SOA and Web Technology for Building BSE Market Map

Claudiu VINŢE[1], Alexandru JURUBIŢĂ[2]
[1]Bucharest Academy of Economic Studies
[2]Blue BitBox
claudiu.vinte@ie.ase.ro, alexandru.jurubita@gmail.com

Visual representation as a map of the stock market data can offer access, in a quick and relevant manner for human participants, to the overall state of the market at a given point in time. The purpose of this paper is to present the results of our academic research upon building the market map for Bucharest Stock Exchange (BSE). We will focus on the algorithm for generating the market map, the system architecture, and web technology employed for capturing the required data and making the map publicly available through the portal www.bursa.ase.ro.
Mathematics Subject Classification: 68M14 (Distributed Systems)
Keywords: *Service-Oriented Architecture (SOA), Message-Oriented Middleware (MOM), Java Message Service (JMS), Tree-Maps, Stock Market Map, Recursive Algorithms, Divide-and-Conquer Strategy*

1 Introduction

Having to face the current complexity of the financial markets in general, and the capital market in particular, the general public, the investors, and the specialists alike are continuously requesting various tools designed to assist them in assessing the evolution of the market as whole, as quickly and as accurately as possible.

There is a large spectrum of instruments, graphical and non-graphical, conceived to capture the evolution of a given stock on the market, over a certain time window, most of them generically falling in one of the following category:

- 2-D charts - evolution of stock price over time - in a variety of representations (line charts, bar charts, Japanese candlesticks etc.);
- 3-D charts, more or less, representing, for instance, the evolution in time of stock price, connected with the corresponding traded volume;
- statistical indicators, arranged in tables, arrays etc., and which are to be interpreted as numbers.

In 1991, Ben Shneiderman proposed a novel way for visualizing tree structures that turnout to have multiple applications since then [1]. Shneiderman was looking in particular for a more economical visual representation of tree structures with a large number of nodes. He proposed a 2-D space-filling approach for tree nodes representation, in which each node is a rectangle whose area is proportional to some attribute such as node size. Shneiderman's approach marks a departure from the traditional tree structures representation, as a rooted, directed graph with the root node at the top of the page, and children nodes below the parent node with lines connecting them. Donald Knuth has a long discussion about this standard representation, especially why the root is at the top, and he offers several alternatives including brief mention of a space-filling approach [2].

Previously, there had been research on relationships between 2-D images and their representation in tree structures, which had focused on node and link representations of 2-D images. The work on this path includes quad-trees and their variants, which are important in image processing [3]. The goal of quad-trees is to provide a tree representation for storage compression and efficient operations on bit-mapped images. XY-tree [4] are a traditional tree representation of 2-D layouts found in newspaper, magazine, or book pages [5]. Related concepts include K-D trees [6], which are often explained with the help of a 2-D rectangular drawing [7], and hB-trees [8], which are a more advanced multi-attribute indexing method that has a useful 2-D representation. Nevertheless, as

Shneiderman noted in his paper, none of these projects sought to provide human visualization aids for viewing large tree structures. The original Shneiderman's goal for his work was to gain a better representation of the utilization of storage space on a hard disk as viewed from the perspective of a multiple level directory of subdirectories and files, as in modern OS file managers.

Knuth had proposed an approach to this problem, by choosing a 1-D representation of a multi-colored line, with the length of a portion representing the length of each file [2]. This is impractical because the line would be too long to view a great number of segments. On the other hand, a 3-D or higher dimensional approach might be difficult to draw and view. Shneiderman proposed a 2-D space-filling approach, in which each file was to be represented as a small rectangle. This method of tree visualization, called tree-maps, offers a very compact and suggestive view, and provides opportunities for many other applications.

Perhaps one of the most popular implementation of tree-maps, particularly related to the field of stock market is the one which shows around 600 the most popularly held stocks listed on the American market, organized by industry groups, size-coded by market capitalization, and color-coded to show price variation: www.smartmoney.com/map-of-the-market/.

2 The algorithm

In a previous paper [12], we briefly introduced the results of our research and proposed a visual way for capturing the evolution of the entire Bucharest Stock Exchange market (BSE), over a given time span [9]. The concept is driven chiefly by the following three principles:

- concentrate on an unique board all, or most of the companies listed on BSE, designated section of the exchange, or grouped them by industrial sector etc.;
- the size of each company represented on the board is given by its market capitalization, relative to the entire market capitalization;
- the evolution of a stock price is to be displayed based on a color scheme.

