter option for the reference price means that if we are to consider for the entropy calculation the trade $\boldsymbol{i}$, then the VWAP price is computed up to trade $\boldsymbol{i\text{-}1}$ (inclusive):

$$\bar{p}_{i-1} = \frac{\sum_{k=1}^{i-1} q_k p_k}{\sum_{k=1}^{i-1} q_k} \qquad (4)$$

Where $\boldsymbol{q_k}$ and $\boldsymbol{p_k}$ are the quantity and the price, respectively, corresponding to the $\boldsymbol{k}$ ordinal transaction (execution) of the day.

The fractions $\frac{\boldsymbol{q_i}}{\boldsymbol{Q}}$ are assimilated to probabilities, the probability of having the execution $\boldsymbol{i}$ at a certain price level, and the following condition has to be satisfied:

$$\sum_{i=1}^{n} \frac{q_i}{Q} = 1$$

3 System architecture, components and functionality

One of desiderates that we embraced from the initial phase of this research was the commitment for employing open source technologies throughout the entire system environment.

The architectural design is one of *service-orientation* (SOA). Each component of the system exposes its functionality, as a service provider, to the other components [12]. The requests for services and the replies are flowed through a *message-oriented middleware* (MOM). A MOM makes use of a message provider (broker) to mediate the messaging operations [13]. In this parading, the elements of a MOM-based system are the client applications, the messages, and the message provider. Under the broad umbrella of client applications, can be in fact identified certain applications that functionally play the role of a client, and others that have the functional role of a server. All the system applications are perceived as clients of the MOM message broker [14]. Within a MOM-based system, a client makes an API call by sending a message to a destination managed by the message provider. The call triggers message provider services to route and deliver the message to the consumer. Once the message was sent, the producer can continue the processing flow, relying on the fact that the message provider retains the message until a consumer component is available to process it. In this manner, the MOM-based model, in connection with the message provider, open the possibility of creating an architecture with loosely coupled components. Such a system can continue to function reliably, without downtime, even when individual components or connections fail. The client applications are consequently effectively relieved of every communication issue, except that of sending, receiving and processing messages [15].

We will briefly describe the functionality of each system component, as they were illustrated in Figure 1.

