*The development of Moving Averages algorithm for E-FAST*

|  |  |
| --- | --- |
| **<<*TORGIE ALEXANDRU*>>** | **<< *Fotografie student* >>** |

Cuprins

[Introducere 3](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619228)

[1. Lucrări asemănatoare 5](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619229)

[2. Analiza sistemului 6](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619230)

[2.1 Definirea Problemei 6](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619231)

[2.2 Context 7](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619232)

[2.2.1 Subiect 7](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619233)

[2.2.2 Utilizare 10](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619234)

[2.2.3 Dezvoltare 13](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619235)

[2.2.4 IT 13](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619236)

[2.3 Necesar de calitate 14](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619237)

[3. Proiectarea sistemului 14](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619238)

[3.1 Proiectare logică 14](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619239)

[3.2 Proiectare fizică 17](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619240)

[4. Dezvoltare 18](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619241)

[5. Validare 18](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619242)

[6. Concluzii si dezvoltare ulterioara 18](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619243)

[7. Bibligrafie 18](file:///C:\Users\alexandru\Downloads\Ionel-Mihai%20Motoc.docx#_Toc390619244)

# Introduction

*~2 pages*

What is E-FAST?(*1 para*)

E-fast is a financial application that will help assist financial decisions by using statistical and financial indicators. E-fasts key features are: speed, accuracy, performance.

Investing in something starts to be more and more difficult today because there are lots of different opportunities. Choosing the best opportunity that comes up is something very difficult. Some may argue that you would even need a special consultant in order to choose the best investment opportunity. Trading financial instruments is still one of the most profitable investment that one may choose to invest in, but with great profits, great risk are sure to follow. Financial analyst and brokers are always in struggle to give assurance that their techniques are up top and profitable. E-fast application aims to become one of the most useful instrument that will help an analyst take profitable decisions and also E-fast will assure confidence for an investor due to its features.

Competition is high in the financial fields, and an analyst must always be with one step in front of the competition. E-fast will boost the performance of an analyst through the fast execution of two consecrated indicators. E-fast will give the analyst the possibility to study the market on a large number of years in order to accurately discover the trends of the analyzed financial instruments.

E-fast will implement Moving Averages method which is one of the most used and popular statistical indicator that shows the trend of a series of values. Moving Averages strategy is used for over 40 years and it hasn`t failed investors yet. Using MA an analyst can easily predict the trend of a certain title. Not only will E-fast implement MA algorithm but will also improve this strategy by giving the analyst the opportunity to use this indicator over a large period of time without any lack in execution speed. Doing so the analyst can gather even more information about past trends of a title and this will help lead him to a better understanding of the future trends.

When it comes to calculus of financial indicators, speed is one key factor because analysts need to have their decisions taken as fast as possible due to the fact that on a financial market the most profitable decision is the one taken at the best moment and due to the volatility of the market that moment last for a few moments.

E-fast platforms basic usage is to assist a financial analyst in his decisions in a financial portfolio management. E-fast will offer two different indicators which if used together will represent a powerful financial indicator that will launch the user of this application far in front of others.

Moving Averages component that will be developed in this application will implement two types of formulas first will be a Simple Moving Average and the second will be an Exponential Moving Average, that will show the trends of the title. Evolutionary algorithm will complete this strategy by showing the correlations between different title. Using this two computational methods together will lead to a more complex result and decision. The application will also keep a high standard of execution speed by using a grid system to distribute this operations.

As a secondary component the application will implement a scheduler that will manage the execution and output of each task. The application can receive a configuration file and will continue to compute the needed analysis.

E-fast wants to offer all the benefits that a counterproductive system couldn`t offer, such as :

1.Possibility of working with historic data

2.Implementing to complementary financial indicators

3.Using a grid computing to make the execution run faster

# Related work

Similar applications from the following viewpoints:

* Applications that use distributed computing platforms for finance

ScaleOut software is a software that provides financial analysis over big data, historical data and also using time costing algorithms in a record time due to the use of a grid system that helps distribute the operations. “Using ScaleOut Analytics Server, customers can maintain investment positions in the IMDG, update them with streaming market data and trades, and simultaneously run continuous risk analytics on the positions according to defined business rules. Alerts can be raised when out-of-range conditions occur. Benefit: real-time response is possible to changing risk profiles during the trading day.”[1]

“Digipede Technologies delivers grid computing software that dramatically improves the performance of financial applications by distributing execution across a network of Windows desktops, servers, and cluster nodes. Built entirely on .NET, the Digipede Network works with Microsoft tools including Excel and Visual Studio, and is radically easier to buy, install, learn, and use than other grid systems”[2]

* Applications that implement financial computational methods

Yahoo finance uses Moving Averages to express indicators to help investors take good decisions when it comes to managing their portfolio. Yahoo finance is a web site sponsored by yahoo that provides basic information and analysis article with focus on the USA markets. This application also offers many financial indicators including Simple Moving Average and Exponential Moving Average.

Google finance also uses Moving Averages algorithms to give investors a good market analysis. This application offers stock information along with headlines news containing decisions taken by financial corporations. Google finance is oriented towards showing the possible trends in the market (Google Trends) and for getting these trends it uses Moving Averages methods.

# System analysis

## Problem definition

What is missing for the researchers and possibly for (…?) and the application will provide.

Most popular applications that use financial indicators such as Moving Averages usually don`t use distributed processes and they do not use a concurrent algorithm. Financial analyst deals with problems when computing MA algorithms because when they use MA over historical data they get the output only after a long period of time because the popular implementation algorithm lacks in computational efficiency.

MA specific applications can`t build a thorough research for one title because it would take due to the lack in efficiency. Differential usage of a MA over a specific title will lead to more profits for an investor. Common application that use MA strategies by comparing two MA with two different numbers of days (one over a smaller number short term MA of days and one over a bigger number of days long term MA) use the same strategy for all the titles that will lack in profits. Using a different strategy for each specific title will give a larger amounts of profits and will also give more confidence to an investors. “There are studies which use profit optimization in order to identify the best MA strategy” [3]. Each title has its own specific atributes that define the series of values and trying to use the same strategy for all series will mean less profits.

Today, most companies who want to analyze the stock market use popular indicators such as SMA/EMA and doing so they can make a decision whether to trade or not.

The methods used in order to foresee the trend usually are very inefficient and they have great costs when it comes to time. Some times the cost of time tends to such extent that the needed time for the applications to compute makes a trend unprofitable because the indicators output is given only after the indicated moment of trade has already gone. Applications that currently develop market indicators run on a very long period of time making some transactions impossible to attend.

The output of some market analyses applications can sometimes be corrupt because the numbers used aren`t shown with the needed numbers of decimals. What certifies a good result is a comparison between two methods.

E-fast will provide the needed speed of computation and also will have two methods implemented with almost no errors for the numbers decimals points. E-fast applications purpose is to inspire confidence for any investor and doing so, it implements two stock market analyses indicators.

Within E-fast application I will develop the Moving Average computational method that aims to give an exact and fast result. Moving Averages methods is used to express a good strategy for future transactions. When applying this statistical method, we usually use it on large numbers of days (historical data) because this way we can find the best strategies for these methods. Sometimes the execution of Moving Averages if it is applied on a large number of days it can run very slow making it inaccurate and not very useful for the decision of selling or buying. E-fast aims to eliminate this problem by using multiple threads within this algorithm and also when there is historical data used (a large number of days), the application will distribute the tasks of this method to a grid computing architecture that will output the results faster, making the algorithm perfectly accurate and usable.

E-fast application gathers daily information from the stock market exchange, doing so data always grows making it increasingly more difficult to give the output in usable time, but due to the using of the grid that is available for the application there will be no problem in analyzing the data. Moving Averages algorithm can be used on a large number of days without lacking in accuracy or speed.