Each of the above principles generates a certain perspective over the stock market. Considered all of them together, they lay out the fundamentals for designing a map of the market, that has the ability to provide, at a glance, the position on the market (or within a certain industrial sector) of each listed company, the size of a company (relative to the size of the others, and to the market as a whole), along with the stock price variation. The color-coded price variation, adds in fact a new informational dimension to a 2-D map otherwise.

On the market map, there will be a rectangle associated to each listed company. The size of the rectangle is proportionally determined, based on the relative weight (ratio) of the individual market capitalization of a considered company, in the overall market capitalization.

The algorithm, which we proposed for such a map of the stock market, relies on the availability from the market place of the following input data:

- number of the companies to be included in the map;
- market capitalization of each company - computed by multiplying the number of the shares, issued by the company on the market, with the close price of the stock from the previous trading day, or from the reference date that we may want to take as base;
- stock price variation during the day, or over the considered time window.

Another input parameter for the algorithm is the size of the board on which the map is to be drawn. It is given as the length and the height of the map board, in number of pixels.

In order to fill the map, the algorithm has to determine the dimensions of each rectangle, in connection with the market capitalization of the corresponding company.

We would like point out that this space-filling approach makes for a different problem than the one concerning 2-D rectangular cutting, in which the size of the rectangular pieces is a prerequisite and, consequently,

not adjustable. Therefore, as long as relative sizes of the companies are preserved, and the drawing board offers enough resolution to fit the number of the companies considered, the algorithm is able to determine a distribution for drawing the map.
Figure 1 illustrates the concept that we have proposed, sampling a handful of companies put together within a stock map, and revealing the above-mentioned perspectives.
Each rectangular is identified based on the coordinates of its topmost left corner C_x, C_y, and the lengths on X (abscise) and Y (ordinate) axis, L_x and L_y, respectively.

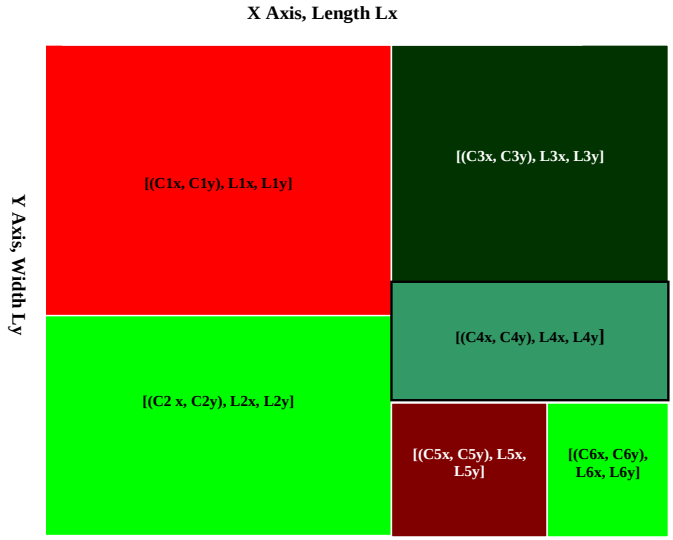

Fig. 1. A rectangular based distribution of market capitalization

In practice, there may be great discrepancies among the listed companies, when it comes to market capitalization. The gap between the bigger companies and the smaller ones may end up in the range of thousands of times. That introduces the impossibility of preserving the real weight (ratio) of an individual company on the market, when its size has to be translated into the dimensions of a rectangle to be drawn on a computer screen. We addressed this issue by normalizing the market capitalization of each company and, consequently, of the stock market as a whole.
The normalization process is conducted by computing a logarithmic factor that is designed to bring the market capitalization of the companies into a narrower interval of values, preserving, to a certain extent, the original proportions among the companies. Formalized, the normalization process is described below.

$$mc_i = q_i * p_i, \quad i = \overline{1,n} \tag{1}$$

$$MC = \sum_{i=1}^{n} mc_i \tag{2}$$

$$\overline{mc} = \frac{MC}{n} \tag{3}$$

$$\begin{cases} f_i = \ln\left(\frac{MC}{mc_i}\right) + \ln\left(1 + \frac{\overline{mc}}{mc_i}\right) \\ mc'_i = mc_i * f_i \end{cases}, \quad i = \overline{1,n} \tag{4}$$

$$MC' = \sum_{i=1}^{n} mc'_i \tag{5}$$

where:

n number of listed companies on the market

q_i number of shares issued on the market by company *i*

p_i last close price for the stock of company *i*

mc_i market capitalization of company *i*

$\overline{mc}$ the average market capitalization of a company on the considered market

MC overall market capitalization

f_i normalization factor

mc'_i normalized market capitalization, corresponding to company *i*

MC' normalized overall market capitalization

The logarithmic factor takes into account the weight of a company in the overall market capitalization, and the distance of an individual market cap from the average market capitalization. Once the input data is adjusted to the needs of the graphical representation on a computer screen, the algorithm can begin to determine the position and the size of each rectangle on the map, associated to the company considered.