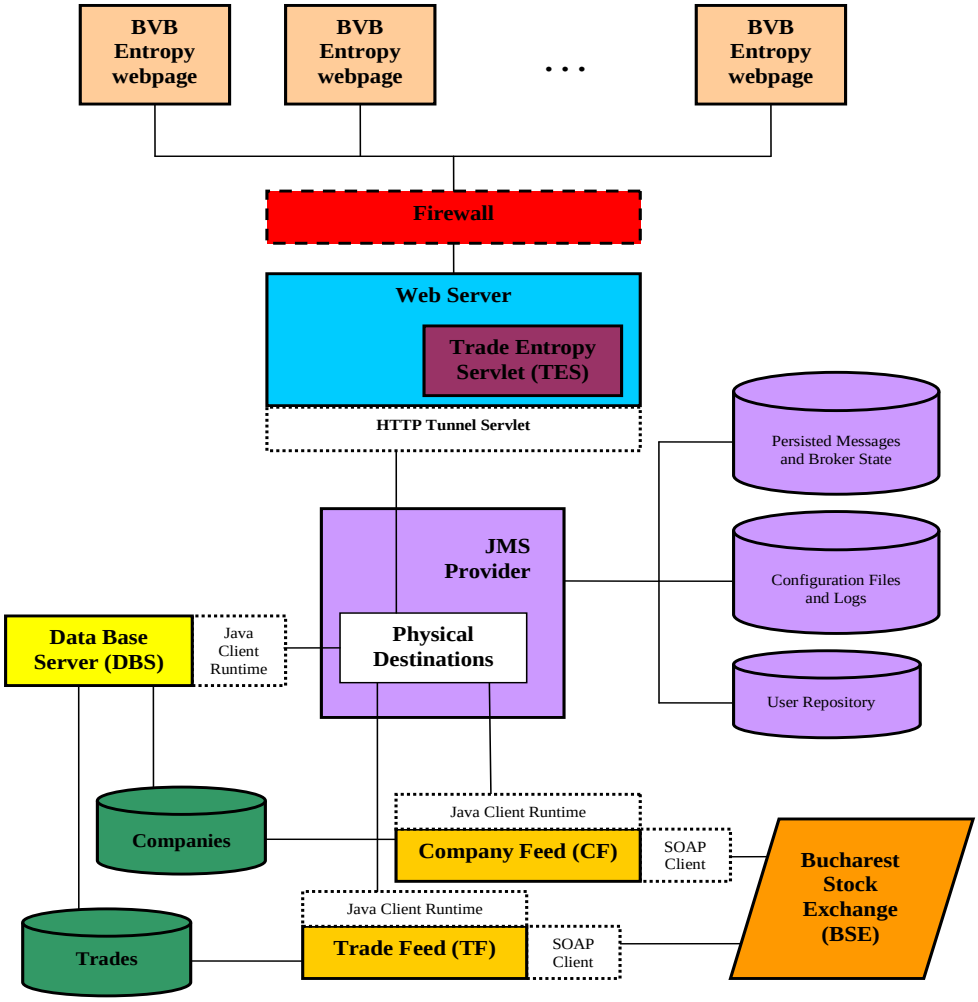

Fig. 1. System architecture for computing the intrinsic entropy of BSE listed stocks

Company Feed (CF) captures data regarding the financial instruments traded on Bucharest Stock Exchange (BSE), listed companies and their status, close price, volume etc.

The web service made available by BSE is accessible through SOAP formatted messages [16]. Once the company data is captured, CF stores it in the system database.

Data Base Server (DBS), in the context of stock intrinsic entropy computation, is the component that provides the historic trade data to web GUI, based on a request-reply model (in asynchronous fashion). In order to improve the overall response of the system, DBS employ prefects historical trade data from the database at launch, for a time interval of 90 days, and has it readily available in memory for the client requests.

Trade Feed (TF) is the application responsible with retrieving market data from the Bucharest Stock Exchange. BSE disseminates market data to different types of end-users, from individual investors to professional trading firms, financial analysts or academic institutions, through different channels and by utilizing different techniques

Web services are *de facto* standard for interoperability over internet. A web service is a software system designed to support machine-to-machine interaction over a network. Web service interactions are described through a XML-based format, called *Web Service Description Language* (WSDL) [17]. The WSDL provides a machine-readable description of the service interface, making all the information available for interaction easily accessible to any system acting as a web service client [18]. The information exposed through WSDL include the address or connection point of the web service, usually rep-

resented by an HTTP URL; the web service operations which can be performed and the messages needed by each operation; the data structures used for input and output in each operation and the transport protocol [19]. The data structures used by the web services are described by using the *XML Schema Definition* (XSD) syntax, which provides a unified way of expressing both simple and complex data types. The data collected from the BSE market data service encompasses mainly information related to the trades that took place throughout the trading day on the primary equity market. For each trade, the traded volume, the execution price and the trade time are included, so the data structure used for this purpose is fairly simple. The description of this structure in XSD format, as it appears in the BSE web service description file (WSDL file) is enclosed below:

```
<s:complexType name="TypeTrade">
  <s:sequence>
       <s:element name="TradeID"
              type="s:long"/>
       <s:element name="Symbolcode"
              type="s:string"/>
       <s:element name="Marketcode"
              type="s:string"/>
       <s:element name="TradeTime"
              type="s:dateTime"/>
       <s:element name="Volume"
              type="s:long"/>
       <s:element name="Price"
              type="s:decimal"/>
  </s:sequence>
</s:complexType>
```

The web service operation for retrieving the trading data will return an XML document containing a list of elements that follow the above specifications. Below is a fragment of an actual response to a web service call:

```
<ArrayOfTypeTrade>
  <TypeTrade>
    <TradeID>10830174</TradeID>
    <Symbolcode>SIF1</Symbolcode>
    <Marketcode>REGS</Marketcode>
    <TradeTime>2012-06-
14T10:15:27.877</TradeTime>
    <Volume>5000</Volume>
    <Price>0.8510</Price>
  </TypeTrade>
  <TypeTrade>
    <TradeID>10830402</TradeID>
    <Symbolcode>SIF1</Symbolcode>
    <Marketcode>REGS</Marketcode>
    <TradeTime>2012-06-
14T10:57:25.547</TradeTime>
    <Volume>4500</Volume>
    <Price>0.8520</Price>
  </TypeTrade>
</ArrayOfTypeTrade>
```