E-fast aims to acquire a plus of efficiency among the existing systems by resolving the existing problems that are on this specific market through a different approach to these problems.

Now, there is a need for an application to be oriented towards performance and accuracy and this needs give the E-fast application an opportunity to make these processes more efficient.

E-fast aims to research all posible time frames strategyes for every title and will output th

e strategy that works best for one title. An investor can than choose the best strategy that works for the title he want to transactionate.

Making the application more speed efficient will give a financial benefit:

* Financial markets are very volatile and if a decision isn`t taken in the first moments it is signaled, there is a high risk that the decision will produce a reduced profit because the prices has already grown/shrunk by the time of the transaction.

Giving the calculus more accuracy will facilitate the users:

* After executing the algorithm, if the result has suffered a small error this could make a transactions profit to be smaller than expected.

Figure 1.1 shows the exact purposes of the Application. As it is shown in the picture here whe the current needs that E-fast has identifyed and will solutionate:

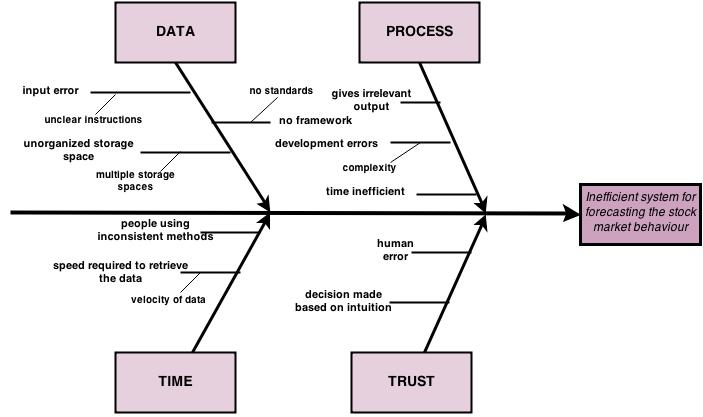


Fig 1.1 Fishbone chart

Figure 1.2 is better represented vision of the current and furute implementations of E-fast application:

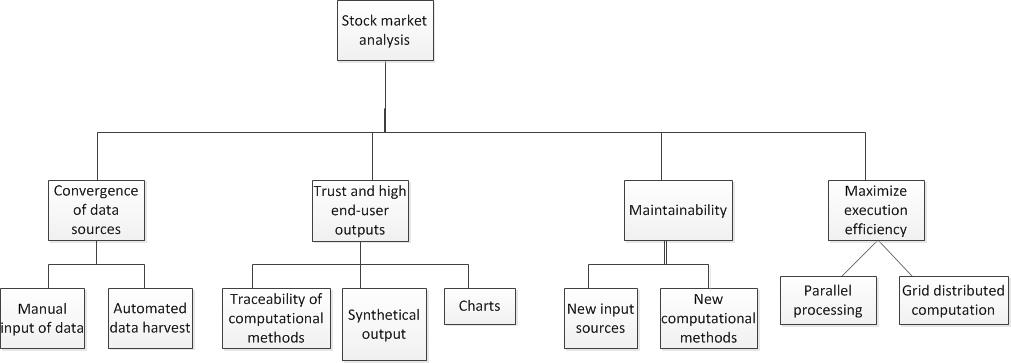


Fig 1.2 E-fast application targets

## Context description

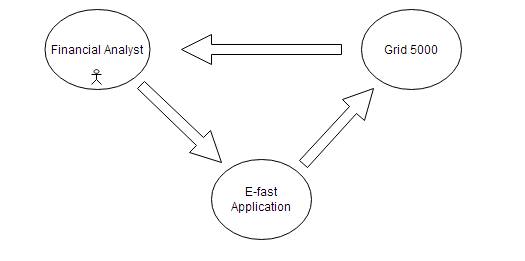
Document the system context using the facets approach.

### Subject

Stakeholders:

The most important stakeholder that takes part in this application is the financial analyst who uses the most of the applications benefits. He will study the output of the application and use this data for future financial transactions.

The grid system is also an important stakeholder that aims to give the output of the operations as fast as possible. In order to use it we must keep in mind the specifications of its interface.



Data:

The Data which E-Fast application will analyze is very important aspect for the app in general. It has to consist of the titles name, value and the date in which that certain titles had that value. Data must be characterized by clarity and proximity and because of this very reason E-Fast platform provides support for historical data which means that the app will have to gain data every day and store it for any need of the customer. For our computational methods to work with their full potential of profits marking than the app will have to gain data from large amount of time. The historical data keeps past information prices of each titles and helps us to forecast the titles future based on the two computational methods (Moving Averages and Evolutionary algorithm). This data is sensible because any mistake made when introducing it might cause a different result in the indicators that E-fast Provides. For this very reason data that will be used within the application are divided into two types of data gathering:

🡪 Data that is provided by the customer: this data is suspect to mistakes but it is the best way to gather data when there is no internet connection available or when the data that the client has got is in hand and he wants to use that certain data. This method will take the data from a file and introduce it into the apps database.

🡪Data that is provided by an online vendor that gathers daily information about each title such as yahoo finance. The E-fast app will have a module that can connect through the internet and introduce automatically all the needed information from such vendors. This way there is no human interference thus making it safer to introduce data this way.

Also what characterizes the data is their format and what titles data should describe is first of all the name of the title using its market name which is an abbreviation from the full name. The price must be expressed in monetary units. The date should be concise meaning it must show for titles that are transactionated by hourly the exact hour, minute, second in which the title had that price, and also for the same title the application will store the daily price of the title which is the closing price at the end of the day (the price of the last ended transaction).

When the application gets all of the needed data this has to be stored in a way that it will ease the access of the user. Data will be added into database tables which are categorized in such a manner that data will be explicit when needed. The customer will have to his linking all the data that the application has stored.

Financial Markets:

What matters most for an investor is the possible profit he makes and what E-fast aims to do is to give an investor an index that will help him take a decision.

“The term markets mean an aggregate of possible buyers and sellers of a certain good or service and transaction between them. A financial market is a market where people, entities can trade financial goods such as securities, commodities or fungible items at prices that reflect the supply and demand of the market at the time when the transaction is made. Markets work by placing many interested buyers and sellers, including households, firms, and government agencies, in one "place", thus making it easier for them to find each other.”[4] Within the financial sector, the term "financial markets" is often used to refer just to the markets that are used to raise finance: for long term finance, the Capital markets; for short term finance, the Money markets. The E-Fast application aims to be applicable for each type of financial market be it stock market (which provide financing through the issuance of shares or common stock, and enable the subsequent trading thereof) or money (which provide short term debt financing and investment) market because the techniques used to gather information for the economic decision is the same for all financial markets. But first of all this application aims to be helpful for the stock market due to the fact that it is used the most. Within the stock market what interest us the most is the way a title of value that represent a stock fluctuates because this fluctuation will mean profits for us. These Fluctuations of a stock in time will give a general direction of its value which is called market trend. These trends are classified as secular for long time frames, primary for medium time frames, and secondary for short time frames. The terms bull market and bear market describe upward and downward market trends, respectively and can be used to describe either the market as a whole or specific sectors and securities. E-fast can define which way a trend of a stock is headed through the computational method that it implements whether a stock is a bull or a bear.

Bull:” A bull market is associated with increasing investor confidence, and increased investing in anticipation of future price increases. For a title this will mean that investor’s confidence is in a continual increase due to the ascending trend of the title. For an investor this means that he will continue to buy more titles because their value will grow.