The algorithm allocates the rectangles on the market board based on a *divide-and-conquer* strategy, implemented in a recursive manner (Figure 2).

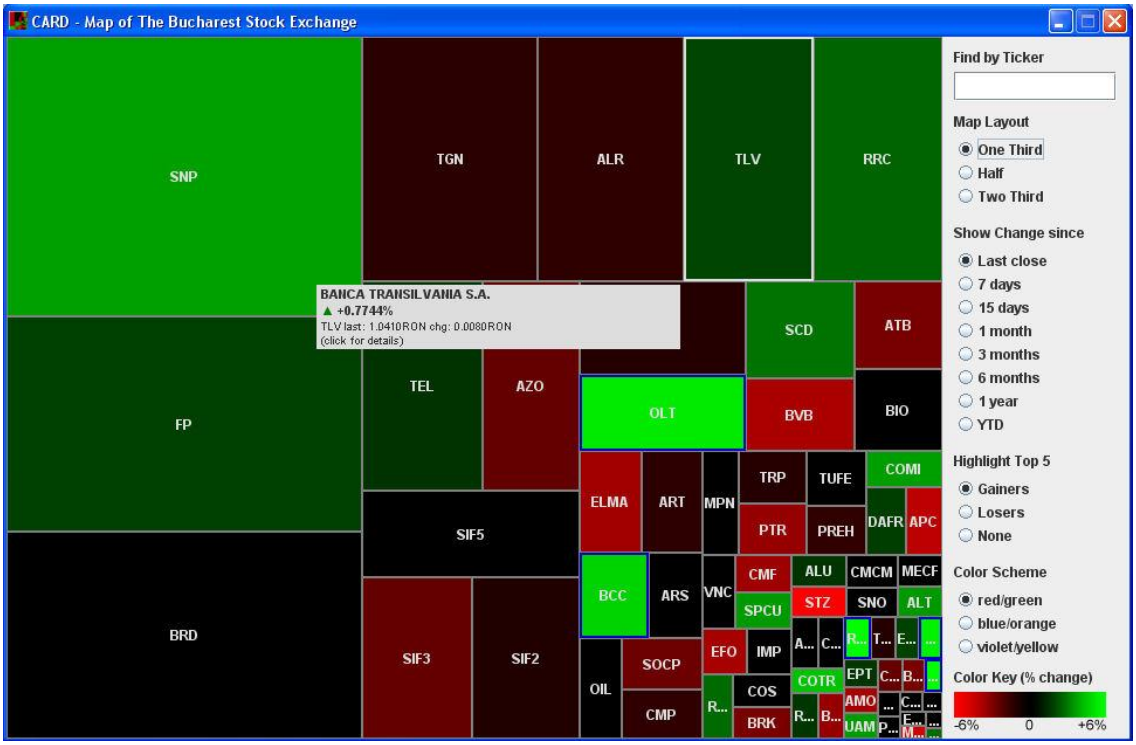

Fig. 2. BSE market map built on *One-Third* distribution model

At each step, the algorithm distributes the companies in two groups: the companies that are to be allocated on the board at the current stage, and the companies that are to be allocated on the remaining area of the board in the subsequent steps. The procedure continues recursively for the latter group of companies, until the companies are exhausted, along with the available drawing area of the board [10] [11].

The rectangles are colored, based on the price variation of the stock within the considered interval of time. In Figure 2, the chosen color scheme has the following interpretation:

- nuances of red for loses;
- black for stagnation of stock price;
- nuances of green for price gains on the market.

The companies are allocated beginning with the biggest company from the topmost left corner of the map, and continuing toward the smallest company listed on the market, which will end up in the bottommost right corner of the map. There are multiple ways to fill the area of the drawing board, or distribute the rectangles on the map.