The advantage of using a WSDL file is that a software system that interacts with the web service will have all the information it needs to parse the response and extract the data; the structure of the response and the type of each element are all included in the definition file.
Also specified through the WSDL are the transport protocols, or how the data will be transferred from one system to another as part of the interaction. Traditionally the web services have been using SOAP as their transport, but other protocols such as HTTP can be used as well. The web service provided by the Bucharest Stock Exchange accepts both HTTP and SOAP protocols. SOAP (initially an acronym of *Simple Object Access Protocol* and later used as a stand-alone term) represents a formal set of conventions governing the format and processing rules of a web service message [20]. These conventions include the interactions among systems in a decentralized, distributed environment, generating and accepting messages for the purpose of exchanging structured and typed information. Although HTTP transport might be easier to use and in some cases might yield faster response times, the SOAP transport has the advantage of providing an extensible framework for message exchange. Therefore, the processing logic necessary to transmit, receive, process or relay a message will follow the conventions defined by the SOAP protocol, for all peers involved in the information exchange. Another advantage of SOAP is that messages can be carried within or on top of different underlying protocols for the purpose of exchange. For example, for a web-based access, a SOAP message could be sent as the body of an HTTP request. In other setups, it can be sent over a TCP stream, inside an email message (through SMTP), or as a JMS message.
The message framework imposed by SOAP, along with all the application specific extensions and the underlying transport protocols

are specified in the web service description file. By including this type of information in the service interface a software system will know how to connect to the web service and how to exchange information. This approach will also hide the implementation details from the peers, allowing each system to be developed independently from both a hardware and software standpoint.

Although the BSE web service offers both HTTP and SOAP for message exchange, we chose to use the SOAP transport for DTF application. Requests and responses to and from the web service are formatted as SOAP messages, from which the actual data is extracted.

Trade Feed service runs as a continuous process, which collects intraday trading data from the BSE corresponding web service: traded volume, execution price and trade time for all traded symbols. Trading data is actively captured at an interval of a few minutes. Due to the architecture of the web services in general, and the BSE implementation in particular, our application needs to periodically poll for data.

Although recent developments in the SOAP specification allow for the asynchronous web service calls, this feature is not widely spread yet. The BSE web service itself does not offer the ability to asynchronously retrieve the data; therefore, all requests must be synchronous. In synchronous mode, any request must be followed immediately by a response from the web service. The drawback of such a paradigm is that the client platform must poll the data it needs. Other channels for data dissemination provided by the BSE offer real-time updates but have other disadvantages, like higher cost and complexity of communication model, or the use of proprietary data exchange protocols.

For the current research, where the real-time data is not critical, we chose the web service channel because of the reduced complexity and implementation costs.

Once TF component retrieves new trading data from the BSE, it extracts it from the SOAP message and passes it on to two different modules: the persisting module, which stores the information to a local database for later use, and the stock entropy module.

Trade Entropy Servlet (TES) is responsible with computing the values of the intrinsic entropy for all the stocks, and with publishing them.

TES is a web application which runs within a web server. It is built upon the Java Servlet specification, which states that the application remains active as long as the web server is running, as opposed to other types of web applications, which are short-lived and active only during the life-cycle of a web request [21]. This architecture allows the stock entropy module to wait for new market data to be passed on by the TF service as it becomes available. Once new data is received, the module re-computes the values of the stock entropy in a progressive manner. For each trade that took place, the entropy value is determined based on all the previous trades performed up to that moment in time. Thus, each trade done on the market will have a corresponding value of the stock entropy, which allows for easy tracking of its evolution throughout the day. It is important to note that the maximum number of observations of the entropy value is limited to the number of transactions. Thus, the more heavily a financial instrument is traded, a higher number of values can be calculated for the entropy, and more accurate its evolution can be determined during the day. TES servlet keeps all the computed values in memory and disseminates them through public web pages, which display the evolution of the entropy for all stocks. The computed values of the entropy are published in a structured format (using JavaScript object notation) and are then used as the input data for graphical charts and tables.

The entire system has been designed and implemented in conjunction with Java Message Service (JMS) API. JMS specification captured, from its conception, the essential elements of a generic messaging system, namely:

- the concept of a messaging provider that routes and deliver messages;

- distinct messaging patterns, or domains such point-to-point messaging and publish/subscribe messaging;
- facilities for synchronous and asynchronous message receipt;
- support for reliable message delivery;
- common message formats such as text, byte and stream.