Bear: A bear market is a general decline in the stock market over a period of time. It is a transition from high investor optimism to widespread investor fear and pessimism. For a title this will mean that investor`s confidence is in a continual decrease due to the descending trend of the title. For an investor this will result in him selling titles because their value will plunge.”[5]

An investor always want to make profits out of his investment and due to this reasons he can use the two trends bear and bull to make a decision in buying more stocks (bull) or selling more stocks (bear). This 2 methods use two types of positions that are recommended by brokers short and long positions:

Shot positions:

“1. The sale of a borrowed security, commodity or currency with the expectation that the asset will fall in value.

2. In the context of options, it is the sale (also known as "writing") of an options contract.

For example, an investor who borrows shares of stock from a broker and sells them on the open market is said to have a short position in the stock. The investor must eventually return the borrowed stock by buying it back from the open market. If the stock falls in price, the investor buys it for less than he or she sold it, thus making a profit.”

Long positions:

“1. The buying of a security such as a stock, commodity or currency, with the expectation that the asset will rise in value.

2. In the context of options, the buying of an options contract.

For example, an owner of shares in McDonald's Corp. is said to be "long McDonald's" or "has a long position in McDonald's." ”[6]

Maximum drawdown:

It is one of the key indicators that shows the risk that an investor takes while using a kind of strategy for transactions.

“The maximum cumulative loss from a peak to a following bottom, commonly denoted the maximum drawdown MDD, is a measure of how sustained one’s losses can be. Large draw-downs usually lead to fund redemptions, and so the MDD is the risk measure of choice for many money management professionals – a reasonably low MDD is critical to the success of any fund.”[7] It is calculated as a percentage value and offers the investor a worst case scenario.

The formula is:

 \text{MDD}(T)=\max_{\tau\in (0,T)}\left[\max_{t \in (0,\tau)} X(t)- X(\tau) \right]

Profits through Computational methods (Moving Averages and Evolutionary Algorithms)

Moving Averages:

What SMA and EMA algorithms aim to acquire is the best interval of days in order to maximize the profits. „In statistics, a moving average (rolling average or running average) is a calculation to analyze data points by creating a series of averages of different subsets of the full data set. It is also called a moving mean (MM) or rolling mean and is a type of finite impulse response filter. Variations include: simple, and cumulative, or weighted forms. Given a series of numbers and a fixed subset size, the first element of the moving average is obtained by taking the average of the initial fixed subset of the number series. Then the subset is modified by "shifting forward"; that is, excluding the first number of the series and including the next number following the original subset in the series. This creates a new subset of numbers, which is averaged. This process is repeated over the entire data series. The plot line connecting all the (fixed) averages is the moving average. A moving average is a set of numbers, each of which is the average of the corresponding subset of a larger set of datum points. A moving average may also use unequal weights for each datum value in the subset to emphasize particular values in the subset. .” [8]

“Whether you use one, two, or many MAs, the concepts and applications are essentially similar. Either the market price must close above or below its MAs to signal a buy or a sell, or the MAs themselves must change their relationship to one another in order to signal a trade. Richard Donchian popularized this approach in the 1950s although it was probably being used well before then. In the typical MA-based system, signals are generated in either of several ways:

• Price closes above or below its MA. Closing above the MA is considered a buy signal whereas closing below the MA is considered a sell signal.

• In the case of multiple MAs, the approach signals buy or sell signals when the various lengths of MAs cross one another.

• In the case of MAs of closing/opening, or high or low prices, signals are generated when crossovers of the MAs occur as defined by the theory or method.”[9]

This application considers both long and sells positions in trading all types of titles, obligations, equities, etc. “Formally, an investor is considered to be long and holding a title if the short term moving average (STMA) is higher than the long term moving average (LTMA) and sells a title otherwise. Buying and selling is assumed to occur at the closing price the first day after the signal is observed”.[3]

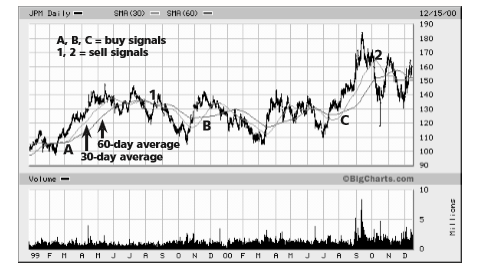


Fig 2.1 Example of SMA chart

We are going to use two types of Moving Averages: SMA (Simple Moving Average) and EMA (Exponential Moving Average)

SMA is a unweighted mean of the previous n data; in our case data will represent the prices of a title in the past n days. The formula for an example of a SMA for the past n’ days closing prices (p_M, p_{M-1},\dots,p_{M-(n-1)}) will be:

\textit{SMA} = { p_M + p_{M-1} + \cdots + p_{M-(n-1)} \over n } [9]

EMA is weighted mean of the previous n data; it has a type of infinite impulse response filter that applies weighting which decrease exponentially. The weighting for each older data decreases exponentially. The formula for an example of EMA for the past n` days of closing prices is:

\text{EMA}_{\text{today}} = \text{EMA}_{\text{yesterday}} + \alpha \times

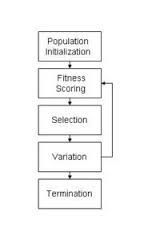
(\text{price}_{\text{today}} - \text{EMA}_\text{yesterday})

\text{EMA}_{\text{today}} = { \alpha \times (p_1 + (1-\alpha) p_2 + (1-\alpha)^2 p_3 + (1-\alpha)^3
p_4 + \cdots ) }

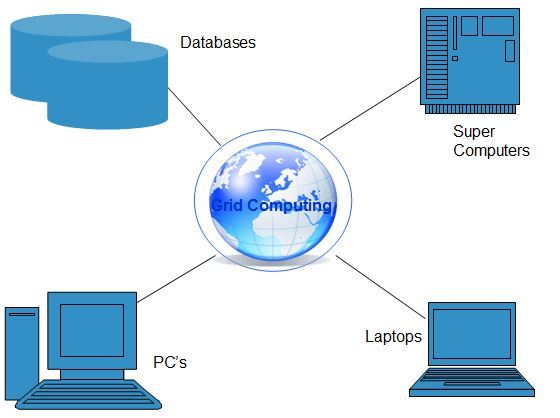
p_1is \text{price}_{\text{today}} p_2is \text{price}_{\text{yesterday}}

Where  \alpha = 2 / (N + 1)  [9]

Evolutionary Algorithm:

Evolutionary Algorithms are stochastic, iterative, population based approaches, which are inspired by the mechanisms of natural evolution. Charles Darwin ﬁrst postulated that natural evolution is based on the principle of the ‘survival of the ﬁttest’ and improvements are caused by random alterations in the reproduction cycle. In biology these random alteration are either caused by random mutations or by recombination. EAs start with a population P of randomly initialized solutions and iteratively apply directed selection and reproduction using recombination and mutation until a termination criterion is met. This way the EA gradually produces more and more adapted/optimized solutions and converges to close to optimal solutions. Contrary to other optimization algorithms the EA requires only the output of the target function for a given solution to guide the selection process and a suitable solution representation to apply the evolutionary operators of recombination and mutation to. Therefore, EAs can be applied to many kinds of optimization problems, where classical optimization approaches often fail. The target function may be noisy, non-linear, non-diﬀerentiable, discontinuous, multi-modal, and high dimensional and may be subject to multiple kinds of constraints.

Within E-fast EV algorithm aims to give a result in which after a certain number of fitness functions that are based on relative value of the growth or decrease in price of a title.