The algorithm that we propose offers the flexibility for tailoring the ratio between the length and the height of the rectangles. This form factor layout is achieved through an additional parameter supplied to the recursive function, parameter that instructs where the cut is to be executed on X-axis, or Y-axis, respectively.

At each stage of the recursion, the algorithms choose to make a cut on the axis that offers the greater remaining length. Then, based on the desired form factor, the cut divides the remaining area into the area to be filled at the considered stage, and the area to be filled later on.

For instance, the layout shown in Figure 2 was obtained by using a form factor of 1/3 (*One Third*), meaning that up to one third of the map area from the topmost left corner is filled first.

After extended tests, we opted for three predefined form factors that proved to offer the most pleasing visual differences: *One Third*, *Half*, *Two Third*. Figure 3 bellow shows a 1/2 form factor layout, along with the violet/yellow color scheme and highlighted gainers (the rectangles with red borders).

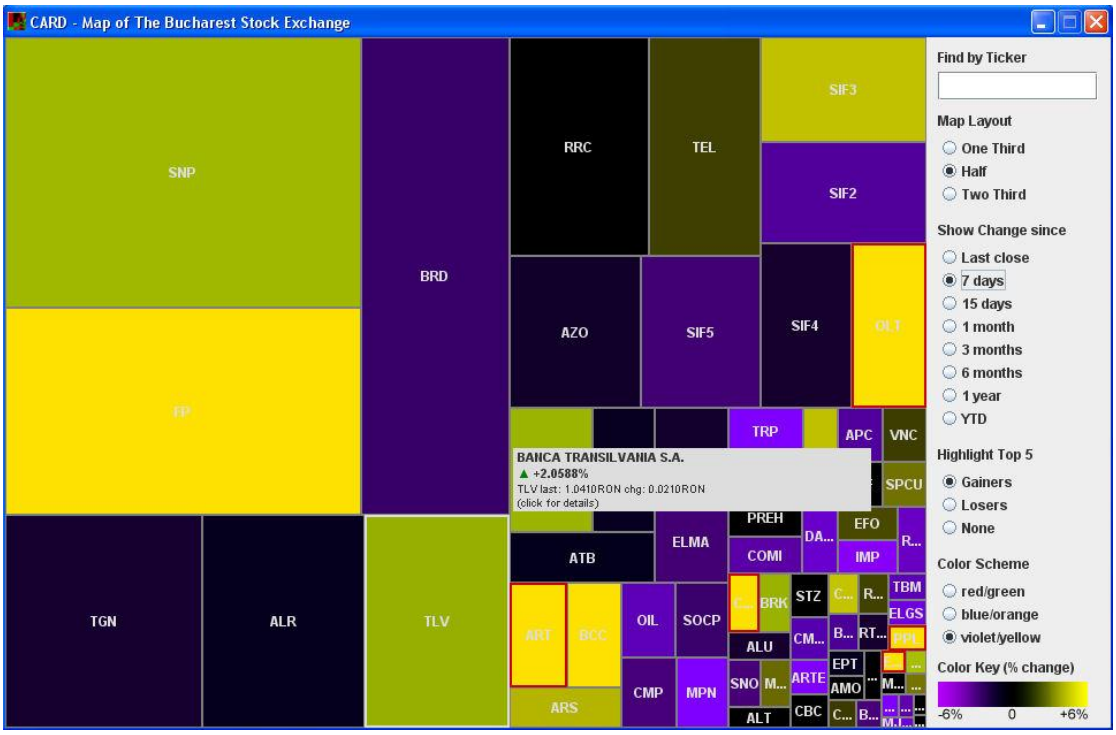

Fig. 3. BSE market map built on *Half* distribution model with highlighted gainers

3 CARD system architecture and implementation

One of desiderates that we embraced from the initial phase of this research was the commitment of employing open source technologies throughout the entire system environment. Our goal has been to provide for the users a convenient way of accessing BSE market map from the internet, through the means of employing a servlet responsible for HTTP tunneling [12] [13]. From the end user perspective, CARD (Capital Allocation through Rectangular Distribution) system consists in a collection of services accessible from a single entry point offered by a web-based graphical user interface (Figure 4).