Messaging among system components is achieved through the following JMS destinations:

- **`TRADE_TOPIC`** – destination topic on which TF module publishes the captured trades from BSE, and at which DBS and TES subscribe;
- **`COMPANY_TOPIC`** – destination topic on which CF module publishes the captured company data from BSE, at which DBS subscribes;
- **`INSTRUMENT_LIST_REQUEST_QUEUE`** – destination queue at which TES sends the requests for the list of instruments traded on BSE, and which is listened by DBS;
- **`INSTRUMENT_LIST_REPLY_QUEUE`** – destination queue at which DBS places the reply with list of instruments, to be retrieved by TES;
- **`TRADE_RANGE_REQUEST_QUEUE`** – destination queue at which TES sends the request for the range of trades corresponding to a certain traded symbol, request which is listened by DBS;
- **`TRADE_RANGE_REPLY_QUEUE`** – destination queue at which DBS places the reply with the range of trades corresponding to the requested symbol, to be retrieved by TES.

Illustrated in Figure 2 are the system components, and manner in which the messages flow among them.

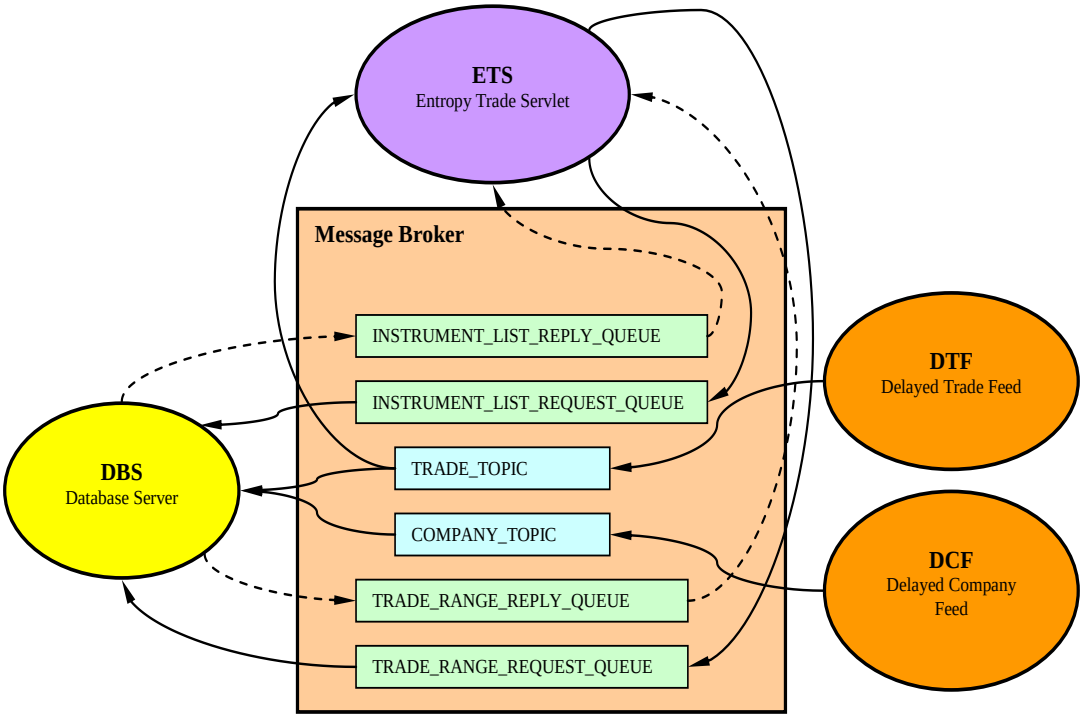

Fig. 2. The message flow among system components

The graphics are created using a specialized library, called HighCharts. It is written in pure JavaScript, and is based on native web-browser technologies. It does not require any third-party client plug-ins like Flash or Java, and it does not need to interact with a server as long as the input data is available locally in the web-browser [24]. The library offers numerous chart types, which can be easily customized through a simple configuration syntax based on JavaScript object notation (JSON). It offers features like range selec-

tors, filters, event markers, data grouping, zooming and others, which can be used out-of-the-box with any type of chart.