Grid Computing:

Grid Computing or distributed computing is a field of Computer science that studies flexible solutions to coordinate the gathering of resources of different entities or organizations. The grid represents the capacity, using only a set of opened standards and protocols, to have access to its applications, data, computational power, storage capacity and other calculation resources through a network. Grid uses a type of system that is parallel and distributed, that enables sharing, selection and aggregation of resources distributed across different domains based on their performance, availability, capability and performance. The E-fast application will use a grid in order to make the computational process run faster.

Output:

The output of the application will be represented by the value of profit for the Moving Average method and by the clusters if using the Evolutionary Algorithm.

Profits value:

The profits that an investor can make if he is using one of the two Moving Average methods is made from each successful transaction that the indicator has signaled us. We are using a strategy in which every time a SMA or EMA with a window opening of few days surpasses a SMA or EMA with a window opening of more days than the other will mean for the investor a signal for a long transaction. In case the SMA or EMA with a fewer window of days gets lower than the SMA or EMA with longer larger window opening of days than this will mean for the investor a short transaction. The output will consist of the profit gained from all the transactions made for a title within the window of days limits (all values between 2-100) and for each profit its maximum drawdown will also be available. The output will be represented by a text file.

Clusters output:

The Evolutionary Algorithm will output the values of each cluster that can be visualized by the investor and let him take a decision. Each cluster will be made from titles with similar evolution within the time given. In case of the clasterization method the result will be made from an array of values of each cluster, for each value there is a moment in time that corresponds to it. The output will be chronologically arranged. Also in the result the name of the title will be included. Having the titles name in a cluster can be crucial important for the analyst because he can correlate the result of the Moving Average with the cluster in which a title is. This way he can take similar decisions for titles in the same clusters as the decisions taken for those titles analyzed with the Moving Averages indicators.

### Functional

Financial Analyst is going to be the most important entity that uses this application. Before any investor takes an action on the market he first has to consult an analyst. The financial analyst is a person who analysis the market to such extent that a decision of investment or not can be made due to his expertize. Financial analysis uses for a study of the market the methods that e-fast implements: Moving Averages and Evolutionary Algorithm. What the analyst must do to use the application is to decide the number of days he wants to analyze and after that he can run the program, it is crucially important to have a good idea about the necessary number of days on which the algorithms must run. The analyst can choose whether to use the data he introduces or the data that is available on the applications database. Also the analyst is responsible to make a configuration which the application will run with. It is up to the analyst to decide for the Moving Averages the number of days that will be processed the value of the exponential value in case of the EMA method, the starting and ending number for the window that contains the number of days those averages keeps. After the execution of the program, the analyst can read the output that will be given to him in a text file format. Within the application there are some decisions made internally depending on the parameters that the analyst chooses, for example if the data he wants to study is relatively a small number of days than the application doesn’t need to distribute the execution. There are some parameters that can be let for the application to decide upon, as for the default value of windows sizes or the value of the exponential factor.

The needed parameters for the Moving Average method is:

* The practiced commission for the type of title
* The Exponential factor value
* Short and long terms for the MA number of days
* Information about the title : name, starting day of the analyzed period and the closing day of the analyzed period

Sending the data to the grid is a decision that is taken within the application and if the decision is taken than the application performs some extra tasks. The application has a feature that allows it to distribute the process among a grid computing (grid 500). This feature allows the application to build a configuration file from the parameters introduced by the analyst and together with the configuration file it is able to build a file with the needed data for the analysis. These files among with the jar file will be allocated by the scheduler to run on the grid. After the methods have finished execution, the result is returned to the user in the same manner as the one executed locally.

 Fig 2.2 Use case scenario

The basic processes that compose the application can be expressed through three containers that interact between them :

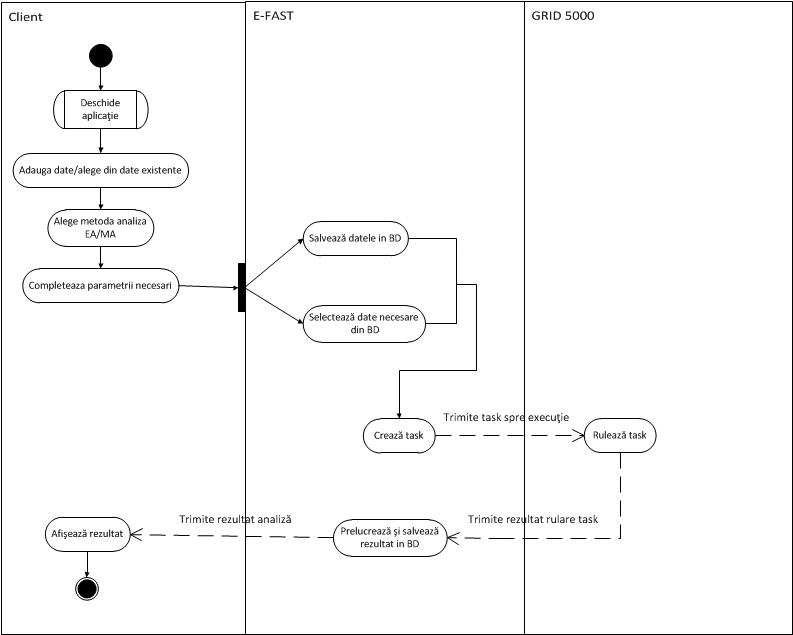


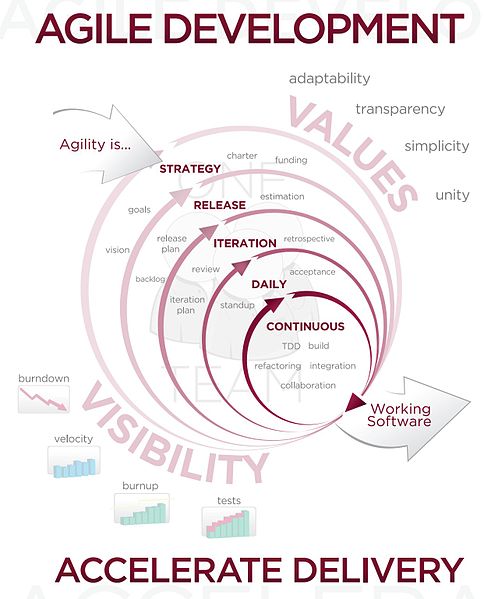
Fig 2.3

 Fig 2.3

### Development

The future development of this application will be oriented towards a user interface that will facilitate the usage of the entire application. The user will be able to choose the tasks he wants to build through a guided interface. Also future development will include some extra features for VIP users. Data snooping will also be included in the process of data preparation that will speed up the whole computational process.

For the development of this application we are going to use Agile methodology developments and for this we are going to define for the application three major iterations process in which we are going to write code and afterwards it is going to be analyzed by the stakeholders involved. The main stakeholder of this application is the financial analyst and due to this reason he will test the application after each iteration is finished.



The first iteration aims to implement the computational algorithm. This iteration will build an algorithm that implements the SMA and EMA formulas. Also this iteration deals with building a data structure that will provide a faster execution. The first business logic layer will provide the profit of a certain combination of short term with long term MA over some values of a title.

The aim of the second iteration is to identify what means an atomic operation for this algorithm in order to change the execution of the algorithm with a multithread algorithm.

The third and last iteration for the CoMeth component will provides an architecture that makes the algorithm eligible to distribute tasks on the grid. The application will build a configuration file that will contain the specific parameters that will be distributed on the grid.

The financial analyst will evaluate the iterations result in the light of three main objectives: execution speed, accuracy of the result, ease of usability.

Using Agile methodology allows developers to keep a strong boundary with the financial analyst. Due to the nature of this project, close relationship with the main stakeholder proves to be benefic because all procedures that composes Moving Averages implementation will be tested gradually and they will be optimized in order to meet the needs of a high quality financial analyst.