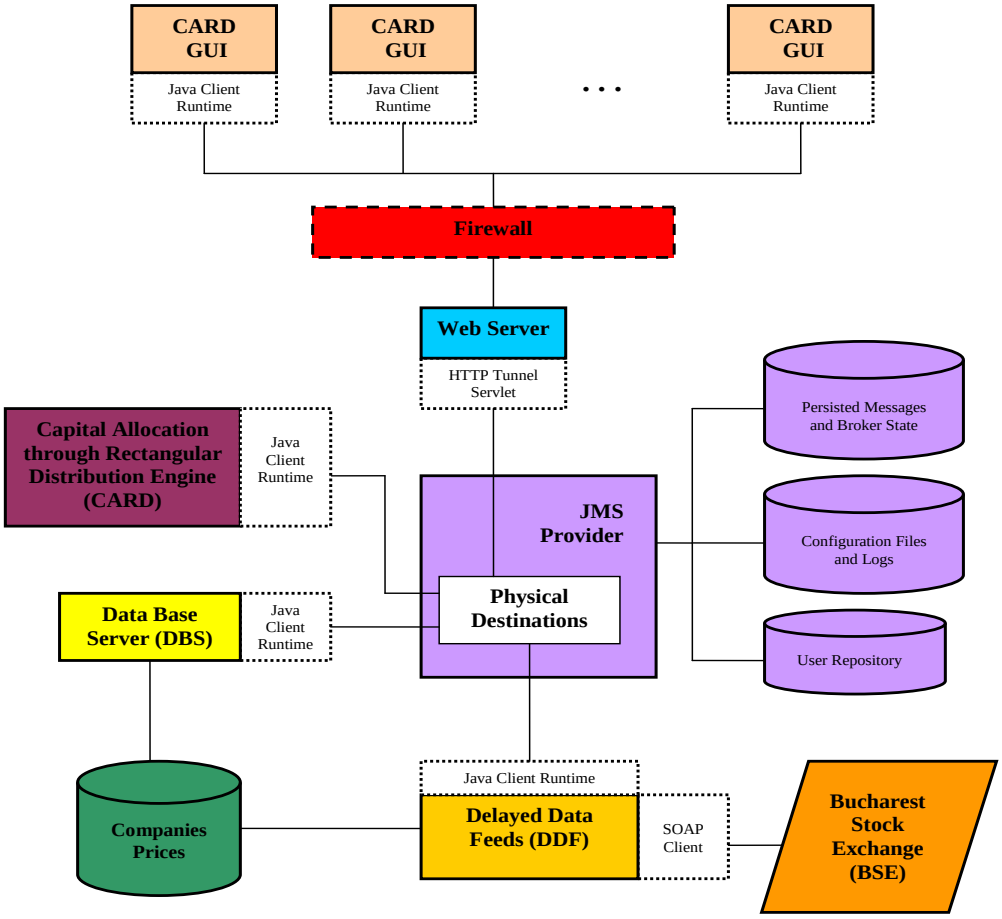

Fig. 4. CARD system architecture

Essentially, CARD GUI has been implemented as a Java applet, which can be launched from a web browser. The applet is the only component of the system that the users come in contact with. All the other components of the system are transparent to the end-user, and create an environment that replicates the perspective of having access to a pool of services.

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 [14] [15]. 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. 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 [16]. 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 [17].

CARD API 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.

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 CARD engine) 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 CARD architecture based on it.

We will briefly describe the functionality of each system component, as they were illustrated in Figure 4.

Delayed-Data Feed (DDF) consists of a collection of web-clients that connect to corresponding web services, intended to capture delayed market-data disseminated by The Bucharest Stock Exchange (BSE). The feed gathers data regarding the financial instruments traded on BSE, listed companies and their status, prices, volumes, exchange indices etc.

The web services made available by BSE are accessible through SOAP formatted messages [19]. Once the market data is captured, DDF stores it in the system database, and delayed prices are also published to a specific topic within the messaging provider; topic at which the system components interested in these prices can subscribe, as is the case with CARD GUI.

Data Base Server (DBS), in the context of CARD system, is the component that provides the historic price data to CARD GUI, based on a request-reply model (in asynchronous fashion). The historical prices support the feature of showing on the map the price variations since last close, 7, 15 days, 1, 3, 6, 12 months back, and year-to-date (YTD). In order to improve the overall response of the system, DBS prefetchs the historical price data from the database at launch, for the predefined time references, and has it readily available in memory for the client requests. A similar mechanism has been implemented in CARD GUI as well, in a multithreaded fashion.

Capital Allocation through Rectangular Distribution engine (CARD) is the component that generates the rectangular distribution based on which the map of BSE market is drawn. This component encapsulates the recursive algorithm that computes the coordinates of each rectangle on the map board.