The advantage of using a charting library is that we can reuse logic which has already been implemented, dramatically reducing the time needed to present the results of our research. Another advantage is that the library is easy to use and compatible with all modern web-browsers. It primarily uses Scalable Vector Graphics (SVG) for rendering images [25]. SVG is a specification of XML-based file format for two-dimensional vector graphics, both static and animated. With SVG, images and their behavior are defined in XML text files. This means that they can be searched, indexed, and, if needed be compressed. Also, as XML files, SVG images can be created and edited on the fly, through a scripting language, like JavaScript. SVG creates powerful, dynamic content because it tightly integrates front-end graphics to back-end business processes and data including corporate databases, and other rich sources of information. SVG files use existing and proven Web standards such as Cascading Style Sheets (CSS) and Extensible Style Sheet Language so that graphics can be easily customized. Moreover, the graphics created in SVG can be scaled without loss of quality across various platforms and devices [26]. SVG enables Web developers and designers to create dynamically generated, high-quality graphics from any type of data with precise structural and visual control. It can be used on the Web, in print and even on portable devices while retaining full quality.

The HighCharts takes full advantage of the SVG technology to render the charts on the client browser. This approach has benefits for both the developers, by reducing the load on the web server, and for end-users by dramatically improving the user experience. Waiting times to see high-quality graphics are reduced, since the images are constructed on the fly on the client-machine. Zooming and other operations are extremely fast because data is resident on the client. Therefore, interactivity is almost instantaneous since there is no need to retrieve additional data from the server.

On our web pages, the entropy data is retrieved only one time, when the page is loaded. Once this step being completed, there is no other interaction with the web server. The data is provided in JSON format, which makes it easy to be loaded through JavaScript by the charting library. Based on this data, the library creates graphics which present the intra-day evolution of the entropy values for a certain stock. Another chart is provided which presents the entropy values of all the instruments traded on the Bucharest Stock Exchange.

4 Methodology and interpretation

The intraday stock entropy $\boldsymbol{H_X^t}$ is computed solely based on the actual trading data:

a) number of transactions (generated executions by the exchange matching engine);
b) quantity executed per trade;
c) overall traded volume per symbol during the day;
d) execution prices;
e) volume weighted average price.

From the above perspective, we can refer to $\boldsymbol{H_X^t}$ entropy measure as intrinsic entropy of a stock traded on exchange. There is no exogenous factor to influence the level of entropy. If there are external influences they are imbedded, already contained in the trading data (volume, price, number of transactions), and quantified as such fig 3 shows the intraday evolution of the entropy for symbol SIF4, during June 15, 2012. It can be seen that the entropy evolution sustains the price variation, the price staying during the day above the open.

Fig. 3. Entropy and price variation for symbol SIF4 on June 15, 2012

In the entropy model that we propose, the probabilities represented by the fractions $\frac{q_i}{Q}$ (executed quantity of trade i, relative to the overall executed quantity) seem to be weighted with the price variation, in terms of weighted entropy concept proposed by Nawrocki and Harding. Our perspective is slightly different though: we would rather interpret these probabilities as degrees of confidence that the market give for a certain price level of the stock, and subsequently for the price variation from the chosen reference price. Furthermore, we need the entropy H_X^t values not to be impacted by the price level of a stock or another, and opted for the relative price variation rather than the price value itself, or the absolute variation from the reference.

One of the particular features of this entropy calculation is that it can generate both positive and negative values for the entropy. We are still interested in the absolute value of the entropy, but in context of an entropy measure associated to a traded stock, allowing the entropy to have negative values has the significance of price decrease for the considered stock, and consequently, the associated interest of the market for rather selling it.

Taking the closing price of the previous day as reference generated an evolution of entropy values over the trading day that was decoupled from the evolution of the price. For example, as long as the execution prices during the day are above the last closing price, the entropy values continue to increase, regardless the fact that the prices can actually decrease over the day, from a high level in the opening, but still stayed above the last close. Similarly, the execution prices could stay the entire day under the previous close, while they were in fact going up from a low opening price. Considering the opening price as reference, or any other pegged value for that matter, could generate a similar decoupling that we have mentioned between entropy evolution and price variation over the day.