### IT

Grid computing:

“The most common resource is computing cycles provided by the processors of the machines on the grid. The processors can vary in speed, architecture, software platform, and other associated factors, such as memory, storage, and connectivity. There are three primary ways to exploit the computation resources of a grid. The first and simplest is to use it to run an existing application on an available machine on the grid rather than locally. The second is to use an application designed to split its work in such a way that the separate parts can execute in parallel on different processors. The third is to run an application that needs to be executed many times, on many different machines in the grid. Scalability is a measure of how efficiently the multiple processors on a grid are used. If twice as many processors makes an application complete in one half the time, then it is said to be perfectly scalable. However, there may be limits to scalability when applications can only be split into a limited number of separately running parts or if those parts experience some other interdependencies such as contention for resources of some kind.”[10]

Computational grids can be recognized by these primary characteristics:

🡪 Made up of clusters of clusters

🡪Enables CPU scavenging to better utilize resources

🡪 Provides the computational power to process large-scale jobs

🡪Satisfies the business requirement for instant access to resources on demand

Technologies used: JAVA, JDBC

JAVA: is a computer programing language that is Object Oriented completely, there are no function nor variables that are not members of one class. Object orientated Programming concepts are the same as in C++, the programs are organized as collections of objects that cooperate, each object is an instance of a class. Each class represents an abstractization of a type of entity from reality, and classes are members of a class hierarchy correlated between them through inheritance relationships. Each object is encapsulated, that means that its representation (its internal structure) isn`t visible to the user but which have access just to the methods that are capable of execution.

JDBC: (Java Database Base Connectivity) is a standard SQL interface of access to the database. It is made from a set of classes an interface written in java, that provides a standard mechanism for database design.

Development environment:

Eclipse IDE (Integrated Development Environment) is an open source integrated programing environment that facilitates the work space management, creation, execution, debugging of applications.

Services used:

Yahoo finance is one of the services that the application uses and it aims to provides stock exchange data which will became useful for the financial analyst if he wants to let the application find the stock data online.

Interactions between the components:

The application knows two different protocols of communication within it: communication with the grid and communication with the database.

The grid has its own interface that accepts communication through http protocol and it must be organized in a specific manner.

The database can communicate with the application through the DBMS. The application can go through a file in which the user can put the stock data and after the application introduces the data into the database.

## Quality requirements

E-fast application aims to achieve some quality requirements that will make the application much more scalability and usage.

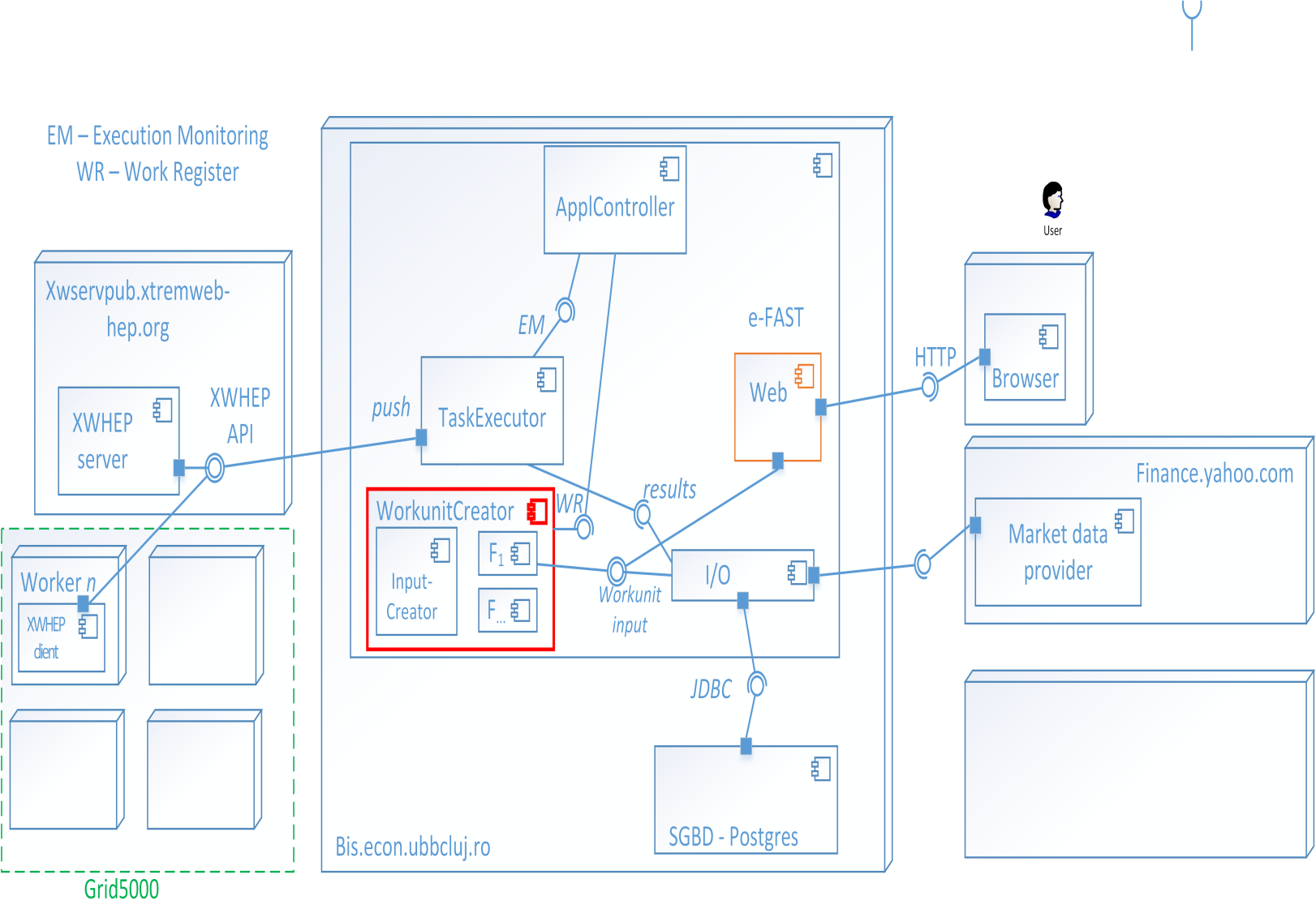
Reduced time of execution is most important goal of this application. Because the markets are moving fast, a financial analyst needs all the information he can gather with some extent of time in such a manner in which he can analyze all the information and lead him to a decision. Traditional software gives the needed indexes values only after a considerable span of time and this because time isn`t one of their priorities. E-fast tries to reduce the time of execution through two methods. First of all, each method runs using parallel algorithms which means that each jar produces more than one thread which means that within the algorithms are designed in such a way that there can be identified some places where the problem can be launched into to two or more separate threads (concurrently). Secondly, the application can use a distributed architecture that will split the execution between all the resources of that are available in the grid. Parts of the execution can be divided between multiple computers and afterwards reattached so that it will give a unitary output.

Precision of the result is another requirement that will give the investor confidence in the results provided by the application. To do so the application will use generally big data types that can store number with double floating-points and have a wide range of value. This type of data type will give more precision throughout the calculus of the application.

Using only indicators with good performance assures success for the analyst decisions. The quality of the output that the application gives is actually the quality of the indicator. E-fast will use only consecrated indicators like those given by the Moving Averages and Evolutionary Algorithm.