The entire CARD system has been implemented in Java.

4 Conclusions and directions for further research

The CARD system (Capital Allocation through Rectangular Distribution) that we have developed, is able to build various visual market maps for BSE. Hovering the pointer over the map, allows for choosing the rectangle to be in focus. For the company in focus is shown a *tooltip*, which supplies the following information: full name of the company, stock symbol, price of the last market transaction, percentage and absolute price variation since the chosen reference. Figure 5 shows the market map built on *Two-Third* distribution model, with the focus on symbol SIF5, and having the color-coded price variation for the last month.

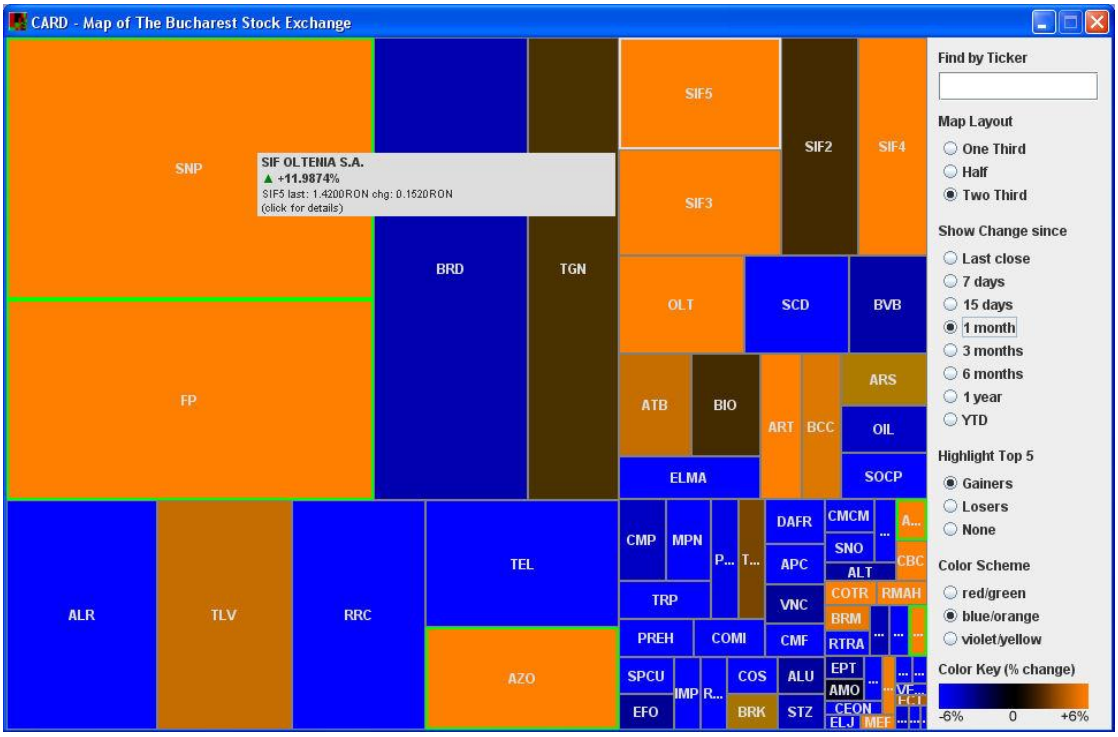

Fig. 5. BSE market map built on *Two-Third* distribution model

In conclusion, the visual representation as a map of the stock market data that we propose offers access, in a quick and relevant manner for human participants, to the overall state of the market at a given point in time. In this paper, we have presented the results of our academic research upon building the market map for Bucharest Stock Exchange (BSE). Our implementation of BSE market map is publicly available through the portal www.bursa.ase.ro.

The graphical user interface available online is transparently updated with fresh market data from BSE every minute and, therefore, provides a quick and accurate view upon the changes occurred on Bucharest stock market, over a desired period, to all the interested parties: market analysts, brokers, investors etc.

CARD engine provides algorithms for constructing, along with 2-D maps, 1-D bars with color-coded segments that can be used for capturing the weight and the evolution of each market sector within an exchange section, or on the stock market as a whole, along with corresponding price variation, for instance. These graphical bars with multicolored segments are essentially similar with what Donald Knuth proposed as a 1-D representation of a multicolored line, with the length of a portion representing the length of each file [2].