As mentioned above, trading data is captured from Bucharest Stock Exchange (BSE) through a delayed data feed. At each fetch (corresponding to t in H_X^t entropy), the new trades are appended to the already received ones, and the value of entropy is recalculated for each stock listed on the market. We envision this entropy model as measure of the market confirmation for the movement of the stock price into one direction or the other. In other words, the intrinsic stock entropy H_X^t should be higher, in absolute terms, if:

a) execution prices move consistently into a certain direction (up or down) during the day;
b) stock is heavily traded, as more transac-

tions suggest a high interest of the market in the considered stock, and the higher number of transactions (states and their associated probabilities) will support the direction of price evolution.

Coming back to the reference price, we consider that a moving reference will reflect most accurately the price change in the entropy measure. The volume weighted average price (VWAP), computed up to the considered trade, give the best indication for the direction of price change with each executed trade (state, in terms of Gibbs entropy).

Price variation $\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right)$ provides both:

a) anchoring to the probability value;
b) direction for the trading activity of a given security, up to the point in time when the entropy is computed.

Without the price variation weight, the sum $-\sum_{i=1}^{n} \frac{q_i}{Q} ln\left(\frac{q_i}{Q}\right)$ is to be positive.

The price variation $\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right)$ has the following implications on the entropy $\boldsymbol{H_X^t}$, leaving apart the associated probability for the price change to occur:

a) if, preponderantly, $\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right) > 0$, then there are chances for entropy to be positive, $\boldsymbol{H_X^t} > 0$;
b) if, preponderantly, $\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right) < 0$, then there are chances for entropy to be negative, $\boldsymbol{H_X^t} < 0$.

The value of the entropy $\boldsymbol{H_X^t}$ is an indication of the interest that the market (investors) manifests, relative to security $\boldsymbol{X}$, up to the moment $\boldsymbol{t}$. A higher entropy value, in absolute terms, suggests a greater market interest in a given security (either in buying or selling it). Conversely, a lower absolute value of the stock intrinsic entropy gives an indication regarding a lower interest of investors in the considered security. Small or inexistent price variation makes for $\left(\frac{p_i}{\bar{p}_{i-1}} - 1\right)$ to approach zero, signifying a greater indetermination fig 4 shows the distribution of $\boldsymbol{H_X^t}$ entropy at the end of the day (June 15, 2012) for a selected set of stocks listed on Bucharest Stock Exchange (BSE).