# System design

## 2.1 Logical design

 Fig 3.1 System architecture

# Components:

* **Config** – read execution configuration
  + System designer – configure the way the application behaves
  + End-user – configure parameters that are relevant for the end-user: ex: the way he constructs the input data for the experiments.
* **Input**/**Output (I/O)** – read input data, write results (text files)
  + Reading and writing text files with different formats
  + **Formatter**
    - **FormatReader** – is capable of reading textfiles with particular format (input data for the experiments – datasets)
    - **FormatWriter** – is capable of creating the format for the results (that will be written into textfiles) – for different types of result analysis – for the finance expert analyst.
      * It is used with a specific CoMeth in order to present the results.
* **WorkunitCreator**
  + This class intensively uses different types of Workunit factories (F1, F…).
    - For each computational method, build a set of specific factories. Each such factory must create workunits according to a unique approach (i.e. for EMA&SMA – factory **f1** creates a set of workunits, where a workunit **wi** executes the .jar for 5 shares, 2-100, 4000 hist. vals.)
    - The Scheduler uses WorkunitCreator to construct the workunits.
  + Reads from configuration (or from user parameters).
  + Based on user’s parameters for a specific computational method builds a user’s bag of work units from:
    - A particular computational method (holds only the name & ID),
    - The name of the input file
    - The name of the configuration file (contains paris of key=value) --- these are the execution parameters (i.e. for GA: number of clusters to be generated) !!! NO MORE command line
* **ApplController**
  + Coordinates the execution of applications for all users
  + Uses the WorkunitCreator to create workunits
  + Uses the TaskExecutor to execute the existent workunits
* **TaskExecutor**
  + Is *singleton*
  + Receives / accepts new workunits in 2 ways:
    - using a WorkunitCreator (i.e. **f1**)
    - workunits from users (Users) and stores them into its own datastructure (ArrayList)
  + For each workunit it creates a task and adds it to its own datastructure (ArrayList):
  + Threads:
    - 1. Manage workunits: receive new workunits and report complete BoW to user
    - 2. Manage the execution of the tasks:
      * Communicate with the DCI: to phisically send tasks and retreive the computed results (for now, it simply stores the output(s) into a local folder).
    - The scope of the threads must be identical with the Scheduler class instantiation
  + Receives the results from the XWHEP server and stores them into the database using the I/O component through its *results* interface.
  + Is capable of simultaneously managing independent tasks from multiple end-users and applications:
    - Handles and insures the execution and completion of all tasks
  + Uses a prioritization mechanism for the task execution (selection method – first approach: FCFS/FIFO)
  + Reports at any time the completion process per: end-user, application, work unit.
  + Is not responsible for synthesizing the results of tasks in any way.
  + Stores the obtained results and corresponding work units.
  + Can trigger the execution of tasks in two ways: locally & delegated on foreign (external) computing infrastructure.
  + For each application the scheduler logs the moment in time when it began and completed the execution of the associated BoW.
  + *Intensively uses an Executor (for future developments).*
* ***ExecutionManager***
  + ***TRIES*** *to execute tasks either on the local machine or on the XWS.*
  + *Communication with xwhep server (XWS) – HTTP client connectors.*
* **CoMeth (Computational method)**
  + **TRIES** to execute tasks either on the local machine or on the XWS.
  + Communication with xwhep server (XWS) – HTTP client connectors.
  + Uses a TaskCreator. The created tasks are handled to the scheduler in order to be executed and from which it expects the corresponding results.
  + Aggregates the results of the tasks and by using an appropriate FormatWriter saves the results to a text file.

## Key words:

**User**

A human user that is interested in having her BoW executed by the E-FAST framework. The user selects from her UI the computational method and the appropriate input/exec. parameters.

**Workunit (wiw)**

The end user aims at accelerating the execution of her own application (i.e. a particular computational method). Usually the end user executes (reruns) the method on a particular input for a set of key parameter combinations. Each such rerun but with different parameters/input may constitute a workunit. For each workunit the end user expects a corresponding output (this is the result produced by the computational method for the given input and key parameters combination).

Attributes:

* User (the identity of the end user interested by the completion of a BoW) - integer
* ID - a unique identifier – integer
* Exe – the name of the executable file representing the computational method (application)
* Parameters – a string indicating the parameters for the execution of the application

For a computational method generate a .jar file that expects the following parameters:

* Path – the absolute path with i/o & eventual configuration files
* Parameters - a string indicating the parameters for the execution of the application

This is the Exe attribute of the workunit.

**Task**

For each workunit that must be completed, the scheduler creates minimum one corresponding task. The task is an instantiation of a workunit. Each task *knows* the identity of the workunit it belongs to. The completion of a task leads to registering the completion of the corresponding workunit. This event is accounted by the scheduler.

* The constructor for the Task must accept a Workunit parameter, like:
  + Workunit w = new Workunit(………);
  + Task t = new Task(w);

**BagOfWorkunits (BoW)**

All workunits received at scheduler:

* Considered for all applications managed by the scheduler or
* may be considered per end user application.

The completion of a BoW must trigger the generation of the final output of an application (and further notice the end user).

**BagOfTasks (BoT)**

All tasks created for the workunits:

* Considered for all applications executed by the scheduler or
* may be considered per end user application.

**Task**

* Operating system/platform dependecies, JRE,...?
* Executable file
* Execution command (the exact line that when provided to XWS, the executable file is run)
* The input file associated with the task
* *State*:
  + New (NEW)
  + Transferring
  + Sent (SNT)
  + Executed (EXE)
  + Completed (CMP)
  + Received (REC)S

**XWS**

* The Xtreme-Web HEP server running at <https://xwservpub.lal.in2p3.fr:4326>

CoMeth (Computational Method) unit implements Moving Averages Method:

The Moving Averages computational method will be implemented to provide the application the needed algorithm to output the result of this method. This method will implement to approaches toward Moving Averages: First method will implement a Simple Moving Average; Second method will implement an Exponential Moving Average. The financial analyst can see a varied result given by this method due to the fact that he can compare the two approaches and choose the one that is more rentable for a certain title. Each implementation of the calculus is then used to aggregate the result of profits it can generate while using these methods. The two Moving Averages implementations will start calculating the results with the provided data for a title and retain the profit of each transaction.



Fig 3.2 Profits and MovinAverages relationship

Within the application there will be provided a factory algorithm implemented in order to use the full extent of the grid. The factory method aims to build different types of work units depending on the need of the user. This will build a configuration file that will be send together with the jar that contains the Moving Averages implementation to the grid that will execute each unit on separate resource on the grid.

We will use Factory design patter which comes under creational pattern as this pattern provides one of the best ways to create an object. 1. “In theory, your code is completely isolated from the implementation of the interface, thus making it possible to transparently swap one implementation for another” [11] We have used a factory design for this implementation due to the following reasons:

1. Swapping one implementation for another in our case swapping from one WorkUnit to another. This way depending on the demands of the user the application will execute one of the three types of WorkUnits.
2. We are not constrained into using only one type of WorkUnit, the application can build any combination of the three types of WorkUnits. It will be up to the scheduler to decide the combination of WorkUnit .
3. The concept of bag of tasks is scalable for the grid architecture by using the parameters the user choose will be implemented in three different types making the execution faster because a bag of tasks cannot be the same for each type of architecture on the grid.

 Fig 3.3 WorkUnits Factory

Moving Averages implementation has an algorithm that permits it parallelized execution leaving an execution manager to choose when and how many threads to execute.

 Fig 3.4 Execution Management

Interfacing with other components:



Fig 3.5 Components that interact with MA CoMeth

Moving Averages implementation is in close relationship with the I/O mechanism. The I/O unit is responsible for the preparation of the data that will be analyzed and also it is responsible for to save the output of this method.