Our undergoing research and development for improving the visual experience offered by tree-maps of the stock market are focused on three directions:

- expand the visual depth of the map, by employing data regarding the structure of the market, within the filling algorithm, for driving, in a flexible manner, the distribution of companies within an industrial sector, along with the allocation of the sectors on the overall market map;
- mechanisms for embodying the stock market map with enriched company specific financial data, comments and analyses from the market specialists;
- identify other company specific attributes that can be displayed based on the market map framework, such as stock entropy, and integrate them within CARD system.

Acknowledgements

We would like to extend our thanks and appreciations to Andrei JURUBIȚĂ and Alexandru LIXANDRU, who follow the 2009-2010 Master's program series, and

willingly devoted their time and energy for a substantial contribution in the final integration phase of CARD project.

References

[1] B. Shneiderman, "Tree visualization with Tree-maps: A 2-d space-filling approach", *Department of Computer Science & Human-Computer Interaction Laboratory, University of Maryland*, June 18, 1991 (in ACM Transactions on Graphics).

[2] D.E. Knuth, *The Art of Computer Programming: Volume 1 / Fundamental Algorithms*, Addison-Wesley Publishing Co., Reading, MA, 1968, pp. 305-313, 435.

[3] H. Samet, *Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing Co., Reading, MA, 1989.

[4] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents", *Proc. of the IEEE 7th International Conference on Pattern Recognition*, Montreal Canada, 1984, 347-349.

[5] J. Ha, R.M. Haralick, I.T. Phillips, "Recursive X-Y Cut using Bounding Boxes of Connected Components", *Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR '95)*, IEEE 0-8186-7128-9/95, 1995.

[6] J. L. Bentley and J.H. Friedman, "Data structures for range searching", *ACM Computing Surveys*, Vol. 11, No. 4, 1979, pp. 397-409.

[7] J.L. Bentley, "Multidimensional Binary Search Trees in Database Application", *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 4, July 1979.

[8] D.B. Lomet and B. Salzberg, "The hB-tree: A multiattribute indexing method with good guaranteed performance", *ACM Transactions on Database Systems*, Vol. 15, No. 4, December 1990, pp. 625-658.

[9] C. Vinţe, "Upon a Tridimensional Perspective of the Stock Market", *Proc. of the 9th International Conference on Informatics in Economy*, Bucharest, May 7-8, 2009.

[10] D.L. Kreher and R.S.R. Douglas, *Combinatorial Algorithms – Generation, Enumeration and Search*, CRC Press LLC, 1999.

[11] Z. Michalewicz and B.D. Fogel, *How to Solve It: Modern Heuristics*, Springer, Berlin, Heidelberg, 2000.

[12] C. Vinţe, "Upon a Trading System Architecture based on OpenMQ Middleware", *Open Source Scientific Journal*, Vol. 1, No. 1, 2009, Available at: http://www.opensourcejournal.ro/

[13] Sun Microsystems, Inc. – Java Message Service, Available at: http://java.sun.com/products/jms/

[14] Sun Microsystems, Inc. - Open Message Queue: Open Source Java Message Service (JMS) - https://mq.dev.java.net/

[15] T. Erl (with additional contributors), *SOA Design Patterns*, Prentice Hall by SOA Systems Inc., New Jersey, NY, 2009.

[16] M. Richards, R. Monson-Haefel, A.D. Chappell, *Java Message Service (Second Edition)*, O'Reilly Media Inc., Sebastopol, CA, 2009.

[17] C. Vinţe, "Upon a Message-Oriented Trading API", *Informatica Economica Journal*, Vol. 14, No. 1/2010.

[18] M. Kalin, *Java Web Services: Up and Running*, O'Reilly Media Inc., Sebastopol, CA, 2009.

[19] Bursa de Valori Bucureşti, http://www.bvb.ro, there are links to web services that provide data regarding listed companies, trading activity, delayed price data etc.

[20] ASETS portal, http://www.bursa.ase.ro, within The Bucharest Academy of Economic Studies.

Claudiu VINŢE has over fifteen years of experience in the design and implementation of software for equity trading systems and automatic trade processing. He is currently CEO and co-founder of 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.

Alexandru JURUBIŢĂ is a web developer and co-founder of Blue BitBox, a young software company based in Bucharest. He holds a master's degree in Economic Informatics from the Academy of Economic Studies in Bucharest. His fields of interest are user experience design, human–computer interaction and artificial intelligence.