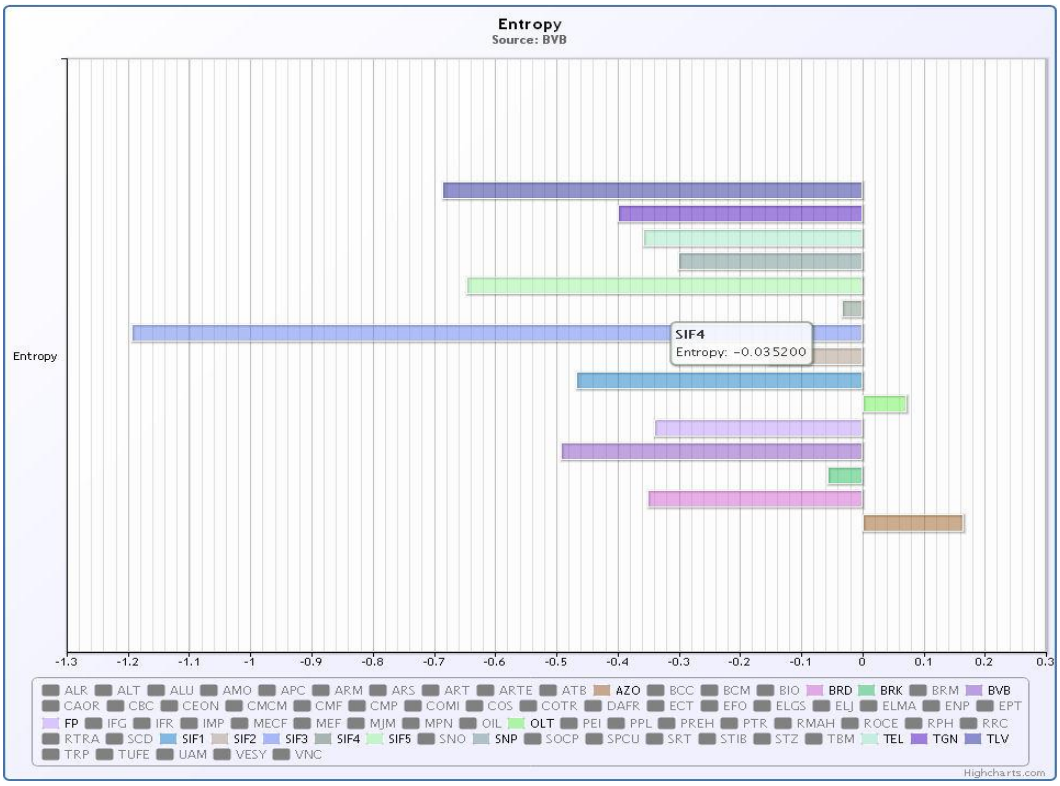

Fig. 4. Entropy distribution for a set of selected stocks listed on BSE, June 15, 2012

5 Conclusions and further research

Summarizing, messaging is a very effective means of building the abstraction layer within SOA, needed to fully abstract a business service (functionality) from its underlying implementation. Through business messaging, the business service does not need to be concerned about where the corresponding implementation service (say, the TES servlet) is located, what language it is written in, what platform it is deployed on, or even the name of the implementation service. All the above-mentioned elements have equally constituted the reasons why we turned to Open Message Queue (OpenMQ), as the open source MOM implementation of JMS, for designing the architecture for computing the intrinsic entropy of BSE listed stocks.

Our research has been conducted based on the transactions executed on the Bucharest Stock Exchange (BSE), and aimed to prove that price variation weighted entropy can offer a synthetic and readily computed indicator, for evaluating direction and intensity of trading activity. We are currently looking into adapting the proposed model to compute the stock entropy for a longer time frame: 3, 7, 15, 30, 60, 90 days. Although we consider that this intrinsic entropy does not have much relevance for extended time frames, having given the relative low number of transactions per day executed on BSE for each listed stock, a longer period would provide more data for supporting the more recent activity. For example, even for intraday computed $\boldsymbol{H_X^t}$ entropy, it would make more sense to not start from zero value at the beginning of trading day, but to carry over some market information from the previous day(s), with the focus still on the most recent data, which we consider to be more relevant for stock intrinsic entropy. We are also investigating whether $\boldsymbol{H_X^t}$ entropy measure can provide a consistent and reliable indication regarding the direction of intraday price variation.

Our ongoing research project aims, at theoretical level to fully explore the implications of the constituents of the proposed calculation of the intrinsic stock entropy and, at applicative level to integrate the entropic model within a trading algorithm that could react decisively, and in a timely fashion, to the evolution of trading results.

References

[1] R. Frigg and C. Werndl, "Entropy – A Guide for the Perplexed", Probabilities in Physics, Oxford University Press, Oxford, 2010.

[2] E.T. Jaynes, C.R. Smith, G.J. Erickson, "The Gibbs Paradox", in Maximum Entropy and Bayesian Methods, Kluwer Academic: Dordrecht, 1992, pp. 1–22.

[3] E.C. Shannon, "A Mathematical Theory of Communication", in Bell System Technical Journal, vol. 27, pp. 379–423, 623-656, July, October 1948.

[4] G.R. Nicholas, "The Entropy Law and the Economic Process", Harvard University Press, Cambridge, MA, USA, 1971.

[5] A. Horowitz and I. Horowitz, "The real and illusory virtues of entropy-based measures for business and economic analysis", Decision Science 7, 121-36, 1976.

[6] G. Philippatos and C. Wilson, "Entropy, market risk and the selection of efficient portfolios", Applied Economics 4, 209-20, 1972.

[7] G. Philippatos and C. Wilson, "Entropy, market risk and the selection of efficient portfolios: reply", Applied Economics 6, 76-9, 1974.

[8] D. White, "Entropy, market risk and the selection of efficient portfolios: comment", Applied Economics 6, 73-5, 1974.

[9] N.D. Nawrocki and H.W. Harding, "State-value weighted entropy as a measure of investment risk", Applied Economics, 1986, 18, pp. 411-419.

[10] G.S.L. Shackle, "Expectations in Economics", Cambridge University Press, Cambridge, UK, 1952.