## 2.2 Physical design

Data structures used within the development of the Moving Averages:

In the E-fast application the physical design will consist of the following implementations:

1. PostgresSQL database;
2. Evolutionary Algorithm;
3. Moving Averages ;

For the Moving Averages method it is important to keep the value of title together with the date when the title was sold with that value. For this reasons we have developed a data structure that will provide this feature:



Fig 3.6 ValueSet datestructure

Also within the application we have used specific collections such as Maps, Sets, Lists. Collections are an object that groups more elements into one entity. Through collections we will have access to different data types. Collections are used to memorize and manipulate data, and also to transmit data from a method to another.

In Java there are some interfaces that represent the basic mechanism of the management of a certain type of collection. The main purpose of an interface is to provide usage of the collection independently of the way they are implemented:

Set: Provides implementation for the concept of the mathematical notion named also “set”. It is a collection that contains no duplicate elements.” The Set interface places additional stipulations, beyond those inherited from the Collection interface, on the contracts of all constructors and on the contracts of the add, equals and hashCode methods.” [12 docs.oracle.javase/7/]

List: Describes list (sequences) of indexed elements. Lists can contain duplicates and provides precise control of the position of an element using the index of the element. “The List interface places additional stipulations, beyond those specified in the Collection interface, on the contracts of the iterator, add, remove, equals, and hashCode methods. Declarations for other inherited methods are also included here for convenience.”[13 docs.oracle.javase/7/]

Maps: Implementations of this interface are objects that associate for each element a unique key. It can`t contain a duplicate key and each key has associated a single element.” The Map interface provides three collection views, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings. The order of a map is defined as the order in which the iterators on the map's collection views return their elements. Some map implementations, like the TreeMap class, make specific guarantees as to their order; others, like the HashMap class, do not. “[14 docs.javase/7/]

Moving Averages algorithm:

1. The application reads all the needed parameters from the configuration file: input file, output file, number of periods short term/ long term, commission, exponential factor (if the user want to he can use default values also).
2. Data structures are populated with the data that will be analyzed.
3. The algorithm starts two separate threads each of them processing either SMA or EMA on the given data. And also each thread calculates the profit for the given parameters.
4. The algorithm runs until all the possible combinations of days terms are executed and each thread can return a list with the results.
5. Each thread returns results and the result is ordered ascending by the profit. The output is written in a file mentioned in the configuration file.

 Fig 3.7 Computational method flow

SMA:



Fig 3.8 SMA algorithm

EMA:



Fig 3.6 EMA algorithm

Profits algorithm:

 Fig 3.9 Profits algorithm

# Development

Interesting code samples and implementation solutions.

One of the most challenging problems that I have encountered is how to make the implementation of the Moving Averages method concurrent. For this I have tried comparing two different types of interfaces that java SE 7 has offered. First of all I have tried implementing the Runnable interface giving each thread common memory usage and the possibility to communicate through common objects. The code would have looked this way:

**public** **class** Execution2 **implements** Runnable{

**private** **static** ValuesSet *val* = **new** ValuesSet();

**boolean** s=**true**;

**private** Lock lock = **new** ReentrantLock();

**private** ValuesSet sma1 = **new** ValuesSet();

**private** ValuesSet sma2=**new** ValuesSet();

String file, fileO;

**private** **int** nperiods1, nperiods2;

**double** commission,aplha;

**private** **static** String *prof*;

**public** Execution2(String file, String fileO, **int** nperiods1,

**int** nperiods2, **double** comission, **double** aplha) **throws** IOException {

**this**.aplha=aplha;

**this**.comission=comission;

**this**.file=file;

WritetoFiles a =**new** WritetoFiles();

*val* = a.Values(file);

**this**.fileO=fileO;

**this**.nperiods1=nperiods1;

**this**.nperiods2=nperiods2;

System.*out*.println(nperiods1+" "+nperiods2);

}

@Override

**public** **void** run() {

**if**(s){

movinaverages m = **new** movinaverages();

ValuesSet v = **new** ValuesSet();

Profits profit = **new** Profits();

File f =**new** File(fileO);

WritetoFiles wf= **new** WritetoFiles();

**try** {

sma1=m.SMA(nperiods1, *val*);

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

sma2=m.SMA(nperiods2, *val*);

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

setProf(profit.doCompare(sma1, sma2, *val*, comission));

} **catch** (IOException e) {

e.printStackTrace();

}

s=**false**;

}

}

**public** **void** stop(){

**this**.s=**false**;

}

**public** String getProf() {

**return** *prof*;

}

**public** **void** setProf(String string) {

**this**.*prof* = string;

}

**public** **int** getNperiods1() {

**return** nperiods1;

}

**public** **void** setNperiods1(**int** nperiods1) {

**this**.nperiods1 = nperiods1;

}

**public** **int** getNperiods2() {

**return** nperiods2;

}

**public** **void** setNperiods2(**int** nperiods2) {

**this**.nperiods2 = nperiods2;

}

}

This implementation for this certain method would give the user a relatively good running time but, would leave the Thread controller to access each object through the setters provided by the implementation.

Also I have tried a different approach to solve this problem, this time I have used a different interface: Callable that suits this types of problems better that the Runnable interface because this certain algorithm uses threads only to return an isolated result that will be later used in the execution manager class also each thread uses one of the moving averages algorithm to implement a single result depending on different parameters but not communicating for an atomic result( an atomic result for this algorithm is basically a profits margin resulted from the calculation of a SMA with a window of certain size with another SMA with a different window size). For these reasons I have implemented the algorithm in this manner:

/\*\*This class aims to build one execution of a thread

\* that will output the profits with the given

\* parameters

\* **@author** alexandru

\*

\*/

**public** **class** ExecutionEma **implements** Callable<ArrayList<String>>, ExecuteService {

**protected** ValuesSet val = **new** ValuesSet();

Thread t;

ArrayList<String> profits=**new** ArrayList<String>();

**protected** ValuesSet ema1 = **new** ValuesSet();

**protected** ValuesSet ema2 = **new** ValuesSet();

String file;

String fileO;

**int** nperiods1;

**int** nperiods2;

**double** comission;

**double** alpha;

String prof;

String prof2;

/\*\*The constructor of this class is used

\* to pass values of the parameters that

\* are defined as global variables

\* **@param** file: file where the values are

\* **@param** fileO: file where the values will be written

\* **@param** nperiods1: the first SMA window of days

\* **@param** nperiods2: the second SMA window of days

\* **@param** comission: the practiced commission for this type of transactions

\* **@param** aplha: the exponential factor

\* **@throws** IOException: in case the reading file or the output file is corrupt

\*/

**public** ExecutionEma(String file, String fileO, **int** nperiods1,

**int** nperiods2, **double** comission, **double** alpha) **throws** IOException {

**this**.alpha = alpha;

**this**.comission = comission;

WritetoFiles a = **new** WritetoFiles();

val = a.Values(file);

**this**.fileO = fileO;

**this**.nperiods1 = nperiods1;

**this**.nperiods2 = nperiods2;

}

/\*\*This method will provide the needed implementation to output

\* the profits of a EMA method with a certain input

\*/

@Override

**public** ArrayList<String> call() **throws** Exception {

movinaverages m = **new** movinaverages();

ValuesSet v = **new** ValuesSet();

Profits profit = **new** Profits();

File f = **new** File(fileO);

ValuesSet a = **new** ValuesSet();

**try** {

**for** (**int** i = 2; i < 100; i += 2)

**for** (**int** j = i + 2; j < 100; j += 2) {

ema1 = m.EMA(i, alpha, val);

ema2 = m.EMA(j, alpha, val);

prof = profit.doCompare(ema1, ema2, val, comission);

profits.add(prof+" "+i+" "+j+" EMA");

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** profits;

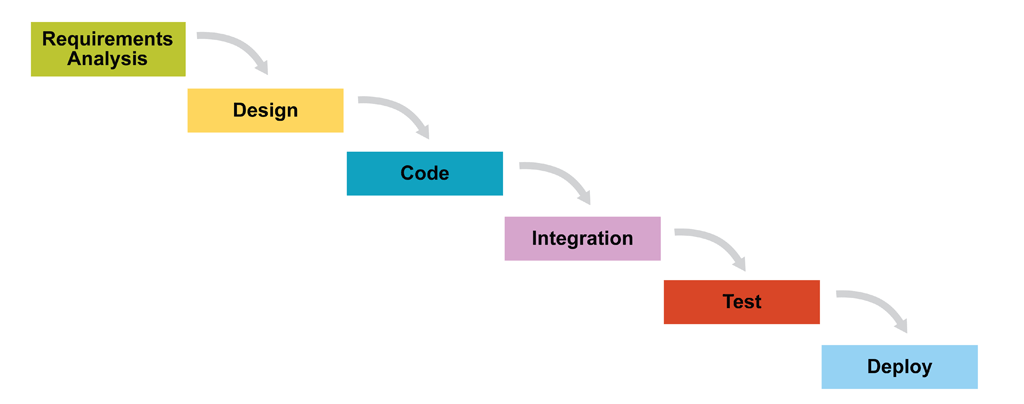
}

}

# Validation

Testing scenarios to prove the accomplishment of the documented requirements.

As it was mentioned earlier we are going to develop this application using Agile methodology. The testing was done according to Agile methods, once with every iteration the testing belonging to it was also made. Testing the application was made gradually, leading to well tested and well integrated application (due to the Agile methodology used).



For each component implemented firstly the requirements were analyzed, afterwards a design was conceived to integrate the component, code was written, the component was integrated and then it was tested and submitted to be a functional part of the entire application.

The CoMeth component, Moving Averages component has been divided into 3 parts that were treated as a subcomponent of it and all of the subcomponents were dealt with as a single component that after being developed and tested must be integrated in the CoMeth component. The 3 subcomponents also represent each a iteration as mentioned earlier:

🡪Implementation of the Moving Averages strategy

🡪Multithreading of the algorithm

🡪Implementation of the grid interfacing

For each subcomponent a testing plan was build that was to be respected in the coding and testing of the component:

**Testing plan for the Implementation of the Moving Averages strategy:**

This component aims to offer two types of MA indicators: EMA and SMA. It must implements accurate and trustful results.

This component is formed by two different ways of calculating the MA indicator and also by a method of calculating the profits resulted from applying the indicator.

Test plan objectives:

* Testing of the calculation implementation, through comparison with other methods of calculating the indicators
* Testing the correctness of the resulted profits, through a verification of each calculation
* Testing of the execution time and interaction with values from the databse

Aspects that will be tested:

* MA indicators
  + Correctitude of the results
  + Strategy accuracy of the profits strategy
  + Execution time
  + Interaction with the database

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name of test case | Steps | Expected results | Functionality | Status |
| 1 | Interaction with the database | 1.Opening the database  2.Querying the database through a parametrized function  3.Using the resulted values | A datastructure with all the values of a title | The values are extracted as desired | good |
| 2 | Correctitude of the results | 1.Calling one of the moving averages method SMA or EMA  2.Checking the results with a auxiliar device | A correct result of the indicator | Method is implemented correctly and the result are correct |  |
| 3 | Strategy accuracy of the profits strategy | 1.Running a MA of short term and a MA of long term and each interssection means a short or a long position | The overall profits gained | 1.algorithm was running only long positions  2.ok, algorithm was alternating long and short | solved |
| 4 | Execution time | 1.Time frames are integrated for the begining and the end of the execution | The running time is as expected under 10 minutes for 5 titles | 1.ok |  |

**Testing plan for the Implementation of Multithreading:**

This subcomponent aims to acquire a better execution time through the usage of two threads instead of one.

This subcomponent is composed from two different types of executions that will became two different threads and a execution manager that gathers the results given by each thread and orders it ascending

Test plan objectives:

* Testing the parallel execution management
* Testing the improvement of using multithreading

Aspects that will be tested:

* Threads
  + Return values of each thread
* Thread Management
  + The management of the threads, the moment of splitting and the moment of reuniting
  + Order of the result
  + Execution time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name of test case | Steps | Expected results | Functionality | Status |
| 1 | Return values of each thread | 1.Running the new  Parallel algorithm and comparing the result with the serial algorithm result | It is expected that both results are the same | The values are equal |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name of test case | Steps | Expected results | Functionality | Status |
| 1 | The management of the threads, the moment of splitting and the moment of reuniting | 1.Running the execution manager class and releasing different threads at a certain point, aftweards calling the result of a thread | Each thread start and ends when it is said to by the execution manager class | Each thread begin and ends when expected |  |
| 2 | Order of the result | 1.Each thread return its execution result  2.Call a function that must order the results | The result will be ordered correctly | The values are ordered |  |
| 3 | Execution time | 1.Running the parallel algorithm  2.Using a time frame to tell us the execution time | The execution time must be smaller than the serial execution | The parralel execution is faster |  |

**Testing plan for the Implementation of the grid interfacing:**

This subcomponent aims to use the parameters that the analyst introduces it build a configuration file that is going to be read by the application and it will execute the mentioned computational method as desired by the user. Than provide the needed execution logic that will be distributed on the grid.

This subcomponent is composed from a file reader that will call the methods as mentioned in the configuration file and from method that creates work unites.

Test plan objectives:

* Testing the usage of the configuration files
* Testing the execution of different work units

Aspects that will be tested:

* Configuration file
  + Building of different configuration files
  + Reading and execution the application according to the configuration file specifications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Name of test case | Steps | Expected results | Functionality | Status |
| 1 | Building of different configuration files | 1.Running the methods that builds the configuration files. | Each build must respect a certain type of workunit and build the configuration file as such. | The configuration files are build corectly |  |
| 2 | Reading and execution the application according to the configuration file specifications | 1.Executing a method that reads the configuration file and start the execution of the application with the specifications mentioned in the file | The program will execute and give the expected output | The application ouputs what was expected |  |

# Conclusion and future work

# Bibligrafie

[1] Q&A: ScaleOut Software's Dave Brinker

[2] Grid Computing in Financial Services--Toward True IT Agility Stevan D. Vidich, PhD, MCSD, SCJP, Industry Architect, Financial Services Group, Microsoft Corporation

[3] Technical Analysis and Stochastic Properties of Exchange Rate Movements: Empirical Evidence from the Romanian Currency Market Alexandru TODEA ,Adrian ZOICAŞ-IENCIU

[4] WILLIAM D Gann - Stock Market Forecasting Courses

[5] Edwards, R.; McGee, J.; Bessetti, W. H. C. (2007). Technical Analysis of Stock Trends. CRC Press. ISBN 978-0-8493-3772-7.

[6] http://www.investopedia.com/terms/s/short.asp

[7] An Analysis of the Maximum Drawdown Risk Measure Malik Magdon-Ismail Dept. Computer Science Rensselaer Polytechnic Institute Troy, NY 12180, USA.

[8] Hydrologic Variability of the Cosumnes River Floodplain (Booth et al., San Francisco Estuary and Watershed Science, Volume 4, Issue 2, 2006)

[9] Moving Averages Simplified By Clif Droke MARKETPLACE BOOKS Columbia, MD

[10] A Library Grid Network for Enhancing Learning in Nigerian Universities. O.K. Boyinbode R.O .Akinyede, and O.S. Adeola Department of Computer Science, Federal University of Technology, Akure, Nigeria.

[11] Thinking in Java – Bruce Eckel /Thinking in patterns – Bruce Eckel