[11] C. Vinţe, I. Smeureanu, I.A. Lixandru, "An Entropic Model for Equity Trading Analysis", in the Proceedings of The Eleventh International Conference on Informatics in Economy IE 2012, Bucharest, May 10-11, 2012.

[12] E. Thomas "SOA Design Patterns", Prentice Hall by SOA Systems Inc., New Jersey, NY, 2009.
[13] Sun Microsystems, Inc. - Open Message Queue: Open Source Java Message Service (JMS) - https://mq.dev.java.net/
[14] R. Mark, R. Monson-Haefel and A.D. Chappell , "Java Message Service (Second Edition)", O'Reilly Media Inc., Sebastopol, CA, 2009.
[15] C. Vinţe, "Upon a Message-Oriented Trading API", Informatica Economica Journal vol. 14, no. 1/2010.
[16] Bursa de Valori Bucureşti, http://www.bvb.ro
[17]Web Service, http://en.wikipedia.org/wiki/Web_service, retrieved June 14, 2012.
[18]WSDL, http://en.wikipedia.org/wiki/Web_Services_Description_Language, retrieved on June 14, 2012.
[19] M. Kalin, "Java Web Services: Up and Running", O'Reilly Media Inc., Sebastopol, CA, 2009.
[20] SOAP over Java Message Service 1.0, *W3C Recommendation*, http://www.w3.org/ *SOAP Version 1.2, W3C Recommendation*, W3C, 2009.
[21]Sun Microsystems, Inc., *Java Servlet 2.5 Maintenance Release 2 Specification*, 2007.
[22] Sun Microsystems, Inc., *Java Message Service* - http://java.sun.com/products/jms/
[23] Sun Microsystems, Inc., *Open Message Queue: Open Source Java Message Service (JMS)* - https://mq.dev.java.net/
[24] HighCharts, http://www.highcharts.com/products/highcharts, retrieved on June 16, 2012.
[25]SVG Zone, Adobe, http://www.adobe.com/svg/overview/svg.html, retrieved on June 16, 2012.
[26]Scalable Vector Graphics, http://en.wikipedia.org/wiki/Scalable_Vector_Graphics, retrieved on June 16, 2012.

Claudiu VINŢE has over fifteen years of experience in the design and implementation of software for equity trading systems and automatic trade processing. In 2007 Claudiu co-founded Opteamsys Solutions, a software provider in the field of securities trading technology and equity markets analysis tools. Previously, he was for over six years with Goldman Sachs in Tokyo, Japan, as Senior Analyst within the Trading Technology Department. Claudiu's expertise in trading technologies also includes working in Tokyo with Fusion System Japan, and Simplex Risk Management as Software Engineer, and Senior Software Engineer, respectively. Since 2009, Claudiu has been given lectures and coordinated the course and seminars upon *The Informatics of the Equity Markets*, within the Master's program organized by the Department of Economic Informatics. Claudiu graduated in 1994 The Faculty of Cybernetics, Statistics and Economic Informatics, Department of Economic Informatics, within The Bucharest Academy of Economic Studies. He holds a PhD in Economic Cybernetics and Statistics from The Bucharest Academy of Economic Studies. His domains of interest and research include combinatorial algorithms, middleware components, algorithmic trading and web technologies for equity markets analysis.

Ion SMEUREANU has graduated the Faculty of Planning and Economic Cybernetics in 1980, as promotion leader. He holds a PhD diploma in "Economic Cybernetics" from 1992 and has a remarkable didactic activity since 1984 when he joined the staff of Bucharest Academy of Economic Studies. Currently, he is a full Professor of Economic Informatics within the Department of Economic Informatics and the dean of the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies. He is the author of more than 16 books and an impressive number of articles. He was

also project director or member in many important research projects. He was awarded the Nicolae Georgescu-Roegen diploma, the award for the entire research activity offered by the Romanian Statistics Society in 2007 and many others.

Ionuţ-Alexandru LIXANDRU graduated from the Bucharest Academy of Economic Studies in 2008. He is a Ph.D. candidate in the field of Economic Informatics at the Bucharest Academy of Economic Studies. Alexandru is currently working at the Bucharest Stock Exchange, as a Software Developer within the Trading System Development department. Previously he was for 5 years with TechTeam Global within the Global Business Applications department. His main areas of interest are system integrations, web technologies, and low-latency technologies for trading systems.